DEVELOPING ENSEMBLE METHODS FOR PREDICTING TRAIN DELAYS IN REAL-TIME

Masoomeh Golchehreh



A thesis presented for the degree of Doctor of Philosophy
School of Computing Sciences
University of East Anglia
United Kingdom JULY, 2026

© This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with the author and that use of any information derived there from must be in accordance with current UK Copyright Law. In addition, any quotation or extract must include full attribution

Abstract

Train delays are a common problem for railway systems in the United Kingdom and many other countries. They affect passengers, railway services, and daily operations. More accurate predictions can help railway operators prevent or reduce delays. Although many studies have sought to improve prediction accuracy, most have focused on individual models or simple ensemble methods and have relied on a limited number of evaluation metrics. Consequently, the accuracy and reliability of these models remain unsatisfactory under different conditions. In addition, despite the importance of real-time prediction, research in this area remains impractical.

To address these limitations, this research develops two ensemble methods for train delay prediction: the Multi-Metric Weighted Ensemble Method (MWEM) and the Multi-Metric Bayesian-Optimised Ensemble Method (BOEM). MWEM assigns model weights based on performance across multiple evaluation metrics and validation rounds, whereas BOEM uses Bayesian optimisation during each validation round to identify an effective combination of model weights. BOEM is then integrated into a software prototype for real-time delay prediction.

To evaluate the proposed methods, historical and live data from several UK railway routes were used. Statistical analysis showed that the proposed ensemble methods performed significantly better than the individual machine learning and deep learning models and the ensemble baselines. In the historical five-fold cross-validation experiments across both railway routes, BOEM reduced MAE by approximately 7% compared with MWEM and by approximately 8% compared with the strongest individual model. In the real-time evaluation, BOEM reduced MAE by approximately 11% compared with the available railway estimates and by approximately 9% in recursive downstream prediction. Further experiments showed that the proposed method remained more reliable under missing-value and cross-dataset conditions.

Overall, the findings show that the proposed multi-metric ensemble methods can improve train delay prediction across historical, real-time, and additional evaluation settings. This research, therefore, provides a practical framework for combining different prediction models and applying them to railway delay prediction.

Access Condition and Agreement

Each deposit in UEA Digital Repository is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the Data Collections is not permitted, except that material may be duplicated by you for your research use or for educational purposes in electronic or print form. You must obtain permission from the copyright holder, usually the author, for any other use. Exceptions only apply where a deposit may be explicitly provided under a stated licence, such as a Creative Commons licence or Open Government licence.

Electronic or print copies may not be offered, whether for sale or otherwise to anyone, unless explicitly stated under a Creative Commons or Open Government license. Unauthorised reproduction, editing or reformatting for resale purposes is explicitly prohibited (except where approved by the copyright holder themselves) and UEA reserves the right to take immediate ‘take down’ action on behalf of the copyright and/or rights holder if this Access condition of the UEA Digital Repository is breached. Any material in this database has been supplied on the understanding that it is copyright material and that no quotation from the material may be published without proper acknowledgement.

Acknowledgements

I am deeply grateful to my primary supervisor, Professor Wenjia Wang. My academic journey with him began when I was a student in his Artificial Intelligence module, which first introduced me to the fields of artificial intelligence and data science. Since then, his guidance, encouragement, and support have been a constant part of my studies and PhD research. I greatly appreciate his patience and the considerable time and effort he devoted to reviewing my work, discussing my ideas, and helping me work through the challenges I encountered along the way. His support has played an important role in helping me reach this stage.

My sincere thanks also go to my secondary supervisor, Professor Gerard Parr, for his valuable advice, encouragement, and support. His experience and thoughtful feedback contributed greatly to the development of this research.

I am grateful to the University of East Anglia, particularly the School of Computing Sciences, for providing the academic environment, facilities, and support that made this research possible. I am also thankful to Professor Beatriz De La Iglesia, Head of the School of Computing Sciences, and Ben Milner for their support and encouragement during my time at UEA.

A very special thank you goes to my family for their unconditional love, patience, and constant support. In particular, I am deeply grateful to my mother, Zahra, my father, Manouchehr, my brother, Ali, and my sister, Reyhaneh. They believed in me throughout every stage of this journey and gave me the strength to keep going, especially during the more difficult moments. This achievement would not have been possible without them.

I am also very thankful to my colleagues, Debbie Taylor, Eleanor Leist, for their kindness and support throughout my PhD journey, and to my mentor, Jeannette Chin, for her guidance and encouragement. My special thanks go to Kazhan Misri, who has been not only a colleague but also a dear friend. Her kindness, friendship, and support have meant a great deal to me.

I would also like to thank my dear friends, Professor Mike and Gill Hulme, for their kindness, care, and support throughout this journey. I am also grateful to all my friends who encouraged me and stood by me during my PhD.

I also gratefully acknowledge National Rail Open Data and the Rail Delivery Group for providing access to the data used in this research.

This journey has not always been easy, but I have been fortunate to have so many supportive people around me. Their encouragement, kindness, and belief in me helped me reach the end of this journey, and I am sincerely grateful to all of them.

Acronyms

Abbreviation	Meaning
AB	AdaBoost
ACC@1	Accuracy within one minute
ACC@k	Accuracy within a tolerance of k minutes
AE	Average Ensemble
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
API	Application Programming Interface
BO-XGBoost	XGBoost with Bayesian optimisation
BOEM	Multi-Metric Bayesian-Optimised Ensemble Method
BOEM-DL	Deep-learning-only BOEM configuration
BOEM-ML	Machine-learning-only BOEM configuration
BR	Bayesian Ridge
CD	Critical Difference
CFD	Coincident Failure Diversity
CNN	Convolutional Neural Network
COR	Correlation
DL	Deep Learning
DT	Decision Tree
ELM	Extreme Learning Machine
ELM-PSO	Extreme Learning Machine with Particle Swarm Optimisation
ENet	Elastic Net
ET	Extra Trees
FTT	Feature Tokenizer Transformer (FT-Transformer)
GB	Gradient Boosting Decision Tree
GiB	Gibibyte
GK	Gaussian Kernel
GPU	Graphics Processing Unit
HSP	Historic Service Performance
KNN	k-Nearest Neighbours
LightGBM	Light Gradient Boosting Machine
LR	Linear Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
Markov-GK	Markov model with Gaussian-kernel estimation
MiB	Mebibyte
ML	Machine Learning
MLP	Multilayer Perceptron
MSM1-WE	Model-selection method 1 using a weighted ensemble
MSM2a-COR-WE	Model-selection method 2a using correlation and a weighted ensemble
MSM2b-CFD-WE	Model-selection method 2b using coincident failure diversity and a weighted ensemble
MWEM	Multi-Metric Weighted Ensemble Method
RAM	Random Access Memory
RF	Random Forest

Continued on next page

Abbreviation	Meaning
RID	Real-Time Train Information Identity
RMSE	Root Mean Squared Error
RMSD	Root Mean Square Deviation
RNN	Recurrent Neural Network
RQ	Research Question
SD	Standard Deviation
SHAP	SHapley Additive exPlanations
SQL	Structured Query Language
ST-NN	Stacking ensemble with a neural-network meta-learner
STOMP	Simple Text Oriented Messaging Protocol
SVM	Support Vector Machine
SVR	Support Vector Regression
TabICL	Tabular In-Context Learning
TabM	Parameter-efficient tabular model combining multiple MLP predictors
TabNet	Attention-based deep-learning model for tabular data
TabPFN	Tabular Prior-Data Fitted Network
TOC	Train Operating Company
UK	United Kingdom
WE	Weighted Ensemble
XGB	XGBoost (Extreme Gradient Boosting)
Anonymised route and dataset labels	
`-R`	Reverse-direction suffix for an anonymised route label
EB	Anonymised railway route label
LK	Anonymised railway route label
NL	Anonymised main railway route label
NL-P1	NL route dataset from Phase 1
NL-P2	NL route dataset from Phase 2
P1	Phase 1
P2	Phase 2
WW	Anonymised main railway route label
WW-P1	WW route dataset from Phase 1
WW-P2	WW route dataset from Phase 2

Publications

The following publications have resulted from the research presented in this thesis:

1. Golchehreh, M., Parr, G., and Wang, W. (2026). Multi-Metric Weighted Ensemble Method for Train Delay Prediction. In *2026 11th International Conference on Machine Learning Technologies (ICMLT)*, pp. 511–517. IEEE.

2. Golchehreh, M., Parr, G., and Wang, W. (2026). Bayesian-Optimised Hybrid Ensembles for Predicting Train Delays Accurately and Robustly. [Under Review]

3. Golchehreh, M., Parr, G., and Wang, W. (2026). Real-Time and Recursive Train Delay Prediction Using a Bayesian-Optimised Ensemble Framework. [In Progress]

Table of Contents

Abstract ii

Acknowledgements iii

Acronyms iv

Publications vi

Table of Contents vii

List of Tables x

List of Figures xii

1 Introduction 1
- 1.1 Research Background and Motivation . . . 1
- 1.2 Aim, Objectives and Research Questions . . . 4
- 1.3 Scope of the Research . . . 5
- 1.4 Contributions of the Research . . . 6
- 1.5 Outline of the Thesis . . . 8

2 Literature Review 10
- 2.1 Introduction . . . 10
- 2.2 Train Delay Prediction Research . . . 11
- 2.3 Ensemble Learning . . . 20
- 2.4 Evaluation, Robustness, and Generalisation . . . 23
- 2.5 Research Gaps and Research Positioning . . . 24
- 2.6 Summary . . . 26

3 Research Methodology 31
- 3.1 Introduction . . . 31
- 3.2 Overall Research Strategy . . . 33
- 3.3 Railway Journey Representation for Delay Prediction . . . 35
- 3.4 Delay Representation . . . 39
- 3.5 Problem Formulation and Prediction Framework . . . 41
- 3.6 Overview of Data and Feature Use . . . 42
- 3.7 Experimental Design . . . 44
- 3.8 Evaluation Metrics . . . 47
- 3.9 Model Interpretability Using SHAP . . . 50
- 3.10 Experimental Setup . . . 51

3.11 Summary . . . 53

4 Data Collection, Preprocessing and Analysis 54
4.1 Introduction . . . 54
4.2 Historical Train Movement and Timetable Data . . . 55
4.3 Live Real-Time Train-Running Data . . . 56
4.4 Selected Routes, Dataset Phases, and Data Coverage . . . 56
4.5 Construction of Route-Section Records and Delay Values . . . 58
4.6 Data Cleaning and Quality Control . . . 60
4.7 Feature Engineering for the Final Dataset . . . 61
4.8 Model-Ready Preprocessing . . . 62
4.9 Final Dataset Summary . . . 63
4.10 Data Limitations . . . 64
4.11 Overall Comparison of Delay Variables . . . 64
4.12 Distribution of Arrival and Departure Delays . . . 66
4.13 Section-Level Variation in Delay Variables . . . 67
4.14 Hour of Day Variation in Delay Variables . . . 69
4.15 Correlation Between Variables . . . 70
4.16 Implications for Modelling and Evaluation . . . 72
4.17 Summary . . . 73

5 Proposed Ensemble Frameworks for Train Delay Prediction 74
5.1 Introduction . . . 74
5.2 Overview of the Proposed Modelling Framework . . . 77
5.3 Base Models . . . 78
5.4 Heterogeneous and Hybrid Ensemble Framework . . . 80
5.5 Average Ensemble (AE) . . . 82
5.6 Multi-Metric Weighted Ensemble Method (MWEM) . . . 83
5.7 Computational Complexity of MWEM . . . 91
5.8 Multi-Metric Bayesian-Optimised Ensemble Method (BOEM) . . . 92
5.9 Computational Considerations . . . 101
5.10 Final Model Training and Ensemble Prediction . . . 101
5.11 Prediction Results Using Historical Train Movement Data . . . 104
5.12 Discussion of the Historical Evaluation . . . 133
5.13 Answers to Research Questions 1–4 . . . 145
5.14 Summary . . . 147

6 Real-Time Ensemble Framework for Train Delay Prediction 149
6.1 Introduction . . . 149
6.2 Overview of the Real-Time Framework and prototype dashboard . . . 150
6.3 System Architecture of the Real-Time Prediction Framework . . . 154
6.4 Live Data Preparation for Real-Time Prediction . . . 157
6.5 Departure Delay Estimation for Real-Time Prediction . . . 162
6.6 Preparation and Integration of BOEM into the Real-Time Prototype . . . 168
6.7 Service-Level Prediction Correction Using Confirmed Observations . . . 175
6.8 Recursive Downstream Delay Prediction . . . 176
6.9 Implementation Considerations for Live Operation . . . 179
6.10 Practical Challenges and Solutions . . . 180
6.11 Time-Aware Model Evaluation for Real-Time Framework . . . 183

6.12 Real-Time Prediction Results . . . 189
6.13 Discussion of the Real-Time Evaluation . . . 202
6.14 Answer to Research Question 5 . . . 207
6.15 Summary . . . 208

7 Conclusion and Further Work **209**
7.1 Main Findings . . . 209
7.2 Synthesis of Findings and Contributions to Knowledge . . . 210
7.3 Scope and Limitations of the Findings . . . 213
7.4 Further Work . . . 213
7.5 Final Remarks . . . 214

8 Appendix **216**
8.1 Use of Generative Artificial Intelligence . . . 216
8.2 Introduction . . . 216
8.3 Description of Final Dataset Features . . . 216
8.4 Base Model Descriptions and Hyperparameter Settings . . . 221
8.5 Example of a Raw STOMP Message . . . 233
8.6 Operational Dashboard Prediction Fields . . . 234
8.7 Summary . . . 234

List of Tables

2.1 Comparative summary of reviewed train delay prediction studies. . . . 27

3.1 Relationship between the research questions, investigations and evaluation evidence. 34
3.2 Main software and computing specifications used in the experiments. . 51

4.1 Overview of the railway data sources used in this research. 55
4.2 Anonymised railway routes and data periods. 58
4.3 Main variables in the final datasets. 62
4.4 Summary of the final route datasets. 63
4.5 Arrival-delay characteristics of the four datasets. 65

5.1 Principal mathematical notation used in this chapter. 76
5.2 Base model groups used in the proposed ensemble framework. 79
5.3 Model-pool configurations used in the BOEM framework. 81
5.4 Mean test performance of all models across all sections. 105
5.5 Cost-based evaluation of BOEM and the individual baseline models. Average cost is reported with asymmetric penalties ($c_{\text{under}} = 3$, $c_{\text{over}} = 1$). Lower is better. 121
5.6 Mean MAE across all evaluated missing-value levels, with standard deviations shown in parentheses, and mean MAE degradation. 125
5.7 Paired Wilcoxon signed-rank comparisons of MAE degradation slopes between BOEM and the base models across the 18 route sections. BOEM achieved a mean degradation slope of 0.015. 127
5.8 Comparison with published train delay prediction methods on the NL route. 131
5.9 Comparison with published train delay prediction methods on the WW route. 132
5.10 Overall comparison with published train delay prediction methods across all route sections. 132

6.1 Mean and standard deviation of inference time per prediction. 189

6.2 Summary of the samples used for evaluating predictions generated during the 40-day live-data collection period. 191

7.1 Mapping between research questions, investigations, evidence and contributions to knowledge. 212

8.1 Description of the main features used in the final modelling dataset. . . 217
8.2 Hyperparameters considered for the base models. 226
8.3 Description of the fields displayed in the operational dashboard prediction tables. 234

List of Figures

1.1 UK passenger rail performance between 2015 and 2025. Source: Author's analysis of Office of Rail and Road data (Office of Rail and Road, 2026). 2

3.1 Workflow supporting model development, experimental evaluation and real-time prediction. 32
3.2 One-to-one station multi-model representation for consecutive station-to-station delay prediction. 36
3.3 All-to-next stations single-model representation using one global model for different station pairs. 37
3.4 All-to-next stations multi-model representation using separate models for all possible station pairs. 38
3.5 Examples of departure delay, arrival delay, and dwell delay based on planned and actual train timings. 40

4.1 Example of raw station-level railway records. 58
4.2 Example of processed route-section records. 60
4.3 Positive-delay rates for arrival, departure, and dwell delays across the four datasets. 65
4.4 Mean arrival, departure, and dwell delays across the four datasets. . . . 66
4.5 Arrival and departure delay distributions across the four datasets. . . . 67
4.6 Positive-delay rates across route sections in the four datasets. 68
4.7 Mean delay values across route sections in the four datasets. 68
4.8 Mean arrival delay by hour of day for Phase 1 and Phase 2. 69
4.9 Positive arrival-delay rate by hour of day for Phase 1 and Phase 2. . . . 70
4.10 Correlation matrices for the NL and WW datasets in Phase 1 and Phase 2. 71

5.1 Overview of the proposed modelling framework and its connection to the real-time prediction framework. 77
5.2 Simplified overview of the Multi-Metric Weighted Ensemble Method (MWEM). 85

5.3 Simplified overview of the Multi-Metric Bayesian-Optimised Ensemble Method (BOEM). 95
5.4 Performance of the BOEM-ML variants using single-metric validation and the proposed multi-metric validation strategy across both routes. . 107
5.5 Section-level prediction performance on the NL route. 108
5.6 Section-level prediction performance on the WW route. 109
5.7 Prediction error across cumulative positive-delay bins for both routes. . 111
5.8 Prediction performance across actual arrival-delay intervals. 112
5.9 Global SHAP analysis for the Gradient Boosting model using the top 15 features. 115
5.10 Average ranking and rank variability of models across sections. Lower average rank indicates better relative performance. 116
5.11 Taylor diagram comparing the individual base models and BOEM across both routes. 117
5.12 Nemenyi Critical Difference diagrams for model performance. Connected models do not differ significantly at $\alpha = 0.05$. 119
5.13 MAE performance of the proposed method and the individual ML models as the proportion of missing departure-delay values increases from 0% to 50%. 124
5.14 Cross-dataset evaluation by section. BOEM, shown by the highlighted line, and the base models were trained on the source route and tested on the target route. 130
5.15 Nemenyi Critical Difference diagram for RMSE in the comparison with published train delay prediction methods. 133

6.1 Main stages of the proposed real-time train delay prediction framework. 151
6.2 Prototype dashboard for monitoring real-time BOEM-based train delay predictions. 153
6.3 System architecture of the proposed real-time train delay prediction framework. 154
6.4 Hierarchical departure delay estimation strategy used when actual departure information is unavailable. 165
6.5 Ordered model selection and fallback strategy for real-time ensemble prediction. 174
6.6 Overall time-aware evaluation performance of the models prepared for the real-time prediction framework. 185

6.7 Route-level time-aware evaluation performance of the real-time prediction models. Error bars show the standard deviation across sections within each route. 186
6.8 Section-level time-aware evaluation results comparing BOEM with the median-based summary references 188
6.9 Overall direct real-time prediction performance comparing BOEM predictions with railway estimates. Error bars indicate variation around the mean estimates. 192
6.10 Route-level direct real-time prediction performance under actual and estimated departure information. Error bars indicate route-level variation. 194
6.11 Overall recursive downstream prediction performance based on mean chain-level errors. 195
6.12 Route-level recursive prediction performance compared with simple baselines. 196
6.13 Overall recursive downstream prediction performance compared with railway next-station estimates. 197
6.14 Route-level recursive prediction performance compared with railway next-station estimates. 198
6.15 Recursive prediction performance across downstream chain horizons. . . 199
6.16 Route-specific recursive chain prediction performance across downstream horizons. 201

8.1 Example of a raw railway message received through the STOMP connection before filtering and transformation. 233

Chapter 1

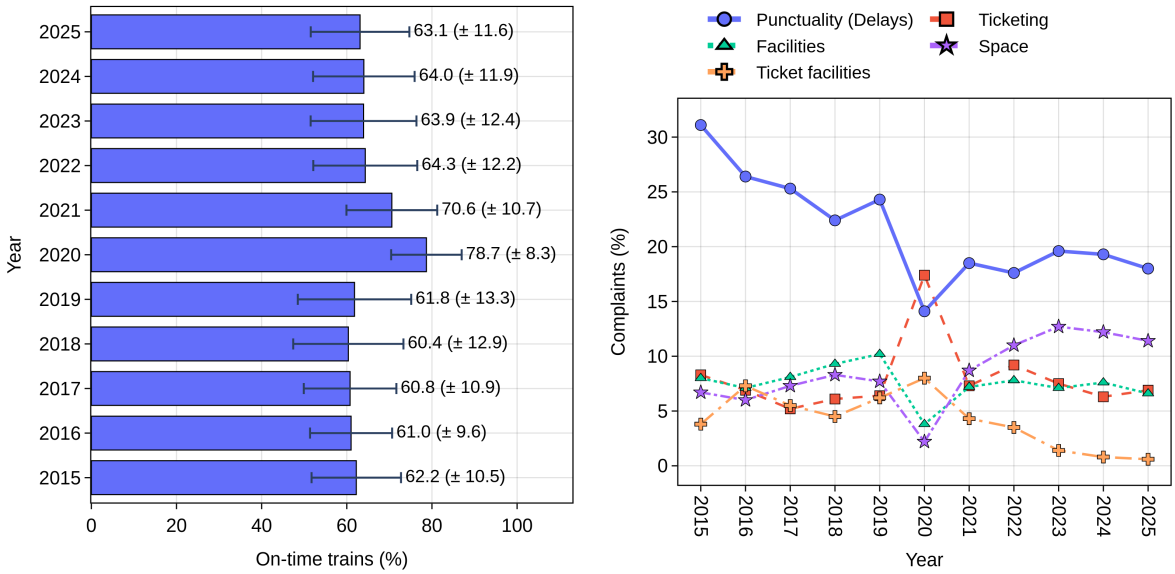
Introduction

1.1 Research Background and Motivation

Railway networks are an important part of transport systems in the UK and many other countries. They support daily travel, economic activity, and more sustainable forms of mobility. Many passengers depend on rail services for work, education, leisure, and other essential journeys. Reliable services allow passengers to plan their journeys with confidence, while delays and cancellations can cause uncertainty, inconvenience, and reduced trust in the railway system (Network Rail, 2026).

The importance of reliable services means that railway organisations need to monitor how closely trains follow their planned timetables. In the UK, railway performance is commonly measured using indicators such as On Time, the Public Performance Measure, delay minutes, and cancellations (Network Rail, 2022). Recent data show that train delays remain a continuing problem in the UK. In this study, data published by the Office of Rail and Road were used to examine annual punctuality and the main categories of passenger complaints between 2015 and 2025. As shown in Figure 1.1(a), approximately 65% of trains arrived on time on average during this period, where *on time* means arriving within one minute of the scheduled arrival time. Figure 1.1(b) also shows that complaints related to delays remained among the main passenger complaints. Together, these findings show that train delays continue to be an important operational and passenger-related problem in the UK railway network.

Since trains share tracks, platforms, junctions, rolling stock, staff, and timetable slots, predicting train delays and their effects is difficult. A problem affecting one train can delay later parts of its journey and may also affect other services. This risk becomes



(a) Average punctuality (b) Five main passenger complaint categories

Figure 1.1: UK passenger rail performance between 2015 and 2025. Source: Author's analysis of Office of Rail and Road data (Office of Rail and Road, 2026).

greater when the network is busy and there is little spare time in the timetable to absorb disruption (Xu et al., 2023).

As a result of these connections, train delays can change as a train moves along its route. A delay may increase, decrease, or spread from one route section to the next. Therefore, predicting only the final delay may not provide enough information. Predicting the delay for each route section can show how it develops during the journey and where further disruption may occur.

Producing section-level prediction requires models that can learn how delays change from one part of a journey to the next. Railway systems produce large amounts of historical timetable and performance data that can be used to train these models. Machine learning and deep learning models can identify patterns between previous delays and the delay at the next station. Therefore, they provide a useful data-driven approach to train delay prediction.

Deep learning (DL) is a subset of machine learning (ML), while ensemble learning is a strategy for combining multiple predictive models. In this thesis, the distinction between ML and DL models is used as a practical naming convention for comparing two predefined model pools. The ML pool comprises established baseline methods, including regression, tree-based, kernel-based, instance-based and Multilayer Perceptron models, whereas the DL pool comprises modern neural models developed for tabular

data. Both pools therefore belong to machine learning; their separate labels identify the experimental groups rather than mutually exclusive theoretical categories. This distinction allows the research to examine whether combining established baseline predictors with modern tabular neural predictors improves performance compared with using either pool alone. Ensemble learning can operate within either pool or across both, and some baseline methods, such as Random Forest and Gradient Boosting, are themselves ensembles. The term "hybrid ensemble" specifically denotes the combination of predictors from both pools through prediction aggregation.

However, different models may work better in different situations. No single model gives the best results for every route or route section. Some models may be more accurate, while others may produce more consistent results or learn complex patterns better. Heterogeneous ensemble learning can help address this problem by combining different types of machine learning and deep learning models. Simply combining more models, however, does not always improve prediction performance. The results depend on which models are selected and how much weight each model receives. Simple averaging gives all models the same weight, while fixed weights may not work equally well across different routes or route sections.

Another limitation is that ensemble models are often selected or weighted using only one performance metric, such as MAE or RMSE. However, a single metric cannot represent all aspects of prediction quality. A model may achieve a low average error but still produce large errors or less stable predictions. This creates a need for ensemble methods that use multiple performance measures when selecting models and assigning their weights.

To address this limitation, multi-metric weighting provides a way to consider different aspects of model performance when building an ensemble. The weights can be calculated directly from several evaluation metrics or optimised using Bayesian optimisation. The first approach is simple and transparent, while the second can search for a better combination of model weights. Comparing these approaches can show whether optimisation provides a clear improvement over direct metric-based weighting.

However, improving the weighting process alone is not enough for practical railway use. A model that performs well on one dataset may not achieve the same results

under different conditions. Train delay patterns can vary across routes, route sections, delay levels, and operating conditions, while real-time values may also be incomplete or unavailable. Therefore, an ensemble model should not only be accurate, but should also remain reliable across different routes and less complete data values conditions.

The practical value of a prediction model depends on whether it can be used while a train is operating. In this setting, new information becomes available as the train moves between stations, and the prediction needs to change with it. Historical records are useful for training the models, but live operation also requires current movement data and a process that can work when some information is missing. This makes it possible to estimate the delay at the next station and then extend the prediction to later sections of the journey.

Taken together, these issues show that the main problem is not simply to predict whether a train will be delayed. The challenge is to follow how arrival delay changes along a journey and to keep the predictions reliable under different route and data conditions. This research addresses this problem by combining machine learning and deep learning models through multi-metric ensemble methods and applying them to section-level prediction using historical and live railway data.

1.2 Aim, Objectives and Research Questions

The main aim of this research is to develop and evaluate an ensemble-based framework for route-section-level train arrival delay prediction. The research combines machine learning and deep learning models and uses multiple performance metrics to support model selection and weighting. It also examines whether the proposed methods can provide accurate and reliable predictions across different railway routes, data conditions, and real-time operational settings. To achieve this aim, the main objectives of the research are as follows:

- To collect, process, and prepare historical and real-time railway data and develop a reusable data preparation pipeline for multiple routes and route sections.
- To develop and evaluate a range of individual machine learning and deep learning models for train arrival delay prediction under consistent experimental conditions.

- To develop multi-metric heterogeneous ensemble methods that combine machine learning and deep learning models using metric-based and Bayesian-optimised weighting strategies.

- To compare the proposed ensemble methods with individual models and baseline ensemble methods in terms of prediction accuracy.

- To examine the generalisation and reliability of the proposed models under different evaluation conditions.

- To develop and evaluate a real-time prediction framework that processes live railway updates, generates next-section and recursive downstream predictions, and presents the prediction outputs through a prototype monitoring dashboard.

Based on these objectives, this research seeks to answer the following research questions:

RQ1. Can multiple performance metrics be used to select and weight models to improve the accuracy of a heterogeneous ensemble?

RQ2. How can a multi-metric ensemble be optimised to improve train delay prediction?

RQ3. Does a hybrid machine learning–deep learning ensemble perform better than machine learning-only and deep learning-only ensembles?

RQ4. Does the proposed ensemble provide more reliable performance than other models across different railway routes and evaluation settings?

RQ5. Can the proposed ensemble framework predict train arrival delays in real time?

1.3 Scope of the Research

This research focuses on route-section-level train arrival delay prediction for selected passenger railway routes in the UK. Each journey is represented as a sequence of sections between consecutive stations, and the main prediction target is the arrival delay at the next station. Historical railway records and timetable data are used to develop and evaluate the models, while live operational updates are used for real-time prediction.

The research examines individual machine learning and tabular deep learning models, together with the proposed heterogeneous ensemble methods. Their accuracy and reliability are evaluated under different experimental conditions. The real-time framework generates next-section and downstream delay predictions, and a prototype dashboard is developed to present the prediction outputs.

The real-time framework and dashboard developed in this thesis are research prototypes evaluated using live railway data. In this context, model deployment refers to integrating offline-trained models into the prototype. Potential applications in passenger information and railway decision support are discussed, but industrial deployment and operational benefits were not evaluated.

The research is limited to predicting the amount of arrival delay. It does not identify the causes of delays or distinguish between primary and reactionary delays. Train cancellation prediction, timetable optimisation, service rescheduling, and direct railway traffic control are outside the scope of the study. The selected routes are treated as case studies, so the findings are limited to the routes, datasets, time periods, and evaluation conditions examined.

1.4 Contributions of the Research

The main contributions to knowledge are organised around the research questions. They arise from the investigation of the Multi-Metric Weighted Ensemble Method (MWEM) and the Multi-Metric Bayesian-Optimised Ensemble Method (BOEM), and concern the findings obtained about weighting, optimisation, model combination, reliability and real-time prediction. The route-section formulation, data preparation pipeline and model implementation provide the methodological foundation for these investigations. Table 7.1 maps each contribution to its research question, investigation and supporting evidence. The claims apply to the datasets and evaluation conditions examined in this research.

- **C1. Effect of multi-metric ensemble weighting (RQ1).** The research provides sufficient empirical evidence to show that using complementary validation metrics across validation rounds can improve ensemble prediction compared with

using the same single metric throughout. In the controlled BOEM-ML comparison, the multi-metric configuration achieved stronger overall results across all four reported evaluation measures than the single-metric configurations. The MWEM comparison with equal-weight averaging provides additional evidence about the value of validation-based weighting. The supporting results are presented in Sections 5.11.1 and 5.11.2.

- **C2. Predictive performance of Bayesian ensemble-weight search (RQ2).** The research provides statistically strong evidence about the effectiveness of searching complete ensemble-weight combinations rather than deriving weights solely from individual model performance. In the reported comparisons, BOEM achieved lower overall test errors than MWEM and the other evaluated ensemble configurations. This establishes the predictive effectiveness of the proposed Bayesian weight-search configuration for the route-section prediction settings investigated. The supporting comparisons are presented in Section 5.11.1 and Table 5.4.

- **C3. Benefit of combining the defined ML and tabular DL model pools (RQ3).** The comparison of BOEM-ML, BOEM-DL and hybrid BOEM provides empirical evidence that combining the two predefined model pools improved prediction compared with using either pool alone. The configurations used the same ensemble-weight optimisation procedure, and hybrid BOEM achieved stronger overall results across all four main evaluation measures. This establishes a benefit of combining these particular model pools under the historical evaluation design used in this research. The supporting evidence is presented in Table 5.4.

- **C4. Reliability under missing-information and cross-route conditions (RQ4).** The research provides evidence about the behaviour of the proposed ensemble beyond aggregate prediction accuracy. BOEM-ML achieved lower average error and variation, together with slower error deterioration, when departure information was unavailable and reconstructed using the evaluated procedure. It also retained useful prediction performance in both transfer directions between the NL and WW routes. These findings establish performance

characteristics under the specific missing-information and route-transfer conditions investigated. The supporting evidence is presented in Sections 5.11.9 and 5.11.10, alongside the section-level analysis in Section 5.11.3.

- **C5. Predictive value under live railway-data conditions (RQ5).** The live evaluation provides evidence that offline-trained section-specific ensemble models can generate useful direct and downstream arrival-delay predictions when operational information is incomplete or changing. BOEM achieved lower prediction errors than the corresponding railway-provided estimates under both confirmed and estimated departure-information conditions, and in the evaluated recursive comparison. This establishes predictive value within the selected routes and live monitoring conditions. The supporting evidence is presented in Sections 6.12.2 and 6.12.6.

1.5 Outline of the Thesis

The remainder of this thesis is organised as follows.

Chapter 1:Introduction. This chapter presents the research background, motivation, and main problem addressed in the research. It defines the research aim, objectives, research questions, scope, and main contributions of the study.

Chapter 2: Literature Review. This chapter reviews the main research related to train arrival delay prediction. It discusses individual, hybrid, and ensemble prediction methods, real-time prediction, model evaluation, robustness, and generalisation. It concludes by identifying the research gaps addressed in this thesis.

Chapter 3: Research Methodology. This chapter presents the overall research design and methodological framework. It explains the railway journey representation, delay measures, prediction formulation, experimental design, evaluation metrics, and implementation environment.

Chapter 4:Data Collection, Preprocessing and Analysis. This chapter describes the railway data sources, dataset construction, data cleaning, feature engineering and preprocessing procedures. It then examines delay distributions, variation across routes, dataset phases, route sections and operating hours, and

relationships between variables. The findings provide the data context for the subsequent modelling and evaluation.

Chapter 5: Proposed Ensemble Methods for Train Delay Prediction. This chapter develops AE, MWEM and BOEM and presents their historical evaluation. It discusses model comparisons, weighting strategies, reliability and generalisation, and answers RQ1–RQ4.

Chapter 6: Real-Time Train Delay Prediction. This chapter presents the research prototype, live-data processing, departure-delay estimation and direct and recursive prediction. It reports the time-aware and live evaluations, discusses their implications and answers RQ5.

Chapter 7: Conclusions and Future Work. This chapter summarises the findings and contributions to knowledge, considers the overall limitations and identifies directions for future work.

Chapter 2

Literature Review

2.1 Introduction

This chapter reviews the main research related to train delay prediction. It provides the background needed to understand the methods used in this thesis and examines how train delays have been studied using statistical approaches and machine learning (ML) methods, including deep learning (DL), together with ensemble strategies for combining predictors. It also reviews real-time prediction, model evaluation, robustness, and generalisation. The chapter then identifies the main limitations of existing studies and defines the research gaps addressed in this thesis.

As discussed in Chapter 1, train delays affect service reliability, passengers, and railway operations. They may be caused by infrastructure problems, signalling failures, train faults, bad weather, passenger incidents, or operational decisions (Zilko et al., 2016; Hauck and Kliewer, 2020). A delay may also spread to other services that use the same tracks, platforms, or junctions (Cacchiani et al., 2014; Corman and Kecman, 2018). These effects have led to a wide range of studies on train delay analysis and prediction.

Different methods have been used for train delay prediction. Earlier studies mainly used statistical, probabilistic, and simulation-based methods. With the wider availability of railway data, machine learning (ML) models, including deep learning (DL) models, have become more common (Spanninger et al., 2022; Tiong et al., 2023b). More recent studies have also examined graph-based, transformer-based, and tabular learning models. However, the performance of an individual model may vary across datasets, routes, and operating conditions. Ensemble learning has therefore been studied as a way of

combining predictions from several models (Mienye and Sun, 2022; Al Ghamdi et al., 2024).

Real-time train delay prediction introduces further challenges because live railway data arrive continuously and may be incomplete, delayed, or inconsistent (Zilko et al., 2016; Hauck and Kliewer, 2020). Prediction models must process these updates and produce results within a suitable time (Leng and Corman, 2022).

The remainder of this chapter is organised as follows. Section 2.2 reviews individual, hybrid, and ensemble approaches to train delay prediction, together with real-time prediction and its operational challenges. Section 2.3 discusses ensemble construction, model selection, diversity, weighting, and optimisation. Section 2.4 reviews model evaluation and validation, with a focus on robustness and generalisation. Section 2.5 identifies the main research gaps and explains how this research addresses them. Finally, Section 2.6 summarises the chapter.

2.2 Train Delay Prediction Research

This section reviews the main approaches used for train delay prediction, including individual, hybrid, ensemble, and real-time methods, with attention to their prediction problems, input data, modelling choices, and reported limitations. In this review, hybrid approaches are methods that combine a prediction model with another method or learning component.

2.2.1 Individual and Hybrid Prediction Approaches

Earlier research on train delay prediction focused mainly on understanding how delays develop and change during railway operations. To support this, many studies used statistical and probabilistic methods based on historical delay observations (Hallowell and Harker, 1996; Gaurav and Srivastava, 2018; Xu et al., 2023). Common approaches included regression, time-series models, and Markov-based methods. These methods were generally easy to understand and could describe changes in delay over time. However, they often relied on assumptions about railway operations and had limited ability to represent complex nonlinear relationships between trains, stations, and

infrastructure. Probabilistic methods were also used to represent uncertainty and the propagation of delays through railway services. Bayesian networks and stochastic models allowed relationships between train movements, infrastructure, and operational conditions to be included in the prediction process (Corman and Kecman, 2018; Huang et al., 2020a). These methods provided useful information about the probability of future delays. However, they often required detailed railway knowledge for model design and calibration. Their computational requirements could also increase when they were applied to larger railway networks.

A more recent probabilistic approach was proposed by Xu et al. (2023). The authors first examined whether delay changes across consecutive stations followed a Markov process. Their results supported a first-order representation in which the delay state at the next station depended mainly on the current delay state. Based on this result, they developed a non-homogeneous Markov model for short-term delay prediction. A Gaussian-kernel method was used to estimate transition values when only a small number of observations were available. The resulting Markov-GK method provided a simple and efficient approach for predicting delay changes between consecutive stations. It was also easier to interpret than many complex learning models. However, it represented delays using a set of discrete states and depended on transition patterns learned from a particular railway network. It may therefore have limited ability to use a wider range of operational features or represent relationships that are not included in the current delay state.

With the increasing availability of historical and operational railway data, ML models have been used more widely for train delay prediction. For example, Yaghini et al. (2013) developed an Artificial Neural Network model and compared it with a Decision Tree and multinomial logistic regression. Marković et al. (2015) compared Support Vector Regression (SVR) with an Artificial Neural Network (ANN). Later studies examined weather-aware ML methods, Extreme Learning Machines (ELM), Random Forest (RF), Gradient Boosting (GB), and related approaches (Oneto et al., 2016, 2018; Li et al., 2021b; Lapamonpinyo et al., 2022). These models became more popular because they can learn nonlinear relationships between railway features and train delays without requiring a complete mathematical description of delay behaviour.

ML studies have used different types of input information. These include timetable variables, previous arrival and departure delays, dwell times, traffic levels, weather conditions, and infrastructure characteristics (Barbour et al., 2018; Ardıç, 2024). The selected features depend on the railway system, the available data, and the prediction target. As a result, models developed for one railway route may not provide the same performance on another route. Tree-based models have been widely used because they can capture interactions between railway features and work well with structured data. Tree-based methods such as Random Forest and Gradient Boosting have also been applied to train delay prediction (Lapamonpinyo et al., 2022). Boosting methods build models in sequence, where each new model focuses on the errors made by the previous models. SVR has also been used for train delay prediction, particularly when the available dataset is not very large (Oneto et al., 2016). Shallow neural networks have been applied when more flexible nonlinear relationships need to be modelled (Oneto et al., 2018). Although these methods have produced useful results, their performance depends on feature preparation, hyperparameter selection, sample size, and validation design. Reviews of train delay prediction studies have shown that no individual model performs best in every dataset or prediction setting (Spanninger et al., 2022; Tiong et al., 2023b). This makes controlled comparison important when models are evaluated for a new railway route or prediction problem. An optimisation-assisted XGBoost method was proposed by Shi et al. (2021). Their BO-XGBoost method used Bayesian optimisation to select suitable XGBoost hyperparameters for next-station arrival delay prediction. The model included operational, spatial, temporal, and infrastructure-related variables and was evaluated using data from two high-speed railway lines. The study reported that Bayesian optimisation improved the performance of XGBoost compared with the model settings used as baselines. The authors also examined results across stations, delay ranges, and abnormal operating conditions. This provided a wider analysis than reporting only one overall performance value. However, BO-XGBoost remained an individual predictor based on one model family. It was also evaluated on high-speed railway data, which have different operational characteristics from the conventional UK intercity services examined in this research.

DL methods have been introduced to model more complex temporal and spatial

relationships. Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) models can learn from ordered train movement records and represent how delays change over time (Wen et al., 2020). These models are useful when previous events in a train journey affect later predictions. Hybrid DL architectures have also been developed. For example, convolutional neural network (CNN) and recurrent components have been combined to learn local patterns and temporal relationships from railway data (Wang et al., 2023a). Gao et al. (2023) proposed a model for short-term train delay prediction that combines Wavelet Transform, a Fully Connected Neural Network, and a Long Short-Term Memory network. Wavelet Transform was used to reduce random interference in the input data, while the neural components were used to learn temporal and nonlinear relationships. The model was evaluated using data from two Chinese high-speed railway lines and outperformed the RF, GBRT, XGBoost, and SVR baseline models. However, it remains one integrated DL architecture rather than an ensemble of independent predictors. Attention mechanisms have been added to some models to give more importance to relevant parts of the input sequence. In these methods, the term hybrid refers to the combination of learning components within one predictive architecture rather than the combination of several independent predictors. Railway systems can also be represented as networks. This has led to the use of Graph Neural Networks and spatio-temporal graph models. In these approaches, stations or railway locations are represented as nodes, while railway links or operational relationships are represented as edges. This allows the model to learn how delays may move across connected parts of the railway network (Ding et al., 2021; Li et al., 2021a).

Transformer-based models have also been studied because they can represent relationships across longer input sequences without relying only on recurrent processing. These methods may be useful for modelling train movements across several locations. However, their performance depends on the amount and structure of the available dataset. DL models usually need larger datasets, more computational resources, and careful hyperparameter tuning. Their outputs can also be more difficult to interpret than those of simpler models (Spanninger et al., 2022; Tiong et al., 2023b). In addition, DL models do not always perform better than classical ML models in every railway prediction task (Yang et al., 2024). More recently, several DL models have been

designed specifically for tabular data. Examples include TabNet, FT-Transformer, TabPFN, TabM, and TabICL (Arik and Pfister, 2021; Gorishniy et al., 2021; Hollmann et al., 2025; Gorishniy et al., 2025; Qu et al., 2026). These models use different methods for learning relationships between structured input features. TabNet uses an attention mechanism to select important features during training (Arik and Pfister, 2021). FT-Transformer applies a transformer structure to numerical and categorical variables (Gorishniy et al., 2021). TabPFN and TabICL use in-context learning to make predictions from tabular data (Hollmann et al., 2025; Qu et al., 2026). TabM follows a different approach by combining several Multilayer Perceptron (MLP) based predictors in one parameter-efficient model (Gorishniy et al., 2025). These models have shown strong results on general tabular datasets, but their performance still depends on the dataset and prediction task. These models are relevant to the present research because the route-section datasets used in this research are represented in tabular form. Each record contains operational and contextual information for a train movement between two consecutive railway locations. Tabular DL models may therefore learn relationships between the prepared railway features without requiring a sequential or network representation. However, the use of modern tabular DL models in train delay prediction remains limited. Al Ghamdi et al. (2024) included TabNet as one of the candidate models in a train delay prediction framework. Arthaud et al. (2024) examined transformer-based methods for large-scale train delay forecasting, while Yang et al. (2024) used a transformer-CNN model for train delay recovery-time prediction. These studies show growing interest in attention-based railway models, but more evidence is needed to understand their performance across different routes, sections, and data conditions.

Other studies have combined individual prediction models with optimisation methods. These methods use an optimisation algorithm to select model settings but do not combine the outputs of several predictors. One example is the ELM-PSO method proposed by Bao et al. (2021). The authors combined an Extreme Learning Machine with Particle Swarm Optimisation (PSO) for train arrival delay prediction. In ELM-PSO, each particle represented a possible number of hidden-layer neurons. Prediction error was used to evaluate each candidate setting, and PSO searched for a suitable ELM structure.

The selected ELM then produced the final delay predictions. This method kept the fast training process of ELM while providing a more systematic way to select its main structural parameter. However, ELM-PSO remained a single optimised predictor rather than an ensemble of several model families. It was also evaluated on one high-speed railway corridor. Therefore, its reported performance does not directly show how the method would behave on conventional railway routes with different numbers of sections, traffic patterns, and delay characteristics.

Overall, statistical approaches, conventional ML models, DL models, and optimisation-assisted variants have all provided useful approaches for train delay prediction. Statistical and probabilistic models are often easier to interpret, while ML models can represent nonlinear relationships between structured operational features. DL models can learn more complex temporal, spatial, or feature relationships but often require more data and computational resources. Modern tabular models provide another option for structured railway datasets, although their use in this field remains limited.

The reviewed studies also show that the performance of an individual model can change across datasets, routes, sections, and prediction conditions. This variation provides the main reason for examining ensemble methods that combine the outputs of several predictors. Train delay prediction studies using ensemble methods are reviewed in the next subsection.

2.2.2 Ensemble Approaches for Train Delay Prediction

Ensemble methods combine several predictors to produce a final prediction. They have been used in train delay prediction because individual models may perform differently across routes, delay levels, and operating conditions. By combining several models, an ensemble may reduce the effect of errors made by one predictor (Nair et al., 2019; Mienye and Sun, 2022). Some train delay studies have used homogeneous ensembles. These methods combine learners from the same model family. Random Forest, Gradient Boosting Decision Trees (GB), LightGBM, and XGBoost are common examples. In this research, Random Forest and boosting methods are treated as individual candidate predictors when they are compared with other models, although they are internally

homogeneous ensemble algorithms. Their learners are trained using different samples, features, or sequential error corrections. Zhang et al. (2021) proposed a GBDT-based method for train delay prediction. The study first used clustering and classification methods to construct features related to delay propagation. These features were then used by the GBDT model. The proposed method produced better results than the selected RF, SVR, and MLP baselines. However, its final prediction was still produced by one boosting model family. Laifa et al. (2021) applied an optimised LightGBM regressor to train delay prediction on the Tunisian railway. The model was compared with Support Vector Machine, Random Forest, XGBoost, and Artificial Neural Network models and achieved the strongest reported performance among the evaluated methods. However, the final predictor remained within one boosting model family. Ardıç (2024) used Random Forest regression to predict arrival delays at several downstream stations and updated the predictions as new train movement information became available. The study found that including weather conditions and technical problems improved prediction performance. It also reported stable forecasts for most trains at the target stations. These studies show that homogeneous ensembles can provide strong results for structured railway data. However, homogeneous ensembles use learners with similar structures. Their predictors may therefore learn similar patterns and make related errors. This has encouraged the use of heterogeneous ensembles, which combine models from different algorithm families. Al Ghamdi et al. (2024) developed several heterogeneous weighted ensemble methods for train delay prediction. Their first method, MSM1-WE, ranked the candidate models using validation performance. Models were then added one by one while the performance of the ensemble continued to improve. The MSM2 methods considered both model performance and diversity. MSM2a-COR-WE used the correlation between model predictions to identify models with different prediction behaviour. MSM2b-CFD-WE used a coincident failure diversity measure to examine whether models made large errors on the same observations. After model selection, the outputs of the selected models were combined using weighted averaging. The study found that the heterogeneous ensembles were generally more accurate and consistent than the individual models. However, the accuracy-based MSM1 method produced the best overall results. Among the tested diversity measures, CFD performed better than

the pairwise measures, but diversity-based selection did not outperform accuracy-based selection (Al Ghamdi et al., 2024). It is relevant to this research because it examines how models should be selected before constructing an ensemble. However, the MSM methods use predefined performance and diversity measures. They do not jointly optimise all ensemble weights using several evaluation metrics.

Stacking has also been examined for train delay prediction. In a stacking method, the predictions of base models are used as inputs to another model, known as the meta-model. This allows the meta-model to learn how the base predictions should be combined. Boateng and Yang (2023a) proposed the ST-NN method. Several neural regression models were trained at the first level, and their predictions were passed to an MLP meta-model. The MLP learned a nonlinear relationship between the base-model outputs and the final delay value. The method produced strong results on the railway dataset used in the study. ST-NN differs from a heterogeneous ML and DL ensemble because its base predictors are neural models. Its performance may also depend on the amount of training data, the structure of the base networks, and the design of the meta-model. Training several neural models and a second-level model may also increase the computational cost.

Ensemble methods have also been used in other transport prediction problems. Stacking and hybrid ensembles have improved flight delay prediction in several studies (Dou, 2020; Sahoo et al., 2022). Similar approaches have been applied to traffic forecasting and travel-time estimation (Pavlyuk, 2020; Lu et al., 2020). These studies show that ensemble learning can be useful when different models capture different parts of a transport prediction problem. However, road and air transport data have different structures and operating conditions from railway data.

Overall, previous railway studies show that ensemble performance depends on the selected models and the method used to combine their predictions. Homogeneous ensembles combine learners from the same model family, while heterogeneous ensembles combine different types of models. However, using different model families does not always mean that their prediction errors are different. For this reason, some studies have considered both model performance and diversity when selecting ensemble members (Al Ghamdi et al., 2024). Stacking provides another approach by using a meta-model

to learn how the base predictions should be combined. The roles of model selection, diversity, and weighting are discussed further in Section 2.3.

2.2.3 Real-Time Prediction and Operational Challenges

Most train delay prediction studies train and test their models using historical datasets. In these studies, the data are collected and prepared before model training. This makes it possible to clean the records, calculate the required features, and evaluate the models under controlled conditions. However, live railway prediction has different requirements because new information arrives continuously during train operations (Spanninger et al., 2022; Tiong et al., 2023a).

Real-time inference and online learning are different approaches. In real-time inference, a model is trained offline and is then used to make predictions when new data arrive. In online learning, the model is also updated as new observations become available (Gama et al., 2014; Bifet et al., 2023). Many operational prediction systems use real-time inference because model training and hyperparameter selection are completed before deployment. Live railway data may be incomplete, delayed, duplicated, or inconsistent. Information from different railway systems may also arrive at different times (Zilko et al., 2016; Hauck and Kliewer, 2020). Therefore, all input features may not be available when a prediction is required. This is different from many offline experiments, where complete historical records are used for model evaluation. Another challenge is partial information. The reason for a disruption, passenger demand, or congestion further along the route may not be known at the time of prediction. Probabilistic and network-based methods have been used to represent uncertainty and relationships between railway services (Corman and Kecman, 2018; Zilko et al., 2016). However, many ML and DL models are trained using complete feature sets and may be affected when some inputs are missing. Prediction time is also important in real railway operations. A model must process new updates and return a prediction within a suitable period. Leng and Corman (2022) explained that railway predictions need to meet the time requirements of traffic management activities. A prediction may have limited value if it is produced after an operational decision has already been made.

Recent studies have also treated train delay prediction as a time-sensitive and

network-related problem. For example, Li et al. (2026) considered delay changes across the railway network and produced predictions from the current state of the system. Such studies show that real-time prediction should consider both the latest train information and the relationships between railway locations (Li et al., 2026). Computational cost is another practical issue. Complex DL and graph-based models may require more processing time and memory than simpler models (Spanninger et al., 2022; Yang et al., 2024). Therefore, real-time evaluation should consider not only prediction accuracy but also processing time, model stability, and the ability to work with incomplete inputs.

Previous reviews have examined a wide range of train delay prediction methods and their potential use in real-time railway operations (Spanninger et al., 2022; Tiong et al., 2023a). However, many existing models are developed and evaluated using historical datasets. Some studies use information from the current train journey to update predictions for later stations (Lapamonpinyo et al., 2022). The reviewed literature gives less attention to prediction when confirmed departure information is not yet available and to recursive prediction across several downstream route sections. Direct comparison between model predictions and the estimates already provided by railway systems has also received limited attention (Leng and Corman, 2022). These issues are important because the information available to a prediction system changes as a train moves along its route. A practical framework should therefore continue to provide useful predictions under different levels of available operational information.

In this research, the models are trained offline and applied when new railway updates are received. The framework therefore performs real-time inference rather than online model updating. The framework produces both direct next-station predictions and recursive predictions across later route sections. Its design is presented in Chapter 6, and its direct and recursive prediction outputs are evaluated in Section 6.12.

2.3 Ensemble Learning

The performance of an ensemble depends on the models included and the method used to combine their predictions. Adding more models does not always improve the result, especially when some models have low prediction performance or make similar errors.

Therefore, model selection, diversity, ensemble size, and weighting are important parts of ensemble construction (Mienye and Sun, 2022; Al Ghamdi et al., 2024).

This section discusses the main factors that affect ensemble performance, such as model selection, diversity, and ensemble size. It then reviews the main weighting and optimisation methods used to combine model predictions.

2.3.1 Ensemble Construction, Model Selection, and Diversity

The first step in ensemble construction is to identify a set of candidate models. These models may use different learning methods, feature representations, or training settings. A large number of candidate models can increase the range of available predictions, but including every model does not always improve the ensemble. Weak or very similar models may add little useful information and may reduce the final performance. Model selection is therefore an important part of ensemble construction. One simple method is to rank models using their validation performance and select the best-performing models. However, selecting models only by accuracy may result in an ensemble whose members make similar errors. For this reason, some studies consider both model performance and diversity when choosing ensemble members (Al Ghamdi et al., 2024). Diversity refers to differences in the predictions or errors of ensemble members. It is often examined using measures such as correlation or failure-based diversity (Dutta, 2009; Wu et al., 2021). However, diversity alone is not enough, because a model may be different simply because it performs poorly. Therefore, both model performance and diversity should be considered when selecting ensemble members (Al Ghamdi et al., 2024). Ensemble size can also affect performance. Adding more models may help when they provide useful and different predictions, but weak or similar models may reduce performance and increase computational cost (Wang, 2008; Gashler et al., 2008). Therefore, the suitable ensemble size depends on the quality and behaviour of the selected models. Previous studies have used different selection methods. Some add models one by one while validation performance improves. Others use correlation, failure-based measures, or search methods to find a suitable subset of models (Al Ghamdi et al., 2024; Zhou, 2025). These approaches show that ensemble construction is not only a matter of combining many models. The quality of the selected models and the differences between

their errors are also important.

In this research, both ML and tabular DL models are considered as candidate predictors. Their validation performance is used to examine their individual quality, while their different learning structures provide the basis for constructing heterogeneous ensembles. The method used to assign weights to the selected models is discussed in the next subsection.

2.3.2 Ensemble Weighting and Optimisation

After the ensemble members are selected, their predictions must be combined. A simple method is equal-weight averaging, where every model has the same contribution to the final prediction. This method is easy to apply, but it assumes that all models have similar prediction quality. Performance-based weighting gives larger weights to models that perform better on validation data. The weights may be calculated using prediction error, accuracy, or another evaluation metric (Al Ghamdi et al., 2020; Srinivas et al., 2025). For example, a model with a lower validation error may receive a larger weight than a weaker model. This reduces the effect of poorly performing models on the final prediction. However, weights calculated directly from model performance may not provide the best combination. A model that performs well individually may not add much new information if its predictions are very similar to those of other models. For this reason, optimisation methods have also been used to search for suitable ensemble weights (Shahhosseini et al., 2022; Zhou, 2025; Srinivas et al., 2025). Optimisation-based methods search for a suitable combination of ensemble weights using validation data. Genetic Algorithms and Particle Swarm Optimisation have been used to optimise the contribution of individual models (Ling et al., 2016; Usmani et al., 2024). Bayesian search has also been applied to ensemble weight optimisation in regression problems (Shahhosseini et al., 2022). These methods provide a more systematic approach than assigning weights manually. However, many optimisation methods use one evaluation metric as the objective. Since MAE and RMSE measure prediction errors differently, weights selected using one metric may not provide the same performance under another metric (Willmott and Matsuura, 2005).

Existing ensemble methods often rely on a single evaluation metric or use predefined

weighting strategies to combine predictions from individual models. These approaches may not fully capture the different aspects of predictive performance or identify suitable combinations of model weights. These limitations motivated the investigation of multi-metric weighting strategies and Bayesian optimisation in this research. The resulting ensemble methods, MWEM and BOEM, are presented and evaluated in the subsequent chapters.

2.4 Evaluation, Robustness, and Generalisation

Train delay prediction models are commonly evaluated using regression metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2) (Spanninger et al., 2022; Tiong et al., 2023a). MAE shows the average size of the prediction error, while RMSE gives more importance to large errors. Therefore, the same model may receive different rankings under these two metrics (Willmott and Matsuura, 2005).

Some studies also use accuracy-based measures for delay prediction. In this research, ACC@1 is used as a tolerance-based regression measure. It reports the percentage of predictions with an absolute error of no more than one minute. This measure provides an operational view of how often the prediction is close to the actual delay.

Using only one metric may give an incomplete view of model performance. For example, a model may have a low MAE but still produce some large errors. Another model may provide a better RMSE but lower ACC@1. For this reason, several metrics should be considered together when comparing prediction models.

The validation method is also important. Many studies train and test models using historical data from the same railway line or time period (Spanninger et al., 2022). For time-dependent data, a random split may mix observations from different time periods between the training and test sets. A chronological split preserves the temporal order by using earlier observations for training and later observations for evaluation. Walk-forward validation extends this approach by repeatedly moving the training and validation periods forward in time. These methods provide an evaluation setting that more closely represents the use of past observations to predict future observations

(Tashman, 2000; Cerqueira et al., 2020).

Robustness refers to the ability of a model to maintain its performance when the input data are incomplete, noisy, or delayed. This is important in railway systems because operational data may contain missing values, transmission delays, or inconsistent records (Zilko et al., 2016; Hauck and Kliewer, 2020). However, many studies evaluate their models using clean and complete datasets. As a result, there is still limited evidence about how train delay prediction models behave when some input features are unavailable.

Generalisation is another important issue. A model trained on one route may not provide the same performance on another route because railway lines can differ in timetable structure, number of stations, traffic level, and delay patterns. Reviews of train delay prediction research have shown that many models are evaluated on only one railway line or dataset (Spanninger et al., 2022; Tiong et al., 2023a). Evaluation across different routes and route sections is therefore needed to examine whether the modelling approach provides consistent performance under different operating conditions.

Overall, model evaluation should consider more than one performance metric. It should also examine time-based validation, incomplete inputs, and performance across different routes and route sections. These aspects are used in this research to provide a wider comparison of the individual and ensemble models.

2.5 Research Gaps and Research Positioning

The reviewed studies have made important progress in train delay prediction. However, several limitations remain in the existing literature.

First, many studies focus on one individual model or compare several models without combining them. Homogeneous ensembles such as Random Forest and boosting methods have been widely used, but fewer studies have combined a wide range of ML and DL models within one heterogeneous ensemble (Spanninger et al., 2022; Al Ghamdi et al., 2024). This research addresses this gap by examining conventional ML and modern tabular DL models as candidate predictors for heterogeneous ensemble construction.

Second, the selection of ensemble members is not always systematic. Some studies

select models mainly according to their individual performance, while others use predefined diversity measures (Al Ghamdi et al., 2024). However, a model that performs well alone may not always provide useful additional information to an ensemble. This research therefore examines the performance of the candidate models before combining them and evaluates different ensemble structures.

Third, many weighting methods use equal weights, fixed rules, or one validation metric (Al Ghamdi et al., 2020; Shahhosseini et al., 2022; Srinivas et al., 2025). However, different metrics measure different parts of prediction performance. A model that performs well under MAE may not have the same performance under RMSE or R^2. This research examines multi-metric weighting and optimisation methods for combining model predictions.

Fourth, robustness and generalisation have received less attention than overall prediction accuracy. Many models are evaluated using complete data from one railway route or dataset (Spanninger et al., 2022; Tiong et al., 2023a). This does not fully show how their performance changes across routes, route sections, delay levels, or incomplete input conditions. This research evaluates the models across two railway routes and multiple route sections. It also examines their performance when some input features are unavailable.

Finally, many train delay prediction studies are evaluated offline using prepared historical datasets. Fewer studies examine how ensemble models can be integrated with continuously arriving operational data, incomplete departure information, and prediction across several later route sections (Spanninger et al., 2022; Tiong et al., 2023a; Leng and Corman, 2022). Operational comparisons with railway-provided estimates are also less common in the reviewed literature. This research addresses this gap through an end-to-end real-time inference framework in which offline-trained models are applied to live route-section records. The framework estimates unavailable departure information, supports section-specific model selection and fallback handling, and generates direct and recursive downstream predictions. The evaluated outputs include direct predictions under actual and estimated departure-information conditions and recursive predictions compared with railway estimates and baseline methods

Among the railway delay prediction studies reviewed in this chapter, no method was

identified that jointly optimised the weights of a heterogeneous ML and DL ensemble using an objective based on several validation metrics. This gap provides the main motivation for the ensemble methods developed and evaluated in this research.

2.6 Summary

This chapter reviewed individual and hybrid prediction approaches, ensemble methods, and real-time train delay prediction. The reviewed studies differed in their prediction targets, input data, modelling methods, validation settings, and railway environments. Table 2.1 provides a structured comparison of the main empirical studies and identifies the published methods used as comparison approaches in the later experimental chapters.

The review showed that statistical approaches, conventional ML models, DL models, and ensemble strategies can support train delay prediction, but their performance depends on model selection, weighting, data quality, and the evaluation setting. Real-time studies also highlighted challenges related to incomplete or delayed information and the need to produce predictions within operational time limits.

The main gaps were the limited use of heterogeneous ML and DL ensembles, reliance on fixed or single-metric weighting methods, limited robustness and cross-route evaluation, and limited real-time ensemble studies. There was also limited evidence on prediction when departure information is unavailable, recursive prediction across later route sections, and comparison with railway-provided estimates. These gaps provide the basis for the methods developed in the following chapters.

Table 2.1: Comparative summary of reviewed train delay prediction studies.

Study	Prediction target and setting	Method and design	Reported contribution	Limitation relevant to this research
Xu et al. (2023)	Delay prediction across consecutive stations	**Markov-GK**†: a non-homogeneous Markov model with Gaussian-kernel estimation	Provides an interpretable method for short-term prediction when only a limited number of transition observations are available.	Uses discrete delay states and transition patterns learned from a specific railway network.
Shi et al. (2021)	Next-station arrival delay prediction on high-speed railway lines	**BO-XGBoost**†: XGBoost with Bayesian optimisation for hyperparameter selection	Provides a systematic approach for selecting XGBoost hyperparameters and examines results across stations, delay levels, and operating conditions.	Uses one model family and was evaluated in a high-speed railway setting.
Bao et al. (2021)	Train arrival delay prediction on a high-speed railway corridor	**ELM-PSO**†: Extreme Learning Machine with Particle Swarm Optimisation	Uses PSO to select the number of hidden-layer neurons in the ELM.	Remains a single optimised predictor and was evaluated on one railway corridor.
Yaghini et al. (2013)	Train delay prediction using historical railway information	Classical ML models, including neural and regression-based approaches	Shows that nonlinear learning methods can be applied to structured railway delay data.	The results depend on the selected features and the railway dataset used.
Oneto et al. (2016)	Train delay prediction using timetable and operational information	Comparison of classical ML models for railway delay prediction	Examines how different ML methods learn relationships between operational features and train delays.	Provides limited evidence about performance across different railway routes.

Continued on next page

Table 2.1 – continued from previous page

Study	Prediction target and setting	Method and design	Reported contribution	Limitation relevant to this research
Oneto et al. (2018)	Train delay prediction using historical and contextual features	Nonlinear ML models applied to structured railway records	Shows the value of using operational and contextual variables in train delay prediction.	The reported performance is linked to the specific data and validation setting.
Wen et al. (2020)	Prediction of temporal train delay behaviour	RNN- or LSTM-based sequential prediction model	Models relationships between earlier train movements and later delays.	Requires suitable sequential data and careful model configuration.
Wang et al. (2023a)	Train delay prediction using temporal railway records	Hybrid CNN and recurrent architecture	Combines local feature learning with temporal modelling in one predictive architecture.	Has greater model and computational complexity than many classical ML methods.
Ding et al. (2021)	Network-related train delay prediction	Graph-based DL model representing railway locations and links	Models relationships between connected railway locations and train movements.	Requires a suitable railway graph and detailed network information.
Li et al. (2021a)	Spatio-temporal delay prediction across a railway network	Graph and temporal learning components	Represents both spatial railway relationships and changes in delay over time.	May be difficult to apply when complete network information is unavailable.
Arthaud et al. (2024)	Large-scale train delay forecasting	Transformer-based railway forecasting model	Captures relationships across longer train sequences and railway locations.	Performance depends on the size and structure of the available training data.

Continued on next page

Table 2.1 – continued from previous page

Study	Prediction target and setting	Method and design	Reported contribution	Limitation relevant to this research
Yang et al. (2024)	Train delay recovery-time prediction	Hybrid transformer–CNN model	Combines attention-based learning with local feature extraction.	Predicts recovery time rather than the route-section arrival-delay target considered in this research.
Zhang et al. (2021)	Train delay prediction using delay-propagation information	Homogeneous GBDT ensemble with propagation features constructed using clustering and classification	Combines boosting with information about horizontal and vertical delay propagation.	The final predictor remains within one boosting model family.
Laifa et al. (2021)	Passenger train delay prediction on the Tunisian railway	Optimised LightGBM regressor compared with SVM, Random Forest, XGBoost, and ANN	Reports strong prediction accuracy and computational efficiency for LightGBM.	Uses one railway system and one final boosting model family.
Ardıç (2024)	Dynamic arrival-delay prediction at downstream stations	Random Forest regression using operational, weather, and technical variables	Updates predictions as new train movement information becomes available and examines the stability of the forecasts.	Uses one high-speed railway line and one model family.
Boateng and Yang (2023a)	Train delay prediction using the outputs of several neural models	**ST-NN**†: neural base models combined by an MLP meta-model	Learns a nonlinear combination of the base-model predictions.	Uses only neural predictors and requires a second level of model training.

Continued on next page

Table 2.1 – continued from previous page

Study	Prediction target and setting	Method and design	Reported contribution	Limitation relevant to this research
Al Ghamdi et al. (2024)	Train delay prediction using models from different algorithm families	**MSM1-WE**† uses performance-based selection; **MSM2a-COR-WE**† uses correlation; and **MSM2b-CFD-WE**† uses coincident failure diversity	Examines the effects of model performance, diversity, and ensemble size on heterogeneous train delay ensembles.	Uses predefined model-selection and weighting rules rather than joint multi-metric optimisation of the complete weight vector.
Leng and Corman (2022)	Use of railway predictions to support traffic management	Operational analysis of prediction timing and communication	Shows that predictions need to be produced within railway decision-making times.	Focuses on operational use rather than the construction of advanced ensembles.
Li et al. (2026)	Time-sensitive, network-wide train delay prediction	Integration of current railway information with network relationships	Considers both the latest operational information and relationships between railway locations.	Requires detailed and timely information about the railway network.

† Methods implemented using the methodological details available in the original publications for the controlled comparisons presented in the later experimental chapters. These implementations are not claimed to be exact replications where complete source code or configuration details were unavailable.

Chapter 3

Research Methodology

3.1 Introduction

This chapter explains the research strategy and experimental design used to investigate the five research questions. The research combines the development of ensemble prediction methods with comparative evaluation using historical railway datasets, time-aware evaluation using more recent observations, and evaluation of predictions generated from live railway updates. Figure 3.1 summarises the workflow supporting these investigations. The research methodology is established through the relationship between the research questions, experimental comparisons, validation procedures and evaluation criteria described in this chapter.

The detailed stages of the framework are presented in the following chapters. The collection, processing and preparation of the railway data, together with the exploratory analysis of the prepared datasets, are presented in Chapter 4. The individual prediction models and the proposed ensemble methods are developed in Chapter 5. The real-time prediction framework, including live data processing and recursive downstream prediction, is presented in Chapter 6. The historical results are presented in Chapter 5, while the time-aware and live prediction results are presented in Chapter 6.

This chapter does not repeat the detailed procedures presented in those chapters. Instead, it defines the main methodological basis of the research. It explains how railway journeys and delay values are represented, how the route-section-level prediction problem is formulated, and how the models are developed, validated, and evaluated within the research framework.

The remainder of this chapter is organised as follows. Section 3.2 establishes the

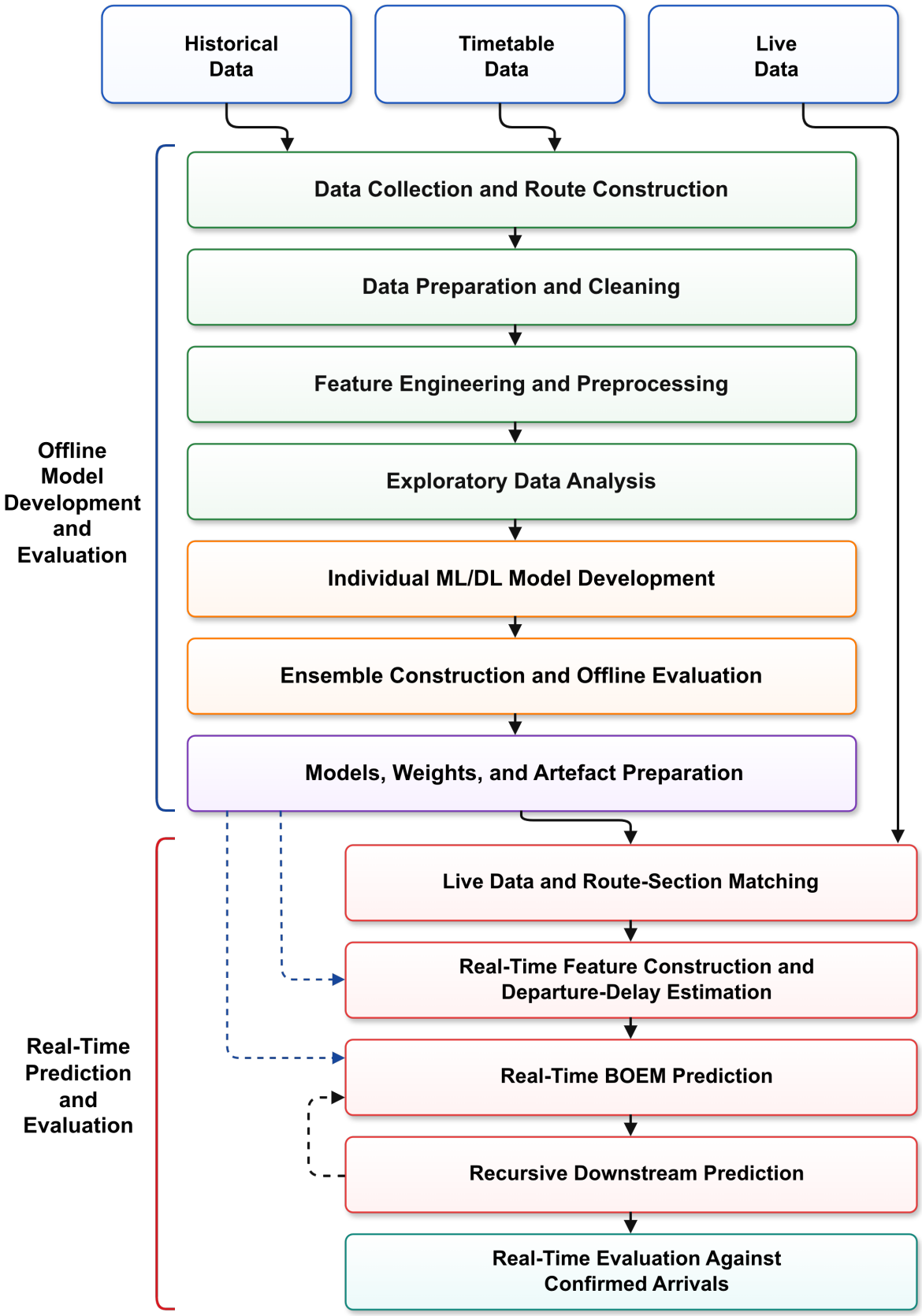


Figure 3.1: Workflow supporting model development, experimental evaluation and real-time prediction.

research strategy and maps the research questions to the individual investigations. Sections 3.3–3.5 describe the journey representation, delay measures, and prediction problem. Sections 3.6–3.8 present the data and features, experimental design, and evaluation metrics. Section 3.9 introduces the SHAP analysis, while Section 3.10 describes the experimental environment. Finally, Section 3.11 summarises the chapter.

3.2 Overall Research Strategy

This research follows a quantitative experimental strategy in which proposed ensemble methods are developed and evaluated through comparative prediction experiments. The purpose is to establish whether the proposed weighting and optimisation approaches improve train arrival delay prediction, whether combining the defined ML and tabular DL model pools provides additional predictive value, and how the resulting methods perform under different route, information-availability and real-time conditions. Developing the prediction framework provides the means of conducting these investigations; the research questions are answered through the evidence obtained from its evaluation.

The investigations are organised into three stages. First, historical experiments compare individual models and ensemble configurations under a common route-section prediction task. Second, targeted experiments examine the weighting strategy, sensitivity to missing departure information and transfer between the two historical route datasets. Third, time-aware model evaluation and live prediction evaluation examine the framework under conditions relevant to real-time use. Together, these stages provide complementary evidence about prediction accuracy, reliability and real-time predictive feasibility within the datasets and evaluation settings examined.

3.2.1 Relationship Between Research Questions and Investigations

The experimental comparisons were selected to provide evidence for the five research questions. Table 3.1 identifies the investigation undertaken for each question and the evidence used to answer it. The historical investigations are reported in Chapter 5, while the time-aware and live prediction investigations are reported in Chapter 6.

Table 3.1: Relationship between the research questions, investigations and evaluation evidence.

Research question	**Investigation and comparison**	**Evidence used to answer the question**
RQ1	Compare MWEM with individual models and the equal-weight Average Ensemble. Compare multi-metric BOEM-ML weighting with corresponding single-metric configurations.	Held-out prediction performance and the controlled weighting-strategy comparison reported in Sections 5.11.1 and 5.11.2.
RQ2	Evaluate BOEM against MWEM, the Average Ensemble and individual models. Include additional comparisons with reconstructed published train delay prediction methods.	Prediction errors, model rankings and statistical comparisons reported in Sections 5.11.1, 5.11.6, 5.11.7 and 5.11.11.
RQ3	Compare BOEM-ML, BOEM-DL and hybrid BOEM using the same ensemble-weight optimisation procedure to examine the effect of combining the two predefined model pools.	Differences in held-out prediction performance between the three model-pool configurations, reported in Table 5.4.
RQ4	Examine performance across route sections, sensitivity to missing departure information and transfer between the two historical route datasets. The targeted missing-information and cross-route investigations use BOEM-ML.	Section-level variation, prediction-error deterioration under missing departure information, and performance in both route-transfer directions, reported in Sections 5.11.3, 5.11.9 and 5.11.10.
RQ5	Conduct chronological model evaluation, then evaluate direct and recursive predictions generated from live railway updates against confirmed arrival outcomes, railway-provided estimates and relevant simple baselines.	Time-aware model performance, live prediction errors, downstream performance across prediction horizons and inference-time measurements, reported in Sections 6.11, 6.12 and 6.11.4.

3.3 Railway Journey Representation for Delay Prediction

An important part of the methodological design is to define how a railway journey is represented. This representation determines the unit of analysis, how the operational data are organised, and how the relationship between successive stations is included in the modelling process.

A train service normally travels through an ordered sequence of calling points from its origin to its destination. During this journey, its operational condition may change between different parts of the route. For example, a train may gain delay, recover part of an existing delay, or maintain a similar delay as it moves between stations (Corman and Kecman, 2018; Spanninger et al., 2022). Therefore, representing the complete journey as a single movement may not fully describe how delay changes along the route.

In this research, a train journey is represented as an ordered sequence of n stations:

$$J = (s_1, s_2, \ldots, s_n), \tag{3.3.1}$$

where J represents the complete train journey, s_1 is the origin station, s_n is the destination station, and s_2 to s_{n-1} are the intermediate stations. The movement between two consecutive stations, (s_i, s_{i+1}), is defined as a route section.

The route section is used as the main unit of analysis in this research. Each train journey is therefore divided into consecutive station-to-station movements. For each route section, the available information about the train and its previous movement is used to predict its arrival delay at the next station. This representation allows changes in delay to be examined and predicted separately along different parts of the route.

Before selecting the final journey representation, three alternative schemes were considered. These schemes represent the railway journey at different levels and differ in the number of datasets and prediction models required. The distinction between local and global railway prediction models has been examined in previous research (Kecman and Goverde, 2015). Previous studies have also predicted train arrival times at downstream stations using information from the train's latest known location and historical travel times (Pongnumkul et al., 2014). More generally, different

prediction locations, horizons, and modelling structures used in train delay prediction are reviewed by Tiong et al. (2023b). Three journey representation schemes can be considered, as described below. In this research, the first scheme, based on consecutive station-to-station sections, was used.

- **One-to-One Station—Multi Model Representation**

  In the first scheme, the journey is divided into consecutive station-to-station sections. A separate prediction model is developed for each adjacent station pair. Therefore, for a journey with n stations, this scheme produces $n-1$ individual models. This scheme allows each model to focus on one specific part of the route. For example, the model for section (s_i, s_{i+1}) learns the delay behaviour between those two neighbouring stations. This can include the effect of the previous delay, the dwell time at the current station, and the possibility of delay recovery before the next station.

  This scheme is also useful for real-time prediction. When the required operational information becomes available at station s_i, the model developed for section (s_i, s_{i+1}) is used to predict the arrival delay at station s_{i+1}. The prediction can then be updated again at the next station. In this way, the model follows the actual movement of the train along the route. Figure 3.2 illustrates the one-to-one station multi-model representation.

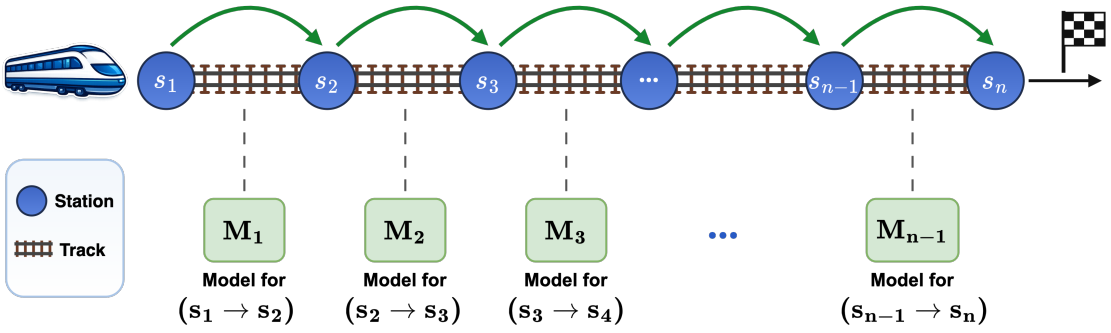


Figure 3.2: One-to-one station multi-model representation for consecutive station-to-station delay prediction.

- **All-to-Next Stations—Single Model Representation**

  In the second scheme, all station-to-next-station records are combined into one dataset, and one global model is trained for the whole route. This means that

the same model is used to predict delay for different station pairs. This scheme is simple because only one model needs to be trained, stored, and maintained. It can also be useful when some individual station pairs do not have enough data for separate models.

However, combining all route sections within one model may make it more difficult for the model to represent section-specific differences. Different sections can have different running times, dwell patterns, congestion levels, and opportunities for delay recovery. A section identifier would therefore be required to help the model distinguish between the different station pairs. Figure 3.3 shows the all-to-next stations single-model representation.

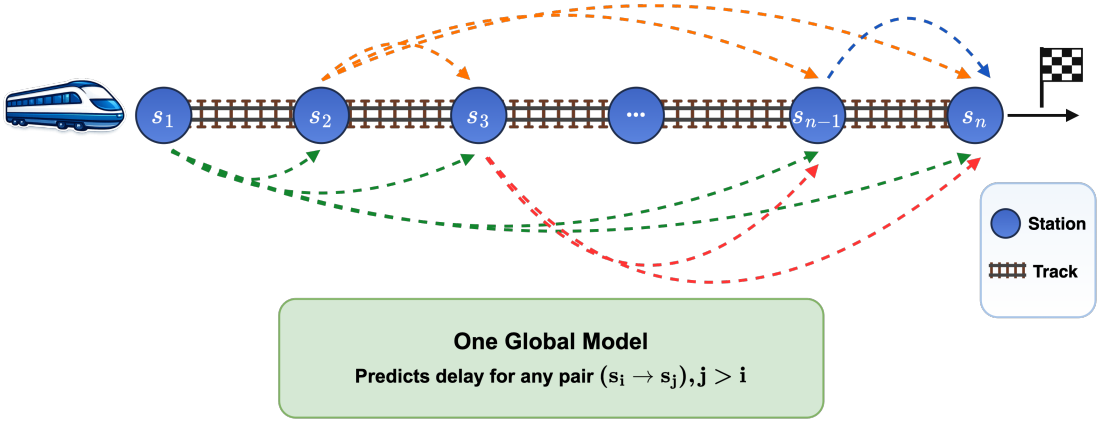


Figure 3.3: All-to-next stations single-model representation using one global model for different station pairs.

- **All-to-Next Stations—Multi Model Representation**

  In the third scheme, a separate prediction model is developed for every possible station pair within a route. The station pairs are not limited to adjacent stations. For example, a model can be developed from s_i to s_{i+2}, from s_i to s_{i+3}, or for any other downstream station pair. For a journey with n stations, station s_1 can be paired with $n-1$ downstream stations, station s_2 can be paired with $n-2$ downstream stations, and this process continues until station s_{n-1}, which can only be paired with s_n. Therefore, the total number of downstream station pairs is:

$$\sum_{i=1}^{n-1}(n-i) = (n-1) + (n-2) + \cdots + 1 = \frac{n(n-1)}{2}. \tag{3.3.2}$$

Only station pairs that follow the direction of the journey are included. Therefore, a reverse pair such as (s_j, s_i), where $j > i$, is not counted separately.

This scheme gives a detailed representation of the route because each station pair can have its own model. It can therefore learn specific delay patterns between different parts of the journey.

However, this scheme requires a large number of models, especially for long routes. This increases the cost of training, tuning, storing, and maintaining the models. Some station pairs may have fewer usable training examples when train services have different calling patterns or when observations are missing for part of a journey. Figure 3.4 illustrates the all-to-next stations multi-model representation.

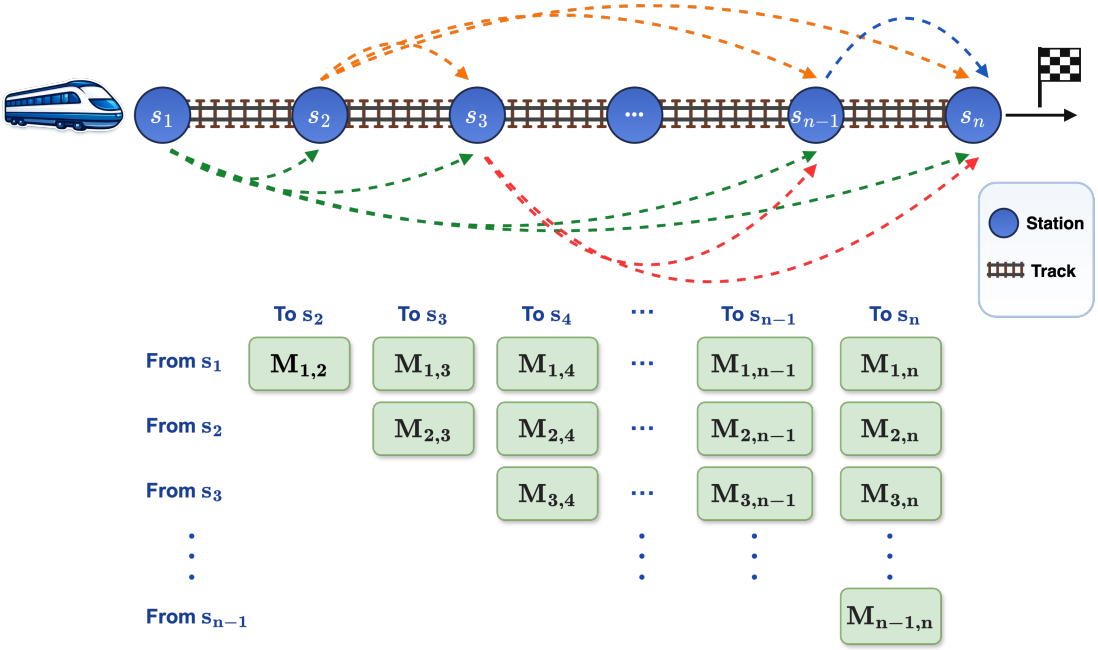


Figure 3.4: All-to-next stations multi-model representation using separate models for all possible station pairs.

Based on these considerations, the one-to-one station multi-model representation was selected for the prediction framework developed in this research. It preserves the differences between individual route sections, provides a direct next-station prediction task, and supports the generation of updated predictions as a train moves along the route. The other two schemes were considered as alternative representations but were not used in the subsequent experiments.

3.4 Delay Representation

To understand the train delay prediction task, it is first necessary to define how delay is measured during a journey. Each train service has planned arrival and departure times specified in the timetable before the journey begins. During operation, the actual arrival and departure times are recorded when the train reaches and leaves each station. The difference between the planned and actual times shows whether the train is early, on time, or delayed.

For each station s_i in a journey J, the planned and actual arrival times are denoted by t_i^{pa} and t_i^{aa}, respectively. Similarly, the planned and actual departure times are denoted by t_i^{pd} and t_i^{ad}. Based on these time differences, the arrival and departure delays at station s_i are calculated as:

$$\text{Arrival delay:} \quad d_i^a = t_i^{aa} - t_i^{pa} \tag{3.4.1}$$

$$\text{Departure delay:} \quad d_i^d = t_i^{ad} - t_i^{pd} \tag{3.4.2}$$

where d_i^a is the arrival delay and d_i^d is the departure delay. All delay values are measured in minutes. A positive value indicates that the train is delayed, whereas a negative value indicates that the train is running ahead of schedule, and a value of zero indicates that it is on time.

In addition to arrival and departure delays, the delay of a train may also change while it is stopped at a station. If the train remains at the station longer than planned, additional delay may be introduced. In contrast, a shorter stop may allow part of the existing delay to be recovered. This change is represented by the dwell delay. Dwell time refers to the time that a train stays at a station between arrival and departure. The planned and actual dwell times at station s_i are defined as:

$$t_i^{pw} = t_i^{pd} - t_i^{pa} \tag{3.4.3}$$

$$t_i^{aw} = t_i^{ad} - t_i^{aa} \tag{3.4.4}$$

where t_i^{pw} is the planned dwell time and t_i^{aw} is the actual dwell time. The dwell delay is

then calculated as:

$$\text{Dwell delay:} \quad d_i^w = t_i^{aw} - t_i^{pw} \tag{3.4.5}$$

A positive dwell delay indicates that the train remained at the station longer than planned, while a negative value indicates that the actual stop was shorter than the planned dwell time.

Consistent with previous railway delay studies, arrival delay, departure delay, and dwell delay are used as the main delay-related measures in this research (Kecman and Goverde, 2015; Pritchard et al., 2021; Kuipers, 2024). The prediction target is the observed arrival delay at the next station.

To make these delay measures easier to understand, Figure 3.5 illustrates the calculation of departure delay, arrival delay, and dwell delay using simple examples. In the departure delay example, the train is scheduled to depart at 09:00 but actually departs at 09:05, producing a departure delay of 5 minutes. In the arrival delay example, the planned arrival time is 10:00 and the actual arrival time is 10:10, resulting in an arrival delay of 10 minutes. For dwell delay, the train is scheduled to remain at the station for 2 minutes but actually remains for 3 minutes, giving a dwell delay of 1 minute. These delay measures are used throughout the thesis for feature construction, model training, prediction, and evaluation.

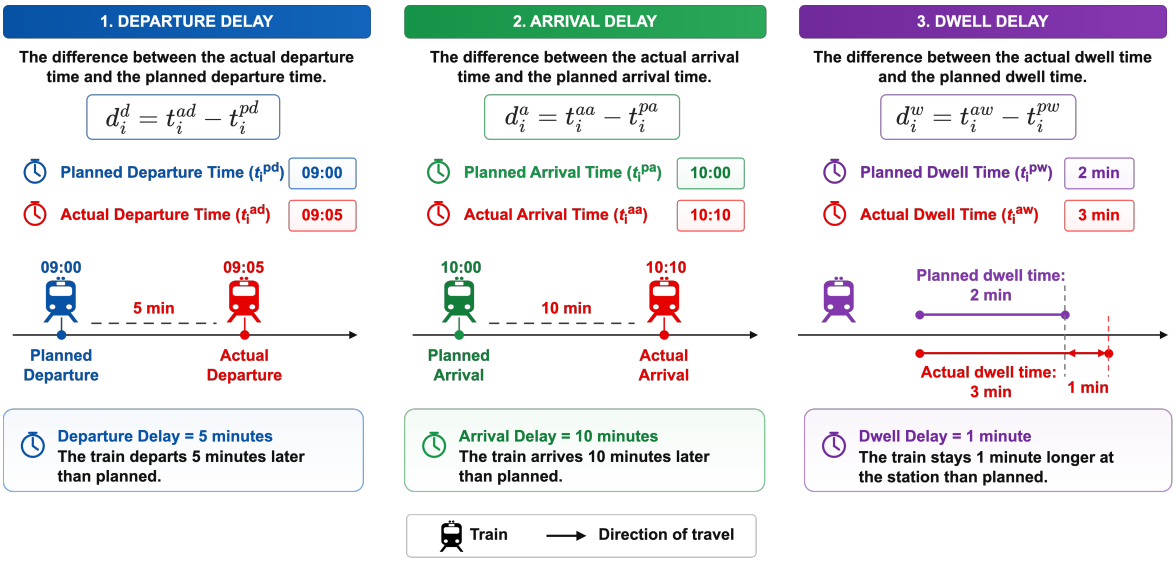


Figure 3.5: Examples of departure delay, arrival delay, and dwell delay based on planned and actual train timings.

3.5 Problem Formulation and Prediction Framework

After defining the railway journey representation and the main delay measures, the next step is to formulate the prediction problem and describe the overall prediction framework used in this research. The problem is treated as a route-section-level regression task. For each route section (s_i, s_{i+1}), the available information at station s_i is used to predict the arrival delay at the next station, s_{i+1}.

Each station-to-station movement is represented by a feature vector. For observation r and route section i, this vector is defined as:

$$\mathbf{x}_{r,i} = [x_{r,i,1}, x_{r,i,2}, \ldots, x_{r,i,p}] \tag{3.5.1}$$

where p is the number of input features and $x_{r,i,k}$ represents feature k for observation r in route section i. The feature vector describes the available state of the train at the prediction point. It may contain delay-related, dwell-related, temporal, route-section, and historical information. The detailed construction of these features is presented in Chapter 4.

For a base learner l_j, the arrival delay at station s_{i+1} is estimated as:

$$\hat{d}^{a(j)}_{r,i+1} = f_{i,j}\left(\mathbf{x}_{r,i}\right), \tag{3.5.2}$$

where $\hat{d}^{a(j)}_{r,i+1}$ is the arrival delay predicted by learner l_j for station s_{i+1}, and $f_{i,j}(\cdot)$ is the prediction function developed by learner l_j for route section i. The observed arrival delay, $d^{a}_{r,i+1}$, is used as the target value during model training and evaluation.

Following the one-to-one station multi-model representation described earlier, a separate model is developed for each consecutive route section. Therefore, all models developed for a given section use the same input records and predict the same next-station arrival delay. This provides a consistent basis for comparing the performance of different prediction methods.

The prediction framework is developed in two main stages. In the first stage, a range of individual ML and DL models is trained and evaluated for the same route-section-level prediction task. These models represent different learning approaches and may capture different relationships within the railway data. Their individual performance provides

the basis for examining their strengths and limitations across different routes and route sections.

In the second stage, the predictions produced by the individual models are combined using ensemble methods. For an ensemble containing m base learners, the combined prediction for route section i is represented as:

$$\hat{d}_{r,i+1}^{a(\mathcal{E})} = \sum_{j=1}^{m} \omega_{i,j} \hat{d}_{r,i+1}^{a(j)}, \tag{3.5.3}$$

subject to:

$$\omega_{i,j} \geq 0, \qquad \sum_{j=1}^{m} \omega_{i,j} = 1, \tag{3.5.4}$$

Three ensemble approaches are considered in this research. The Average Ensemble (AE) assigns the same weight to all included models and is used as a simple ensemble baseline. The proposed Multi-Metric Weighted Ensemble Method (MWEM) assigns weights according to the validation performance of the individual models across multiple evaluation metrics. The proposed Multi-Metric Bayesian-Optimised Ensemble Method (BOEM) uses Bayesian optimisation to determine the combination of models and weights.

The individual models and ensemble methods follow the same route-section-level prediction formulation. This allows their predictions to be compared under the same data, validation, and evaluation conditions. The detailed model descriptions, training procedures, hyperparameter settings, ensemble-construction methods, and experimental results are presented in Chapters 5 and 6.

3.6 Overview of Data and Feature Use

The prediction framework developed in this research uses historical, timetable, and real-time railway data. Each type of data supports a different part of the research, from offline model development to real-time prediction. Historical railway data contain records of train services that operated in the past, including their planned and actual movements. These data are used to prepare the section-level datasets, construct the

prediction features, and train, validate, and evaluate the individual and ensemble models.

Timetable data provide the planned arrival and departure times, the calling order of stations, and the route structure of each train service. These data are used to define the planned journey and to calculate the delay measures described in the previous section. They are also used in the real-time framework to identify the current and downstream route sections of an active service.

Real-time railway data contain live train-running updates received while a service is operating. These updates are used to identify the current state of the journey and to construct the information required for online prediction. The trained models then use these inputs to generate arrival delay predictions for the next station and, where required, for later sections of the route. The input features used in the prediction models are grouped into delay-related, dwell-related, temporal, route-section, and historical features. Together, these features describe the state of the train at each prediction point and provide the input vector defined in Equation 3.5.1.

The offline and real-time parts of the framework use a consistent feature representation. During offline model development, the features are constructed from historical records. During real-time operation, the same required inputs are generated from live updates and timetable information. When a required value is not yet available in the live data, it is estimated or replaced using the procedures defined for the real-time framework. This section provides only an overview of how the data and features are used within the methodology. The data sources, collection process, preprocessing steps, and detailed feature construction are presented in Chapter 4. The construction of real-time features from live railway updates is explained in Chapter 6.

The Phase 1 datasets, NL-P1 and WW-P1, support the historical model comparisons and targeted investigations addressing RQ1–RQ4. The more recent Phase 2 datasets, NL-P2 and WW-P2, support chronological model development and evaluation for the real-time framework. Predictions generated during the 40-day live evaluation period in 2026 provide the live-data evidence for RQ5. The routes were selected because they provided sufficient station-level historical records and differed in route structure. Dataset coverage, sample sizes and preparation procedures are reported in Chapter 4,

particularly Section 4.4 and Table 4.4. The live evaluation samples are described in Section 6.12.1.

3.7 Experimental Design

This section describes how the experiments were designed to develop and evaluate the prediction models. Historical model development was based on five-fold cross-validation, while temporal model development used expanding-window walk-forward validation. The section also outlines the additional evaluation settings used in this research.

- **Historical Model Validation**

  For the historical experiments, each route-section dataset was evaluated separately using random five-fold cross-validation. The available observations for each section were divided into five folds. In each iteration, four folds were used for model development and the remaining fold was used as the held-out test fold. This process was repeated five times so that each fold was used once for testing. The held-out test fold was not used for fitting the prediction models, selecting hyperparameters, or estimating ensemble weights. The performance measures were calculated for each test fold, and the final results were obtained by averaging the results across the five outer folds. Applying the same procedure separately to each route section allowed model performance to be examined across different parts of each railway route.

  Within each outer training fold, an inner five-fold cross-validation procedure was used for model tuning and ensemble development. The inner folds were used to select the hyperparameters of the individual models and to determine the ensemble weights without using information from the outer test fold. The validation predictions generated within the inner procedure were used to calculate the metric-based weights for MWEM and optimise the weights for BOEM. After the model settings and ensemble weights had been determined, the models were fitted using the complete outer training data and evaluated on the corresponding outer test fold. This nested design kept the final test data separate from model tuning and ensemble construction.

- **Temporal Validation for Real-Time Development**

  A separate walk-forward expanding-window validation procedure was used for the models developed for real-time prediction. This procedure followed the chronological order of the railway observations and used an expanding training window. At each step, the model was trained using the observations from the earlier months and evaluated using the following months. After each step, the most recent months were added to the training data before the next period was evaluated. For example, a model may first be trained using the first month and evaluated on the second month. The first two months are then combined for training, and the third month is used for evaluation. This process continues until all previous months are used for training and the final month is retained as the test period. This expanding-window design reflects the way in which the amount of historical information available to a real-time prediction system increases over time. It also prevents observations from future months from being used to train or evaluate predictions for an earlier period.

- **Route and Route-Section Evaluation**

  The main experiments were conducted separately for each railway route. Within each route, model performance was evaluated for every consecutive station-to-station section. This allowed the performance of the individual and ensemble models to be examined across different parts of each route. Route-level summaries were also produced from the section-level results. Using two routes allowed the proposed methods to be examined under different route and operational conditions. In addition to the two main routes, the real-time framework was also evaluated on six further routes, including services operating in the reverse directions. This provided additional evidence of how the framework performed across a wider range of route and directional conditions.

- **Generalisation Evaluation**

  A separate cross-dataset experiment was conducted to examine whether the prediction models could generalise between the two railway datasets. The models were trained using one route dataset and evaluated using the other route dataset.

This experiment was separate from the main within-route evaluation. Additional experiments also examined generalisation across route sections, delay levels, incomplete data conditions, and six further real-time routes, including reverse directions. The historical cross-dataset results are presented in Section 5.11.10. The live route-level results are presented in Sections 6.12.3 and 6.12.5.

The trained models were also evaluated using live railway data. The online predictions were compared with the observed arrival delays when the corresponding actual information became available. The detailed real-time evaluation procedure is presented in Chapter 6.

3.7.1 Baseline Comparisons

Different baselines serve different purposes in the experimental design. Individual prediction models establish whether combining predictions provides an advantage over the evaluated single-model alternatives. The Average Ensemble provides an equal-weight reference for examining the value of model selection and weighting. MWEM provides a reference for evaluating the BOEM weight-search configuration, while BOEM-ML and BOEM-DL provide the comparisons needed to examine the hybrid model pool. Selected methods reconstructed from published studies provide additional historical comparisons under the datasets and evaluation procedure used in this research; these are adapted implementations rather than exact reproductions of the original studies. In the live evaluation, railway-provided estimates provide a practical comparison for direct predictions, while recursive predictions are evaluated against simple baselines and, where available, corresponding railway next-station estimates.

3.7.2 Evaluation and Interpretation of Evidence

The historical random-fold evaluation examines comparative performance within the sampled route-section datasets, whereas chronological evaluation examines prediction on later observations using models developed from earlier observations. The random-fold results are therefore interpreted separately from the evidence concerning future-period and live prediction performance. Historical comparisons use held-out prediction

outcomes under the validation design described above. Performance is examined using complementary evaluation measures and section-level analyses, with Friedman and Nemenyi comparisons used to assess model-ranking differences and paired Wilcoxon comparisons used in the missing-departure investigation. The detailed procedures and results are reported in Sections 5.11.7 and 5.11.9. Live predictions are evaluated against confirmed arrival-delay outcomes, with direct results separated according to whether actual or estimated departure information was available. Recursive results are also examined across prediction horizons. Observed differences in average performance are distinguished from statistically supported differences, and conclusions are restricted to the datasets, model configurations and evaluation conditions examined.

3.8 Evaluation Metrics

This section describes the metrics used to evaluate the prediction models. Since train delay prediction is treated as a regression task in this research, the main evaluation measures are Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). According to recent research, train delay prediction models should be evaluated using standard error-based measures, such as MAE and RMSE, together with broader considerations including generalisability and robustness (Yong et al., 2025). In real-time passenger train delay prediction, Lapamonpinyo et al. (2022) evaluated model performance using R^2, MAE, and RMSE. Similarly, Huang et al. (2020c) used RMSE and MAE to compare deep learning and conventional machine learning models for multi-attribute train delay prediction. Therefore, the use of MAE, RMSE, and R^2 in this research is consistent with established evaluation practice in train delay prediction.

In addition to these standard regression metrics, Huber loss is also used in some experiments. It is less sensitive to very large errors than mean squared error, which is useful in railway delay prediction because some journeys may experience unusually large delays (Huber, 1992; Lapamonpinyo et al., 2022).

This thesis also uses a tolerance-based accuracy measure, denoted as $ACC@k$. This metric measures the percentage of predictions that fall within k minutes of the true

delay value. For example, $ACC@1$ shows the percentage of predictions that are within one minute of the actual delay. This measure is useful from an operational point of view because railway predictions do not always need to be exact to be useful (Gao et al., 2024; Arthaud et al., 2024).

Let y_r and $\hat{y}_r$ denote the observed and predicted arrival delays for observation r, respectively, and let N denote the total number of evaluated observations.

- **Mean Absolute Error (MAE)**

  Mean Absolute Error measures the average absolute difference between the actual and predicted delay values. A lower MAE value means that the model has smaller average prediction errors.

$$\text{MAE} = \frac{1}{N} \sum_{r=1}^{N} |y_r - \hat{y}_r| \tag{3.8.1}$$

  A lower MAE indicates a smaller average prediction error. Since delay is measured in minutes, MAE is also reported in minutes.

- **Root Mean Squared Error (RMSE)**

  Root Mean Squared Error gives more weight to large prediction errors. A lower RMSE value shows that the model makes fewer large errors.

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{r=1}^{N} (y_r - \hat{y}_r)^2}. \tag{3.8.2}$$

- **Coefficient of Determination (R^2)**

  The coefficient of determination, denoted as R^2, measures how well the model explains the variation in the actual delay values. A higher R^2 value indicates better model performance.

$$R^2 = 1 - \frac{\sum_{r=1}^{N} (y_r - \hat{y}_r)^2}{\sum_{r=1}^{N} (y_r - \bar{y})^2}, \tag{3.8.3}$$

  where $\bar{y}$ is the mean of the observed delay values. A higher R^2 indicates that the predictions explain more of the variation in the observed delays.

- **Huber Loss**

  Huber loss is used to reduce the effect of very large errors. It behaves like squared error for small mistakes and like absolute error for larger mistakes. This makes it useful when the data contains unusual delay values.

$$L_\delta(y_r, \hat{y}_r) = \begin{cases} \frac{1}{2}(y_r - \hat{y}_r)^2, & |y_r - \hat{y}_r| \leq \delta, \\ \delta\,|y_r - \hat{y}_r| - \frac{1}{2}\delta^2, & |y_r - \hat{y}_r| > \delta, \end{cases} \tag{3.8.4}$$

  where δ controls the point at which the loss changes from the squared-error form to the absolute-error form. The average Huber loss is calculated as:

$$\text{Huber} = \frac{1}{N}\sum_{r=1}^{N} L_\delta(y_r, \hat{y}_r). \tag{3.8.5}$$

  A lower Huber loss indicates better performance.

- **Tolerance-Based Accuracy**

  Tolerance-based accuracy measures the percentage of predictions that are within a selected time window of the actual delay. In this research, this metric is reported as $ACC@k$, where k represents the tolerance in minutes.

$$\text{ACC@}k = \frac{100}{N}\sum_{r=1}^{N} \mathbb{I}\left(|y_r - \hat{y}_r| \leq k\right), \tag{3.8.6}$$

  where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 when the condition is satisfied and 0 otherwise. A higher $ACC@k$ indicates that a larger percentage of predictions fall within the selected tolerance.

Together, these metrics provide both statistical and operational views of model performance. MAE and RMSE measure prediction error, R^2 measures goodness of fit, Huber loss reduces the effect of extreme errors, and $ACC@k$ shows how useful the prediction is within practical railway time limits.

3.9 Model Interpretability Using SHAP

In addition to evaluating predictive performance, a supporting interpretability analysis was conducted to examine the contribution of the input features. For this purpose, SHapley Additive exPlanations (SHAP) was used. SHAP is a feature-attribution method based on Shapley values from cooperative game theory and assigns a contribution value to each input feature (Lundberg and Lee, 2017; Lundberg et al., 2020).

For observation r in route section s, the prediction of model f_s can be represented as:

$$f_s\left(\mathbf{x}_{r,s}\right) = \phi_{0,s} + \sum_{j=1}^{p} \phi_{r,s,j}, \tag{3.9.1}$$

where $\phi_{0,s}$ is the baseline prediction for the model developed for section s, p is the number of input features, and $\phi_{r,s,j}$ is the SHAP value of feature j for observation r. A positive SHAP value indicates that the feature increases the predicted delay relative to the baseline, while a negative value indicates that it decreases the predicted delay. In this research, SHAP was used as a supporting analysis rather than as a direct explanation of BOEM. It was applied to the Gradient Boosting (GB) model to examine the features influencing the predictions of a strong non-linear baseline model. A separate GB model was developed for each route section, so feature importance was first calculated independently for every section. The importance of feature j in section s was calculated using the mean absolute SHAP value:

$$I_{s,j} = \frac{1}{N_s} \sum_{r=1}^{N_s} \left|\phi_{r,s,j}\right|, \tag{3.9.2}$$

where N_s is the number of evaluated observations in section s. The feature importance values were then normalised within each section and averaged across the section-level models to produce an overall importance summary:

$$\bar{I}_j = \frac{1}{S} \sum_{s=1}^{S} \widetilde{I}_{s,j}, \tag{3.9.3}$$

where $\widetilde{I}_{s,j}$ is the normalised importance of feature j in section s, and S is the total number of evaluated route sections. The variation in feature importance across sections

was also reported using the standard deviation. The SHAP results are presented in Section 5.11.5.

3.10 Experimental Setup

The experiments were implemented in Python using standard machine learning, deep learning, optimisation, and data analysis libraries. The main software and computing specifications used in this research are summarised in Table 3.2. Details of the datasets, experimental design, evaluation metrics, and proposed ensemble methods are presented in the relevant sections.

Table 3.2: Main software and computing specifications used in the experiments.

Component	**Specification**
Operating system	Rocky Linux 8.10 (Green Obsidian)
Processor	Intel Xeon Platinum 8562Y+ (one logical processor allocated)
Memory	128 GiB total RAM
Graphics processor	NVIDIA L40S with 46,068 MiB of GPU memory
Python version	Python 3.11.15
Main libraries	NumPy (1.26.4), pandas (2.3.3), scikit-learn (1.6.1), SciPy (1.17.0), SHAP (0.52.0), XGBoost (2.1.4), PyTorch (2.12.0), scikit-optimize (0.10.2), Optuna (4.9.0), joblib (1.5.3), TabM (0.0.3), TabPFN (7.1.1), PyTorch TabNet (4.1.0), PyTorch Tabular (1.2.0), and TabICL (2.1.1)
Development environment	Visual Studio Code, Jupyter Notebook, and Google Colab

3.10.1 Programming Language, Packages and Tools

The experimental work in this thesis was implemented mainly using Python. Python was selected because it provides strong support for data processing, machine learning, deep learning, statistical analysis, and visualisation. It also has a wide range of open-source libraries that are suitable for developing reproducible data-driven research. SQL was also used for storing, querying, and managing structured railway data. This was important because both historical and real-time railway data contain many records, and database queries were needed to extract route-level, section-level, and service-level information efficiently. Several Python packages were used during the research. The main packages include:

- **Pandas**: used for data loading, cleaning, transformation, feature construction, and tabular data analysis (McKinney et al., 2011).
- **NumPy**: used for numerical computation, array operations, and efficient data processing (Harris et al., 2020).
- **Scikit-learn**: used for implementing several machine learning models, preprocessing methods, model evaluation procedures, and regression metrics (Pedregosa et al., 2011).
- **XGBoost**: used to implement the extreme gradient boosting regression model (Chen et al., 2015).
- **Scikit-optimize (skopt)**: used to conduct the Bayesian optimisation procedure for searching continuous ensemble-weight combinations.
- **Optuna**: used for automated hyperparameter optimisation and searching the model configuration spaces (Akiba et al., 2019).
- **PyTorch**: used for implementing and evaluating the deep learning and modern tabular learning models (Paszke et al., 2019). The original publications describing the individual model architectures were also cited where the models were introduced.
- **Matplotlib** and **Plotly**: used for producing figures, exploratory data analysis plots, and result visualisations.
- **SHAP**: used for post-hoc model interpretation and feature attribution analysis of the selected tree-based baseline model (Lundberg and Lee, 2017).
- **SQLite**: used for local storage and querying of structured railway data during the historical and real-time experiments.

These tools supported different stages of the research, including data preparation, feature engineering, model training, ensemble construction, evaluation, and visualisation. The purpose of this section is only to introduce the main implementation environment. Detailed descriptions of the data processing pipeline, ensemble implementation, and real-time prediction system are provided in the next chapters.

3.11 Summary

This chapter established the quantitative experimental strategy used to investigate the five research questions. It linked each question to the relevant comparisons and evaluation evidence, explained the roles of the historical, more recent and live datasets, and described the prediction formulation, validation procedures, baseline comparisons and evaluation criteria.

These elements provide the methodological basis for interpreting the findings reported in Chapters 5 and 6. The historical investigations address ensemble weighting, optimisation, model-pool combination and reliability, while the chronological and live evaluations address real-time predictive performance. The next chapter presents the railway datasets, their preparation and exploratory analysis in greater detail.

Chapter 4

Data Collection, Preprocessing and Analysis

4.1 Introduction

This chapter explains how the railway data used in this research were collected, prepared and analysed for train arrival delay prediction. The study used three main data sources: Historic Service Performance (HSP), railway timetable data, and live data from Darwin PushPort. After describing the data sources, dataset construction and preprocessing procedures, the chapter examines the main delay patterns across routes, dataset phases, route sections and operating hours. It also analyses relationships between the variables and explains the implications of these observations for the subsequent modelling and evaluation.

HSP provided historical records of train services, including planned and actual station times (National Rail, 2026). The timetable data provided information about the planned structure of each service. This included station order, stopping patterns, and planned arrival and departure times (Open Rail Data Wiki, 2023). Darwin PushPort provided live updates about train services during operation, including actual and estimated times, cancellations, platform changes, and timetable changes (Rail Delivery Group, 2026).

The historical and timetable data were used to construct the datasets for model development and evaluation. They were also used to prepare the more recent datasets for training the models used in the real-time prediction framework. The live Darwin data were used to create model inputs during train operation and to support the prototype dashboard. Table 4.1 summarises the main data sources and their roles in

this research. The detailed implementation of the live data pipeline is presented in Chapter 6.

Table 4.1: Overview of the railway data sources used in this research.

Data source	Main information	Use in this research
Historic Service Performance (HSP)	Historical planned and actual arrival and departure times, station records, service dates, and service identifiers.	Used to collect historical train movement records and calculate delay values.
Timetable data	Planned station order, stopping patterns, route information, and planned arrival and departure times.	Used to check the journey structure and organise station records in the correct order.
Darwin PushPort	Live and estimated train times, service and station identifiers, cancellations, and changes to train services.	Used to prepare real-time model inputs, produce live predictions, and support the prototype dashboard.

The rest of this chapter is organised as follows. Section 4.2 introduces the historical train movement and timetable data. Section 4.3 describes the live train-running data. Section 4.4 presents the selected routes, data periods, and dataset phases. Section 4.5 explains how the raw station records were organised into train journeys and converted into route-section records. Section 4.6 describes the data cleaning and quality checks. Section 4.7 presents the feature construction process, and Section 4.8 explains the model-ready preprocessing steps. Section 4.9 describes the final datasets, and Section 4.10 discusses their main limitations. The exploratory analysis then compares delay variables in Section 4.11, examines delay distributions in Section 4.12, and explores variation across route sections and operating hours in Sections 4.13 and 4.14. Section 4.15 examines relationships between the variables, and Section 4.16 discusses the implications for modelling and evaluation. Finally, Section 4.17 summarises the chapter.

4.2 Historical Train Movement and Timetable Data

The historical datasets were created mainly from Historic Service Performance (HSP) data. HSP provides past train-running records from Darwin and allows train services to be searched through an application programming interface (API) (Open Rail Data, 2023; National Rail, 2026). The search used the origin station, destination station, date

range, time range, and Train Operating Company (TOC). The RID field identifies a specific train service on a given date.

To access the data, two HSP API requests were used. The *serviceMetrics* request was first used to identify the relevant train services. The *serviceDetails* request was then used to collect the station-level records for each service using its RID. The collected fields included the service date, RID, station code, planned times, and actual times. These fields were used to construct journeys and calculate train delays.

Timetable data were used to check the planned station order, stopping pattern, route, and travel direction. The selected routes and data periods are described in Section 4.4.

4.3 Live Real-Time Train-Running Data

This research uses live data to support real-time train arrival delay prediction. These data allow the system to receive current train information and prepare model inputs while a train is operating. Chapter 6 explains the full real-time prediction process in detail.

Live train-running data were collected from Darwin PushPort (National Rail Enquiries, 2023; Rail Delivery Group, 2026). The data were received through the Simple Text Oriented Messaging Protocol (STOMP). The main fields used in this research included the service identifier, station code, planned times, and actual or estimated times. These fields were used to identify the train, find its current route section, and prepare the model inputs. To use the live records correctly, the system matched them with timetable information. This step helped identify the correct route, station order, and travel direction, and ensured that the system used only the information needed for live delay prediction.

4.4 Selected Routes, Dataset Phases, and Data Coverage

This research mainly focuses on two railway routes. To avoid showing the real route and station names, the routes are reported using anonymised two-letter labels. The main routes are labelled NL and WW. NL includes 8 stations and 7 route sections,

while WW includes 20 stations and 19 route sections. The same anonymised labels are used throughout the thesis.

The selection of these routes was based on two main considerations. First, both routes provided sufficient historical station-level records for constructing the required route-section datasets. Second, they differed in route length and in the numbers of stations and route sections. This allowed the proposed prediction methods to be examined on two different route structures rather than on a single route. The two routes provide contrasting study cases for evaluating the proposed methods, but they are not treated as a representative sample of the wider railway network. The sample counts in Table 4.4 describe station-to-station observations within these routes; a larger number of observations or route sections does not imply a corresponding increase in the diversity of distinct railway operating environments. The findings are therefore interpreted within the selected routes and data periods. The allocation of observations to model development and evaluation is described in Section 3.7.

As explained earlier, this research collected the historical data at two different stages. The first stage provided the earlier datasets for the offline modelling experiments. The second stage provided newer datasets for further analysis and real-time model training. To make the datasets easier to identify and compare, these two stages are called Phase 1 (P1) and Phase 2 (P2). The four main datasets are labelled NL-P1, WW-P1, NL-P2, and WW-P2. The real-time framework also used other routes and reverse travel directions. These routes supported live route matching, prediction processing, and dashboard display. The suffix `-R` refers to the reverse direction of the related route. For example, `NL-R` refers to the reverse direction of `NL`.

Table 4.2 summarises the main datasets, their coverage, and their use in this research.

Table 4.2: Anonymised railway routes and data periods.

Main use	Route label	Number of stations	Data period
Offline modelling and proposed ensemble development	NL-P1	8	2021–2022 (6 months)
	WW-P1	20	2017–2018 (2 years)
Model training for real-time prediction	NL-P2	8	2024–2025 (2 years)
	WW-P2	20	2024–2025 (2 years)
Real-time section mapping and dashboard display	NL-R	8	2025
	WW-R	20	2025
	LK	6	2025
	LK-R	6	2025
	EB	10	2025
	EB-R	10	2025

4.5 Construction of Route-Section Records and Delay Values

The HSP source did not provide a ready-made dataset for route-section-level delay prediction. It provided separate station records for each train service. Therefore, this research designed and implemented a processing pipeline to create the modelling datasets from the raw railway records. Figure 4.1 shows an example of the original data format. Each row contains information about one train service at one station. A complete train journey may therefore appear across several rows.

rid	date_of_service	toc_code	location	planned_arrival_time	planned_departure_	actual_arrival_time	actual_departure_tin	late_canc_reaso
202501018111858	01/01/2025	SW			19:54		19:53	
202501018111858	01/01/2025	SW			20:01		20:02	
202501018111858	01/01/2025	SW		20:20	20:21	20:21	20:22	
202501018111858	01/01/2025	SW		20:29	20:30	20:30	20:31	
202501018111858	01/01/2025	SW		21:23	21:24	21:24	21:23	
202501018111858	01/01/2025	SW		21:39	21:40	21:40	21:40	
202501018111858	01/01/2025	SW		22:01	22:03	22:03	22:05	
202501018111858	01/01/2025	SW		22:16	22:17	22:17	22:19	
202501018111858	01/01/2025	SW		22:24	22:24	22:24	22:29	
202501018111858	01/01/2025	SW		22:35	22:39	22:39	22:42	
202501018111858	01/01/2025	SW		22:44	22:44	22:44	22:46	
202501018111858	01/01/2025	SW		22:47	22:47	22:47	22:51	
202501018111858	01/01/2025	SW		22:50	22:56	22:51	22:51	
202501018111858	01/01/2025	SW		22:56	22:53	22:56	23:00	
202501018111858	01/01/2025	SW		23:02	23:03	23:03	23:07	
202501018111858	01/01/2025	SW		23:09	23:10	23:10	23:14	
202501018111858	01/01/2025	SW		23:16	23:16	23:16	23:19	
202501018111858	01/01/2025	SW		23:23	23:24	23:24	23:29	
202501018111858	01/01/2025	SW		23:31	23:31	23:31	23:37	
202501018111858	01/01/2025	SW		23:36		23:39		

Figure 4.1: Example of raw station-level railway records.

The study first grouped the records by RID and service date. Each group represented one train service on one day. It then checked the station records against the planned route. This step identified the origin, destination, station order, and travel direction of each service.

The order of the raw records did not always provide the required journey structure. The study therefore used the planned station sequence to place the records in the correct order. It kept services that matched the selected route and direction. It removed incomplete journeys, services in other directions, and records that did not follow the required route structure.

The study then converted each valid journey into consecutive route-section records. Each record represents a movement from a current station to the next station. For example, a journey with eight stations produces up to seven route-section records. This process allowed one complete train journey to provide several prediction samples.

For each route section, the study linked the planned and actual times of the two stations. It then calculated arrival delay, departure delay, and dwell delay. Section 3.4 gives the full definitions of these delay measures. The study reported all delay values in minutes.

The arrival delay at the next station formed the prediction target. The input values described the train at the current station and the planned movement towards the next station. This structure ensured that each record followed the route-section prediction task defined in Chapter 3. Figure 4.2 shows an example of the dataset after journey construction and delay calculation.

This process created the route-section datasets used in this thesis. The study built these datasets from raw station-level railway records rather than using an existing modelling dataset. It applied the same main structure to the selected routes and data phases, which allowed the later models to use a consistent prediction format.

Journey Date	Day Of Week	Train	Hcode	Train Uid	Train Class	Peak	Profit Centre	Bugle Profit Centre	Origin	Public Start Time	Actual Start Time	Dest	Public End Time	Actual End Time	Location	Actual Activity	Working Time	Public Time
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		ARRIVE	9:47	9:48
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		DEPART	9:49	9:49
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	11:05	11:08
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	10:41	10:41
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		ARRIVE	10:08	10:09
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		DEPART	10:10	10:10
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		ORIGIN	9:32	9:32
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		X	10:59	10:59
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	11:12	11:12
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	11:10	11:10
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	10:47	10:47
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		ARRIVE	10:28	10:29
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		DEPART	10:30	10:30
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	11:06	12:36
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	11:06	11:06
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	10:00	10:50
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	9:57	9:57
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	10:11	10:12
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	10:06	10:05
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	10:07	10:07
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		PASS	11:17	11:13
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		TERMINUS	11:18	10:18
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		ARRIVE	10:18	10:13
11/12/2021	FRIDAY	1P25	481P25MF	481P25MF12	1	OFF	21781002	21781002		9:32:00	9:31:36		11:18:00	11:17:23		DEPART	10:19	10:20

Figure 4.2: Example of processed route-section records.

4.6 Data Cleaning and Quality Control

The study checked the route-section records before using them for modelling. These checks ensured that each record represented a valid train movement and included the information needed to calculate the input and target values.

For the historical datasets, the study removed records with missing planned or actual times when these values were needed to calculate the input or target delay. It did not estimate these values because this could create artificial training records or change the true prediction target.

The real-time framework followed a different process. During live operation, some input values, such as the current departure delay, were not always available as confirmed values. In these cases, the system estimated the missing input from the available live and historical information. The system did not use these estimates to change the historical target values. Chapter 6 explains this estimation process in detail.

The study also checked the service date, RID, station codes, and journey order. It removed records when the current and next stations did not form a valid route section or when the timing information did not match the journey structure. All time values were converted into the same format before the delay values were calculated. The study excluded cancelled services when no actual arrival time was available for the prediction target. The study treated records as duplicates when the same RID, service date, current station, and next station appeared more than once with the same information.

In these cases, it kept one record and removed the repeated entries.

The cleaning process did not use a delay-based threshold. Large delay values were kept when their station and timing information was valid. The study only removed records with missing, repeated, or inconsistent information. This approach kept real delay events in the dataset and avoided removing unusual but valid train movements. All time values were converted into the same format before the delay values were calculated. The study applied the same main quality checks to each route and data phase. The final sample sizes after cleaning are reported in Section 4.9.

4.7 Feature Engineering for the Final Dataset

After data cleaning, the study created the variables needed for route-section-level prediction. Each record uses information from the current station and route section to predict the arrival delay at the next station.

The model inputs include operational, historical-delay, and temporal features. Operational features describe the current train condition. They include departure delay and dwell delay.

Historical-delay features describe the recent delay history of the same train. They include the latest available arrival, departure, and dwell delays, together with summary values from earlier parts of the journey. The study used these features only when the required information was available.

Temporal features describe when the train service operated. They include day of week, day of month, hour of day, month, season, weekend, holiday, and peak-period indicators.

Table 4.3 summarises the main feature groups. Appendix 8.3 provides a full description of all features. The study did not use the route, station, RID, or service date as direct model inputs. It used these fields to identify services, select route sections, group records, and check the data.

The study also took steps to prevent information leakage. It created each input from information available before the train arrived at the next station. It did not use the target value of the same record or information from later parts of the journey. Historical summary features used only earlier available records.

Table 4.3: Main variables in the final datasets.

Variable group	Example variables	Purpose
Operational features	Departure delay and dwell delay	Describe the current train condition and route section.
Historical-delay features	Last checkpoint arrival delay, last checkpoint departure delay, last checkpoint dwell delay, and mean previous delay values	Describe the recent delay history of the train.
Temporal features	Day of week, hour, month, season, weekend, holiday, and peak period	Describe the time of operation.
Prediction target	Arrival delay at the next station	Provide the value predicted by the models.
Supporting fields	RID, service date, route, current station, and next station	Support service matching, route selection, and data checking.

The real-time models used a smaller feature set. During live operation, some previous-station values were not always available or reliable. Estimating these values could introduce additional noise and error, and during development they were found to increase the prediction error of the models. The study therefore removed these features from the real-time model inputs. It trained the real-time models with the same reduced feature set used during live prediction. This ensured that the offline training and real-time input structures remained consistent.

4.8 Model-Ready Preprocessing

After feature engineering, the study prepared the selected variables for model training. The preprocessing method depended on the type of each variable.

The numerical variables included departure delay, dwell delay, last checkpoint delay values, and mean previous delay values. The study applied `StandardScaler` when a model required scaled numerical inputs (Pedregosa et al., 2011). This step helped scale-sensitive models, such as SVR, KNN, and neural network models.

The study kept day of month, hour of day, and month as ordered numerical variables using ordinal encoding. It used one-hot encoding for day of week and season. It also converted weekend status, holiday status, and peak-period status into binary values.

The current station, next station, and route direction supported route matching, section definition, and data checking. The arrival delay at the next station formed the prediction target and was not included in the model input matrix.

The historical-delay features included the last checkpoint arrival, departure, and dwell delays, together with their mean previous values. The offline models used these features when they were available. The real-time models did not use them because live data did not always provide these values reliably.

The study fitted each preprocessing step only on the training data to avoid data leakage. It then applied the fitted transformations to the validation and test data. During cross-validation, the study fitted the preprocessing steps again within each training fold. This process prevented information from the validation or test data from entering the training stage.

The final output was a model-ready feature matrix for the individual models used in the ensemble framework.

4.9 Final Dataset Summary

The data collection and processing steps produced four final datasets: NL-P1, WW-P1, NL-P2, and WW-P2. The study processed each dataset separately and converted the raw station records into the common route-section format.

The datasets cover two railway routes and two data phases. NL contains 8 stations and 7 route sections, while WW contains 20 stations and 19 route sections. The two phases also cover different time periods and contain different numbers of records. As NL and WW datasets were collected over different time periods, comparisons between the two routes may reflect a combination of route characteristics, timetable conditions, service patterns, and period-specific operating conditions. Table 4.4 summarises the main structure of the final datasets. The sample counts show the records available after journey construction and data cleaning. These prepared datasets are examined in

Table 4.4: Summary of the final route datasets.

Measure	**NL-P1**	**WW-P1**	**NL-P2**	**WW-P2**
Data period	Oct. 2021–Mar. 2022	Jan. 2017–Dec. 2018	Jan. 2024–Dec. 2025	Jan. 2024–Dec. 2025
Stations	8	20	8	20
Route sections	7	19	7	19
Final route-section samples	17,435	185,976	100,895	192,499

the exploratory analysis later in this chapter and are used for model development and evaluation in the subsequent chapters.

4.10 Data Limitations

The datasets include planned and actual train times, station information, route sections, and delay values. However, they do not include all factors that may affect train delays. For example, the data do not provide full information about weather, passenger demand, staff problems, rolling stock faults, or infrastructure failures. The models, therefore, use the available train-running information to predict delay values. They do not identify the exact cause of each delay.

Some historical records did not contain all the times needed for journey construction and delay calculation. The study removed these records during data cleaning. Live data also had some limits because updates could arrive late and sometimes remained estimated until they were confirmed. The datasets cover selected routes and time periods. Other routes or periods may show different delay patterns. These limits define the data scope of this research.

The prepared datasets supported training and comparative evaluation of the selected models under the experimental design described in Section 3.7. However, the information available to each section-specific model depends on its own training records and delay distribution, rather than the total sample count across all routes and sections. Most observations are close to the timetable, while larger delays are less frequent, so the data provide more examples of routine running conditions than severe disruption. The different route structures, collection periods and observed delay patterns provide contrasting evaluation settings, but do not cover every operating condition. Dataset adequacy is therefore assessed through held-out predictive performance, variation across sections, chronological evaluation and the targeted robustness and transfer experiments, rather than sample size alone.

4.11 Overall Comparison of Delay Variables

Table 4.5 presents the main arrival-delay characteristics of the four datasets. The mean values are reported together with their standard deviations (SD). The WW datasets have higher mean arrival delays and greater variation than the NL datasets. This comparison shows that the two routes have different delay patterns and that their delay

behaviour also changes between the two phases.

Table 4.5: Arrival-delay characteristics of the four datasets.

Measure	**NL-P1**	**WW-P1**	**NL-P2**	**WW-P2**
Mean arrival delay (SD), minutes	0.10 (2.99)	1.09 (5.36)	0.10 (3.24)	1.39 (5.28)
Mean positive arrival delay (SD), minutes	2.83 (4.54)	5.01 (7.75)	4.33 (5.56)	5.62 (6.83)

Figure 4.3 compares the positive-delay rates. In this chapter, the positive-delay rate is the percentage of records with a delay greater than zero minutes. This definition differs from the *On Time* measure discussed in Chapter 1, which also includes trains arriving up to one minute late. Dwell delay has the highest rate in all four datasets. The arrival and departure delay rates are generally higher in the WW datasets than in the NL datasets. The comparison between the two phases also shows different patterns for the two routes. The rates are generally lower in NL-P2 than in NL-P1, while several delay rates are higher in WW-P2 than in WW-P1.

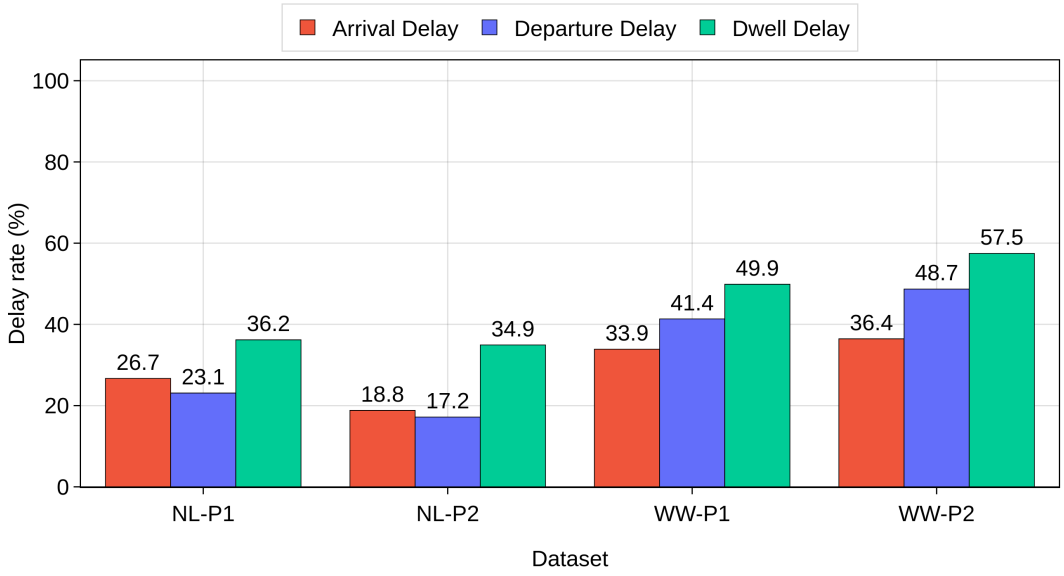


Figure 4.3: Positive-delay rates for arrival, departure, and dwell delays across the four datasets.

Figure 4.4 compares the mean delay values. The WW datasets generally have higher mean delays than the NL datasets, particularly for departure delay. In contrast, the mean values for NL remain closer to zero. The findings also differ between Phase 1 and Phase 2, although the size and direction of these differences vary across the delay variables.

Overall, delays are generally more frequent and larger in the WW datasets, while the

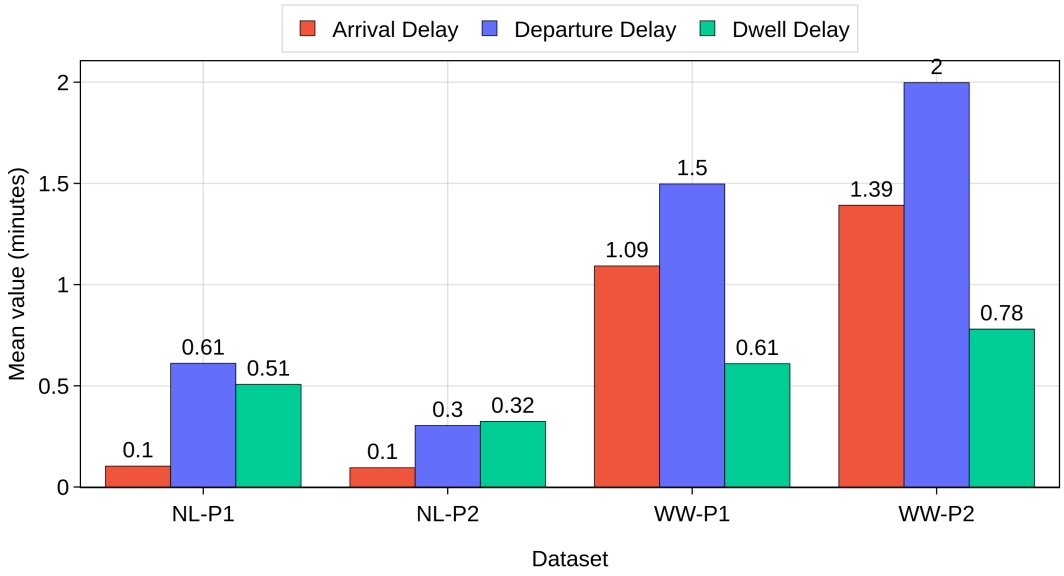


Figure 4.4: Mean arrival, departure, and dwell delays across the four datasets.

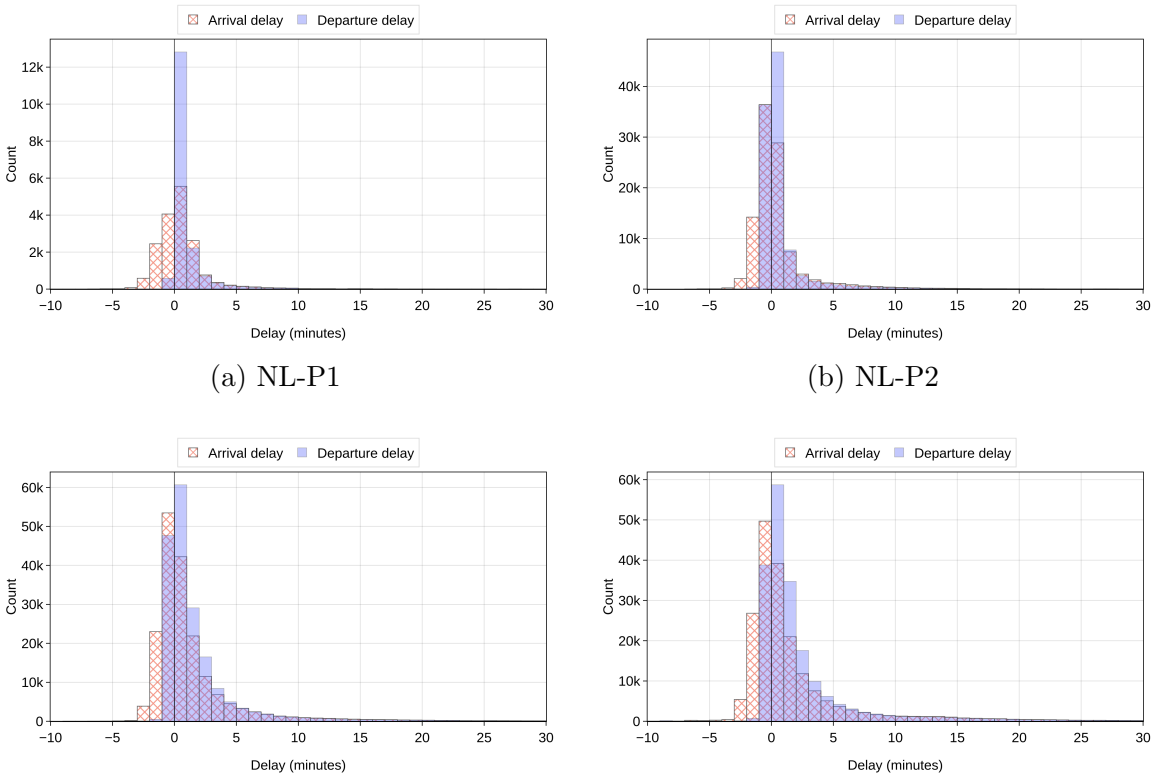
NL datasets remain closer to the planned timetable. The differences between Phase 1 and Phase 2 also show that the delay patterns are not identical across the two data periods.

4.12 Distribution of Arrival and Departure Delays

Mean values and positive-delay rates give only an overall view of the data. This section examines how arrival and departure delays are distributed across different delay ranges. The uneven arrival-delay distribution reflects the naturally occurring frequencies in the recorded railway services. In this regression task, the objective is to predict continuous delay values across the observed train-running conditions, including frequent near-timetable arrivals and less frequent larger delays. Retaining the observed distribution allows model development and full-data evaluation to reflect this operational mix and provides a common basis for comparing the methods. Larger-delay ranges contain fewer observations, so their performance is also examined separately in Section 5.11.4 and interpreted within the available sample coverage. This evaluation strategy combines assessment across the recorded service conditions with explicit examination of the less frequent delay ranges.

Figure 4.5 shows the arrival and departure delay histograms. Most observations are close to zero, while the number of records decreases as the positive delay becomes larger. The WW datasets contain a larger proportion of observations at higher positive

delay values, particularly in Phase 2.



(a) NL-P1 (b) NL-P2

(c) WW-P1 (d) WW-P2

Figure 4.5: Arrival and departure delay distributions across the four datasets.

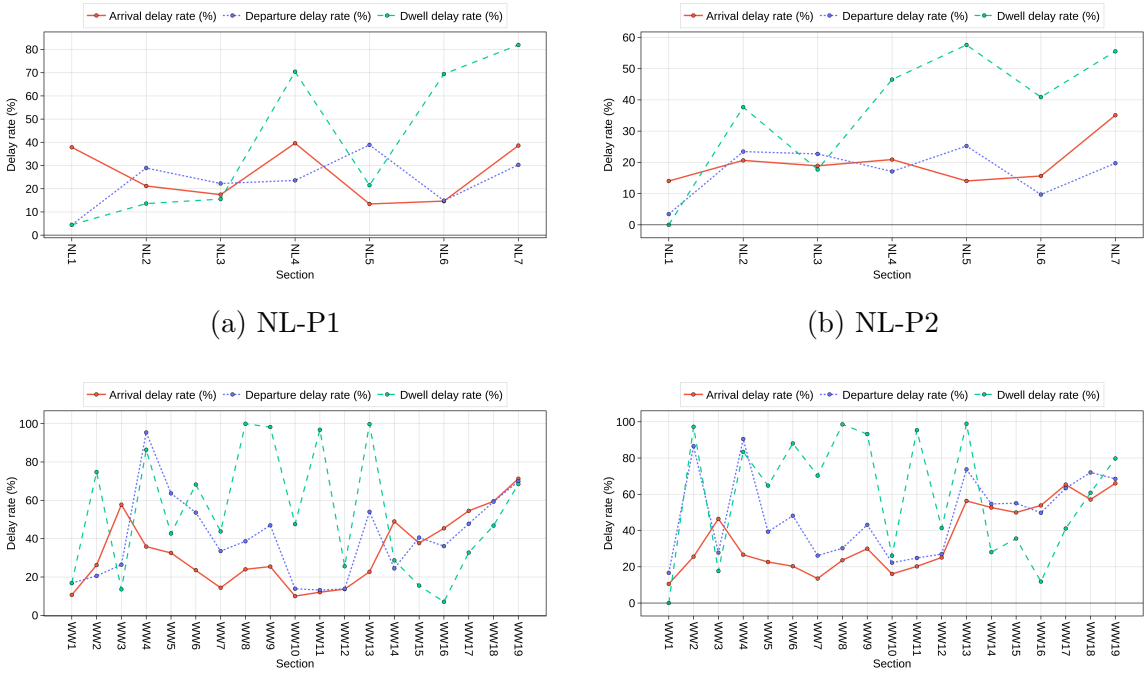
The analysis shows that the datasets contain many small-delay cases but fewer large-delay cases. The later model evaluation therefore examines performance across different delay ranges as well as overall performance.

4.13 Section-Level Variation in Delay Variables

Delay behaviour may differ between route sections. This section compares the positive-delay rates and mean delay values for each station-to-station section.

Figure 4.6 shows the positive-delay rates for arrival, departure, and dwell delay. The rates vary across sections in all four datasets. The NL route has fewer sections, and the differences between them are generally smaller. The WW datasets shows wider variation, particularly in dwell delay rates.

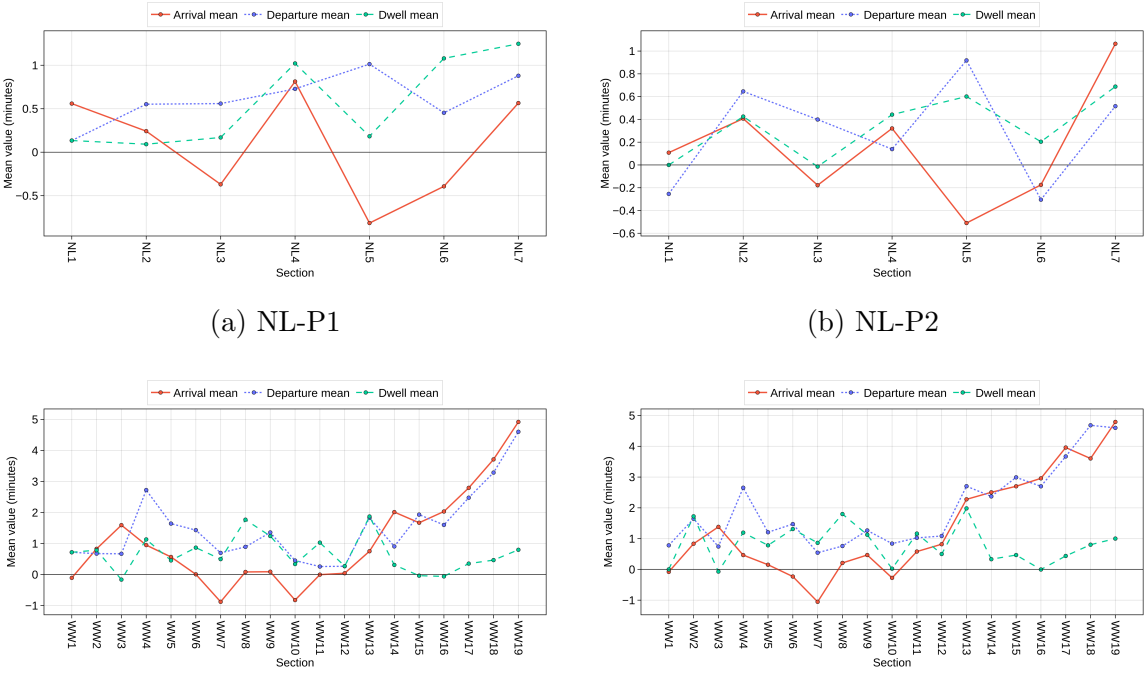
Figure 4.7 compares the mean delay values. A section can have a high positive-delay rate but a small mean delay when most delays are short. A lower delay rate can also be linked to a larger mean value when fewer but longer delays occur. The two measures therefore show different parts of section-level delay behaviour.



(a) NL-P1 (b) NL-P2

(c) WW-P1 (d) WW-P2

Figure 4.6: Positive-delay rates across route sections in the four datasets.



(a) NL-P1 (b) NL-P2

(c) WW-P1 (d) WW-P2

Figure 4.7: Mean delay values across route sections in the four datasets.

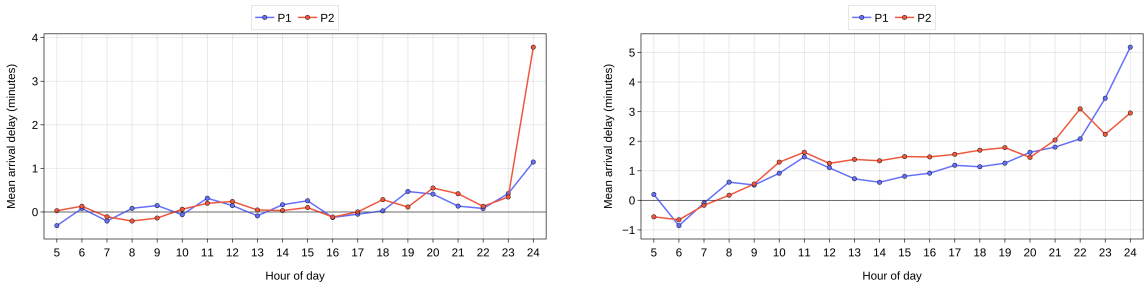
The NL sections generally have small mean delays, although their positive-delay rates are not the same. The WW sections show larger differences and show greater variation in mean arrival and departure delays in both phases. These observations show that delay patterns are not uniform along a route.

The section-level differences are considered in the later model evaluation. The findings describe only the two routes and data periods used in this research and are not intended to represent the whole railway network.

4.14 Hour of Day Variation in Delay Variables

This section examines how arrival delays change across different hours of the day. It considers the mean arrival delay and the percentage of trains with a positive arrival delay.

Figure 4.8 compares the mean arrival delay across the operating day. For NL, the hourly mean values remain close to zero for most hours. A larger value appears near the end of the day in Phase 2. For WW, the mean arrival delay is generally higher during the evening and later operating hours.

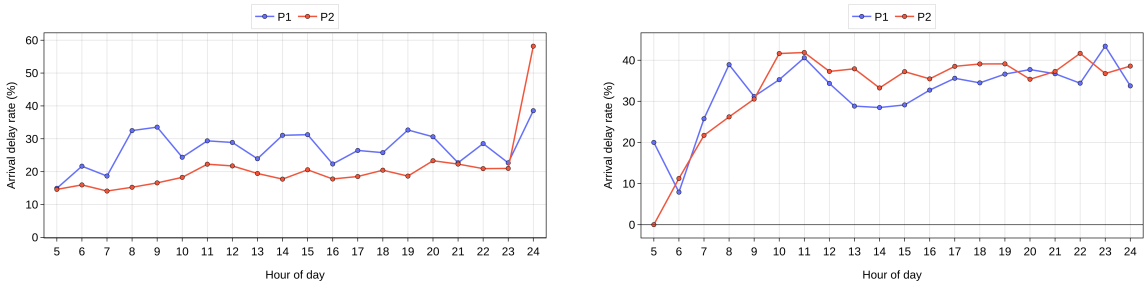


(a) NL (b) WW

Figure 4.8: Mean arrival delay by hour of day for Phase 1 and Phase 2.

Figure 4.9 shows the percentage of records with a positive arrival delay at each hour. NL-P2 has a lower delay rate than NL-P1 during most hours. In contrast, the WW datasets show higher delay rates across much of the operating day.

The hourly observations show that delay levels are not constant across the operating day. The differences also vary between the two routes and data phases.



(a) NL (b) WW

Figure 4.9: Positive arrival-delay rate by hour of day for Phase 1 and Phase 2.

4.15 Correlation Between Variables

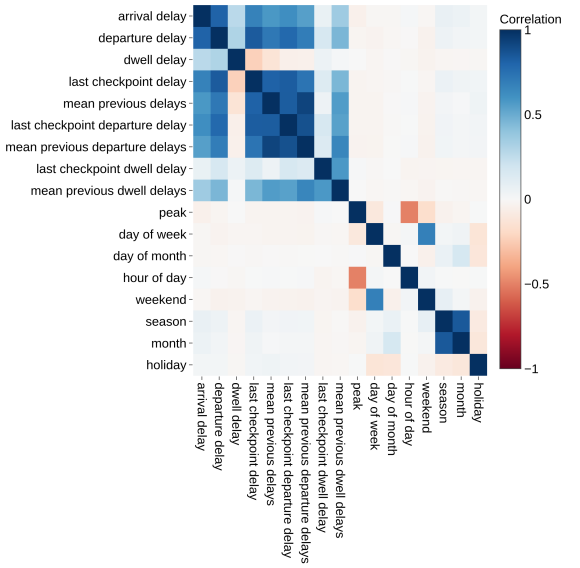
Correlation analysis was used to examine the linear relationships between arrival delay and the main input variables. The analysis was carried out separately for each route and phase. Correlation describes the strength and direction of a linear relationship, but it does not show that one variable causes another.

The variables included in the matrices are not the same in both phases. Phase 1 includes previous-delay variables in addition to the main operational and temporal variables. Phase 2 uses a smaller feature set that is more suitable for the real-time prediction setting. The full feature sets for both phases were described in Section 4.7.

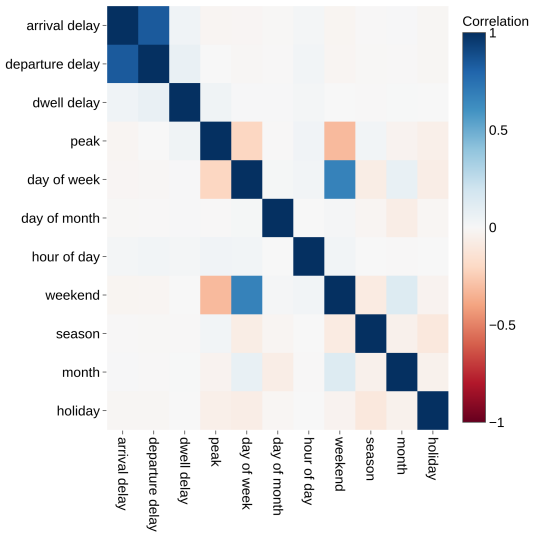
Figure 4.10 shows the correlation matrices for the four datasets. Departure delay has the strongest relationship with arrival delay in all cases. Departure delay at the current station therefore provides strong information about arrival delay at the next station.

In Phase 1, the previous-delay variables also have strong relationships with arrival delay. In Phase 1, the correlations with last checkpoint delay and mean previous delay are also high. These findings show that the earlier delay state of a train is closely related to its delay at the next station.

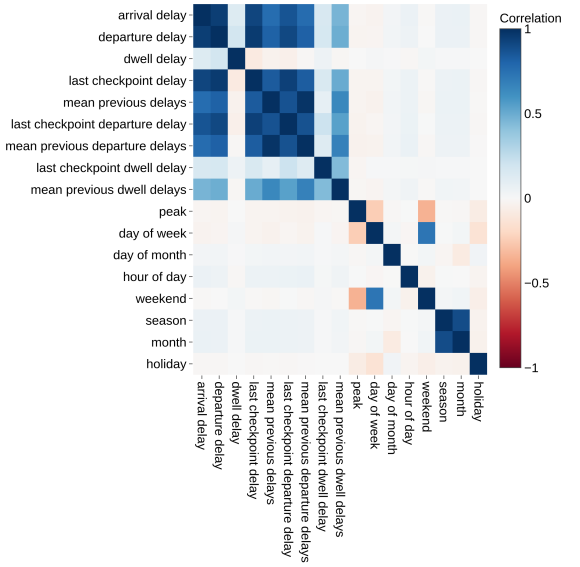
Dwell delay has a weaker relationship with arrival delay. Dwell delay may still provide operational information, but its direct linear relationship with the target is limited in these datasets. Most temporal variables have weak linear correlations with arrival delay. This does not mean that they have no value for prediction. Their effects may be non-linear or may depend on other variables. The later modelling experiments examine whether these variables improve prediction when they are used with the



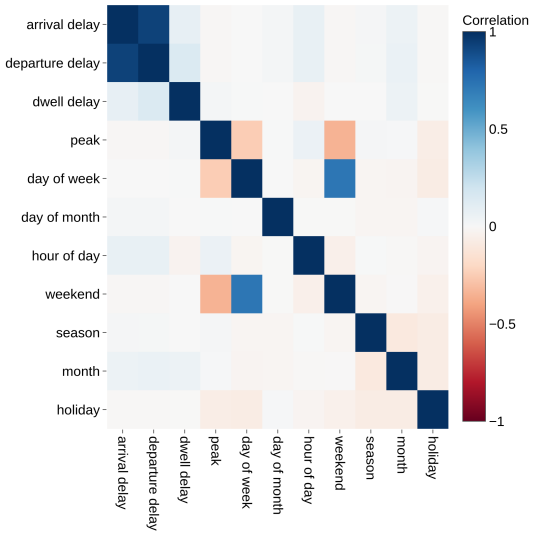
(a) NL-P1



(b) NL-P2



(c) WW-P1



(d) WW-P2

Figure 4.10: Correlation matrices for the NL and WW datasets in Phase 1 and Phase 2.

operational features. Overall, departure delay has the strongest linear relationship with next-station arrival delay. Previous-delay variables are also strongly related to the target in Phase 1, especially for WW. The remaining variables show weaker direct linear relationships.

4.16 Implications for Modelling and Evaluation

The exploratory analysis provides useful information for the modelling and evaluation presented in the following chapters. The main findings are linked to several experiments reported in Chapters 5 and 6.

- The delay distributions show that most records are early, on time, or affected by small delays. Larger delays occur less often, particularly in the NL datasets. Reporting only overall performance could therefore hide poor predictions for these less frequent cases. Section 5.11.4 compares model performance across different delay ranges.

  The section-level analysis shows that delay behaviour is not uniform along a route. Some sections have frequent but small delays, while others have fewer but larger delays. The section-level evaluations in Sections 5.11.3 and 6.11.3 examine whether model performance remains consistent across these local conditions.

- The NL and WW routes have different lengths, numbers of sections, and delay characteristics. Larger delays were observed in the WW datasets. These differences provide two different evaluation settings. They are considered in the route-level results and in the cross-dataset generalisation analysis in Section 5.11.10.

- The comparison between Phase 1 and Phase 2 shows that the delay values and rates are not identical across the two data periods. This means that model performance should be tested on later observations, rather than only through a random division of the data. Time-aware validation is therefore used for the more recent datasets, with the results reported in Section 6.11.

- The hourly analysis shows that arrival delay varies across the operating day. Hour of day has a weak direct linear relationship with arrival delay, but it may

still provide useful information when combined with operational variables. It is therefore retained as a temporal input in the modelling datasets.

- Departure delay has the strongest linear relationship with next-station arrival delay in all four datasets. It is therefore an important input for prediction. However, an actual departure value may not yet be available when a live prediction is required. The estimation methods described in Section 6.5 are used to manage this condition. Estimated departure information also supports the recursive downstream prediction process described in Section 6.8.

Overall, the analysis explains why the later evaluation considers different delay ranges, routes, route sections, data periods, and real-time input conditions. It also provides the data background needed to interpret the model results presented in Chapters 5 and 6.

4.17 Summary

This chapter described the collection, preprocessing and exploratory analysis of the railway datasets used in this research. Historical HSP records, timetable information and live Darwin updates were introduced, and the construction of route-section records, data cleaning, feature engineering and model-ready preprocessing were explained. The exploratory analysis compared the NL and WW datasets across both phases. Most records were close to the planned timetable, while larger delays occurred less frequently. Delay behaviour varied across routes, route sections and operating hours, with the WW datasets generally showing higher delays and greater variation. Correlation analysis identified a strong linear relationship between current-station departure delay and next-station arrival delay. These observations provide the data context for the subsequent modelling and evaluation, including comparisons across delay ranges, route sections and data periods.

Chapter 5

Proposed Ensemble Frameworks for Train Delay Prediction

5.1 Introduction

This chapter presents the development and historical evaluation of the proposed ensemble modelling framework for train arrival delay prediction. The framework uses conventional machine learning and tabular deep learning models to develop route-section-level predictors. It combines their predictions using the Average Ensemble (AE) as a baseline method and the proposed Multi-Metric Weighted Ensemble Method (MWEM) and Multi-Metric Bayesian-Optimised Ensemble Method (BOEM).

The framework uses the modelling datasets collected, prepared and analysed in Chapter 4. These datasets provide the input features and next-station arrival-delay target used for model training. The chapter first explains the candidate models, ensemble configurations, model-selection procedures and calculation of ensemble weights. It then evaluates the resulting predictors using historical railway data under the experimental design described in Chapter 3.

The evaluation compares the proposed methods with individual models, equal-weight averaging and selected methods reconstructed from published train delay prediction studies. Additional analyses examine weighting strategies, section-level performance, delay magnitudes, feature importance, statistical rankings, asymmetric prediction cost, missing departure information and cross-dataset generalisation. The discussion interprets this evidence and explains how it addresses RQ1–RQ4. Chapter 6 subsequently examines the preparation and evaluation of the ensemble framework for real-time prediction.

The chapter proceeds as follows. Section 5.2 presents the overall modelling framework. Section 5.3 describes the base model pool, and Section 5.4 explains the heterogeneous and hybrid configurations. Section 5.5 introduces AE, while Sections 5.6 and 5.8 present MWEM and BOEM. Section 5.10 explains final model training and ensemble prediction. Section 5.11 presents the historical evaluation results. Section 5.12 discusses these findings, and Section 5.13 answers RQ1–RQ4. Finally, Section 5.14 summarises the chapter.

Table 5.1 summarises the principal mathematical notation used in this chapter.

Table 5.1: Principal mathematical notation used in this chapter.

Notation	Meaning
D_{train}	Training data available for one route section.
T_i^D and V_i^D	Internal training subset and internal validation subset, respectively, in validation round i.
Y_i^V	Vector of true target values in validation subset V_i^D.
$L = \{l_1, l_2, \ldots, l_m\}$	Candidate set of m base models.
l_j	The jth base model in the candidate model set.
m	Number of candidate base models.
f	Number of internal validation rounds.
$L_i^{\text{ranked}(E_i)}$	Ordered list of candidate models ranked from best to worst according to evaluation metric E_i in validation round i.
q	Number of candidate weight vectors evaluated by Bayesian optimisation in each validation round.
i, j, and k	Indices used for validation rounds, base models, and model-based summations, respectively.
$E = \{E_1, E_2, \ldots, E_f\}$	Predefined sequence of evaluation metrics assigned to the validation rounds.
E_i	Evaluation metric assigned to validation round i.
$E_i(l_j)$	Validation performance score of model l_j using metric E_i.
$\hat{Y}_j$	Prediction vector produced by model l_j.
$\hat{Y}_{i,j}$	Validation prediction vector produced by model l_j in validation round i.
ω_j^{AE}	Equal weight assigned to model l_j by the Average Ensemble.
$\hat{Y}^{\text{AE}}$	Prediction vector produced by the Average Ensemble.
Φ_i	Set of models retained by the MWEM greedy selection procedure in validation round i.
Ψ_i	Temporary model set formed when a new model is added to Φ_i during greedy selection.
$\hat{Y}_i^{\Psi_i}$	Temporary ensemble prediction vector produced by the models in Ψ_i.
θ_i^{new} and θ_i^{best}	Temporary ensemble score and best ensemble score obtained so far in validation round i.
$\gamma_{i,j}$	Non-negative performance score of retained model l_j for a higher-is-better metric in validation round i.
ϵ	Small positive constant used to prevent division by zero.
$\omega_{i,j}$	MWEM weight assigned to model l_j in validation round i.
$\mathcal{W}_i$	Complete MWEM weight vector obtained in validation round i.
ω_j^*	Final weight assigned to model l_j after averaging the round-specific weights.
$\mathcal{W}^*$	Final weight vector produced by MWEM or BOEM.
$\mathcal{E}$	Ensemble representation; when used as a prediction superscript, it denotes an ensemble prediction.
$\Omega_i^{c,\text{raw}}$	Raw candidate weight vector proposed by Bayesian optimisation in validation round i.
$\omega_{i,j}^{c,\text{raw}}$	Raw candidate weight proposed for model l_j in validation round i.
Ω_i^c	Normalised candidate weight vector evaluated by BOEM in validation round i.
$\omega_{i,j}^c$	Normalised candidate weight assigned to model l_j in validation round i.
$\hat{Y}_i^{\mathcal{E}}(\Omega_i^c)$	BOEM ensemble prediction vector produced using candidate weight vector Ω_i^c.
$\mathcal{L}_i(\Omega_i^c)$	BOEM objective value of candidate weight vector Ω_i^c in validation round i.
$\mathcal{C}_i^{(q)}$	Set of the q normalised candidate weight vectors evaluated in validation round i.
$\Omega_i^{c,\text{best}}$	Best normalised candidate weight vector found in validation round i.
$\omega_{i,j}^{c,\text{best}}$	Weight assigned to model l_j by the best BOEM candidate in validation round i.
$\mathcal{A}$	List of models used by the final ensemble.
$\mathcal{W}$	Weight vector corresponding to the models in $\mathcal{A}$.
z	Number of models used by the final ensemble.
x and X_{test}	One unseen input observation and a set of unseen input observations, respectively.
$\hat{y}_j$ and $\hat{y}^{\mathcal{E}}$	Prediction of base model l_j and final ensemble prediction, respectively, for one observation.
$\hat{Y}_j$ and $\hat{Y}^{\mathcal{E}}$	Prediction vector of base model l_j and final ensemble prediction vector, respectively, for multiple observations.
n_v	Number of observations in an internal validation subset.
T_{train} and T_{full}	Average cost of training one base model during internal validation and on the full training dataset, respectively.
$\mathcal{O}(\cdot)$	Asymptotic computational complexity.

5.2 Overview of the Proposed Modelling Framework

Figure 5.1 shows the overall structure of the proposed modelling framework. The framework connects the prepared section-level railway data to the trained ensemble predictors used for next-station arrival delay prediction.

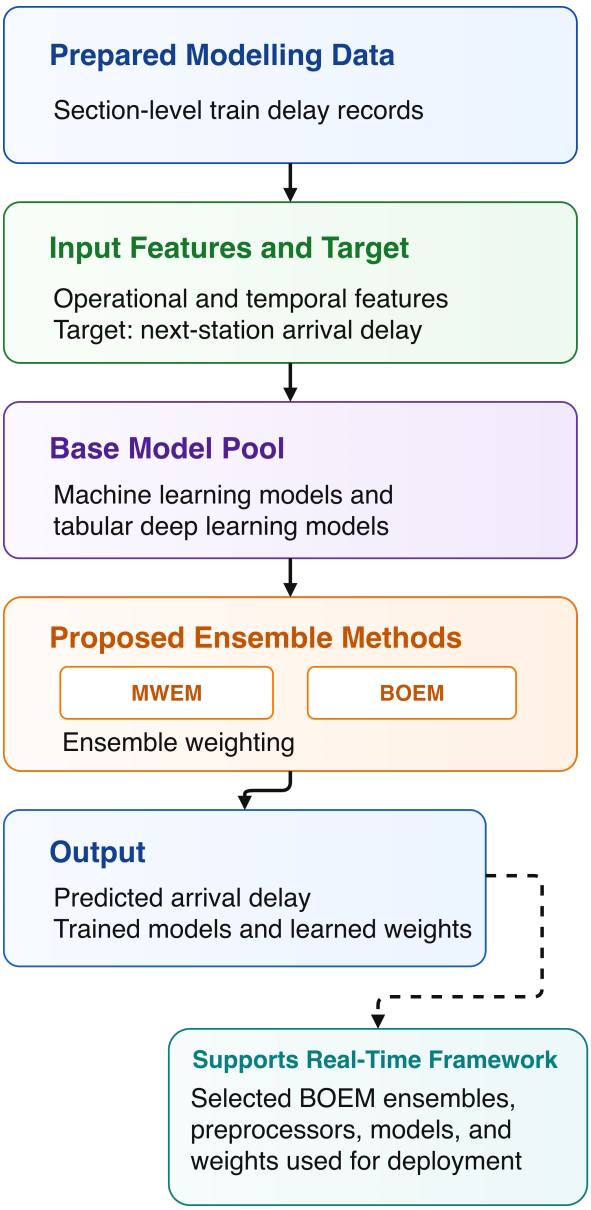


Figure 5.1: Overview of the proposed modelling framework and its connection to the real-time prediction framework.

The first two stages of the framework summarise the prepared modelling data, the input features, and the prediction target. Chapter 4 explain these stages in detail. The remaining stages cover the development of the base model pool, the construction of the ensemble methods, and the production of trained section-level predictors. The framework develops the prediction models separately for each route section. It first

trains the candidate base models and then combines their predictions using AE, MWEM, or BOEM. After training, it stores the fitted models and learned weights for prediction on unseen data. The final stage in the figure shows how these trained predictors connect to the real-time framework discussed in Chapter 6. The following sections explain each modelling component in more detail.

5.3 Base Models

The literature review in Chapter 2 showed that train delay prediction studies have used a wide range of machine learning and deep learning models. These studies reported different levels of performance across datasets, railway networks, prediction tasks, and operating conditions (Spanninger et al., 2022; Tiong et al., 2023b). Recent studies have also started to examine modern tabular deep learning models for railway prediction, although their use in this area remains limited (Al Ghamdi et al., 2024; Arthaud et al., 2024).

The data analysis in Chapter 4 also showed that delay behaviour varies across routes, route sections, time periods, and operating conditions. A model that represents the delay pattern of one section well may perform differently in another section. For this reason, the proposed framework does not rely on one model family. Instead, it uses a broad pool of models with different learning structures.

The base model pool includes conventional machine learning models and tabular deep learning models. The conventional machine learning group covers linear, kernel-based, instance-based, tree-based, boosting, and neural approaches. The tabular deep learning group includes recent neural models developed for structured tabular data. Depending on the ensemble configuration, the framework uses the conventional machine learning pool, the tabular deep learning pool, and both groups together.

Different model families can represent different relationships in the railway data. Linear models provide a simple representation of relationships between the input features and arrival delay. Tree-based and boosting models can represent non-linear relationships and interactions between operational and temporal variables. Kernel-based and instance-based models use different forms of similarity to model complex delay patterns. Neural and tabular deep learning models can learn more complex interactions

between the input features. Table 5.2 summarises these model groups and their main roles in the framework.

Table 5.2: Base model groups used in the proposed ensemble framework.

Model group	**Models**	**Main role in the framework**
Linear models	Linear Regression (LR), Bayesian Ridge (BR), and Elastic Net (ENet)	Represent linear relationships between the input features and arrival delay.
Tree-based and boosting models	Decision Tree (DT), Random Forest (RF), Extra Trees (ET), AdaBoost (AB), Gradient Boosting (GB), and XGBoost (XGB)	Represent non-linear relationships and interactions between operational and temporal features.
Kernel-based model	Support Vector Regression (SVR)	Represent non-linear relationships through a kernel-based learning approach.
Instance-based model	k-Nearest Neighbours (KNN)	Estimate delay from similar section-level observations in the training data.
Neural model	Multilayer Perceptron (MLP)	Learn non-linear relationships through a fully connected neural network.
Tabular deep learning models	FT-Transformer (FTT), TabNet, TabPFN, TabM, and TabICL	Learn complex feature relationships using architectures developed for structured tabular data.

The base models have two roles in this research. First, each model acts as an individual predictor and provides a reference for evaluating the ensemble methods. Second, the models act as base learners whose predictions form the inputs to AE, MWEM, and BOEM. This design allows the experiments to compare the ensemble methods with the same models that they combine. For each route section, the framework trains every base model independently using the prepared section-level training data. All models use the same input features and the same next-station arrival delay target for that section. Each trained model then produces its own prediction. The ensemble stage combines these predictions according to the weighting procedure of AE, MWEM, or BOEM.

As discussed in Section 2.3, the model pool includes several learning families because ensemble methods depend on differences in model behaviour. If all models produced almost the same predictions, combining them would provide little additional information. However, the framework does not assume that model diversity alone leads to better

predictions. The validation procedures used by MWEM and BOEM determine how much each model contributes to the final prediction.

The conventional machine learning models use the configurations selected through hyperparameter tuning during model development. The tabular deep learning models use stable configurations based on their original studies or official implementations, with limited route-specific tuning. The aim is to assess their contribution to the proposed ensemble framework rather than to develop new neural architectures. These models are established methods in machine learning and tabular deep learning. Therefore, this chapter focuses on their roles within the proposed framework rather than repeating their full theoretical descriptions. Appendix 8.4 provides a short description of each model together with its implementation settings and selected hyperparameters.

5.4 Heterogeneous and Hybrid Ensemble Framework

After the base models produce their predictions, the next stage of the framework is to combine their outputs. Ensemble learning provides different ways to perform this combination, depending on the types of models included in the model pool.

A common distinction is between homogeneous and heterogeneous ensembles (Sagi and Rokach, 2018; Ganaie et al., 2022). A homogeneous ensemble combines models from the same learning family. For example, a random forest combines several decision trees. In contrast, a heterogeneous ensemble combines models developed using different learning approaches, such as linear, tree-based, kernel-based, instance-based, and neural models.

The model pool used in this research contains models from several learning families. The proposed framework therefore uses a heterogeneous ensemble structure to combine their predictions. This design is also supported by the literature reviewed in Chapter 2, which shows that heterogeneous ensembles can combine the complementary strengths of different learning paradigms and reduce correlated prediction errors. This choice also follows the findings presented in Chapter 4, where delay behaviour varied across routes, route sections, time periods, and operating conditions. These differences suggest that one model family may not represent all delay patterns equally well. A heterogeneous structure allows models with different learning behaviours to contribute to the same

final prediction. Previous train delay studies have also used combinations of different prediction models to represent complex railway conditions (Nair et al., 2019; Al Ghamdi et al., 2024). The framework developed in this research examines a model pool containing several conventional machine learning and tabular deep learning families and applies different procedures for selecting and weighting their predictions.

The framework also supports a hybrid ensemble configuration. In this research, the term *hybrid ensemble* refers to a model pool that includes both conventional machine learning models and tabular deep learning models. The models remain separate predictors and are trained independently. The framework combines their outputs only at the ensemble stage. Therefore, the hybrid structure describes the composition of the model pool rather than a single integrated model architecture.

The framework uses three main methods to combine the base-model predictions. The Average Ensemble (AE) assigns the same weight to all models and provides a simple ensemble baseline. The proposed Multi-Metric Weighted Ensemble Method (MWEM) uses validation performance to select models and calculate their contributions. The proposed Multi-Metric Bayesian-Optimised Ensemble Method (BOEM) uses Bayesian optimisation to search for suitable combinations of model weights.

The BOEM framework is examined using three model-pool configurations as summarised in Table 5.3. BOEM-ML includes only the conventional machine learning models, while BOEM-DL includes only the tabular deep learning models. The hybrid BOEM configuration combines both groups within one hybrid ensemble. These configurations use the same weight optimisation procedure and differ only in the models included in the candidate pool.

Table 5.3: Model-pool configurations used in the BOEM framework.

Configuration	**Model pool**	**Purpose**
BOEM-ML	Conventional machine learning models	Examines ensemble weighting within the conventional machine learning model pool.
BOEM-DL	Tabular deep learning models	Examines ensemble weighting within the tabular deep learning model pool.
Hybrid BOEM	Conventional machine learning and tabular deep learning models	Examines the combined contribution of both model groups within a hybrid ensemble.

In all configurations, the framework trains the base models independently before combining their predictions. The ensemble method then controls how much each model contributes to the final output. This structure keeps the base-model training process separate from the ensemble weighting process and allows the same prediction models to support different ensemble configurations. The next section introduces AE as the baseline ensemble method. The following sections then present the proposed MWEM and BOEM methods.

5.5 Average Ensemble (AE)

The previous section described how the framework brings together predictions from models with different learning structures. Before applying the proposed weighting methods, a simple method is needed to provide a reference for ensemble comparison. For this purpose, the framework uses the Average Ensemble (AE), also known as uniform averaging. Given a candidate base model pool $L = \{l_1, l_2, \ldots, l_m\}$, each base model l_j produces a prediction $\hat{Y}_j$. AE assigns the same weight to all base models, so that $\omega_j^{\mathrm{AE}} = 1/m$, and the ensemble prediction is computed as:

$$\hat{Y}^{\mathrm{AE}} = \frac{1}{m}\sum_{j=1}^{m}\hat{Y}_j = \sum_{j=1}^{m}\omega_j^{\mathrm{AE}} \cdot \hat{Y}_j \tag{5.5.1}$$

where $\hat{Y}^{\mathrm{AE}}$ denotes the AE prediction and $\hat{Y}_j$ denotes the prediction produced by base model l_j.

This approach is easy to implement. However, it assumes that all base models are equally reliable, even when their validation performance differs (Mohammed and Kora, 2023). Previous studies have noted that simple averaging is not always the best combination strategy and that weighted ensembles can provide superior predictive performance (Shahhosseini et al., 2022). In addition, equal-weight averaging may ignore useful information about the relative accuracy of individual models and the correlation between their prediction errors (Wang et al., 2023b). This limitation motivates the proposed weighted ensemble methods, MWEM and BOEM, which assign different contributions to base models based on validation performance or optimised ensemble weights.

5.6 Multi-Metric Weighted Ensemble Method (MWEM)

The Average Ensemble assigns equal weights to all models, despite differences in their validation performance. To address this limitation, this research proposes two weighted heterogeneous ensemble methods: MWEM and BOEM. MWEM uses validation performance and greedy model selection to determine model weights, while BOEM uses Bayesian optimisation to find suitable weight combinations. These methods are described in the following sections.

The Multi-Metric Weighted Ensemble Method (MWEM) is the first proposed weighted ensemble method in this research. It extends validation-based weighting by using several measures of prediction performance when selecting and weighting the base models. The method follows the weighted ensemble framework introduced in the previous section but provides a direct procedure for constructing the final model set and weight vector.

The purpose of model weighting is to control how much each base model contributes to the final ensemble prediction. A larger weight gives a model more influence, while a smaller weight reduces its contribution. Effective weights are needed because the base models may have different levels of accuracy and may produce similar or complementary errors. Equal weighting may therefore give too much influence to weaker or redundant models.

Validation performance provides a suitable basis for calculating these weights because it evaluates each model using data that were not used to train that model. This provides a more realistic indication of how the model may perform on unseen data and allows the ensemble to be constructed without using the held-out test set. Weighting has been used in previous ensemble learning studies as a data-based alternative to fixed or manually selected weights (Al Ghamdi et al., 2020; Shahhosseini et al., 2022).

However, the result of model selection and weighting can depend on the evaluation metric. A model with a low MAE may not have the lowest RMSE or the highest R^2 or tolerance-based accuracy because these metrics represent different aspects of prediction performance. For example, MAE measures the average size of the prediction errors, RMSE gives greater importance to larger errors, R^2 score measures how well the

predictions represent variation in the target values, Huber loss reduces the influence of very large errors, and tolerance-based accuracy measures how often the prediction error remains within an accepted range. Using only one metric may therefore favour one form of model performance. MWEM uses a predefined sequence of evaluation metrics across the internal validation rounds so that several views of prediction performance contribute to the final ensemble. In other words, one metric is assigned to each validation round, and the resulting round-specific weight vectors are averaged after all rounds have been completed. The metric sequence is defined before model training and remains fixed for all candidate models and route sections.

Within each validation round, MWEM first ranks the models using the assigned metric. It then applies greedy model selection to retain models that improve the combined validation prediction. Finally, it calculates non-uniform weights for the retained models using their validation scores. Figure 5.2 provides a simplified overview of these stages, while Algorithm 1 summarises the complete procedure.

Let f denote the number of internal validation rounds, and let $L = \{l_1, l_2, \ldots, l_m\}$ represent the candidate set of m base models. In validation round i, the available training data D_{train} are divided into an internal training subset T_i^D and an internal validation subset V_i^D. Let Y_i^V denote the true target values in V_i^D, and let $\hat{Y}_{i,j}$ denote the prediction vector produced by model l_j.

A fresh instance of each candidate model is trained on T_i^D and used to generate predictions for V_i^D. The validation performance of model l_j is calculated using the metric E_i assigned to validation round i:

$$E_i(l_j) = E_i\left(Y_i^V, \hat{Y}_{i,j}\right). \tag{5.6.1}$$

Let $L_i^{\text{ranked}(E_i)}$ denote the ordered list of models in L, ranked from best to worst according to the metric E_i assigned to validation round i. For a lower-is-better metric, such as MAE, MSE, RMSE, or Huber loss, a smaller score produces a higher rank. For a higher-is-better metric, such as R^2 or tolerance-based accuracy, a larger score produces a higher rank.

After constructing $L_i^{\text{ranked}(E_i)}$, MWEM applies greedy model selection. The selected model set Φ_i is initially empty, and the models in $L_i^{\text{ranked}(E_i)}$ are considered one at a

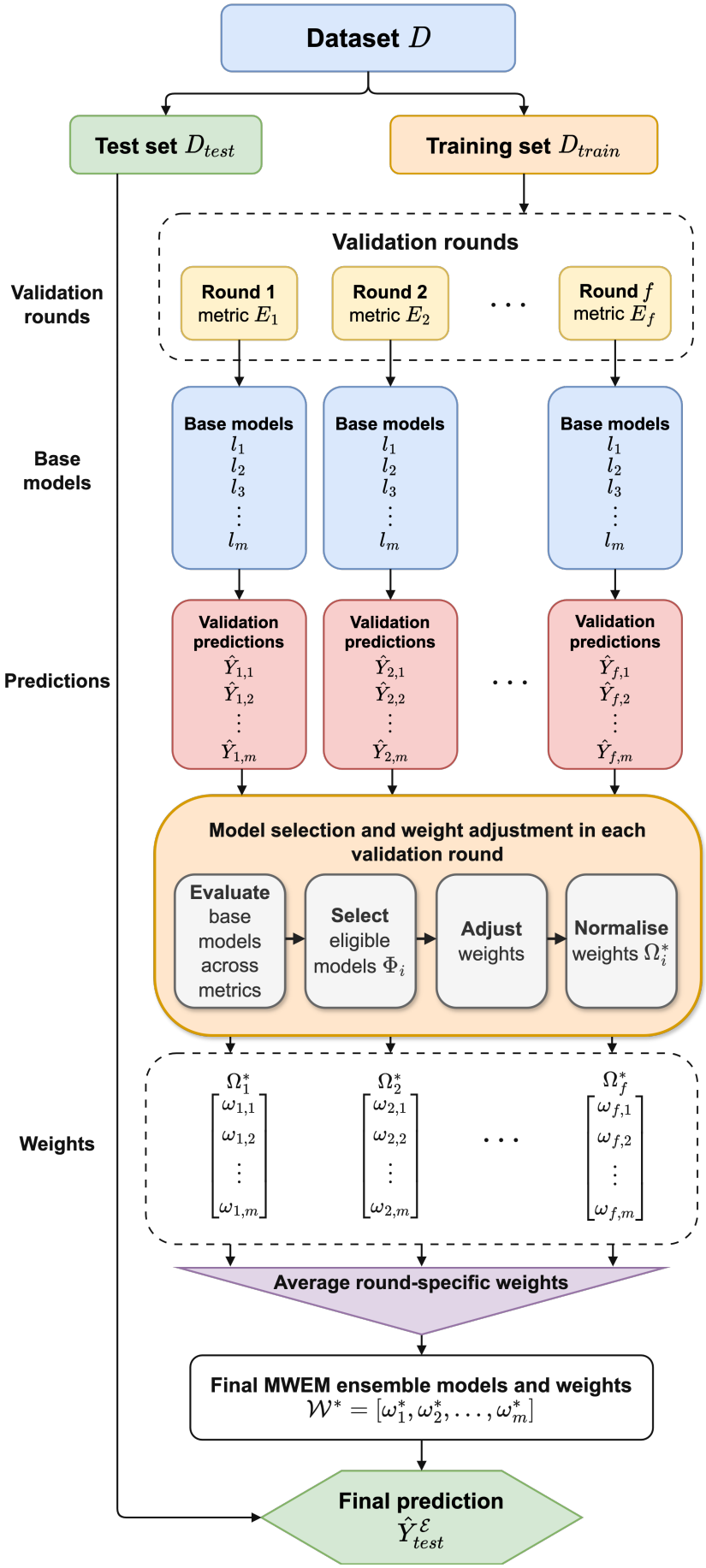


Figure 5.2: Simplified overview of the Multi-Metric Weighted Ensemble Method (MWEM).

time in their ranked order. When model l_j is considered, it is temporarily added to the current selected set to form:

$$\Psi_i = \Phi_i \cup \{l_j\}. \tag{5.6.2}$$

The temporary ensemble prediction is calculated by taking the simple average of the validation predictions produced by the models in Ψ_i:

$$\hat{Y}_i^{\Psi_i} = \frac{1}{|\Psi_i|} \sum_{l_k \in \Psi_i} \hat{Y}_{i,k}. \tag{5.6.3}$$

The temporary ensemble is evaluated using the same metric E_i that was used to rank the models:

$$\theta_i^{\text{new}} = E_i \left(Y_i^V, \hat{Y}_i^{\Psi_i} \right). \tag{5.6.4}$$

The value θ_i^{best} represents the best ensemble score obtained so far in validation round i using the currently retained models in Φ_i. The value θ_i^{new} represents the score of the temporary ensemble formed by adding the next ranked model to the current selected model set. The candidate model is retained only when θ_i^{new} improves θ_i^{best}. For a lower-is-better metric, improvement means that $\theta_i^{\text{new}} < \theta_i^{\text{best}}$. For a higher-is-better metric, improvement means that $\theta_i^{\text{new}} > \theta_i^{\text{best}}$. When an improvement occurs, the temporary model set becomes the new selected model set and θ_i^{best} is updated to θ_i^{new}. Otherwise, the candidate model is discarded. Since Φ_i is initially empty and θ_i^{best} is initialised to $+\infty$ for a lower-is-better metric or $-\infty$ for a higher-is-better metric, the first model in $L_i^{\text{ranked}(E_i)}$ is always retained and establishes the first valid ensemble score.

For example, suppose the current selected ensemble has an MAE of 1.2 minutes. If adding the next ranked model reduces the temporary ensemble MAE to 1.0 minutes, the model is retained. If adding another model increases the MAE to 1.3 minutes, that model is discarded. In this way, greedy selection checks whether each model improves the combined prediction rather than retaining models only because of their individual rankings.

The simple average in Equation 5.6.3 is used only during the greedy selection stage. After the selected model set Φ_i has been determined, MWEM calculates non-uniform

weights from the validation scores of the retained models. Models that are not retained receive a weight of zero for that validation round. For a higher-is-better metric, the non-negative performance score of retained model l_j is defined as:

$$\gamma_{i,j} = \max\left(E_i(l_j), 0\right), \qquad l_j \in \Phi_i. \tag{5.6.5}$$

This transformation is required because a higher-is-better metric such as R^2 can produce a negative value. Negative values are set to zero because negative ensemble weights would not have the same interpretation as the non-negative performance-based weights used in MWEM.

The round-specific weights for a higher-is-better metric are calculated as:

$$\omega_{i,j} = \frac{\gamma_{i,j}}{\sum_{l_k \in \Phi_i} \gamma_{i,k}}, \qquad l_j \in \Phi_i, \tag{5.6.6}$$

This assigns a larger weight to a retained model with a higher validation score. For example, suppose three retained models have non-negative performance scores of $[0.9, 0.8, 0.1]$. Their normalised weights are approximately $[0.50, 0.44, 0.06]$. The model with the highest performance score therefore receives the largest contribution to the ensemble.

For a lower-is-better metric, a smaller validation error should produce a larger weight. The validation error is therefore converted into an inverse performance value, and the round-specific weights are calculated as:

$$\omega_{i,j} = \frac{1/\left(E_i(l_j) + \epsilon\right)}{\sum_{l_k \in \Phi_i} 1/\left(E_i(l_k) + \epsilon\right)}, \qquad l_j \in \Phi_i, \tag{5.6.7}$$

where $\epsilon > 0$ is a fixed small constant. It prevents division by zero and limits very large inverse values when a validation error is close to zero. The value of ϵ was set to 10^{-6} in all experiments. Although no validation error reached zero in this research, this safeguard was considered so that the method remains applicable when a zero or near-zero validation error occurs.

For example, suppose three retained models have MAE values of $[1, 2, 4]$. Their inverse performance values are approximately $[1, 0.5, 0.25]$, producing normalised weights

of approximately $[0.57, 0.29, 0.14]$. The model with the lowest validation error therefore receives the largest weight.

A model that is not retained during validation round i receives a weight of zero:

$$\omega_{i,j} = 0, \qquad l_j \notin \Phi_i. \tag{5.6.8}$$

The complete round-specific weight vector contains one entry for every model in the original candidate set:

$$\mathcal{W}_i = [\omega_{i,1}, \omega_{i,2}, \ldots, \omega_{i,m}]. \tag{5.6.9}$$

The round-specific weights satisfy:

$$\omega_{i,j} \geq 0, \qquad \sum_{j=1}^{m} \omega_{i,j} = 1. \tag{5.6.10}$$

The ranking, greedy selection, and weight calculation are repeated independently for every validation round using the metric assigned to that round. After all validation rounds have been completed, the corresponding model weights are averaged:

$$\omega_j^* = \frac{1}{f} \sum_{i=1}^{f} \omega_{i,j}, \qquad j = 1, 2, \ldots, m. \tag{5.6.11}$$

where ω_j^* is the final weight assigned to model l_j after averaging its round-specific weights across the f validation rounds. The final MWEM weight vector is:

$$\mathcal{W}^* = [\omega_1^*, \omega_2^*, \ldots, \omega_m^*]. \tag{5.6.12}$$

Since every round-specific weight vector is non-negative and sums to one, averaging these vectors preserves the same properties:

$$\omega_j^* \geq 0, \qquad \sum_{j=1}^{m} \omega_j^* = 1. \tag{5.6.13}$$

For example, if two validation rounds produce the weight vectors $[0.6, 0.4, 0]$ and $[0.2, 0.5, 0.3]$, their average is $[0.4, 0.45, 0.15]$, which is also non-negative and sums to one.

Averaging reduces the dependence of the final ensemble on one validation round or

one evaluation metric. However, MWEM does not search all possible model subsets or weight combinations. Therefore, the greedy procedure does not guarantee a globally optimal ensemble. Instead, it provides a practical model set and final weight vector based on the validation evidence collected from the different validation round–metric pairs.

A model belongs to final ensemble when it receives a positive final weight after the round-specific weights have been averaged. Models with a final weight of zero do not contribute to the final ensemble prediction.

All model ranking, greedy selection, and weight calculation steps use only the internal training and validation subsets. The held-out test set is not used to rank models, select the retained model sets, or calculate the final weights. It is used only after the ensemble structure has been fixed. This separation prevents information from the test set from influencing the selected models or their weights. After MWEM determines the final weight vector $\mathcal{W}^*$, they are passed to the final model-training stage described in Section 5.10. Algorithm 1 summarises the complete MWEM procedure.

Algorithm 1: Multi-Metric Weighted Ensemble Method (MWEM)

Input: D_{train} (training data); $L = \{l_1, \ldots, l_m\}$ (candidate base models); f (number of internal validation rounds); $E = \{E_1, \ldots, E_f\}$ (validation-metric sequence); ϵ (small positive constant)

Output: $\mathcal{W}^* = [\omega_1^*, \ldots, \omega_m^*]$ (final MWEM weight vector)

`// Select and weight the models independently in each validation round`

for $i \leftarrow 1$ **to** f **do**

  Construct the internal training subset T_i^D and validation subset V_i^D from D_{train};

  `// Train the base models and calculate their validation scores`

  foreach $l_j \in L$ **do**

    Create a fresh instance of model l_j;

    Train model l_j on T_i^D;

    Generate and store validation predictions $\hat{Y}_{i,j}$ on V_i^D;

    Calculate $E_i(l_j) = E_i\left(Y_i^V, \hat{Y}_{i,j}\right)$;

  Construct the ordered model list $L_i^{\text{ranked}(E_i)}$ by ranking the models in L from best to worst according to E_i;

  `// Start greedy model selection`

  Initialise the selected model set Φ_i as empty;

  if E_i *is lower-is-better* **then**

    Initialise the best ensemble score as $\theta_i^{\text{best}} = +\infty$;

  else

    Initialise the best ensemble score as $\theta_i^{\text{best}} = -\infty$;

  foreach $l_j \in L_i^{\text{ranked}(E_i)}$ **do**

    Form the temporary model set $\Psi_i = \Phi_i \cup \{l_j\}$;

    Calculate the temporary ensemble prediction $\hat{Y}_i^{\Psi_i} = \frac{1}{|\Psi_i|} \sum_{l_k \in \Psi_i} \hat{Y}_{i,k}$;

    Calculate the temporary ensemble score $\theta_i^{\text{new}} = E_i\left(Y_i^V, \hat{Y}_i^{\Psi_i}\right)$;

    if θ_i^{new} *improves* θ_i^{best} **then**

      Retain model l_j by setting $\Phi_i = \Psi_i$;

      Update the best ensemble score as $\theta_i^{\text{best}} = \theta_i^{\text{new}}$;

    else

      Discard model l_j from the temporary model set;

  `// End greedy selection and calculate round-specific weights`

  if E_i *is higher-is-better* **then**

    For every retained model $l_j \in \Phi_i$, calculate $\gamma_{i,j} = \max(E_i(l_j), 0)$;

    For every retained model $l_j \in \Phi_i$, calculate $\omega_{i,j} = \gamma_{i,j} / \sum_{l_k \in \Phi_i} \gamma_{i,k}$;

  else

    For every retained model $l_j \in \Phi_i$, calculate $\omega_{i,j} = \frac{1/(E_i(l_j)+\epsilon)}{\sum_{l_k \in \Phi_i} 1/(E_i(l_k)+\epsilon)}$;

  For every unselected model $l_j \notin \Phi_i$, assign $\omega_{i,j} = 0$;

  Store the round-specific weight vector $\mathcal{W}_i = [\omega_{i,1}, \ldots, \omega_{i,m}]$;

`// Average the weights obtained from all validation rounds`

for $j \leftarrow 1$ **to** m **do**

  Calculate $\omega_j^* = \frac{1}{f} \sum_{i=1}^{f} \omega_{i,j}$;

Construct the final MWEM weight vector $\mathcal{W}^* = [\omega_1^*, \ldots, \omega_m^*]$;

Store $\mathcal{W}^*$ for final model training and prediction;

return $\mathcal{W}^*$;

5.7 Computational Complexity of MWEM

The main computational cost of MWEM comes from training the base models during the internal validation process. Let f be the number of validation rounds and m be the number of candidate base models. Since every model is trained once in each validation round, MWEM requires approximately $f \times m$ model-training operations.

The exact training cost depends on the models included in the candidate pool. For example, a linear model usually requires less training time than a boosting or deep learning model. Therefore, let T_{train} represent the average training cost of one base model. The main validation cost can be expressed as:

$$\mathcal{O}\left(fmT_{\text{train}}\right). \tag{5.7.1}$$

MWEM also ranks the models and applies the greedy selection procedure in each validation round. Model ranking has a small cost compared with model training. During greedy selection, the method tests whether adding each candidate model improves the current ensemble. If the ensemble predictions are recalculated from the beginning for every candidate model, the worst-case cost of this step is:

$$\mathcal{O}\left(fn_v m^2\right), \tag{5.7.2}$$

where n_v is the number of observations in each validation subset. This is a conservative upper bound. The cost can be lower when the implementation stores the current sum of the ensemble predictions and adds only the prediction of the new candidate model.

After the validation process, the selected models are trained again using the full training dataset. If z models receive positive final weights, this stage requires z additional model-training operations. The total computational cost of MWEM can therefore be summarised as:

$$\mathcal{O}\left(fmT_{\text{train}} + fn_v m^2 + zT_{\text{full}}\right), \tag{5.7.3}$$

where T_{full} represents the average cost of training one selected model on the full training dataset. In practice, the repeated training of the base models is expected to form the largest part of the total computational cost. Model ranking, weight calculation,

and the final weighted average require much less computation.

During prediction, MWEM does not repeat the validation or model-selection steps. The trained models and their weights are already fixed. Each active model produces a prediction, and the framework calculates their weighted average. Therefore, the additional cost introduced by the ensemble combination is small compared with the cost of generating the base-model predictions.

5.8 Multi-Metric Bayesian-Optimised Ensemble Method (BOEM)

The previous section introduced MWEM, where model weights are calculated directly from validation scores after greedy model selection which provides a simple data-based weighting method. However, the contribution of one model may also depend on the predictions produced by the other models. Two models with strong individual performance may produce similar errors, while another model with slightly weaker individual performance may provide useful complementary information. Therefore, individual validation scores may not always identify the most suitable joint combination of model weights.

A simple numerical example illustrates this limitation and also shows the difference between AE, MWEM, and BOEM. Suppose the true validation values for two observations are $[10, 10]$, and three models produce the predictions

$$\hat{Y}_1 = [7, 9], \qquad \hat{Y}_2 = [7.2, 8.8], \qquad \hat{Y}_3 = [5, 15].$$

The first and second models each have an MAE of 2.0, while the third model has a higher MAE of 5.0. The first two models are therefore individually stronger, but they produce very similar errors.

In AE, all three models receive equal weights. The resulting prediction is

$$\hat{Y}^{\mathrm{AE}} = \frac{[7, 9] + [7.2, 8.8] + [5, 15]}{3} = [6.4, 10.93],$$

which has an MAE of approximately 2.27. The weaker third model therefore receives too much influence under equal weighting, causing AE to perform worse than the strongest

individual model.

During MWEM's greedy selection, the first ranked model is retained and establishes an MAE of 2.0. Adding the second model produces the temporary prediction

$$\frac{[7, 9] + [7.2, 8.8]}{2} = [7.1, 8.9],$$

which also has an MAE of 2.0. Since the score does not improve, the second model is not retained. Adding the third model to the first produces

$$\frac{[7, 9] + [5, 15]}{2} = [6, 12],$$

which has an MAE of 3.0. The third model is therefore also discarded. MWEM consequently retains only the first model and produces an MAE of 2.0.

However, the third model contains useful complementary information because its higher prediction for the second observation helps correct the underprediction produced by the first two models. BOEM can preserve this contribution while limiting its negative effect by assigning a smaller weight. For example, using the weight vector $[0.45, 0.45, 0.10]$ gives

$$\hat{Y}^{\mathcal{E}} = (0.45[7, 9]) + (0.45[7.2, 8.8]) + (0.10[5, 15]) = [6.89, 9.51],$$

which has an MAE of 1.80. In this example, AE performs worst because it gives equal influence to all models, MWEM performs better by removing models that do not improve the equally weighted temporary ensemble, and BOEM performs best because it searches for a joint weight vector that retains a small but useful contribution from the weaker complementary model.

The purpose of model weighting is to control how much each base model contributes to the final ensemble prediction. A larger weight gives a model more influence, while a smaller weight reduces its contribution. Effective weights are needed because the base models may have different levels of accuracy and may produce similar or complementary errors. Equal weighting may therefore give too much influence to weaker or redundant models, while weights based only on individual model scores may not identify the best joint combination.

To address this issue, this research proposes the Multi-Metric Bayesian-Optimised Ensemble Method (BOEM). Instead of calculating each weight separately from an individual model score, BOEM searches for a complete weight vector that improves the combined ensemble prediction. In this way, the model weights are evaluated together rather than independently.

BOEM uses Bayesian optimisation because the model weights are continuous and many possible weight combinations exist. Testing every possible combination would be impractical, especially when the number of base models increases. Bayesian optimisation uses the results of previously evaluated weight vectors to select new and more promising candidates (Snoek et al., 2012; Shahriari et al., 2016). It therefore guides the search without requiring the full weight space to be tested.

BOEM follows the same internal validation structure and multi-metric strategy used by MWEM. In each validation round, all candidate base models are trained on the internal training subset and evaluated on the corresponding validation subset. One predefined evaluation metric is assigned to each round. The validation predictions are then stored and reused during weight optimisation. Therefore, the base models are not retrained for every candidate weight vector.

Figure 5.3 provides a simplified overview of the BOEM process. The procedures for generating candidate weights, normalising them, calculating the objective value, performing Bayesian optimisation, and averaging the round-specific weights are explained below and summarised in Algorithm 2.

In BOEM, a predefined evaluation metric is assigned to each internal validation round. Bayesian optimisation uses the assigned metric to identify the best ensemble weight vector for that round. After all validation rounds have been completed, the resulting round-specific weight vectors are averaged to obtain the final BOEM weight vector. In this way, several evaluation metrics contribute to the final ensemble construction, reducing dependence on one validation round or one performance measure.

For the BOEM formulation, the elements that must be specified are the decision variables, the search space, and the objective function. The decision variables are the candidate model weights. The search space defines the possible values of these weights, while the objective function measures how closely the resulting ensemble predictions

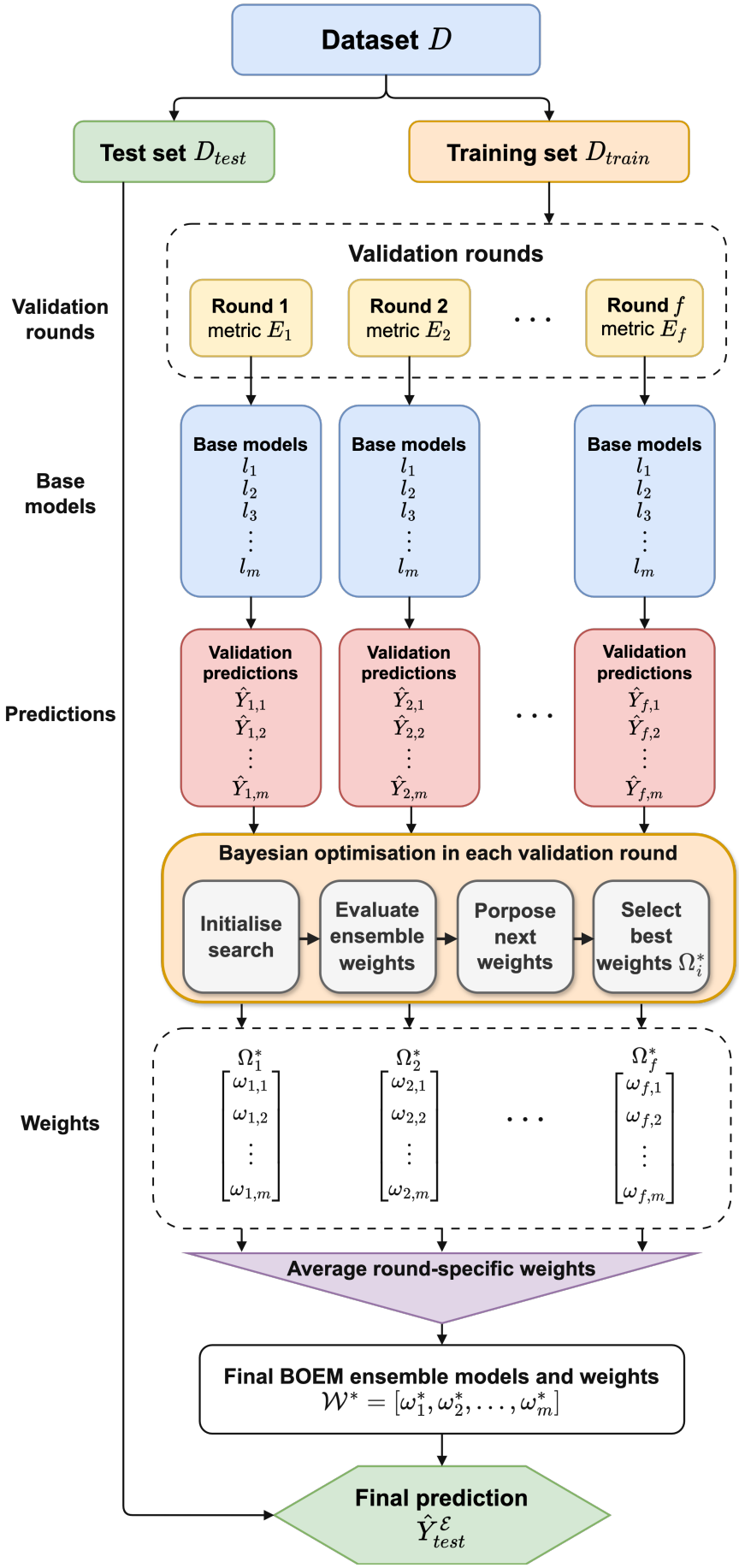


Figure 5.3: Simplified overview of the Multi-Metric Bayesian-Optimised Ensemble Method (BOEM).

match the true validation values.

Let f denote the number of internal validation rounds, and let $L = \{l_1, l_2, \ldots, l_m\}$ represent the candidate set of m base models. In validation round i, let Y_i^V denote the true target values in validation subset V_i^D, and let $\hat{Y}_{i,j}$ denote the prediction vector produced by model l_j.

A raw candidate weight vector proposed by Bayesian optimisation is defined as:

$$\Omega_i^{c,\text{raw}} = [\omega_{i,1}^{c,\text{raw}}, \omega_{i,2}^{c,\text{raw}}, \ldots, \omega_{i,m}^{c,\text{raw}}], \qquad \omega_{i,j}^{c,\text{raw}} \in [0,1], \tag{5.8.1}$$

where $\omega_{i,j}^{c,\text{raw}}$ is the raw candidate weight proposed for model l_j in validation round i. Before a candidate is evaluated, its raw weights are normalised as:

$$\omega_{i,j}^{c} = \frac{\omega_{i,j}^{c,\text{raw}}}{\sum_{k=1}^{m} \omega_{i,k}^{c,\text{raw}}}, \qquad j = 1, 2, \ldots, m. \tag{5.8.2}$$

The resulting normalised candidate weight vector is:

$$\Omega_i^c = [\omega_{i,1}^c, \omega_{i,2}^c, \ldots, \omega_{i,m}^c]. \tag{5.8.3}$$

The normalised candidate weights satisfy:

$$\omega_{i,j}^c \geq 0, \qquad \sum_{j=1}^{m} \omega_{i,j}^c = 1. \tag{5.8.4}$$

Normalisation ensures that the weights represent relative model contributions and that the ensemble prediction is not affected by the overall magnitude of the raw candidate vector. For example, suppose three models produce the prediction vector $[4, 6, 8]$ for an observation with a true value of 6. The raw vectors $[0.2, 0.3, 0.5]$ and $[0.4, 0.6, 1.0]$ represent the same relative contributions, but without normalisation they produce ensemble predictions of 6.6 and 13.2, respectively. After normalisation, both become $[0.2, 0.3, 0.5]$ and produce the same ensemble prediction of 6.6.

Using the normalised candidate weights, BOEM calculates the validation ensemble prediction as:

$$\hat{Y}_i^{\mathcal{E}}(\Omega_i^c) = \sum_{j=1}^{m} \omega_{i,j}^c \hat{Y}_{i,j}, \tag{5.8.5}$$

where $\hat{Y}_i^{\mathcal{E}}(\Omega_i^c)$ denotes the ensemble prediction vector for validation round i using candidate weight vector Ω_i^c.

For example, suppose three models predict delays of $[4, 6, 8]$ minutes for an observation with a true delay of 6 minutes. The candidate weights $[0.2, 0.3, 0.5]$ produce an ensemble prediction of $(0.2 \times 4) + (0.3 \times 6) + (0.5 \times 8) = 6.6$ minutes, giving an absolute error of 0.6 minutes. Using the result of this evaluation, Bayesian optimisation may examine another candidate, such as $[0.3, 0.4, 0.3]$. This candidate produces $(0.3 \times 4) + (0.4 \times 6) + (0.3 \times 8) = 6.0$ minutes and gives zero error for this illustrative observation. For this illustrative observation, the second candidate gives the lower error. In the actual optimisation, however, candidate weight vectors are compared using the assigned metric calculated across all observations in the validation subset, rather than using the error of one observation.

Bayesian optimisation is implemented as a minimisation procedure. For a lower-is-better metric, such as MAE, MSE, or RMSE, the objective function is the metric value itself:

$$\mathcal{L}_i(\Omega_i^c) = E_i\left(Y_i^V, \hat{Y}_i^{\mathcal{E}}(\Omega_i^c)\right). \tag{5.8.6}$$

where E_i is the evaluation metric assigned to validation round i. For a higher-is-better metric, such as R^2 or tolerance-based accuracy, BOEM minimises the negative metric value:

$$\mathcal{L}_i(\Omega_i^c) = -E_i\left(Y_i^V, \hat{Y}_i^{\mathcal{E}}(\Omega_i^c)\right). \tag{5.8.7}$$

Using the negative value for a higher-is-better metric converts it into the same minimisation form. Therefore, a smaller value of $\mathcal{L}_i(\Omega_i^c)$ always represents a better candidate weight vector, regardless of whether the assigned metric is lower-is-better or higher-is-better.

The optimiser uses the objective values from previously evaluated candidates to select further promising raw weight vectors. This process continues until q candidate vectors have been evaluated in validation round i. Here, q represents the optimisation budget and controls the number of candidate weight vectors that can be tested.

Let $\mathcal{C}_i^{(q)}$ denote the set of the q normalised candidate weight vectors evaluated during validation round i. The objective value associated with each candidate is also stored. The best candidate found within the optimisation budget is defined as:

$$\Omega_i^{c,\text{best}} = \underset{\Omega_i^c \in \mathcal{C}_i^{(q)}}{\arg\min}\, \mathcal{L}_i(\Omega_i^c). \tag{5.8.8}$$

The operator arg min returns the argument that gives the smallest objective value. Since the arguments in Equation 5.8.8 are candidate weight vectors, $\Omega_i^{c,\text{best}}$ is the weight vector that produces the lowest objective value among the q evaluated candidates. It is the best solution found within the available optimisation budget and is not assumed to be the global optimum of the full continuous search space. The round-specific best weight vector is written as:

$$\Omega_i^{c,\text{best}} = [\omega_{i,1}^{c,\text{best}}, \omega_{i,2}^{c,\text{best}}, \ldots, \omega_{i,m}^{c,\text{best}}], \tag{5.8.9}$$

where $\omega_{i,j}^{c,\text{best}}$ is the weight assigned to model l_j by the best candidate found in validation round i.

The optimisation is repeated independently for every validation round. Each round therefore produces the best normalised candidate weight vector found within its optimisation budget for its validation data and assigned metric. After all validation rounds have been completed, the corresponding model weights are averaged across the rounds:

$$\omega_j^* = \frac{1}{f} \sum_{i=1}^{f} \omega_{i,j}^{c,\text{best}}, \qquad j = 1, 2, \ldots, m. \tag{5.8.10}$$

where ω_j^* is the final weight assigned to model l_j after averaging its best normalised weights across the f validation rounds. The final BOEM weight vector is:

$$\mathcal{W}^* = [\omega_1^*, \omega_2^*, \ldots, \omega_m^*]. \tag{5.8.11}$$

Since every round-specific weight vector is normalised, all of its components are non-negative and sum to one. Averaging these vectors preserves the same properties. For example, if two validation rounds produce the weight vectors $[0.2, 0.3, 0.5]$ and $[0.4, 0.4, 0.2]$, their average is $[0.3, 0.35, 0.35]$, which is also non-negative and sums to

one. Averaging reduces the dependence of the final weights on a single validation round or evaluation metric and the final vector summarises the best weight vectors obtained from the different validation round–metric pairs.

Unlike MWEM, BOEM does not use MWEM's greedy forward-selection procedure to add models one at a time. Within the candidate model set supplied to BOEM, Bayesian optimisation controls the contribution of each model through its weight. A model with a large weight has a stronger influence on the ensemble prediction, whereas a model with a small weight has less influence. The same BOEM procedure is applied to the three candidate model pools introduced earlier. BOEM-ML uses only the conventional machine learning models, BOEM-DL uses only the tabular deep learning models, and the hybrid BOEM uses both groups together. The optimisation steps remain unchanged across these configurations, while the number and type of candidate models differ.

The validation predictions used during weight optimisation are generated by models trained only on the corresponding internal training subsets. For validation round i, the models are trained on T_i^D and evaluated on the separate validation subset V_i^D. The resulting validation predictions are used to compare candidate weight vectors. The held-out test set is not used to generate the weights, evaluate candidate vectors, or select the final ensemble. It is used only after the ensemble construction process has been completed. This separation prevents information from the test set from influencing the learned ensemble weights. Algorithm 2 summarises the complete BOEM procedure.

Algorithm 2: Multi-Metric Bayesian-Optimised Ensemble Method (BOEM)

Input: D_{train} (training data); $L = \{l_1, \ldots, l_m\}$ (candidate base models); f (number of internal validation rounds); $E = \{E_1, \ldots, E_f\}$ (validation-metric sequence); q (candidate-evaluation budget per validation round)

Output: $\mathcal{W}^* = [\omega_1^*, \ldots, \omega_m^*]$ (final BOEM weight vector)

`// Optimise one weight vector independently in each validation round`

for $i \leftarrow 1$ **to** f **do**

  Construct the internal training subset T_i^D and validation subset V_i^D from D_{train};

  `// Train the base models and store their validation predictions`

  foreach $l_j \in L$ **do**

    Create a fresh instance of model l_j;

    Train model l_j on T_i^D;

    Generate and store validation predictions $\hat{Y}_{i,j}$ on V_i^D;

  `// Start Bayesian optimisation for validation round` i

  Define the raw candidate search space as $\omega_{i,j}^{c,\text{raw}} \in [0, 1]$;

  Initialise a raw candidate weight vector $\Omega_i^{c,\text{raw}} = [\omega_{i,1}^{c,\text{raw}}, \ldots, \omega_{i,m}^{c,\text{raw}}]$;

  Initialise the evaluated candidate set $\mathcal{C}_i^{(q)}$ as empty;

  `// The stored model predictions are reused during optimisation`

  for q *candidate evaluations* **do**

    `// Normalise the current raw candidate weights`

    For every model l_j, calculate $\omega_{i,j}^c = \omega_{i,j}^{c,\text{raw}} / \sum_{k=1}^m \omega_{i,k}^{c,\text{raw}}$;

    Form the normalised candidate vector $\Omega_i^c = [\omega_{i,1}^c, \ldots, \omega_{i,m}^c]$;

    `// Calculate the weighted ensemble prediction`

    Calculate $\hat{Y}_i^{\mathcal{E}}(\Omega_i^c) = \sum_{j=1}^m \omega_{i,j}^c \hat{Y}_{i,j}$;

    `// Evaluate the candidate using the metric assigned to round` i

    if E_i *is lower-is-better* **then**

      Calculate $\mathcal{L}_i(\Omega_i^c) = E_i\Big(Y_i^V, \hat{Y}_i^{\mathcal{E}}(\Omega_i^c)\Big)$;

    else

      Calculate $\mathcal{L}_i(\Omega_i^c) = -E_i\Big(Y_i^V, \hat{Y}_i^{\mathcal{E}}(\Omega_i^c)\Big)$;

    Add Ω_i^c to $\mathcal{C}_i^{(q)}$ and store its associated objective value $\mathcal{L}_i(\Omega_i^c)$;

    Update the Bayesian optimisation model using the weight vector proposed in the search space as the input and $\mathcal{L}_i(\Omega_i^c)$ as the observed objective value;

    Use the updated Bayesian optimisation model to propose the next raw candidate weight vector $\Omega_i^{c,\text{raw}}$;

  `// End Bayesian optimisation after` q `candidate evaluations`

  Select the candidate weight vector with the lowest objective value as $\Omega_i^{c,\text{best}} = \arg\min_{\Omega_i^c \in \mathcal{C}_i^{(q)}} \mathcal{L}_i(\Omega_i^c)$;

  `//` $\arg\min$ `returns the candidate vector with the lowest objective value`

  Store the round-specific best vector as $\Omega_i^{c,\text{best}} = [\omega_{i,1}^{c,\text{best}}, \ldots, \omega_{i,m}^{c,\text{best}}]$;

for $j \leftarrow 1$ **to** m **do**

  Calculate $\omega_j^* = \frac{1}{f} \sum_{i=1}^f \omega_{i,j}^{c,\text{best}}$;

Construct the final BOEM weight vector $\mathcal{W}^* = [\omega_1^*, \ldots, \omega_m^*]$;

Store $\mathcal{W}^*$ for final model training and prediction;

return $\mathcal{W}^*$;

5.9 Computational Considerations

The computational cost of BOEM comes from two main stages. The first stage is the internal validation process. As in MWEM, every candidate base model is trained in each validation round. If the framework uses f validation rounds and m base models, it performs approximately $f \times m$ model-training operations.

The second stage is Bayesian weight optimisation. If the optimiser evaluates q candidate weight vectors in each validation round, it performs approximately $f \times q$ objective evaluations. These evaluations use the stored validation predictions. Therefore, the base models are not retrained each time a new candidate weight vector is tested. Each evaluation only normalises the candidate weights, combines the stored predictions, and calculates the selected validation metric.

The optimisation cost increases when the number of candidate models, the number of validation observations, or the optimisation budget increases. However, all weight optimisation is completed offline. During prediction, BOEM only applies the trained base models and combines their outputs using the stored final weights. The weighted combination itself requires little additional computation. The total prediction time mainly depends on the number and type of models that receive useful weights.

Bayesian optimisation was implemented in Python using `gp_minimize` from `scikit-optimize`. A Gaussian Process surrogate model and the Expected Improvement acquisition function were used. The search range for each raw candidate weight $\omega_{i,j}^{c,\text{raw}}$ was set to $[0, 1]$, and each raw candidate weight vector was normalised before the ensemble prediction was calculated. A fixed optimisation budget was used in each validation round to keep the optimisation process consistent across the experiments, and no additional early-stopping rule was applied. A fixed random seed was used, and the same settings were applied across all sections and validation rounds.

5.10 Final Model Training and Ensemble Prediction

The previous sections described how AE, MWEM, and BOEM determine the contribution of the base models. After the ensemble structure and model weights have been found, the framework trains the final models and uses them to generate

predictions for unseen data. This stage is performed separately for each route section.

The models trained during the internal validation rounds are used only for model selection and weight estimation. They are not used as the final prediction models because each of them was trained on only part of the available training data. After the validation process is complete, the framework creates fresh instances of the required base models and trains them on the full training dataset for the corresponding route section.

The final model set depends on the ensemble method. AE uses all models in its defined model pool and assigns an equal weight to each model. MWEM includes only models with non-zero final weights in the ensemble prediction. BOEM uses the candidate models and the final weights obtained through Bayesian optimisation. Let

$$\mathcal{A} = [l_1, l_2, \ldots, l_z] \tag{5.10.1}$$

denote the list of z models used by the final ensemble, and let

$$\mathcal{W} = [\omega_1, \omega_2, \ldots, \omega_z] \tag{5.10.2}$$

denote the corresponding weight vector. For a new input observation x, each trained model l_j generates an individual prediction:

$$\hat{y}_j = l_j(x). \tag{5.10.3}$$

The final ensemble prediction is calculated as:

$$\hat{y}^{\mathcal{E}} = \sum_{j=1}^{z} \omega_j \hat{y}_j, \tag{5.10.4}$$

where $\hat{y}^{\mathcal{E}}$ is the predicted arrival delay for the next station, $\hat{y}_j$ is the prediction produced by base model l_j, and ω_j is its final ensemble weight. Lowercase notation, such as $\hat{y}_j$ and $\hat{y}^E$, represents a prediction for one observation, whereas uppercase notation represents a prediction vector for multiple observations.

The preprocessing steps fitted during model development are also stored with the trained models. These may include feature ordering, data scaling, and other model-specific transformations. The same fitted preprocessing steps are applied to the

test and real-time input data. They are not fitted again using unseen observations.

The held-out test set is used only after the final models and ensemble weights have been fixed. Each final base model generates predictions for the test observations, and the framework combines these predictions using Equation 5.10.4. The final ensemble and individual model predictions are then evaluated using the metrics defined in Chapter 3. This process keeps the test data separate from model selection and weight estimation.

Since the framework develops models at the route-section level, each section has its own trained base models and ensemble weights. For example, the ensemble used to predict the delay for one section may contain different weights from the ensemble used for another section. This allows the final predictors to reflect the delay behaviour observed in each part of the route.

For each route section, the framework stores the final trained models, the ensemble method, the learned weight vector, and the required preprocessing information. These stored components can then be loaded without repeating the model-selection or weight-estimation process. Chapter 6 explains how the stored predictors are integrated into the real-time train arrival delay prediction framework. Algorithm 3 summarises the final model training and prediction process.

Algorithm 3: Final Ensemble Training and Prediction

Input: D_{train} (full training data for one route section);
$\mathcal{A}$ (final model set);
$\mathcal{W}$ (final ensemble weights);
X_{test} (unseen input data)
Output: $\hat{Y}^{\mathcal{E}}$ (final ensemble predictions)

foreach $l_j \in \mathcal{A}$ **do**
    Create a fresh instance of l_j;
    Fit the preprocessing pipeline required by l_j using D_{train};
    Train l_j using the processed training data;
    Store the fitted preprocessing pipeline and model;
foreach $l_j \in \mathcal{A}$ **do**
    Apply the stored preprocessing pipeline of l_j to X_{test};
    Generate predictions $\hat{Y}_j = l_j(X_{\text{test}})$;
Calculate the final ensemble predictions:

$$\hat{Y}^{\mathcal{E}} = \sum_{l_j \in \mathcal{A}} \omega_j \hat{Y}_j$$

return $\hat{Y}^{\mathcal{E}}$;

5.11 Prediction Results Using Historical Train Movement Data

This section presents the prediction results obtained using the historical train movement data. The aim is to evaluate the performance of the proposed ensemble framework across the two selected UK railway routes. Unless a specific route or section is mentioned, the reported values are averages over both routes and all sections.

5.11.1 Overall Model Performance

Table 5.4 summarises the mean test performance of all models across the 26 route sections. Among the conventional ML baselines, GB, XGB, RF, MLP, LR, and BR produced comparatively good results, although no individual model achieved the best baseline performance for every metric. Weaker overall results were obtained by models such as ENet and KNN. These findings indicate that different model structures capture different aspects of railway delay behaviour and support the use of ensemble learning. Among the tabular DL models, TabPFN and TabICL achieved the highest ACC@1 values, while TabICL also produced the lowest MAE within this model group. TabM achieved a comparatively high R^2, whereas FTT and TabNet produced weaker overall mean results. Therefore, the tabular DL models were competitive with the conventional ML models, but their relative performance differed across the evaluation metrics.

The first proposed ensemble method, MWEM, improved over all conventional ML baselines. Compared with GB, which was one of the strongest individual ML baseline by MAE and RMSE, MWEM reduced MAE from 0.650 to 0.637 and RMSE from 1.213 to 1.119, while also increasing R^2 from 0.890 to 0.903 and ACC@1 from 91.747% to 92.094%. This confirms the finding of the earlier conference study that multi-metric weighted aggregation provides a systematic advantage over individual predictors (Golchehreh et al., 2026).

Building on this first proposed ensemble, the BOEM variants further improved predictive performance. BOEM-ML achieved strong results among the conventional ML-based ensembles, with an R^2 of 0.910, an MAE of 0.604, an RMSE of 1.073, and an ACC@1 of 92.541%. BOEM-DL also achieved competitive results, with an R^2 of 0.909, MAE of 0.612, RMSE of 1.089, and ACC@1 of 92.668%. These results show

Table 5.4: Mean test performance of all models across all sections.

Model	R^2	MAE	RMSE	ACC@1 min
LR	0.895	0.696	1.163	90.991
ENet	0.735	1.164	2.137	81.224
BR	0.895	0.695	1.163	90.999
SVR	0.826	0.761	1.728	89.697
KNN	0.665	1.361	2.438	74.541
MLP	0.896	0.691	1.158	90.722
DT	0.789	0.843	1.689	86.208
RF	0.885	0.664	1.258	91.445
AB	0.860	0.864	1.444	87.645
ET	0.865	0.719	1.342	89.150
GB	0.890	0.650	1.213	91.747
XGB	0.888	0.656	1.230	91.746
TabPFN	0.879	0.661	1.255	92.308
TabM	0.890	0.706	1.192	90.651
FTT	0.792	0.739	1.839	90.415
TabNet	0.827	0.778	1.489	88.566
TabICL	0.877	0.648	1.343	92.308
AE	0.895	0.665	1.208	91.696
MWEM	0.903	0.637	1.119	92.094
BOEM-ML	0.910	0.604	1.073	92.541
BOEM-DL	0.909	0.612	1.089	92.668
BOEM	**0.914**	**0.594**	**1.055**	**92.819**

that multi-metric Bayesian ensemble-weight optimisation was effective for both the conventional ML and tabular DL model pools.

The final hybrid BOEM achieved the best overall performance across all reported metrics, with an R^2 of 0.914, MAE of 0.594, RMSE of 1.055, and ACC@1 of 92.819%. Compared with MWEM, BOEM reduced MAE by approximately 7% and RMSE by approximately 6%, showing the additional benefit of multi-metric Bayesian optimisation over the earlier metric-weighted greedy strategy. Compared with the strongest individual baseline models, BOEM also achieved consistent improvements. In particular, BOEM reduced MAE by approximately 9% relative to GB, 10% relative to XGB, 10% relative to TabPFN, and 8% relative to TabICL. Similarly, RMSE was reduced by approximately 9% compared with MLP, 13% compared with GB, 14% compared with XGB, 16% compared with TabPFN, and 21% compared with TabICL. These results suggest that the research progression from MWEM to BOEM leads to increasingly robust ensemble weights and consistently stronger performance than individual models or simpler ensemble alternatives.

5.11.2 Multi-Metric BOEM Weighting Strategy

To evaluate the proposed multi-metric BOEM strategy, the BOEM-ML configuration was compared with single-metric BOEM variants. In the proposed version, different evaluation metrics are assigned to the validation rounds, and the optimised round-level weight vectors are averaged to obtain the final weights. In each single-metric variant, the same metric is used across all validation rounds. This comparison examines whether using several metrics across the validation process provides more balanced ensemble weights than relying on one metric. This controlled setting isolates the effect of the optimisation objective itself and avoids additional variability introduced by heterogeneous DL architectures. The compared variants were optimised using individual validation objectives, including R^2, MAE, RMSE, ACC@1, and Huber, while the proposed multi-metric version combines different validation metrics when estimating the ensemble weights.

Figure 5.4 shows the results across both routes. The multi-metric variant achieved the best overall result in all four evaluation views. It obtained the highest ACC@1, the lowest MAE, the highest R^2, and the lowest RMSE. This is important because the improvement is not simply caused by optimising one selected metric. Instead, the results indicate that combining complementary metrics leads to more balanced performance across the evaluation measures. The proposed strategy therefore provides a better compromise between error magnitude, explained variance, and minute-level arrival accuracy.

5.11.3 Section-Level Performance

The section-level results provide a more detailed view of model behaviour across the two routes. Figure 5.5 presents the results for the NL route. This route contains seven sections, and the results show that BOEM consistently outperformed or matched the strongest ML and tabular DL models across the NL sections. BOEM provided a clearer advantage in the more unstable sections where weaker models experienced larger increases in MAE and RMSE and lower R^2 values. In contrast, BOEM preserved a more stable error profile.

Figure 5.6 shows the corresponding results for the WW route. This route contains

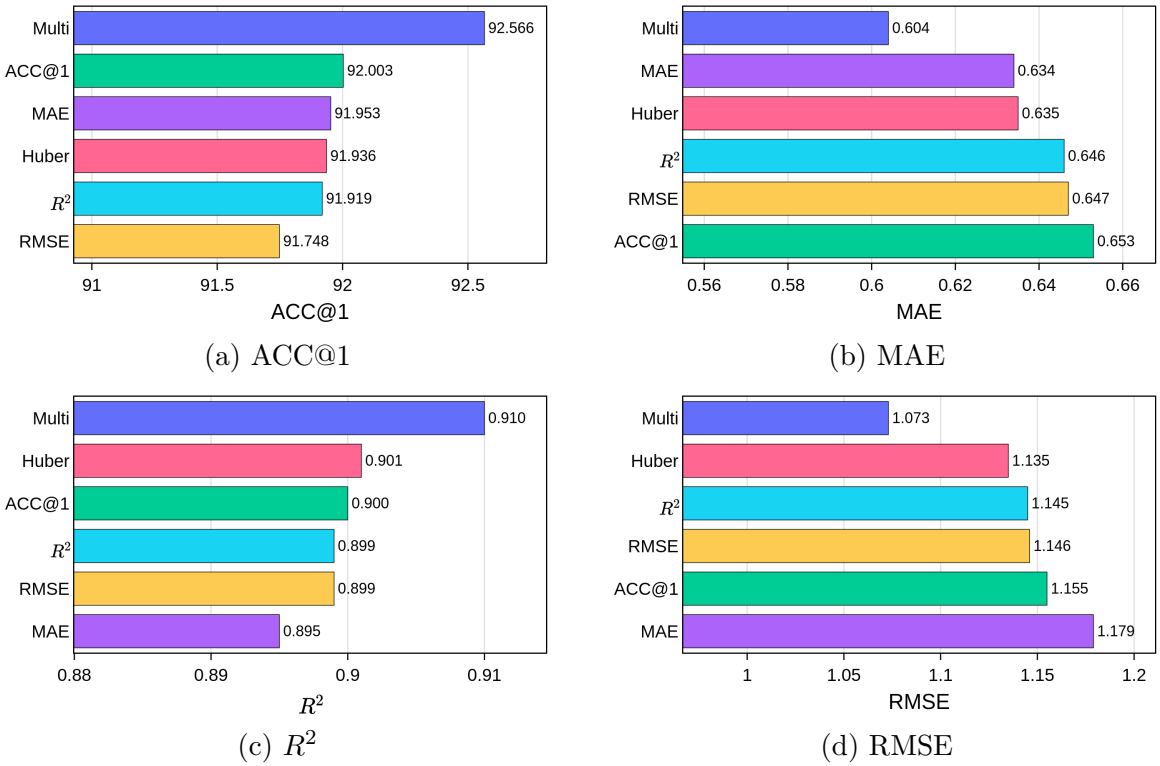
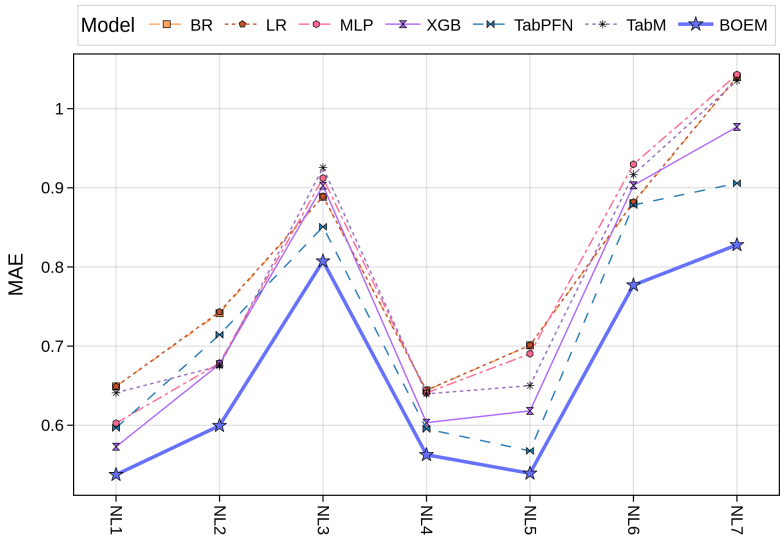


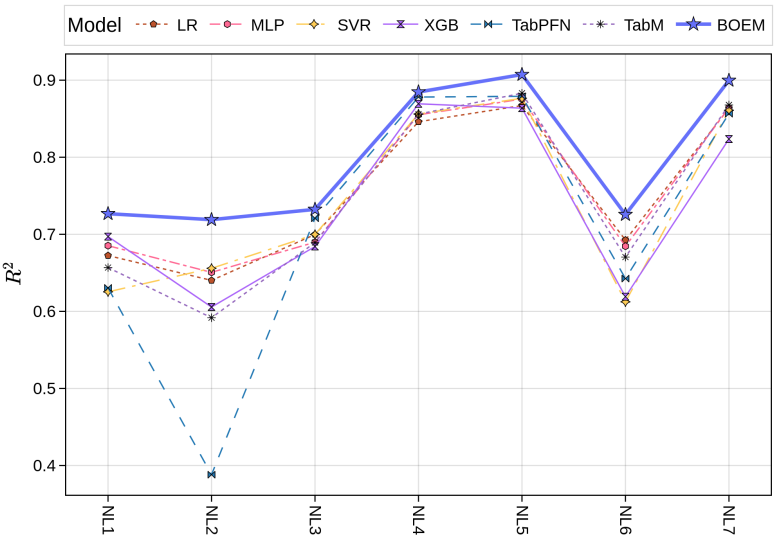
Figure 5.4: Performance of the BOEM-ML variants using single-metric validation and the proposed multi-metric validation strategy across both routes.

nineteen sections and is more heterogeneous than the NL route. The results show larger changes in performance across sections. In the earlier WW sections, most strong models perform well, but the differences become clearer in later sections where delay propagation and operational variability become more difficult to model. The results in some sections are particularly challenging. In these sections, several individual models show large error increases or sharp reductions in R^2. BOEM provides the strongest overall performance and shows less variation across the difficult sections.

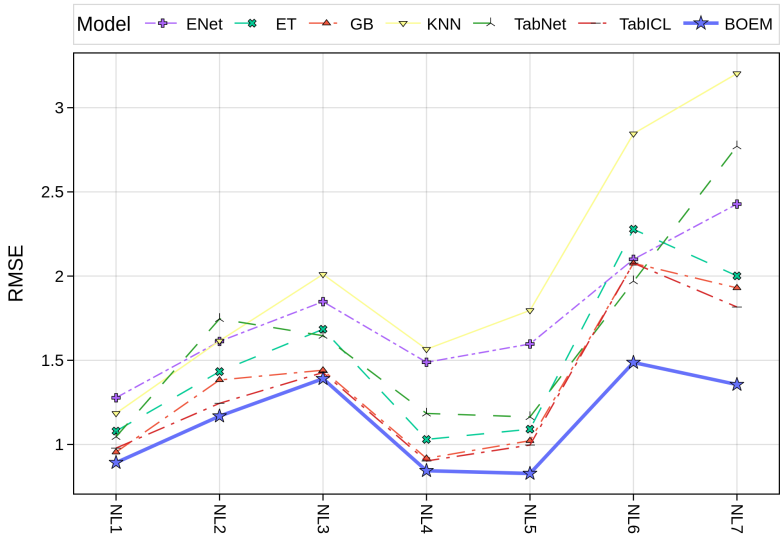
Some sections show weaker performance because some parts of the network are more operationally complex. They may include high-density stations, junctions where several services meet, or segments where delays from earlier stations build up. According to the information provided by (Network Rail, 2016, pp. 100–101) (Network Rail, 2015, pp. 65–66), some of the more difficult sections such as NL3, NL7, WW18, and WW19 are located near busy junctions or mixed-traffic corridors. These areas carry dense passenger services and, in some cases, freight traffic. However, these operational characteristics were not measured directly in this experiment and should therefore be considered possible explanations for the observed section-level differences. These



(a) MAE

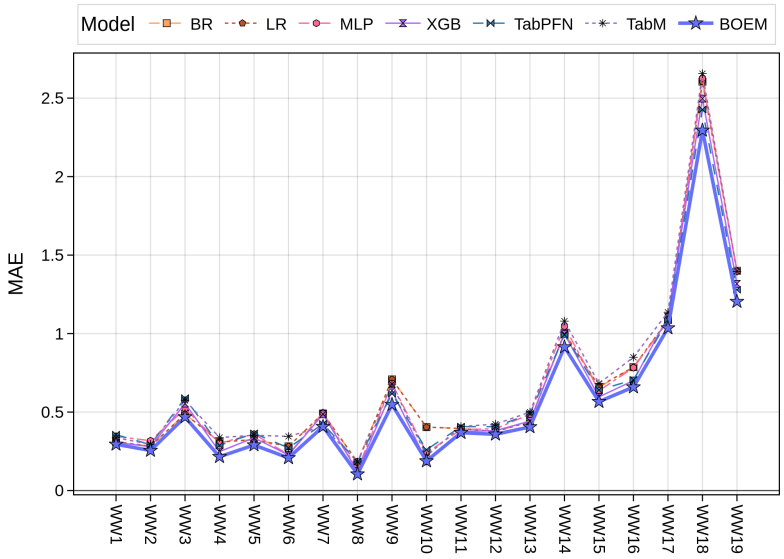


(b) R^2

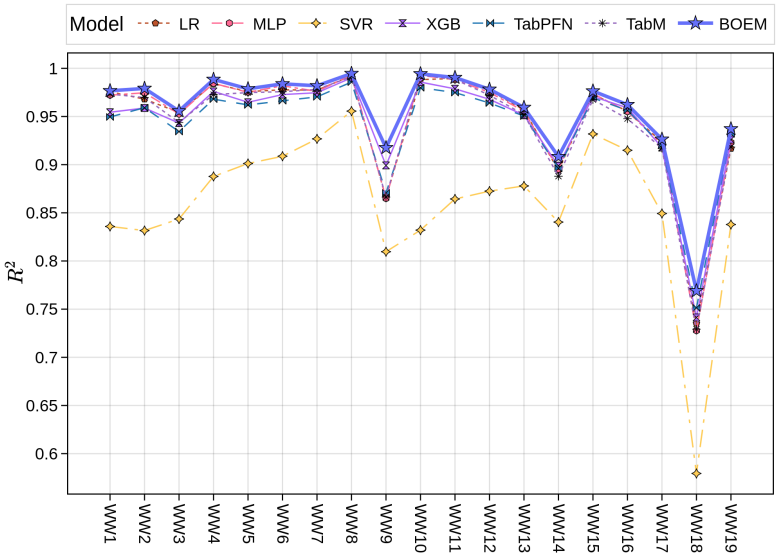


(c) RMSE

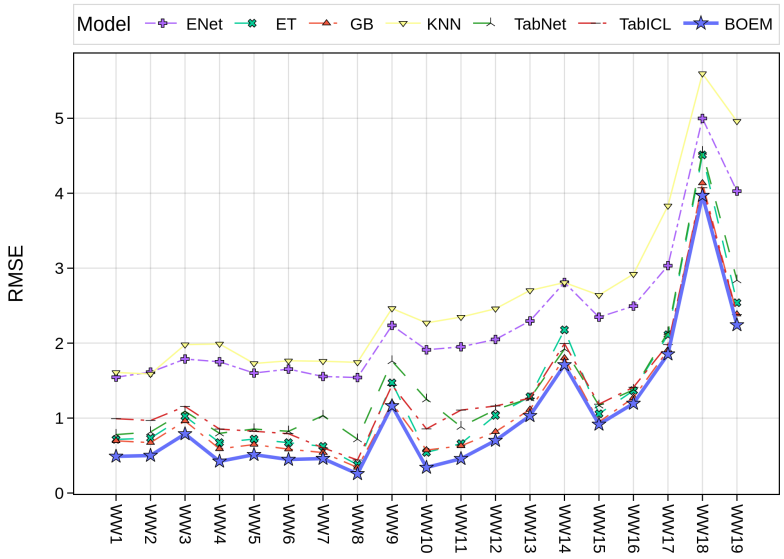
Figure 5.5: Section-level prediction performance on the NL route.



(a) MAE



(b) R^2



(c) RMSE

Figure 5.6: Section-level prediction performance on the WW route.

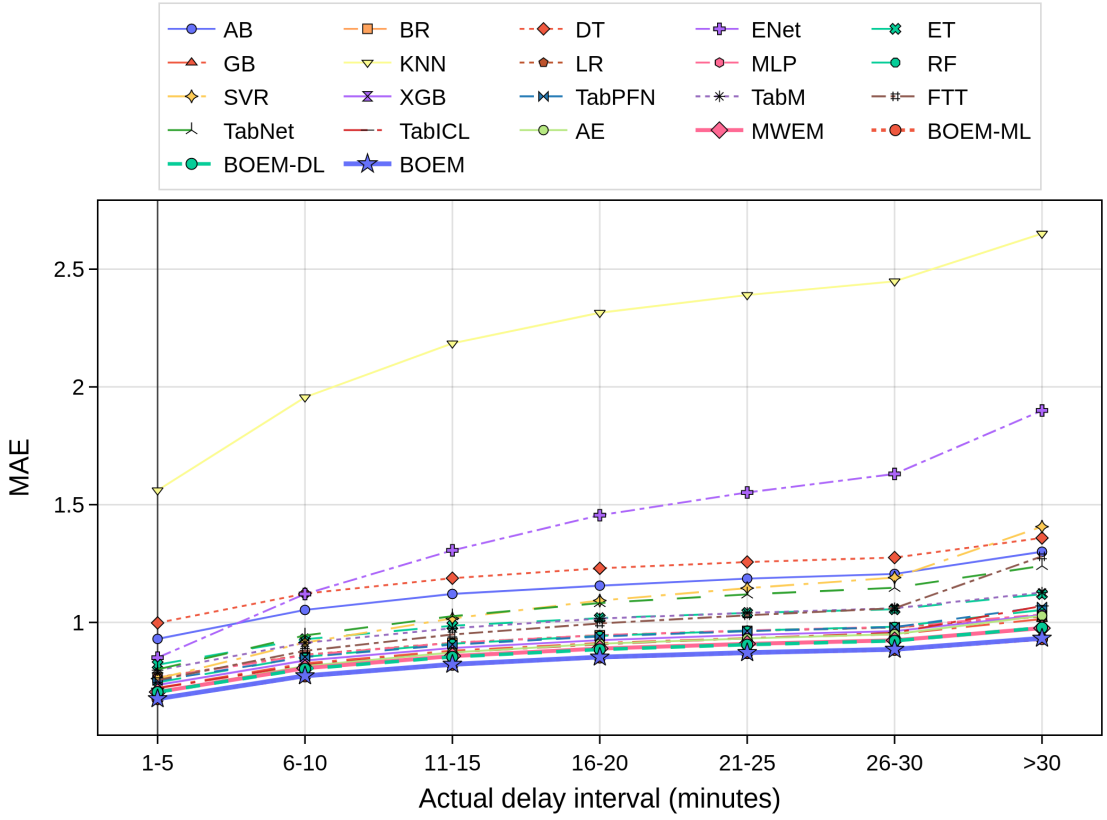
conditions create irregular and highly variable delay patterns, which are harder for individual models to learn accurately. This may contribute to the larger performance variation shown by models such as KNN, ENet, DT, and TabNet across sections. The ensemble methods, especially BOEM, show more consistent performance because the weighted combination reduces reliance on any single model under different section-level conditions.

5.11.4 Performance Across Delay Magnitudes

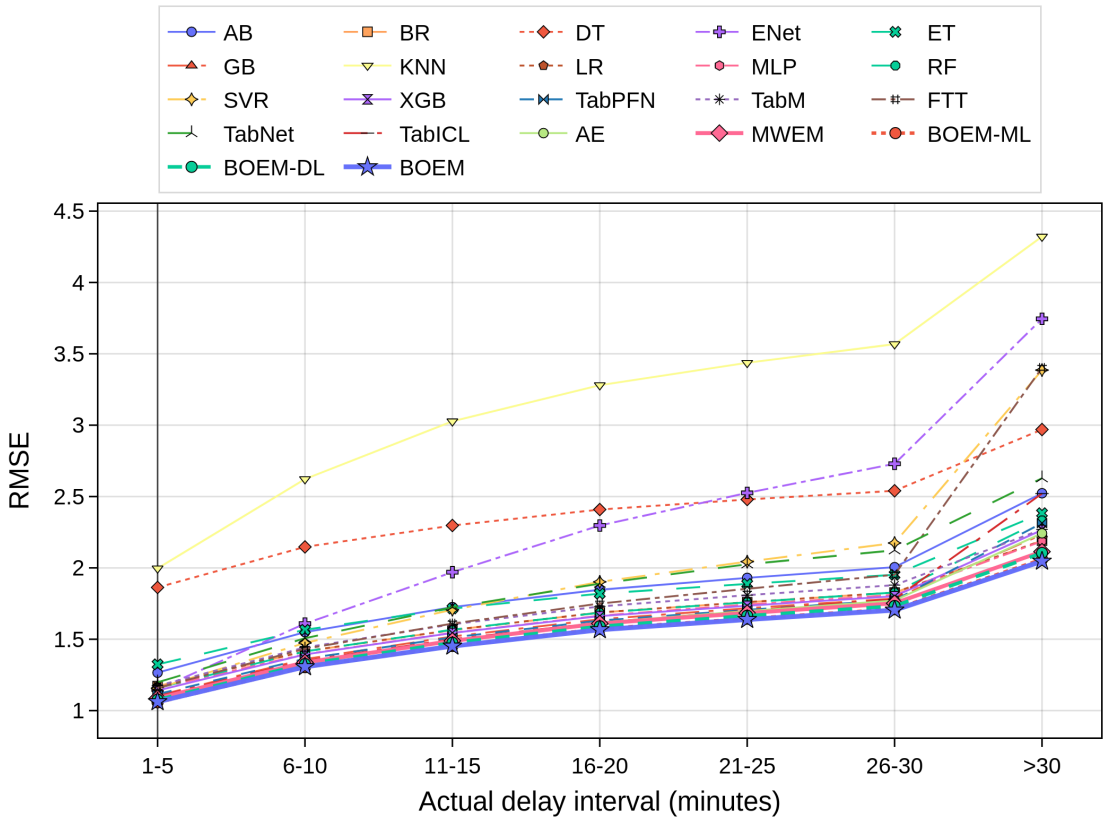
Average performance values can hide differences between small and large delay conditions. This is important because large delays are less frequent, but they are more important from an operational point of view. To examine this behaviour, model errors were analysed across cumulative positive-delay bins, with zero-minute delay representing on-time operation. Positive delays were evaluated using cumulative thresholds from 1 to 30 minutes, with an additional category for delays greater than 30 minutes.

Figure 5.7 shows that prediction error increases as delay magnitude increases. This is expected because larger delays are more irregular and often depend on disruption propagation, congestion, and operational interactions that are harder to model from static features alone. The increase is more pronounced for weaker baselines such as KNN and ENet, while stronger tree-based models and ensemble methods show more gradual degradation. MWEM already limits error escalation compared with the individual baseline models, which supports the value of the first proposed multi-metric aggregation strategy. BOEM provides the strongest behaviour overall, remaining among the best models across the delay ranges. This indicates that the hybrid Bayesian optimised ensemble is not only effective for common low-delay observations, but also maintains comparatively lower error as the prediction task becomes more difficult. Since severe-delay observations are less frequent, the results for the largest delay categories should be interpreted with caution.

To complement the preceding analysis, prediction performance was also examined using the delay intervals identified in the exploratory analysis in Chapter 4. While the previous results show how prediction performance changes more generally with delay magnitude, this additional analysis compares BOEM with the individual ML and



(a) MAE

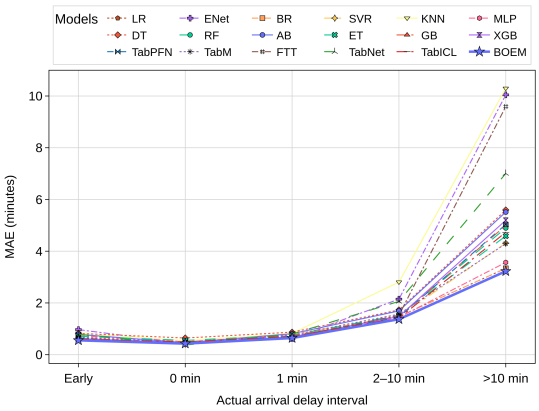


(b) RMSE

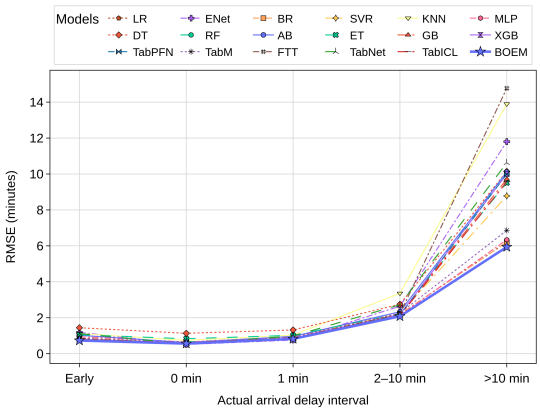
Figure 5.7: Prediction error across cumulative positive-delay bins for both routes.

tabular DL models within five operationally interpretable actual arrival-delay intervals: early, 0 minutes, 1 minute, 2–10 minutes, and more than 10 minutes.

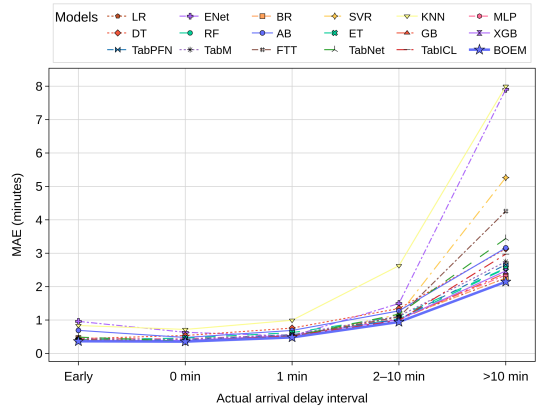
Figure 5.8 shows that MAE and RMSE remained comparatively low for early, on-time and one-minute observations, increased for the 2–10 minute interval, and rose substantially for delays greater than 10 minutes on both routes. The uneven distribution therefore matters when interpreting model performance: the models receive many examples of routine running conditions, while larger-delay behaviour is represented by fewer observations and potentially less coverage of different disruption patterns. The higher errors in the larger-delay intervals show that strong overall performance does not imply uniform prediction accuracy across the delay range. This pattern is consistent with both the smaller representation and greater irregularity of severe-delay conditions; the experiments evaluate these characteristics together within the observed railway data.



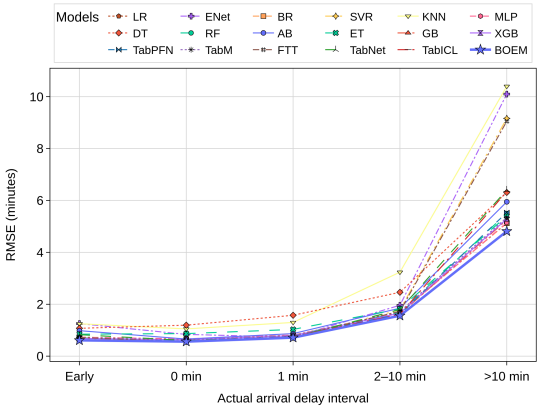
(a) NL route: MAE



(b) NL route: RMSE



(c) WW route: MAE



(d) WW route: RMSE

Figure 5.8: Prediction performance across actual arrival-delay intervals.

BOEM achieved the lowest MAE and RMSE across the reported delay intervals. The

individual models showed greater variation in the larger-delay intervals. In particular, several models that were competitive for near-timetable observations deteriorated sharply when the actual delay exceeded 10 minutes. BOEM showed a more gradual increase in error and remained comparatively stable across both routes. This shows that BOEM's performance advantage was also present in the larger-delay intervals and was not limited to near-timetable observations.

Taken together, these analyses demonstrate that the proposed method retains a predictive advantage as the task moves from frequent near-timetable observations to less frequent larger delays. Its comparatively lower errors across the reported intervals provide evidence that the benefits extend beyond the most common operating conditions. At the same time, the increase in absolute error means that predictions for larger delays should be interpreted with the accuracy demonstrated for those ranges, rather than the full-data average. The smaller samples in the most severe-delay ranges also make their performance estimates more sensitive to the particular disruption events represented. The findings therefore support the method's comparative performance under the observed larger-delay conditions, while evaluation across additional severe-delay events would strengthen the evidence for those less frequent cases.

5.11.5 SHAP-Based Feature Importance Analysis

To provide additional interpretability for the prediction results, SHAP analysis was applied to the Gradient Boosting (GB) model. The methodological basis of SHAP is described in Section 3.9. In this research, SHAP is used as a supporting analysis and is not intended to explain the proposed BOEM model directly. BOEM is a prediction-level weighted ensemble over heterogeneous base learners. Therefore, this analysis examines which input variables were most influential for a strong tree-based baseline model.

Figure 5.9 shows the global SHAP results for the GB model using the top 15 most important features. The bar plot reports the mean normalised absolute SHAP importance across the section-level models, with error bars showing the standard deviation across sections. The beeswarm plot shows both the magnitude and direction of the feature effects on the model output. Higher SHAP values indicate that a feature increased the predicted arrival delay, while lower SHAP values indicate that it reduced

the predicted delay.

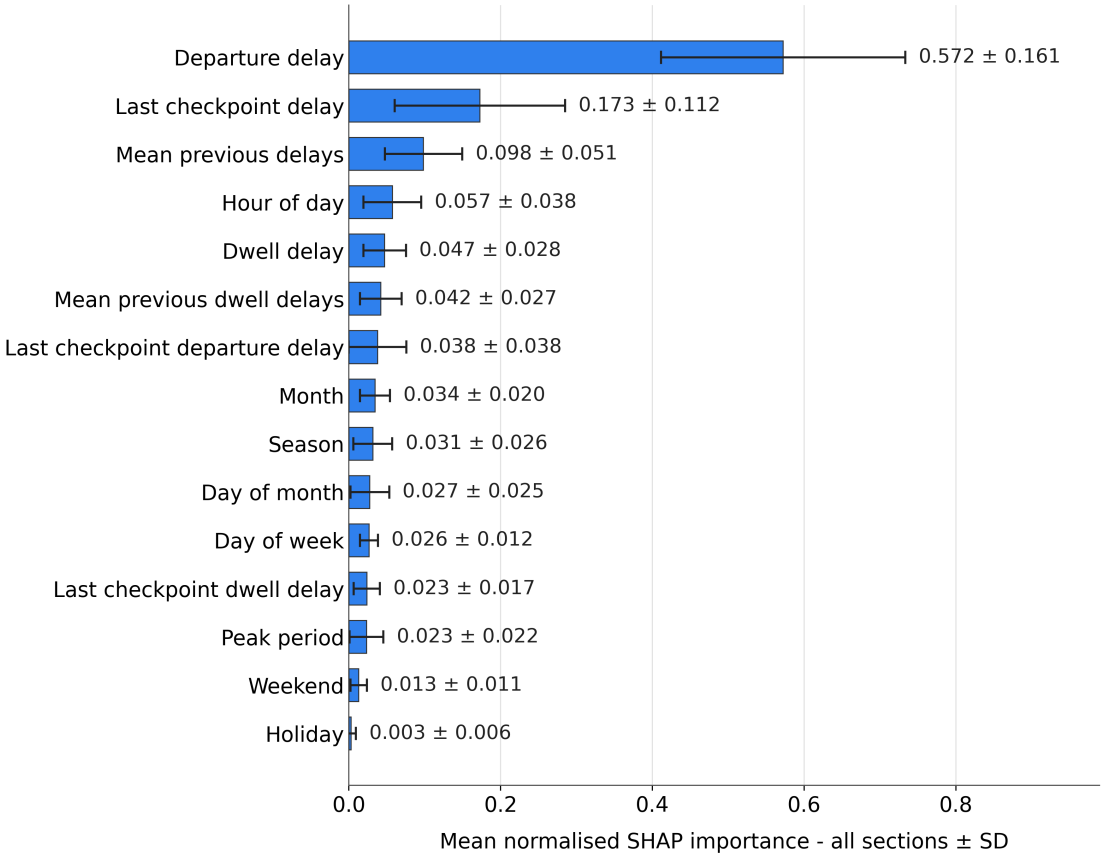
The results show that *departure delay* had the highest mean SHAP importance. This indicates that the current delay state of the train is a major factor in predicting downstream arrival delay. The next most important variables were *last checkpoint delay* and *mean previous delays*. These results are consistent with the delay propagation behaviour of railway services, where current and recent delay history strongly influence subsequent arrival performance.

Other operational and temporal variables, including *hour of day*, *dwell delay*, *mean previous dwell delays*, *month*, *season*, and *peak period*, also contributed to the GB predictions, but their influence was smaller than that of the main delay-related variables. This indicates that the model relied primarily on operational delay information, while temporal and service-context features provided additional supporting information. Overall, the SHAP results provide an interpretable check that the GB model relied mainly on operationally meaningful delay features (Lundberg and Lee, 2017; Lundberg et al., 2020).

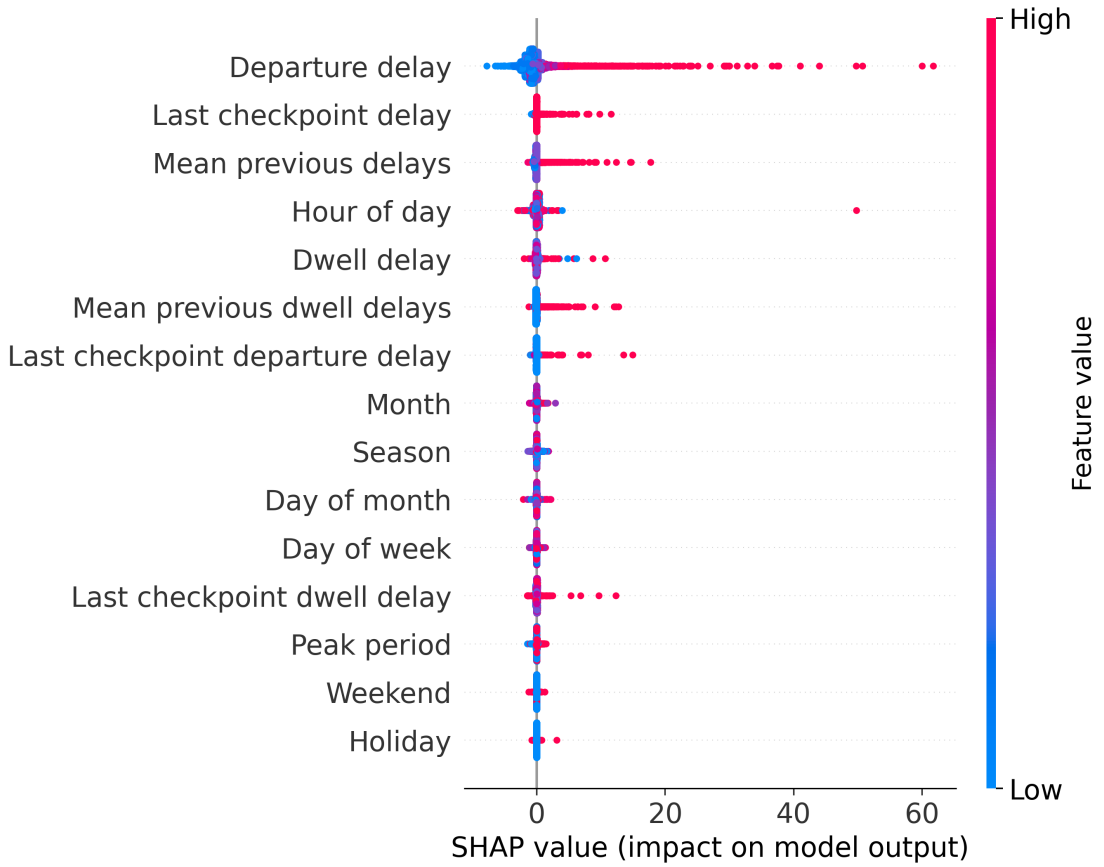
5.11.6 Ranking and Taylor Diagram Analysis

To provide a fair comparison across heterogeneous sections, a ranking-based evaluation was applied. For each section, models were ranked separately according to each evaluation metric, and the resulting ranks were averaged across sections. The standard deviation of the ranks was also computed to assess stability. Figure 5.10 illustrates the results. The analysis confirms that BOEM achieved the strongest average ranking with low variability across metrics. MWEM also ranked ahead of most individual models, confirming the contribution of the first proposed ensemble. BOEM further improved the ranking position and stability, showing that Bayesian weight optimisation provides an additional advantage beyond validation-based metric weighting.

The Taylor diagram in Figure 5.11 provides an integrated comparison of model behaviour by showing three statistical properties in one plot: the correlation coefficient, the standard deviation, and the root mean square deviation (RMSD). The correlation coefficient shows how closely the predicted delays follow the pattern of the observed delays. The standard deviation shows whether the spread of the predicted delay values

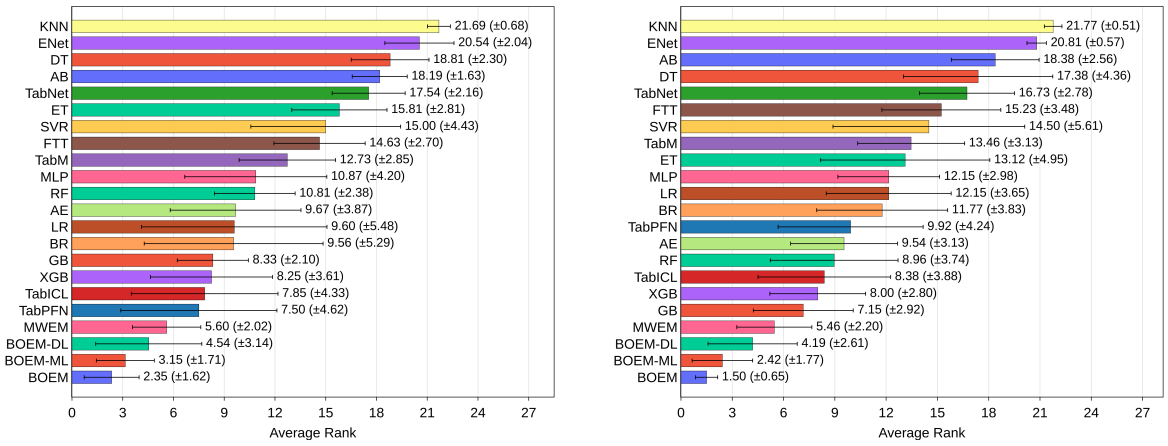


(a) Global mean normalised SHAP importance with standard deviation across sections.



(b) Global SHAP beeswarm plot showing feature effects on GB predictions.

Figure 5.9: Global SHAP analysis for the Gradient Boosting model using the top 15 features.



(a) ACC@1 (b) MAE

(c) R^2 (d) RMSE

Figure 5.10: Average ranking and rank variability of models across sections. Lower average rank indicates better relative performance.

is similar to the spread of the actual delay values. RMSD measures the overall deviation between the predicted and observed delay patterns, where smaller RMSD indicates lower prediction error. In the diagram, the reference point represents the statistical properties of the observed delay values. Therefore, models located closer to the reference point have a better overall balance of high correlation, similar variability, and low deviation from the observed values.

As shown in Figure 5.11, BOEM is positioned closest to the reference point among the compared models. This indicates that BOEM provides the most favourable balance between correlation, variability, and RMSD. Several strong base models are also positioned near the reference area, but they remain farther from the observed delay point because of either higher deviation or less suitable variance representation. This result supports the previous ranking and section-level findings. It shows that BOEM does not only improve one isolated metric, but provides a more balanced prediction profile across different aspects of model behaviour.

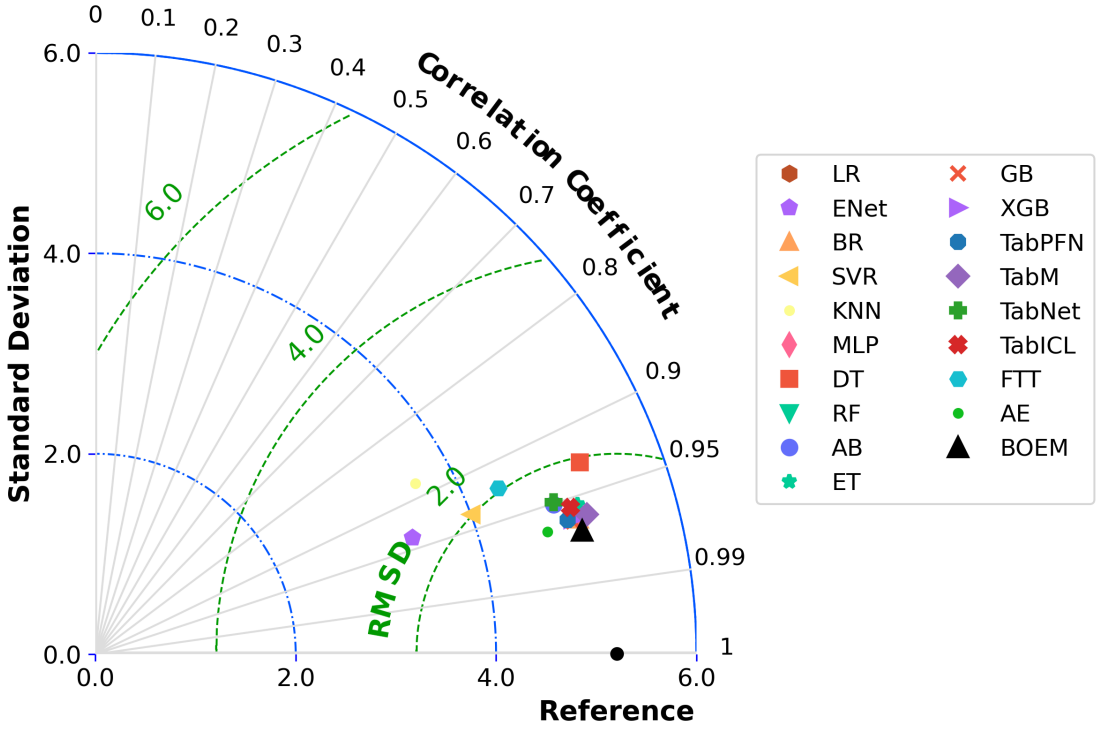


Figure 5.11: Taylor diagram comparing the individual base models and BOEM across both routes.

5.11.7 Statistical Significance Analysis

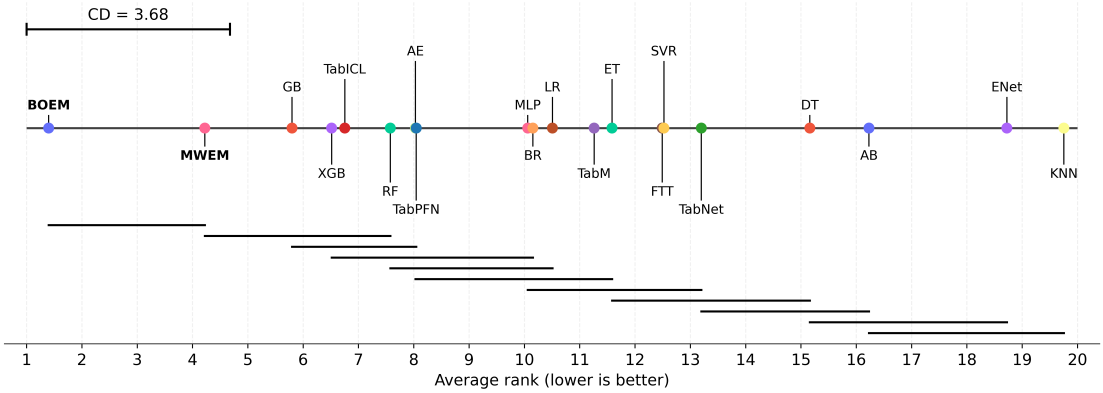
To examine whether the observed performance differences among the models were statistically significant, the Friedman test followed by Nemenyi's post-hoc analysis was applied. Models were ranked within each section-fold evaluation unit, and significance was assessed at $\alpha = 0.05$ using the Critical Difference (CD) threshold. Figure 5.12 presents the CD diagrams for MAE and R^2. Models positioned closer to the left side of each diagram have better average ranks, while horizontal bars connect models that do not differ significantly according to the Nemenyi test. These tests were selected because the study compares more than two models using the same matched section–fold evaluation units. The performance values may also differ in distribution across heterogeneous route sections and validation folds. The Friedman test provides a rank-based comparison of all models without requiring the performance values to follow a normal distribution. When the Friedman test identifies a significant overall difference, the Nemenyi post-hoc test allows the average ranks of the models to be compared using the Critical Difference threshold (Demšar, 2006). This combination therefore helps determine whether the observed performance differences are consistent across the evaluation units rather than being caused by variation in individual sections or folds.

For MAE, BOEM achieved the best overall rank of 1.4, indicating the strongest average performance across sections. MWEM, which is the first proposed ensemble method in this research, achieved the second-best rank of 4.2. The Friedman test confirmed statistically significant differences between the models, with $\chi^2 = 1560.440$ and $p < 0.001$. The Nemenyi critical difference was 3.678. The difference between the average ranks of BOEM and MWEM was smaller than the critical difference, meaning that their difference was not statistically significant at $\alpha = 0.05$.

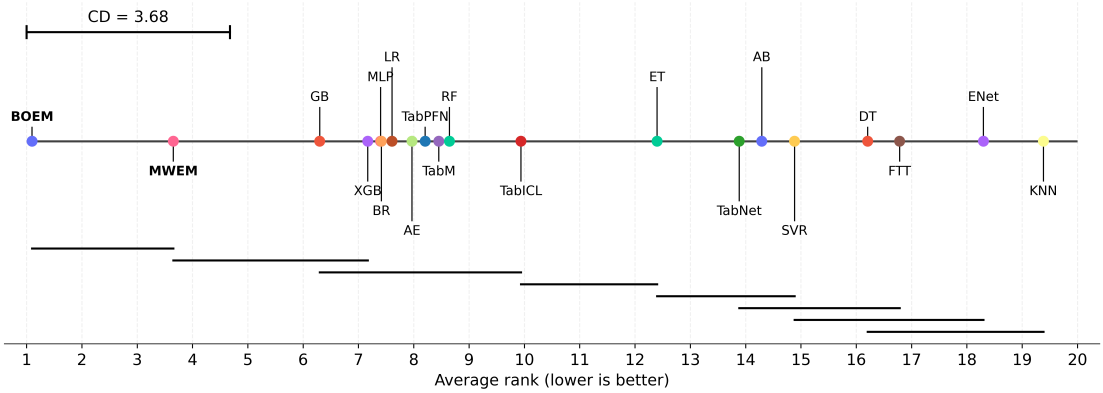
The Friedman test was conducted over 130 section-fold evaluation units, obtained from 26 sections and 5 cross-validation folds, across 20 competing models. The large test statistic reflects highly consistent ranking differences across these repeated evaluations. To avoid over-emphasising the numerical magnitude of the p-value, it is reported as $p < 0.001$.

For R^2, BOEM again achieved the best average rank of 1.1, followed by MWEM with an average rank of 3.7. The Friedman test confirmed statistically significant

differences among the models for this metric, with $\chi^2 = 1751.564$ and $p < 0.001$. The resulting critical difference was also 3.678 for the Nemenyi post-hoc comparison. The difference between BOEM and MWEM remained smaller than the critical difference. These results show that the ranking advantage of BOEM is not limited to one metric, but is repeated across explained variance and error-based measures.



(a) MAE



(b) R^2

Figure 5.12: Nemenyi Critical Difference diagrams for model performance. Connected models do not differ significantly at $\alpha = 0.05$.

A more detailed examination of the ranking structure reveals an important practical distinction. Both proposed ensemble methods occupy the leading part of the ranking distribution. MWEM achieved a better average rank than the individual models and the AE baseline, confirming the strength of the first proposed multi-metric weighted ensemble However, MWEM also overlaps with a second group that includes strong individual models such as GB, XGB, TabICL, and RF in the MAE-based comparison. In contrast, BOEM remains at the leading edge of the ranking distribution and consistently occupies the top position across all reported metrics.

This result suggests that the Bayesian optimisation layer is associated with better

average ranks and more consistent performance across the evaluated units. Therefore, the added value of BOEM lies in its ability to produce more reliable and robust predictions across heterogeneous sections, even when the absolute performance gains over MWEM are moderate. This is important for the research because it shows a clear development path from MWEM as the first proposed ensemble method to BOEM as the final multi-metric Bayesian optimised ensemble framework.

5.11.8 Cost-Based Evaluation

To better reflect operational railway priorities, a cost-based evaluation framework was used. Standard error metrics such as MAE and RMSE treat underestimation and overestimation in the same way if the absolute error is equal. However, in railway operations, these two types of error do not always have the same practical consequence. Underpredicting delays may contribute to missed connections, inaccurate passenger information, poor disruption management, and cascading disruption, whereas overprediction mainly introduces additional scheduling slack. For this reason, delay underestimation was penalised more strongly than overestimation. As no established railway-specific evaluation measure was identified for this purpose, this research adapted an asymmetric linear loss and proposed it as a cost-based evaluation method for train delay prediction. The method assigns different penalties to underprediction and overprediction to reflect their different operational consequences. This asymmetric cost was used only for evaluation and did not affect model training or ensemble optimisation. The cost is defined as:

$$\text{Cost}(y, \hat{y}) = \begin{cases} c_{\text{under}} \cdot (y - \hat{y}), & \hat{y} < y, \\ c_{\text{over}} \cdot (\hat{y} - y), & \hat{y} \geq y, \end{cases} \tag{5.11.1}$$

$$\text{Avg. Asym. Cost} = \frac{1}{N} \sum_{i=1}^{N} \text{Cost}(y_i, \hat{y}_i) \tag{5.11.2}$$

where y_i and $\hat{y}_i$ denote the actual and predicted delays for test instance i, respectively, and N is the number of test instances. The average cost represents the mean asymmetric prediction cost across all test samples. Lower values indicate better operational performance under the selected asymmetric penalty setting. For this evaluation, underprediction was assigned three times the cost of overprediction, with $c_{\text{under}} = 3$ and

Table 5.5: Cost-based evaluation of BOEM and the individual baseline models. Average cost is reported with asymmetric penalties ($c_{\text{under}} = 3$, $c_{\text{over}} = 1$). Lower is better.

Model	Avg. Cost	UnderRate	OverRate
BOEM	**1.133**	0.412	0.588
GB	1.218	0.391	0.609
XGB	1.233	0.416	0.584
TabICL	1.241	0.374	0.626
RF	1.244	0.386	0.614
MLP	1.291	0.426	0.574
TabPFN	1.298	0.460	0.540
AE	1.301	0.430	0.570
BR	1.303	0.423	0.577
LR	1.304	0.424	0.576
ET	1.321	0.363	0.637
TabM	1.324	0.392	0.608
FTT	1.507	0.463	0.537
TabNet	1.519	0.432	0.568
DT	1.541	0.220	0.780
AB	1.633	0.415	0.585
SVR	1.733	0.498	0.502
ENet	2.382	0.311	0.689
KNN	3.222	0.424	0.576

$c_{\text{over}} = 1$, to represent the greater potential consequences of underestimated delays.

Table 5.5 reports the average asymmetric cost together with underprediction and overprediction rates. BOEM achieved the lowest average cost of 1.133, outperforming the strongest individual models, including GB (1.218), XGB (1.233), TabICL (1.241), and RF (1.244). The cost reduction achieved by BOEM was approximately 7–9% compared with the strongest individual models and substantially larger against weaker models. These findings indicate that the proposed hybrid ensemble is not only accurate under standard regression metrics, but also more effective under operationally asymmetric penalty conditions.

The underprediction and overprediction rates also provide useful information about model behaviour. BOEM produced an underprediction rate of 0.412 and an overprediction rate of 0.588. This indicates that BOEM did not simply achieve the lowest cost by always overestimating delays. Instead, it produced both underpredictions and overpredictions while achieving the lowest average asymmetric cost while still reducing the more expensive underestimation errors. This is important because a model that avoids underestimation by always predicting larger delays may reduce one

type of risk but can also reduce the usefulness of the predictions for timetable recovery and passenger information.

The difference between BOEM and weaker models is much larger under the asymmetric cost setting. For example, the cost of KNN is 3.222, ENet is 2.382, and SVR is 1.733, showing that models with weaker predictive stability become much less suitable when delay underestimation is penalised more heavily. BOEM also improved clearly over stronger models. Its lower cost compared with GB, XGB, TabICL, and RF shows that the hybrid ensemble provides an operational advantage beyond standard MAE and RMSE improvements.

A Friedman test based on section-level asymmetric costs confirmed a statistically significant difference among the 19 evaluated models across 26 sections ($\chi^2 = 340.09$, $p < 0.001$). This confirms that the cost performance of the models is not equivalent. The Nemenyi post-hoc comparisons showed that BOEM differed significantly from the competing models whose average-rank differences exceeded the critical difference, supporting the conclusion that BOEM provides a lower operational cost across sections. Overall, the cost-based evaluation demonstrates that the proposed hybrid ensemble can be more suitable for practical delay prediction tasks, where the reliability of predictions under operationally costly errors is as important as average accuracy.

5.11.9 Robustness under Missing Values

The previous experiments evaluated the models when the required input information was available. In this study, robustness refers to how well a model maintains stable predictive performance when some input information is missing. It is quantified by examining how prediction error changes as the level of missingness increases, with a smaller increase in MAE, and therefore a smaller degradation slope, indicating stronger robustness. Railway data may contain missing values at the time a prediction must be produced. This is particularly important for departure delay. The feature-importance analysis showed that departure delay was the most influential feature for predicting arrival delay. At the same time, the real-time experiments showed that confirmed departure information was the input that was most commonly missing when a prediction had to be generated. A train may have arrived at a station while its confirmed departure

update has not yet been received. Therefore, missing values in departure delay represent both an important and realistic source of uncertainty for the proposed framework.

For this reason, the robustness experiment focused specifically on missing values in the departure-delay feature rather than randomly removing values from different input features. This provides a more operationally meaningful missing-value test. Instead of examining arbitrary missing values, the experiment evaluates whether the models can maintain accurate arrival-delay prediction when one of their most important input values is missing and must first be estimated from preceding railway information.

The experiment used the WW route and the BOEM-ML configuration. The WW route contains 19 route sections and provides a relatively long sequence in which preceding-section information can be used. The first section was required to provide preceding information, while the remaining 18 sections, WW2–WW19, were used as the evaluation sections. The same five outer folds were used for all compared models so that the models were evaluated using the same training and test observations.

Ten missing-value levels were examined: 0%, 2%, 4%, 6%, 8%, 10%, 20%, 30%, 40%, and 50%. At each level, the specified proportion refers to the proportion of test observations for which the current departure-delay value was treated as missing. For example, at 20%, departure delay was removed from 20% of the selected test observations. The same observations were treated as missing for all models. This ensured that differences between models were not caused by different missing-value patterns. The 0% condition was also generated within the same experiment, using the original departure information, and therefore provided a directly comparable baseline without missing values.

When the departure-delay value was missing, it was reconstructed from the preceding row in the filtered railway sequence. Consider two consecutive observations representing movements from station A to B and from station B to C. For prediction on the B–C section, the preceding A–B model can estimate the arrival delay at station B. This estimated arrival-delay information at B is then used to obtain an estimate of the departure delay from B. This estimated value replaces the missing departure-delay value before the current B–C arrival-delay prediction is produced.

For the individual models, this reconstruction was performed using the same model

family. For example, when RF was evaluated on the current section, the RF model associated with the preceding section was used to estimate the preceding arrival delay. This evaluates each model as an independent prediction process and allows its sensitivity to missing and estimated departure information to be measured.

For the proposed method, one common departure-delay estimate was first generated from the preceding-section ensemble predictions and then used by all current-section base models. The resulting value was used as the same departure-delay input for all current-section base learners before their predictions were combined by the proposed method. This ensured that the models within the ensemble operated from the same estimated value when departure-delay information was missing.

Figure 5.13 shows the resulting MAE values from 0% to 50% missing departure-delay values. MAE is used as the main robustness measure because it directly represents the average prediction error in minutes and allows the effect of increasing missingness to be compared across the models.

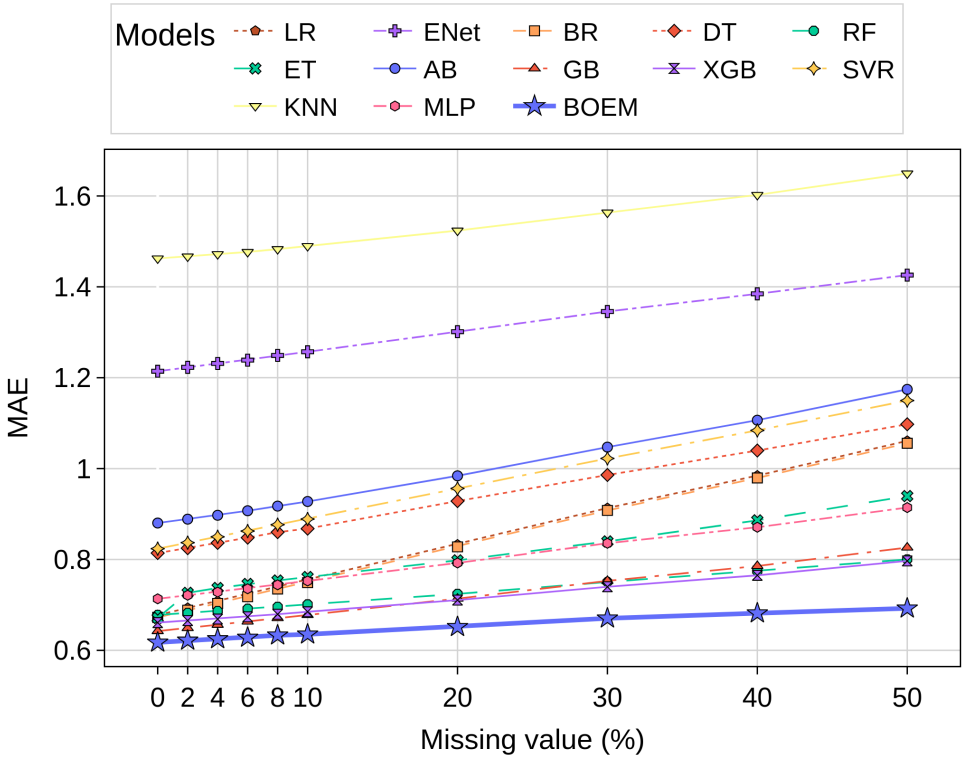


Figure 5.13: MAE performance of the proposed method and the individual ML models as the proportion of missing departure-delay values increases from 0% to 50%.

The results show that the proposed method maintained stronger performance as the proportion of missing values increased. It achieved the lowest mean MAE at every

evaluated level. More importantly, the difference between the proposed method and the individual models generally became larger as the proportion of missing departure-delay values increased. Compared with the strongest individual competitor at each level, the proposed method reduced MAE by approximately 4% at 0%, 6% at 10%, 8% at 20%, 9% at 30%, 11% at 40%, and 13% at 50%. This indicates that its advantage was maintained and became larger as more missing departure-delay values had to be estimated.

Table 5.6 provides additional information about both accuracy and consistency across all ten evaluated missing-value levels. The values report the mean MAE across the 18 evaluation sections, with the corresponding standard deviation shown in parentheses on the second line of each cell. The final column reports the mean degradation slope and therefore shows how quickly each model's MAE increased as the proportion of missing values increased.

Table 5.6: Mean MAE across all evaluated missing-value levels, with standard deviations shown in parentheses, and mean MAE degradation.

Model	0%	2%	4%	6%	8%	10%	20%	30%	40%	50%	Degradation slope
	0.678	0.693	0.708	0.723	0.740	0.754	0.833	0.913	0.985	1.061	
LR	(0.583)	(0.582)	(0.580)	(0.579)	(0.582)	(0.576)	(0.575)	(0.578)	(0.583)	(0.593)	0.077
	1.214	1.223	1.231	1.239	1.249	1.257	1.301	1.346	1.385	1.426	
ENet	(0.652)	(0.655)	(0.657)	(0.661)	(0.665)	(0.666)	(0.681)	(0.696)	(0.712)	(0.729)	0.043
	0.673	0.688	0.703	0.718	0.735	0.749	0.828	0.908	0.979	1.056	
BR	(0.583)	(0.582)	(0.580)	(0.579)	(0.581)	(0.576)	(0.575)	(0.578)	(0.583)	(0.593)	0.077
	0.814	0.825	0.837	0.848	0.860	0.868	0.928	0.986	1.040	1.097	
DT	(0.833)	(0.828)	(0.823)	(0.821)	(0.821)	(0.810)	(0.802)	(0.777)	(0.760)	(0.750)	0.057
	0.678	0.682	0.687	0.692	0.696	0.701	0.724	0.750	0.774	0.800	
RF	(0.583)	(0.585)	(0.584)	(0.585)	(0.587)	(0.586)	(0.591)	(0.594)	(0.599)	(0.604)	0.024
	0.671	0.727	0.737	0.746	0.754	0.761	0.798	0.840	0.886	0.940	
ET	(0.624)	(0.625)	(0.624)	(0.623)	(0.625)	(0.623)	(0.623)	(0.623)	(0.627)	(0.629)	0.046
	0.880	0.889	0.897	0.907	0.917	0.928	0.984	1.047	1.106	1.174	
AB	(0.607)	(0.610)	(0.613)	(0.615)	(0.620)	(0.619)	(0.636)	(0.659)	(0.678)	(0.698)	0.059
	0.642	0.649	0.656	0.663	0.671	0.678	0.714	0.753	0.786	0.826	
GB	(0.579)	(0.580)	(0.580)	(0.581)	(0.583)	(0.582)	(0.584)	(0.591)	(0.596)	(0.604)	0.037
	0.661	0.665	0.669	0.674	0.679	0.685	0.711	0.740	0.765	0.797	
XGB	(0.578)	(0.578)	(0.576)	(0.576)	(0.576)	(0.574)	(0.568)	(0.564)	(0.561)	(0.561)	0.027
	0.824	0.837	0.850	0.863	0.876	0.889	0.956	1.022	1.084	1.150	
SVR	(0.671)	(0.670)	(0.670)	(0.669)	(0.668)	(0.667)	(0.664)	(0.664)	(0.666)	(0.669)	0.065
	1.462	1.467	1.472	1.477	1.483	1.490	1.524	1.563	1.602	1.649	
KNN	(0.752)	(0.754)	(0.755)	(0.757)	(0.758)	(0.762)	(0.768)	(0.773)	(0.780)	(0.791)	0.037
	0.713	0.721	0.728	0.736	0.744	0.753	0.792	0.836	0.871	0.914	
MLP	(0.640)	(0.642)	(0.643)	(0.643)	(0.648)	(0.648)	(0.658)	(0.672)	(0.684)	(0.702)	0.040
	0.617	**0.622**	**0.625**	**0.628**	**0.633**	**0.635**	**0.652**	**0.670**	**0.682**	**0.692**	
BOEM	**(0.562)**	**(0.562)**	**(0.560)**	**(0.560)**	**(0.560)**	**(0.558)**	**(0.556)**	**(0.553)**	**(0.552)**	**(0.557)**	**0.015**

The standard deviations provide further evidence about consistency across the route sections under missing-value conditions. The proposed method had the lowest MAE standard deviation at every evaluated level, and its variation remained relatively stable as the proportion of missing values increased. Its average variation was also lower than that of the individual models. This indicates that the lower mean error of the proposed method was observed consistently across the route sections rather than being produced

by particularly strong performance in only a small number of sections.

Although the mean MAE and standard deviation describe accuracy and consistency, they do not directly show how quickly performance deteriorates as missingness increases. A model may have a relatively low error at 0% but still deteriorate quickly as more values become missing. For this reason, a degradation analysis was also performed.

For each model and each of the 18 evaluation sections, a straight line was fitted through the MAE values across all ten missing-value levels. Let x_i denote the missing-value percentage and e_i denote the corresponding MAE. The slope of the fitted line was calculated as

$$\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(e_i - \bar{e})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{5.11.3}$$

where $\bar{x}$ and $\bar{e}$ are the mean missing-value percentage and mean MAE, respectively, and β_1 represents the change in MAE for a one percentage-point increase in missing values. Since the missing-value levels are expressed in percentage points, the degradation measure was defined as

$$D = 10\beta_1 \tag{5.11.4}$$

where D represents the average increase in MAE for each 10 percentage-point increase in missing values. A smaller value of D therefore indicates slower deterioration and stronger robustness.

For example, a degradation slope of 0.05 means that MAE increases on average by approximately 0.05 minutes for each 10 percentage-point increase in missing values. A smaller value therefore indicates that the model is less affected as the level of missingness increases.

The final column of Table 5.6 shows the mean degradation slopes across the 18 sections. The proposed method achieved the smallest degradation slope of 0.015, while RF was the closest individual competitor with 0.024, followed by XGB with 0.027. Compared with RF, the proposed method achieved an approximately 37% lower degradation slope, indicating a substantially slower increase in prediction error as missingness increased. Together with the increasing performance gap shown in

Figure 5.13, this indicates that the proposed method was not only more accurate but also less sensitive to increasing missing values.

The consistency of the degradation results was also examined across the individual route sections. For each individual model, its degradation slope for each section was paired with the corresponding slope of the proposed method for the same section. A paired Wilcoxon signed-rank test was then applied separately to each proposed method versus individual model comparison. This test was selected because it evaluates whether the paired section-level differences consistently favour one method without requiring the differences to follow a normal distribution. The Wilcoxon statistic represents the sum of the ranks of the paired section-level differences and indicates how consistently the degradation slopes favour the proposed method, while the p-value indicates whether this difference is statistically significant under the null hypothesis of no systematic difference between the two methods.

Table 5.7: Paired Wilcoxon signed-rank comparisons of MAE degradation slopes between BOEM and the base models across the 18 route sections. BOEM achieved a mean degradation slope of 0.015.

Model	Base-model slope	Wilcoxon statistic	p-value
RF	0.024	154	< 0.001
XGB	0.027	165	< 0.001
GB	0.037	171	< 0.001
KNN	0.037	154	< 0.001
MLP	0.040	171	< 0.001
ENet	0.043	159	< 0.001
ET	0.046	169	< 0.001
DT	0.057	165	< 0.001
AB	0.059	171	< 0.001
SVR	0.065	171	< 0.001
BR	0.077	171	< 0.001
LR	0.077	171	< 0.001

As shown in Table 5.7, the proposed method showed significantly slower MAE degradation than every base model ($p < 0.001$ for all comparisons). RF was the closest robustness competitor in terms of mean degradation slope, but its difference from the proposed method remained statistically significant. These results provide statistical evidence that the proposed method was less affected as the level of missingness increased.

These statistical results support the pattern observed in Figure 5.13. As the proportion of missing departure-delay values increased, the proposed method maintained a lower prediction error while the individual models generally deteriorated more quickly.

The results also indicate that the proposed method was more effective at recovering

from missing departure-delay values and limiting the propagation of estimation errors between consecutive sections. When a departure-delay value was missing, its estimate was obtained from preceding-section predictions and then used as an input for the current-section prediction. Any error in this estimate could therefore propagate to the next prediction and increase its error. However, the proposed method showed the smallest increase in MAE as the proportion of estimated values increased, suggesting that it was less sensitive to these propagated errors than the base models. This provides further evidence that the ensemble can maintain more reliable predictions when missing operational information must be reconstructed from earlier predictions.

Overall, the results provide evidence of both lower error and stronger robustness under missing values. The proposed method maintained the lowest MAE across all evaluated missing-value levels, showed the lowest variation in MAE across the route sections, and experienced the slowest increase in MAE as the proportion of missing departure-delay values increased. The section-level statistical comparisons further confirmed that this slower deterioration was consistent across the route. These findings indicate that the proposed method provides more reliable performance under missing-value conditions, particularly when the important departure-delay feature is missing and must be estimated from preceding railway predictions.

5.11.10 Cross-Dataset Generalisation

Operational conditions differ across railway routes and services. A model that performs well on one route may not necessarily perform well on another route, especially when the route length, stopping pattern, station density, and delay propagation behaviour are different. For this reason, cross-dataset generalisation was examined to assess whether the proposed model can transfer predictive behaviour across the two railway routes used in this research.

The training data from one route were used to develop the models, which were then evaluated using the test data from the other route. Two transfer directions were considered: NL→WW, where models were trained using the NL route and tested on the WW route, and WW→NL, where models were trained using the WW route and tested on the NL route. The analysis reports per-section R^2 and RMSE in both directions.

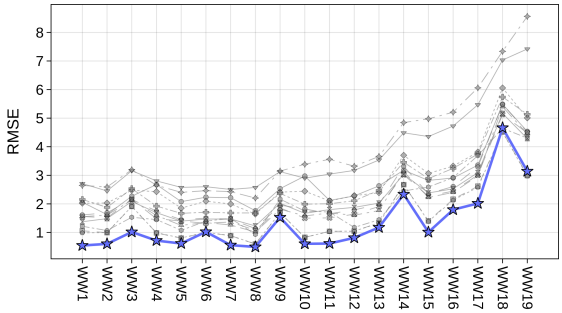
The highlighted curve in each plot corresponds to BOEM-ML, while the grey curves represent the competing models.

The cross-dataset generalisation analysis was performed using the BOEM-ML configuration. This is consistent with the robustness and weighting-strategy analyses, where the objective was to examine the stability of the ensemble weighting mechanism in a controlled setting. Restricting this experiment to the conventional ML model pool allows the transfer behaviour of the proposed Bayesian weighting strategy to be assessed without additional variability from heterogeneous DL architectures.

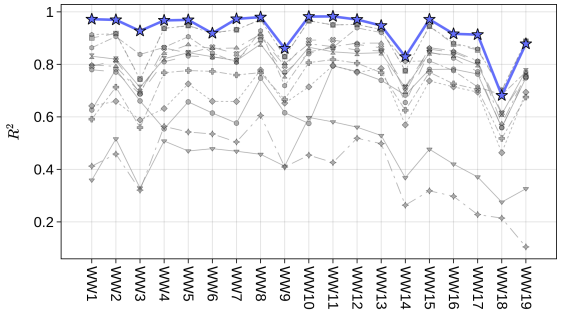
Figure 5.14 presents the cross-route evaluation results. In the NL→WW direction, BOEM achieved the strongest overall behaviour across the WW target sections. It generally produced the highest R^2 values and the lowest RMSE values compared with the competing models. This is particularly important because the WW route contains more sections and shows stronger operational variation across the route. The results indicate that BOEM can preserve strong predictive performance even when it is evaluated on a longer and more heterogeneous route than the one used for training.

The WW→NL direction provides the opposite transfer setting. In this case, models trained on the longer WW route were evaluated on the shorter NL route. BOEM again remained the strongest or among the strongest models across the NL sections. The R^2 results show that BOEM maintains a high level of explained variance across most target sections, while the RMSE results show that it also controls prediction error more effectively than the competing models. These findings indicate that the proposed method captured predictive delay relationships that remained useful across the two contrasting route datasets, providing evidence beyond within-route model performance.

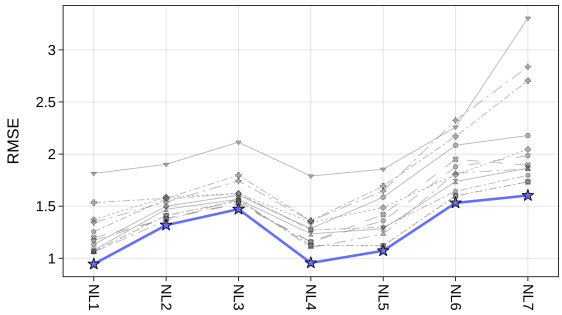
The proposed method retained competitive predictive performance in both transfer directions between NL and WW. The routes differ in station-to-station structure, data coverage and observed delay behaviour, providing contrasting cases for examining cross-route transfer. The results therefore demonstrate that the predictive value of the proposed method extends beyond evaluation within a single route and provide empirical support for transfer between the evaluated datasets. Wider applicability depends on how closely other railway settings share the predictive relationships represented in these data. Differences in network structure, service patterns, traffic density, timetable



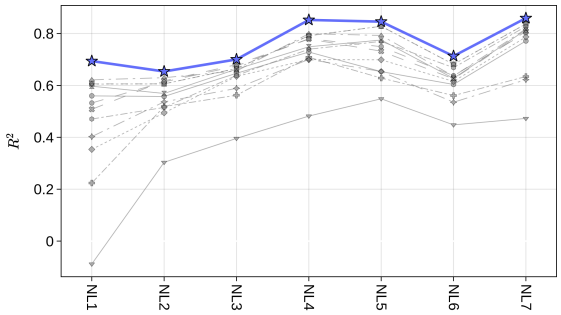
(a) RMSE for models trained on NL and tested on WW



(b) R^2 for models trained on NL and tested on WW



(c) RMSE for models trained on WW and tested on NL



(d) R^2 for models trained on WW and tested on NL

Figure 5.14: Cross-dataset evaluation by section. BOEM, shown by the highlighted line, and the base models were trained on the source route and tested on the target route.

characteristics, station spacing and delay propagation may affect performance, making these important dimensions for further cross-route evaluation.

5.11.11 Comparison with Published Train Delay Prediction Methods

The proposed MWEM and BOEM were further compared with some of the latest and best train delay prediction methods from the studies reviewed in Chapter 2. These methods comprise BO-XGBoost (Shi et al., 2021), ELM-PSO (Bao et al., 2021), the MSM1-WE, MSM2a-COR-WE, and MSM2b-CFD-WE heterogeneous ensembles (Al Ghamdi et al., 2024), ST-NN (Boateng and Yang, 2023a), and Markov-GK (Xu et al., 2023). The numerical results originally reported in these studies could not be compared directly with those obtained in this research. The studies were based on different railway networks, data periods, sample sizes, feature sets, prediction targets, delay distributions, and validation procedures. Each method was implemented based on the algorithmic descriptions, equations, model configurations, optimisation procedures, and experimental settings reported in its publication, and was then applied to the NL and

WW route-section datasets used in this research. Paper-specific settings were retained wherever sufficient implementation details were available. The results below therefore compare the implementations of the published methods under the data and evaluation conditions of this research rather than attempting to reproduce their originally reported numerical values.

Table 5.8 presents the results for the seven NL route sections. BOEM achieved the best mean result for each metric. Among the methods reconstructed from the literature, MSM2b-CFD-WE produced the lowest MAE, while ST-NN produced the lowest RMSE. BOEM reduced both of these errors by around 10%. BOEM also recorded the lowest section-level standard deviation for each metric on the NL route. MWEM also outperformed all reconstructed published methods across all four metrics, achieving higher R^2 and ACC@1 values and lower MAE and RMSE.

Table 5.8: Comparison with published train delay prediction methods on the NL route.

Method	R^2	ACC@1 min (%)	MAE	RMSE
BO-XGBoost (Shi et al., 2021)	0.709 (±0.135)	90.106 (±5.836)	0.784 (±0.180)	1.467 (±0.500)
ELM-PSO (Bao et al., 2021)	0.716 (±0.113)	89.639 (±6.233)	0.791 (±0.176)	1.460 (±0.494)
ST-NN (Boateng and Yang, 2023a)	0.758 (±0.098)	89.925 (±6.276)	0.769 (±0.157)	1.262 (±0.292)
Markov-GK (Xu et al., 2023)	0.751 (±0.093)	88.924 (±7.605)	0.795 (±0.170)	1.287 (±0.290)
MSM1-WE (Al Ghamdi et al., 2024)	0.756 (±0.099)	90.489 (±5.589)	0.754 (±0.162)	1.287 (±0.324)
MSM2a-COR-WE (Al Ghamdi et al., 2024)	0.721 (±0.099)	89.494 (±6.234)	0.792 (±0.192)	1.450 (±0.464)
MSM2b-CFD-WE (Al Ghamdi et al., 2024)	0.761 (±0.102)	91.070 (±5.178)	0.737 (±0.164)	1.298 (±0.376)
MWEM (Golchehreh et al., 2026)	0.770 (±0.097)	91.311 (±5.248)	0.721 (±0.152)	1.258 (±0.341)
BOEM	**0.799 (±0.092)**	**92.685 (±4.349)**	**0.664 (±0.133)**	**1.138 (±0.282)**

Table 5.9 presents the results for the nineteen WW route sections. BOEM again achieved the best mean result for each metric. Among the methods reconstructed from the literature, ST-NN produced the lowest MAE and RMSE. BOEM reduced these errors by 5% and 6%, respectively. BOEM also recorded the lowest standard deviations for R^2, ACC@1, and MAE. Its RMSE standard deviation was 0.905, which was close to the lowest value of 0.903 obtained by MSM2a-COR-WE, although MSM2a-COR-WE had a considerably higher mean RMSE. MWEM achieved higher R^2 and ACC@1 and lower RMSE than all reconstructed published methods, while its MAE of 0.606 was slightly higher than the 0.598 achieved by ST-NN.

Table 5.10 presents the combined results across all 26 route sections. BOEM achieved the strongest overall result for each metric. Among the methods reconstructed from the literature, MSM1-WE and MSM2b-CFD-WE produced the highest R^2, MSM1-WE

Table 5.9: Comparison with published train delay prediction methods on the WW route.

Method	R^2	ACC@1 min (%)	MAE	RMSE
BO-XGBoost (Shi et al., 2021)	0.943 (±0.057)	91.483 (±14.020)	0.627 (±0.571)	1.178 (±0.927)
ELM-PSO (Bao et al., 2021)	0.943 (±0.063)	91.305 (±14.310)	0.651 (±0.576)	1.170 (±0.972)
ST-NN (Boateng and Yang, 2023a)	0.945 (±0.063)	92.093 (±14.148)	0.598 (±0.575)	1.091 (±0.962)
Markov-GK (Xu et al., 2023)	0.944 (±0.065)	91.323 (±14.414)	0.690 (±0.586)	1.150 (±0.992)
MSM1-WE (Al Ghamdi et al., 2024)	0.949 (±0.058)	92.124 (±13.655)	0.609 (±0.555)	1.098 (±0.935)
MSM2a-COR-WE (Al Ghamdi et al., 2024)	0.918 (±0.055)	88.117 (±14.161)	0.784 (±0.558)	1.432 (±0.903)
MSM2b-CFD-WE (Al Ghamdi et al., 2024)	0.947 (±0.056)	91.874 (±13.658)	0.616 (±0.549)	1.127 (±0.918)
MWEM (Golchehreh et al., 2026)	0.952 (±0.055)	92.382 (±13.267)	0.606 (±0.537)	1.068 (±0.923)
BOEM	**0.956 (±0.052)**	**92.869 (±12.472)**	**0.568 (±0.513)**	**1.024 (±0.905)**

achieved the highest ACC@1, and ST-NN produced the lowest MAE and RMSE. BOEM reduced MAE and RMSE by approximately 8% and 7%, respectively. BOEM also recorded the lowest overall standard deviation for each metric. MWEM also outperformed all reconstructed published methods across all four metrics.

Table 5.10: Overall comparison with published train delay prediction methods across all route sections.

Method	R^2	ACC@1 min (%)	MAE	RMSE
BO-XGBoost (Shi et al., 2021)	0.880 (±0.134)	91.112 (±12.251)	0.669 (±0.497)	1.256 (±0.834)
ELM-PSO (Bao et al., 2021)	0.882 (±0.128)	90.856 (±12.543)	0.689 (±0.500)	1.248 (±0.869)
ST-NN (Boateng and Yang, 2023a)	0.895 (±0.111)	91.510 (±12.431)	0.644 (±0.500)	1.137 (±0.832)
Markov-GK (Xu et al., 2023)	0.892 (±0.112)	90.677 (±12.831)	0.718 (±0.507)	1.187 (±0.856)
MSM1-WE (Al Ghamdi et al., 2024)	0.897 (±0.111)	91.684 (±11.929)	0.648 (±0.482)	1.149 (±0.813)
MSM2a-COR-WE (Al Ghamdi et al., 2024)	0.865 (±0.112)	88.487 (±12.413)	0.786 (±0.483)	1.437 (±0.799)
MSM2b-CFD-WE (Al Ghamdi et al., 2024)	0.897 (±0.109)	91.658 (±11.869)	0.648 (±0.476)	1.173 (±0.804)
MWEM (Golchehreh et al., 2026)	0.903 (±0.106)	92.094 (±11.557)	0.637 (±0.464)	1.119 (±0.805)
BOEM	**0.914 (±0.095)**	**92.819 (±10.795)**	**0.594 (±0.443)**	**1.055 (±0.782)**

The consistency of these results was examined using the Friedman test followed by the Nemenyi post-hoc test. The statistical analysis was conducted over 130 section-fold evaluation units, obtained from 26 route sections and five cross-validation folds, across the nine compared methods. Statistically significant differences were identified for MAE ($\chi^2 = 527.337$, $p < 0.001$), RMSE ($\chi^2 = 541.771$, $p < 0.001$), and R^2 ($\chi^2 = 546.469$, $p < 0.001$). The Nemenyi critical difference was 1.490 for each metric. BOEM achieved the best average rank for all three metrics, with ranks of 1.662 for MAE, 1.392 for RMSE, and 1.369 for R^2. MWEM achieved the second-best average rank. The difference between the average rank of BOEM and that of every other method exceeded the critical difference for all three metrics, indicating that BOEM achieved significantly better ranks than the other compared methods under the Nemenyi comparison. The differences between MWEM, MSM1-WE, ST-NN, and MSM2b-CFD-WE were generally below the critical difference.

Figure 5.15 presents the Critical Difference diagram for RMSE. BOEM achieved the lowest average rank and is positioned separately from the other methods. MWEM achieved the second-lowest rank and was included in the same non-significant group as MSM1-WE, ST-NN, and MSM2b-CFD-WE. The MAE and R^2 diagrams produced the same main result, with BOEM ranked first and MWEM ranked second, and are therefore not repeated.

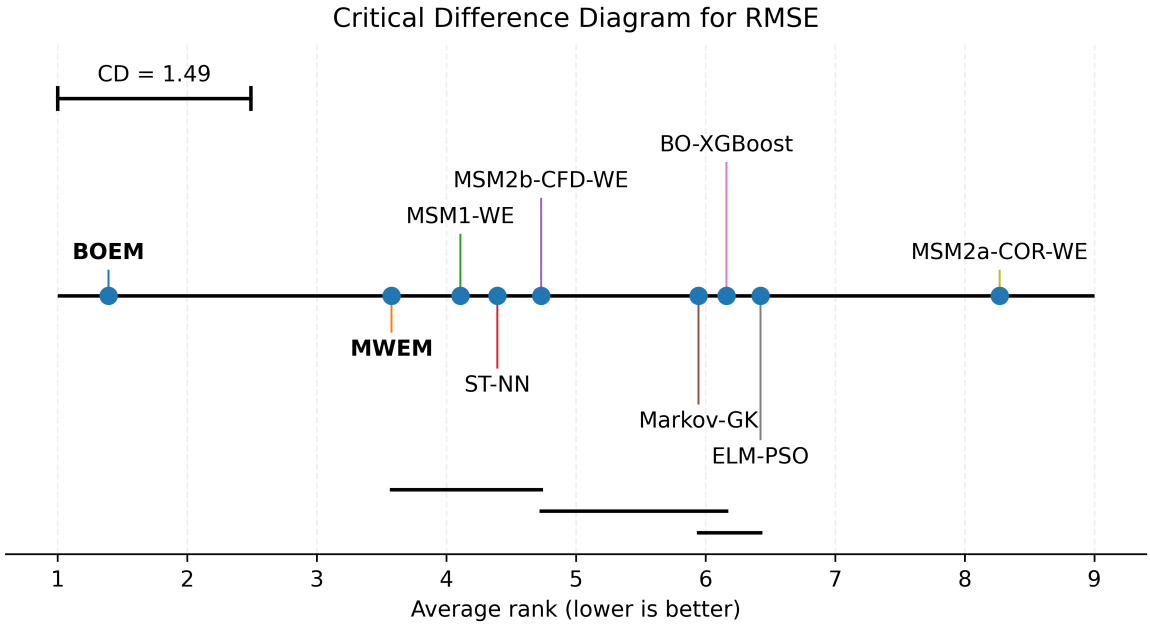


Figure 5.15: Nemenyi Critical Difference diagram for RMSE in the comparison with published train delay prediction methods.

The results demonstrate that both proposed ensemble methods provided stronger overall performance than the methods reconstructed from the literature, with BOEM achieving the best mean values, the lowest overall variability, and the highest statistical ranking. A broader interpretation of these findings is presented in Section 5.12.4.

5.12 Discussion of the Historical Evaluation

5.12.1 Performance of the Individual Models and the Need for Ensemble Learning

The individual models used different learning approaches and therefore responded differently to the same railway features. Some models were better at representing complex relationships between the current delay, previous delay values, and the next arrival delay. Other models produced stronger results only for particular routes, route

sections, or evaluation measures. It was therefore expected that the models would not have the same level of performance under every prediction condition.

The results confirmed this difference. Several conventional ML and tabular DL models achieved good prediction performance, but no single model remained the strongest across all experiments. The tree-based ML models, including GB, RF, XGB, and ET, generally performed well. Some tabular DL models, particularly TabPFN, TabM, and TabICL, also produced strong results. However, the DL models did not consistently outperform the conventional ML models.

This suggests that a more complex model does not automatically produce a better result for tabular railway data. The usefulness of each model depended on the structure and amount of the available data and on the prediction setting. Similar findings have been reported in studies of tabular data, where tree-based models often remain competitive with deep learning methods (Gorishniy et al., 2021; Grinsztajn et al., 2022; McElfresh et al., 2023).

The strong performance of simpler baselines provides an important reference for assessing the additional value of the proposed method. A modest difference in R^2 should be interpreted alongside the corresponding prediction errors. For example, Table 5.4 reports R^2 values of 0.895 for LR and 0.914 for the proposed method, while MAE decreases from 0.696 to 0.594 minutes and RMSE from 1.163 to 1.055 minutes, corresponding to reductions of approximately 14.7% and 9.3%, respectively. These results demonstrate an improvement in prediction error despite the relatively small difference in explained variance. The strength of the comparison is assessed using complementary metrics, section-level results and the statistical analysis in Section 5.11.7. The comparative findings are therefore interpreted alongside the robustness, transfer and live-evaluation results, rather than treating model complexity or a small improvement in one metric as sufficient evidence of superiority.

The justification for the additional ensemble and optimisation stages also concerns performance when prediction conditions become less favourable. The correlation analysis shows that departure delay provides strong linear information about next-station arrival delay, which is consistent with the competitive performance of regression-based baselines. However, confirmed departure information is not always available during live prediction.

Under the reconstruction procedure evaluated in Section 5.11.9, the tested configuration of the proposed method achieved an MAE of 0.692 minutes at 50% missingness, compared with 1.061 minutes for LR and 0.797 minutes for the strongest individual competitor. The asymmetric-loss evaluation in Section 5.11.8 provides complementary evidence under the proposed penalty setting. These benefits are assessed alongside the computational requirements: ensemble-weight optimisation is completed offline, while measured model-level inference averaged 0.7665 milliseconds in the evaluated parallel implementation, as reported in Section 6.11.4. The additional complexity is therefore justified through the demonstrated combination of accuracy, resilience to incomplete inputs and prediction-time feasibility.

The differences between the evaluation measures also affected the model comparison. MAE represents the average size of the prediction errors, while RMSE gives more importance to larger errors. In contrast, R^2 measures how much of the variation in arrival delay is represented by the model, and ACC@1 measures how often the prediction is within one minute of the observed delay. Because these measures represent different aspects of performance, a model with the lowest MAE did not always achieve the lowest RMSE or the highest R^2 and ACC@1. Selecting a model using only one measure could therefore give an incomplete view of its overall performance.

The model rankings also changed between the NL and WW routes and across individual route sections. The two routes differed in their length, number of sections, amount of data, and delay behaviour. Some sections also contained more variable delay patterns than others. These differences affected the difficulty of the prediction task and explain why a model that performed well on one section did not always provide the same result on another. Prediction also became more difficult for larger delays, which were less common and may have been linked to more unusual operating conditions. The SHAP analysis of GB provided further support for these results. Departure delay and the features related to previous delays had the strongest effects on its predictions. This indicates that the recent operational state of the train was important for predicting delay at the next station.

Overall, the results showed that the individual models had different strengths and weaknesses. No single model provided the strongest and most consistent performance

across all routes, sections, delay levels, and evaluation measures. This provided the main reason for developing a heterogeneous ensemble. By combining models with different prediction behaviours, the ensemble could use their complementary strengths. However, the models also needed to be selected and weighted carefully, because giving the same importance to every model could allow weaker predictions to reduce the quality of the final result. The following section discusses how MWEM addressed this problem using several validation measures.

5.12.2 Multi-Metric Weighting and the MWEM Framework

The results for the individual models showed that their rankings changed across MAE, RMSE, R^2, and ACC@1. This was expected because each measure represents a different part of prediction performance as mentioned in previous section. Therefore, selecting models using only one measure could favour one aspect of performance while ignoring others.

MWEM was developed to use these measures together when selecting and weighting the models. Instead of giving every model the same importance, it used their validation performance to determine their contributions to the final prediction. Models that performed well across the selected measures received greater influence, while weaker models received lower weights.

The experimental results showed that MWEM overally performed better than the strongest individual models and the Average Ensemble. This suggests that the improvement did not come only from combining several predictions. The way in which the models were selected and weighted was also important. In the Average Ensemble, a weak prediction can have the same influence as a strong prediction. MWEM reduced this problem by linking the weights to the validation results.

Using several measures also gave a wider view of model performance. A model with a low average error could still produce some large errors or perform less well under another measure. By considering several measures, MWEM was less dependent on the result of one evaluation criterion. This provided a more balanced basis for constructing the ensemble.

Another advantage of MWEM was its relatively simple weighting process. The

weights were calculated directly from the validation results, without requiring an additional optimisation stage. This made the method easier to apply and interpret than BOEM, and it required less training time. These features may be useful when separate ensembles need to be trained for many route sections.

However, the direct weighting process also had a limitation. MWEM mainly used the individual validation performance of each model. It did not test a wide range of complete weight combinations. A model that was not one of the strongest individual predictors could still improve the ensemble when combined with other models, but direct metric-based weighting might not fully identify this contribution.

The findings are consistent with previous railway and transport studies in which model weights were determined from validation performance (Al Ghamdi et al., 2020; Boateng and Yang, 2023a,b; Al Ghamdi et al., 2024). In this research, the use of several measures extended this idea by considering different aspects of prediction performance within one heterogeneous ensemble.

Overall, the results showed that several validation measures could be used together to select and weight models. MWEM provided stronger performance than individual prediction and equal-weight averaging while keeping the weighting process simple. This provides the main evidence for RQ1. The next section examines whether Bayesian optimisation could further improve the ensemble weights.

5.12.3 Bayesian Optimisation and the BOEM Framework

MWEM showed that several validation measures could be used to build a strong weighted ensemble. However, its weights were calculated directly from the validation results of the individual models. This approach is simple and clear, but it does not search different weight combinations to find the combination that gives the best ensemble result.

The Multi-Metric Bayesian-Optimised Ensemble Method (BOEM) was developed to improve this process. BOEM uses Bayesian optimisation to search for model weights under a multi-metric validation setting. Instead of assigning weights directly from the individual model scores, it evaluates different weight combinations and identifies combinations that improve the overall ensemble performance.

This search process provides greater flexibility than direct metric-based weighting. A model that is not the strongest individual predictor may still improve the ensemble when it is combined with other models. BOEM can take this effect into account because it evaluates the combined prediction rather than considering only the separate performance of each model.

The results showed that BOEM improved on MWEM and the hybrid BOEM reduced MAE by approximately 7% and RMSE by approximately 6% compared with MWEM. Compared with the strongest individual models, it reduced MAE by approximately 8% and RMSE by approximately 9%. It also achieved stronger overall rankings across the main evaluation measures. These results show that Bayesian optimisation found more effective weight combinations than the direct weighting method used by MWEM.

The controlled weighting experiment also showed that the multi-metric BOEM configuration provided a better overall balance than the single-metric configurations. A weight search based only on MAE, RMSE, R^2, or ACC@1 could improve performance under that measure, but the result was not always as strong under the other measures. The multi-metric setting reduced this dependence on one evaluation criterion and produced a more balanced result.

The comparison between the ML-only, DL-only, and hybrid BOEM configurations provided another important finding. The hybrid BOEM achieved stronger results than both the ML-only and DL-only versions across the four main evaluation measures. This indicates that combining the complementary strengths of ML and DL models produced stronger overall performance than relying on either model group alone. The ML models, especially the tree-based methods, provided strong predictions for the structured railway data. The tabular DL models also captured useful patterns that were not always represented in the same way by the conventional ML models. When both groups were combined, their different prediction behaviour provided additional value to the ensemble. The results therefore suggest that tabular DL models are most useful in this research when they support, rather than replace, strong conventional ML models.

The statistical ranking results provided further support for BOEM. In the comparison with the reconstructed published methods, BOEM achieved the best average rank for

MAE, RMSE, and R^2. The rank difference between BOEM and the other methods, including MWEM, exceeded the Nemenyi critical difference for these measures. This shows that the leading rank of BOEM was observed across the section-fold evaluation units and was not based only on a small number of favourable results.

The performance improvements should also be considered alongside their variability across the evaluation units. Table 5.10 reports an MAE of 0.594 ± 0.443 minutes for the proposed method, compared with 0.637 ± 0.464 for MWEM, and an RMSE of 1.055 ± 0.782 compared with 1.119 ± 0.805. The corresponding R^2 values are 0.914 ± 0.095 and 0.903 ± 0.106. The lower mean errors and smaller standard deviations provide complementary evidence of accuracy and more consistent performance across the evaluated section-fold units. The section-level results and rank variability in Sections 5.11.3 and 5.11.6 further describe this consistency. Sections 5.11.7 and 5.11.11 report the Friedman and Nemenyi analyses, distinguishing overall differences between models from the specific comparisons supported by the post-hoc results.

The predictive benefits involve additional computational requirements. The proposed method requires offline training of the candidate models and an additional ensemble-weight optimisation stage, while the hybrid configuration also includes tabular DL models. These operations are completed before live prediction. Table 6.1 reports mean model-level inference times of 0.0028 milliseconds for LR, 0.0047 milliseconds for GB and 0.7665 milliseconds for the proposed method in the evaluated computing environment. The ensemble therefore requires more prediction-time computation than the simpler baselines, while its measured mean inference time remains below one millisecond in the parallel implementation. This provides a concrete accuracy–computation trade-off: improved prediction errors, the reported consistency across evaluation units and stronger performance under missing departure information are obtained with additional offline computation and sub-millisecond mean model-level inference. The timing results concern the measured inference stage; the live evaluation provides complementary evidence of the complete prototype operating with railway updates.

Previous railway ensemble studies have often used fixed structures, equal weighting, or weights based mainly on one performance measure (Al Ghamdi et al., 2020, 2024;

Zhou, 2025). BOEM extends these approaches by searching for ensemble weights using several validation measures and rounds. Similar benefits from Bayesian ensemble optimisation have also been reported in other prediction fields (Shahhosseini et al., 2022; Wadsworth and Niemi, 2025).

Bayesian optimisation produced more effective weight combinations than the direct weighting used in MWEM. The hybrid configuration also performed better than the ML-only and DL-only configurations, which supports the use of both model groups in the final ensemble. These findings provide the main evidence for the second and third research questions.

5.12.4 Comparison with Published Train Delay Prediction Methods

A direct comparison with the numerical results reported in previous studies would not provide a fair evaluation. Earlier studies used different railway networks, datasets, prediction targets, input features, delay definitions, sample sizes, and validation methods. Their reported results therefore depend on both the proposed method and the data used in each study. For this reason, the comparison in Section 5.11.11 did not use the reported results from different papers directly. Instead, selected methods from previous train delay prediction studies were reconstructed and evaluated using the same datasets and experimental procedure used in this research.

Seven method configurations from five published studies were included in the comparison. These were BO-XGBoost (Shi et al., 2021), ELM-PSO (Bao et al., 2021), MSM1-WE, MSM2a-COR-WE, and MSM2b-CFD-WE (Al Ghamdi et al., 2024), ST-NN (Boateng and Yang, 2023a), and Markov-GK (Xu et al., 2023). These methods were selected because their main algorithms and training procedures were described in enough detail to allow them to be adapted to the route-section prediction task used in this research.

The reconstructed methods were evaluated using the same NL and WW datasets, prediction target, test folds, and evaluation measures. This provided a more controlled comparison than using results taken directly from different papers. It reduced the effect of differences in data and evaluation design and allowed the methods to be compared

under the same conditions.

However, the comparison should not be treated as an exact reproduction of the original studies. Complete source code, preprocessing steps, and all model settings were not available for every method. The implementations were therefore based on the equations, model descriptions, hyperparameters, and experimental details reported in the papers. The results show how these methods performed after adaptation to the datasets and prediction setting of this research.

Under these common experimental conditions, both MWEM and BOEM achieved stronger combined performance than the reconstructed published methods. BOEM achieved the strongest mean result across the reported evaluation measures and obtained the best average rank for MAE, RMSE, and R^2. Compared with ST-NN, which was the strongest reconstructed method for the overall error measures, BOEM reduced MAE by approximately 8% and RMSE by approximately 7%.

The statistical analysis also supported the leading performance of BOEM. Its rank difference from the reconstructed methods exceeded the Nemenyi critical difference for MAE, RMSE, and R^2. This indicates that its stronger ranking was observed across the section-fold evaluation units and was not caused only by a small number of favourable results.

The reconstructed methods represent different approaches to train delay prediction. BO-XGBoost and ELM-PSO optimise the settings of an individual model. ST-NN combines neural model predictions through a stacking process. Markov-GK represents delay changes through transition states. The MSM methods select heterogeneous models using accuracy or diversity measures. BOEM differs from these methods because it combines a wider group of ML and tabular DL models and searches for their weights using several validation measures. The results suggest that this wider model pool and the multi-metric weight search provided a better balance across the main evaluation measures in the prediction setting used in this research.

Several other published studies were not reconstructed. Some studies mainly applied standard models such as RF, GB, XGB, or neural networks (Li et al., 2021b; Taleongpong et al., 2022; Lapamonpinyo et al., 2022). These model families were already included in the individual model pool of this research. Reconstructing these studies as separate

methods would therefore have repeated models that had already been evaluated. Other studies depended on input information that was not available in the same form in the NL and WW datasets. For example, Zhang et al. (2021) used a specialised delay-propagation feature, while Sarhani and Voß (2024) used position and weather data. The main value of these studies is linked to their data and feature design. Removing those inputs would have changed the original methods and would not have provided a fair comparison. The study by Borin et al. (2026) was also not included as a separate comparator. Its main contribution was the development of an integrated operational and weather dataset, while XGBoost was used as a prediction baseline. Using that dataset would have changed the railway network, prediction target, and data conditions. It would therefore not have provided a controlled method comparison for the route-section task used in this research.

Graph-based and sequence-based methods were excluded for a different reason. Sequence models require ordered histories of several train events, while graph models require a wider network representation. This may include station connections, train interactions, simultaneous train states, and different node and edge relationships (Huang et al., 2020b; Li et al., 2021a; Tang et al., 2025; Li et al., 2024a,b). The historical datasets in this research were prepared as independent station-to-station records. They did not contain a complete simultaneous network state or the continuous event sequences needed by these models. Reconstructing these inputs would have required a different data structure, prediction target, and experimental design. Their exclusion was therefore based on a difference in the research problem rather than an assumption that graph or sequence models are not useful.

Overall, this comparison provides a wider evaluation of the proposed ensemble methods. It shows that MWEM and BOEM performed strongly against several optimisation, stacking, probabilistic, and heterogeneous ensemble approaches under the same experimental conditions. However, the findings are limited to the datasets, route-section prediction task, and reconstructed implementations used in this research.

5.12.5 Robustness, Generalisation, and Operational Cost

The main model comparisons showed how the methods performed under the standard historical prediction setting. However, a model used in practice may also face missing values, different route conditions, and errors with different operational effects. For this reason, the additional experiments examined robustness, cross-route transfer, and asymmetric prediction cost.

The missing-value experiment examined how the models performed when departure-delay information was unavailable and had to be estimated from preceding-section predictions. This experiment used the BOEM-ML configuration and evaluated 18 sections of the WW route. Ten missing-value levels were considered: 0%, 2%, 4%, 6%, 8%, 10%, 20%, 30%, 40% and 50%. The same test observations were treated as missing for all compared models at each level.

The results in Section 5.11.9 showed that prediction error increased as more departure information had to be estimated. However, the proposed method maintained the lowest mean MAE across the evaluated levels. Its MAE increased from 0.617 minutes with complete departure information to 0.692 minutes at 50% missingness, as reported in Table 5.6. Its mean degradation measure was also the lowest among the compared models. The paired Wilcoxon comparisons supported its slower deterioration across the evaluated route sections.

These findings provide evidence that the ensemble was less sensitive to missing departure information and the resulting propagation of estimation errors than the compared individual models. They apply to the specific reconstruction procedure, route sections and missing-value conditions evaluated in this experiment. They do not establish robustness to every form of incomplete or unreliable live railway data.

The cross-route experiments examined whether models trained on one route could remain useful when evaluated on the other route. These experiments also used the BOEM-ML configuration. The NL and WW historical datasets covered different years and different durations, with six months of data for NL and two years for WW. The two routes also differ in length, number of route sections, and delay behaviour. Therefore, transferring a model between them provided a more difficult test than evaluating it on later records from the same route. The proposed method remained competitive in

both transfer directions. When trained on NL and evaluated on WW, it preserved useful prediction performance on the longer and more varied route. When trained on WW and evaluated on NL, it also maintained strong behaviour on the shorter route. These results suggest that the ensemble learned some delay relationships that were shared between the two routes. This shows that the proposed method transferred more accurately and consistently across the evaluated section transfers in both directions.

The transfer results are useful because a new route may not always have a large amount of clean historical data for model training. A model that retains useful performance across routes may reduce the amount of route-specific data needed at the early stage of model development. However, the results do not show that one model can be applied to any railway route without retraining.

The cross-route results show that the proposed method retained useful performance when trained on one route and evaluated on the other. This suggests that some delay patterns may be shared across routes with different lengths and operating conditions. Previous studies have mainly focused on prediction within one route or dataset, while less attention has been given to incomplete inputs and cross-route evaluation (Spanninger et al., 2022; Tiong et al., 2023b). The experiments in this research provide additional evidence under these two conditions. Further evaluation on more routes would help to examine how widely these findings can be applied.

The cost-based evaluation examined another practical issue. Standard MAE and RMSE treat underprediction and overprediction in the same way. However, underpredicting a train delay may have a greater operational effect because it can provide passengers or railway staff with an overly optimistic estimate. For this reason, the asymmetric cost analysis applied a higher penalty to underprediction than to overprediction. BOEM achieved the lowest average cost and improved by approximately 7–9% compared with strong individual models such as GB, XGB, TabICL, and RF. This shows that the advantage of BOEM remained when the direction of the prediction error was also considered.

The underprediction and overprediction rates also showed that BOEM did not reduce cost only by producing larger delay estimates. This indicates that the model did not always overpredict delays to avoid the higher underprediction penalty. The cost analysis

provides a useful operational view of model performance. A model with a slightly lower average error may not always be the best choice if it frequently underestimates important delays. Considering both the size and direction of the error therefore gives a wider view than using MAE or RMSE alone. At the same time, the cost function used in this research represents one possible penalty setting. Different railway organisations may place different costs on underprediction and overprediction. The numerical cost results should therefore be interpreted within the penalty values used in this research.

Overall, the additional experiments showed that the proposed approach remained useful under more difficult conditions. The proposed method showed robustness to simulated missing values in input data, transferability between the two selected routes, and achieved the lowest asymmetric operational cost. These results provide the main evidence related to the fourth research question. However, broader evaluation would be needed before making conclusions about generalisation across other routes or railway networks.

5.13 Answers to Research Questions 1–4

The historical evaluation provides the following answers to the first four research questions introduced in Chapter 1. The answers are interpreted within the datasets and experimental conditions evaluated in this chapter.

RQ1. Can multiple performance metrics be used to select and weight models to improve the accuracy of a heterogeneous ensemble?

Yes. In this research, multiple validation metrics were successfully used to select models and determine their contributions within heterogeneous ensembles. MWEM assigned greater weights to models with stronger performance across several measures and reduced the influence of weaker models. It performed better than the strongest individual models and the Average Ensemble, showing that multi-metric weighted aggregation improved prediction accuracy. BOEM extended this approach by evaluating complete weight combinations and assigning each model a weight according to its contribution to the combined validation prediction, even when it was not the strongest individual predictor. A weighting strategy experiment showed that the multi-metric BOEM configuration achieved the best overall performance across all evaluation metrics

compared with the corresponding single-metric configurations. These findings show that using complementary performance measures provides a more accurate, reliable, and balanced basis for model selection and weighting than relying on a single metric. The supporting evidence is presented in Sections 5.11.1 and 5.11.2.

RQ2. How can a multi-metric ensemble be optimised to improve train delay prediction?

In this research, a multi-metric ensemble was optimised by using Bayesian optimisation to search for effective combinations of model weights during repeated validation rounds. Within each validation round, candidate weight vectors were evaluated using one predefined performance metric. Different metrics were used across the rounds, and the optimised round-specific weight vectors were averaged to obtain the final ensemble weights. The results from the validation rounds were then used to determine the final model weights. Compared with the strongest individual model for each error measure, BOEM reduced MAE by approximately 8% and RMSE by approximately 9%. BOEM also achieved stronger average rankings across the selected routes, route sections, and evaluation measures. These findings demonstrate that Bayesian optimisation can improve multi-metric ensemble weighting and train delay prediction under the datasets and experimental settings used in this research. The supporting evidence is presented in Sections 5.11.1, 5.11.6 and 5.11.7.

RQ3. Does a hybrid machine learning–deep learning ensemble perform better than machine learning-only and deep learning-only ensembles?

In this research, the hybrid BOEM performed better than both the machine learning-only BOEM-ML and the deep learning-only BOEM-DL ensembles. It achieved stronger performance across all four main evaluation measures. The ranking analysis also placed BOEM first for all evaluation metrics across the 26 route sections, with low variability in its rankings. These findings show that the ML and tabular DL models provided complementary predictive information. Combining both model groups within the hybrid ensemble therefore produced more accurate and consistent train delay predictions than using either model group alone. The comparison between the three model-pool configurations is reported in Table 5.4.

RQ4. Does the proposed ensemble provide more reliable performance

than other models across different railway routes and evaluation settings?

Within the routes, datasets, and evaluation settings considered in this research, the proposed method provided more reliable performance than the compared models. Under simulated missing values experiment, it achieved lower average error and variation than the individual models, while its error increased more slowly as the amount of missing data increased. In cross-dataset testing, it also achieved better results across both transfer directions. These findings support reliability; however, evaluation on additional routes and railway systems is required before network-wide generalisation can be established. The supporting evidence is presented in Sections 5.11.3, 5.11.9 and 5.11.10.

5.14 Summary

This chapter developed and evaluated an ensemble framework for route-section-level train arrival delay prediction. It combined conventional machine learning and tabular deep learning models within heterogeneous and hybrid ensemble configurations. AE provided an equal-weight baseline, MWEM used multi-metric validation results and greedy model selection, and BOEM used Bayesian optimisation to search for effective combinations of model weights.

The historical experiments compared these methods with individual models and reconstructed published approaches. Hybrid BOEM achieved the strongest overall mean performance across the four main evaluation measures. The additional analyses examined whether these improvements were maintained across route sections and delay magnitudes, and considered statistical rankings and asymmetric prediction cost. The interpretation of statistical differences depended on the particular comparison and evaluation units used.

The weighting-strategy, missing-departure and cross-dataset experiments used BOEM-ML to examine specific properties of the weighting mechanism. Their results provided evidence of balanced multi-metric performance, slower deterioration under missing departure information and useful transfer between the two evaluated routes. Together, the results and discussion addressed RQ1–RQ4 within the scope of the historical datasets and experimental conditions.

Chapter 6 extends the investigation to time-aware model preparation and prediction using live railway updates.

Chapter 6

Real-Time Ensemble Framework for Train Delay Prediction

6.1 Introduction

The ensemble framework developed in Chapter 5 was evaluated using prepared historical railway datasets. Its application to real-time prediction requires railway information to be processed while a train journey is still in progress. In this setting, live information may be incomplete, revised or received at different stages of the journey. It must therefore be organised and transformed into the route-section and feature structure required by the trained models. Previous studies have shown that current train information and historical delay patterns can support predictions for later stations. Lapamonpinyo et al. (2022) identified the most recently observed delay as an important predictor of future delay. Similarly, Wen et al. (2020) described real-time delay prediction as a process in which predictions are updated when new operational information becomes available.

This chapter presents the development and evaluation of the real-time prediction framework. It explains the processing of live updates, construction of model inputs, estimation of unavailable departure information, selection of trained ensembles, service-level prediction correction and recursive downstream prediction. It then presents the time-aware evaluation of models prepared for real-time use and the evaluation of predictions generated from live railway updates. The discussion interprets these results and answers RQ5.

The framework is a research prototype evaluated using live operational railway data. Its evaluation provides evidence of prediction performance and technical feasibility under the observed conditions, rather than industrial validation or operational adoption by a

railway operator. Where model deployment is discussed, the term refers to integrating trained models and their associated artefacts into this research prototype. Base-model training and ensemble-weight optimisation are completed offline; the framework performs real-time inference rather than online retraining of these models.

The chapter proceeds as follows. Section 6.2 introduces the framework and prototype dashboard. Section 6.3 describes the system architecture, and Section 6.4 explains live-data preparation. Section 6.5 presents departure-delay estimation. Section 6.6 explains the preparation and application of BOEM. Sections 6.7 and 6.8 describe service-level correction and recursive prediction. Sections 6.9 and 6.10 discuss implementation requirements and practical challenges. Section 6.11 presents time-aware model evaluation, and Section 6.12 reports the live prediction results. Section 6.13 discusses the findings, and Section 6.14 answers RQ5. Finally, Section 6.15 summarises the chapter.

6.2 Overview of the Real-Time Framework and prototype dashboard

Before describing the architecture and implementation of the real-time framework, this section provides a high-level view of its main stages and shows how the resulting predictions are presented during live operation.

As illustrated in Figure 6.1, the framework follows six main stages. It receives live railway updates, organises the information by train service, constructs prediction records for the relevant route sections, generates the required online features, applies the appropriate trained ensemble model, and stores the resulting predictions. These stages allow the models developed from historical data to be applied to the latest available information during an active train journey.

To provide a clearer understanding of the framework and demonstrate how the models developed in Chapter 5 can be integrated and applied in a real-world setting, a live prototype dashboard was developed. The dashboard also provides an environment for testing the proposed framework using live railway data and observing its behaviour during active train journeys. Also, the prototype dashboard provides a practical view of the outputs produced by this process. As shown in Figure 6.2, it presents the current

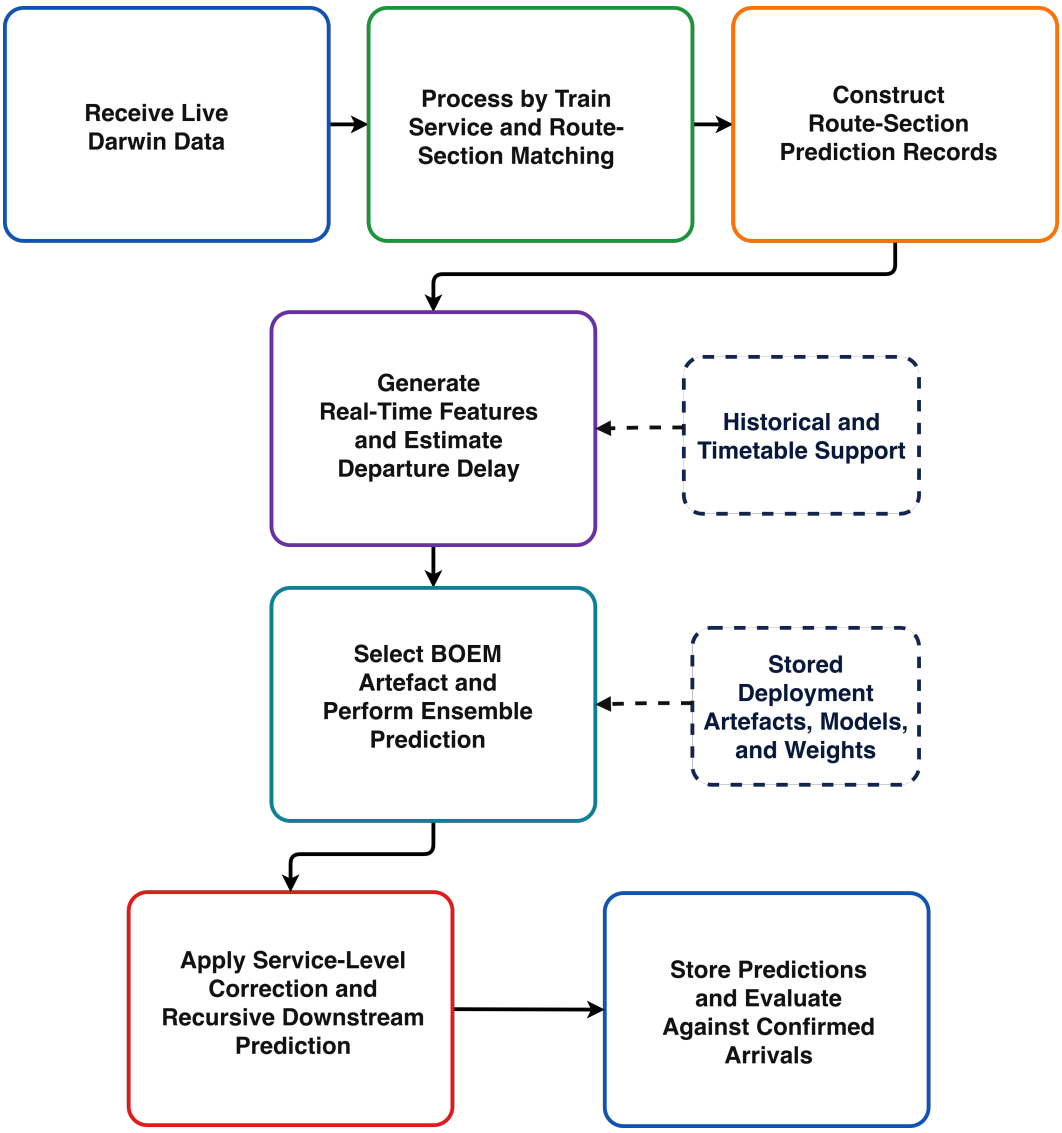


Figure 6.1: Main stages of the proposed real-time train delay prediction framework.

train services, predicted arrival delays, route sections, and downstream predictions within a single interface. It also distinguishes between predictions based on confirmed departure information and those produced using estimated departure information.

The upper part of the dashboard provides summary information about the monitored services, while the map shows the route sections where predictions are being generated. Filters allow the user to examine a particular route, section, train service, or delay condition. The prediction tables provide further information about individual services, including the departure information used, the selected prediction strategy, and the predicted arrival delay.

For services with confirmed departure information, the dashboard can also display predictions for later sections of the journey. This provides an overview of how the current delay may continue along the remaining route. The dashboard therefore supports the monitoring and interpretation of predictions produced by the real-time framework. A detailed description of the fields displayed in the dashboard prediction tables is provided in Appendix 8.6. The dashboard is introduced here only to provide an overall view of the framework in operation. Further details of the framework and its implementation are presented in the following sections.

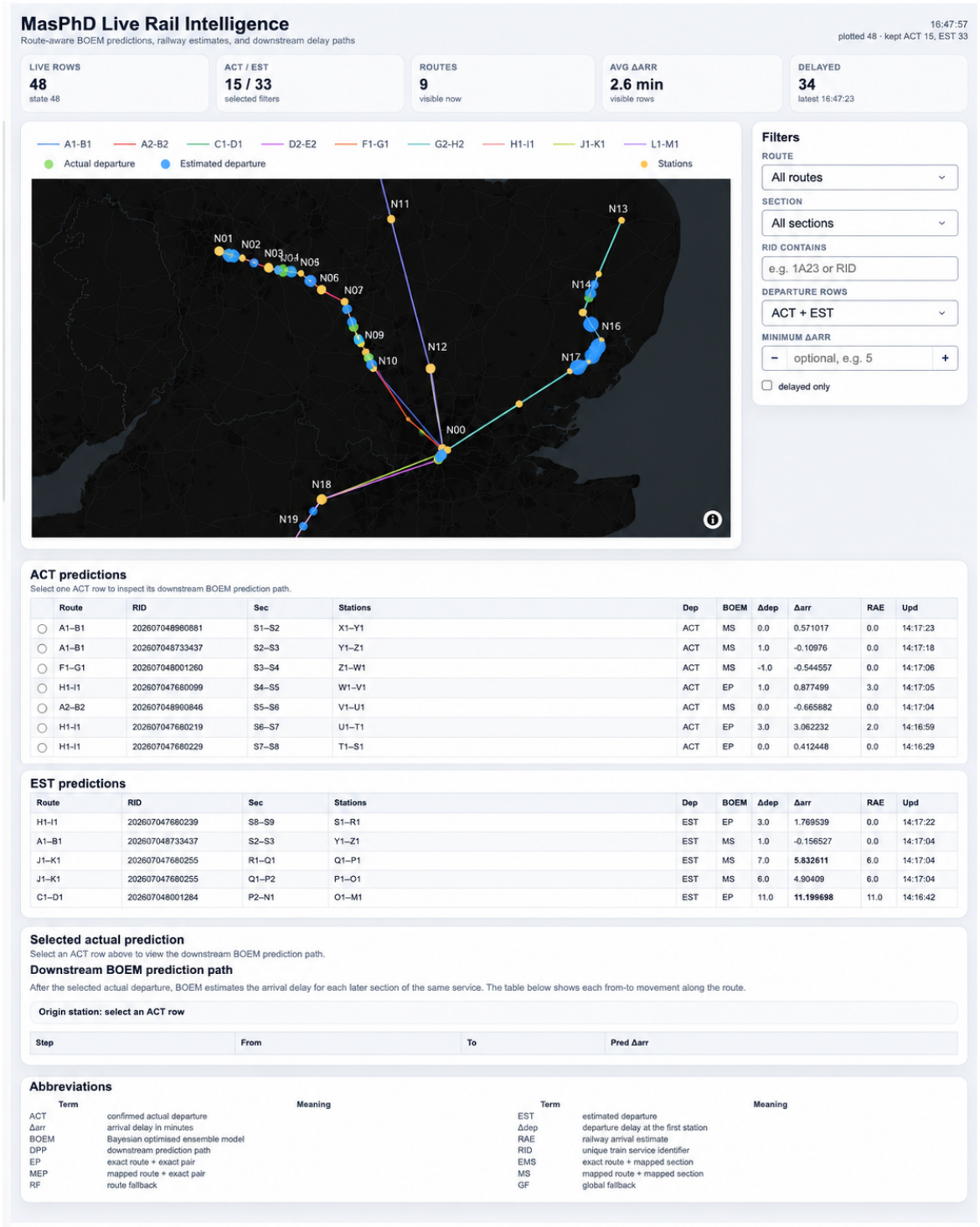


Figure 6.2: Prototype dashboard for monitoring real-time BOEM-based train delay predictions.

6.3 System Architecture of the Real-Time Prediction Framework

As shown in Figure 6.3, the architecture consists of three connected components: the offline modelling environment, the online prediction environment, and the operational knowledge base. The offline environment prepares the trained models and ensemble weights before deployment, while the online environment processes live railway information and applies the trained models during active train journeys. The operational knowledge base, developed as part of this research, maintains an organised and continuously updated representation of train services, operational information, model inputs, and prediction outputs. In other words, the system architecture and operational workflow present two views of the proposed framework. The three architectural components show where the main activities are performed, while the six operational stages show how live railway information moves through the system.

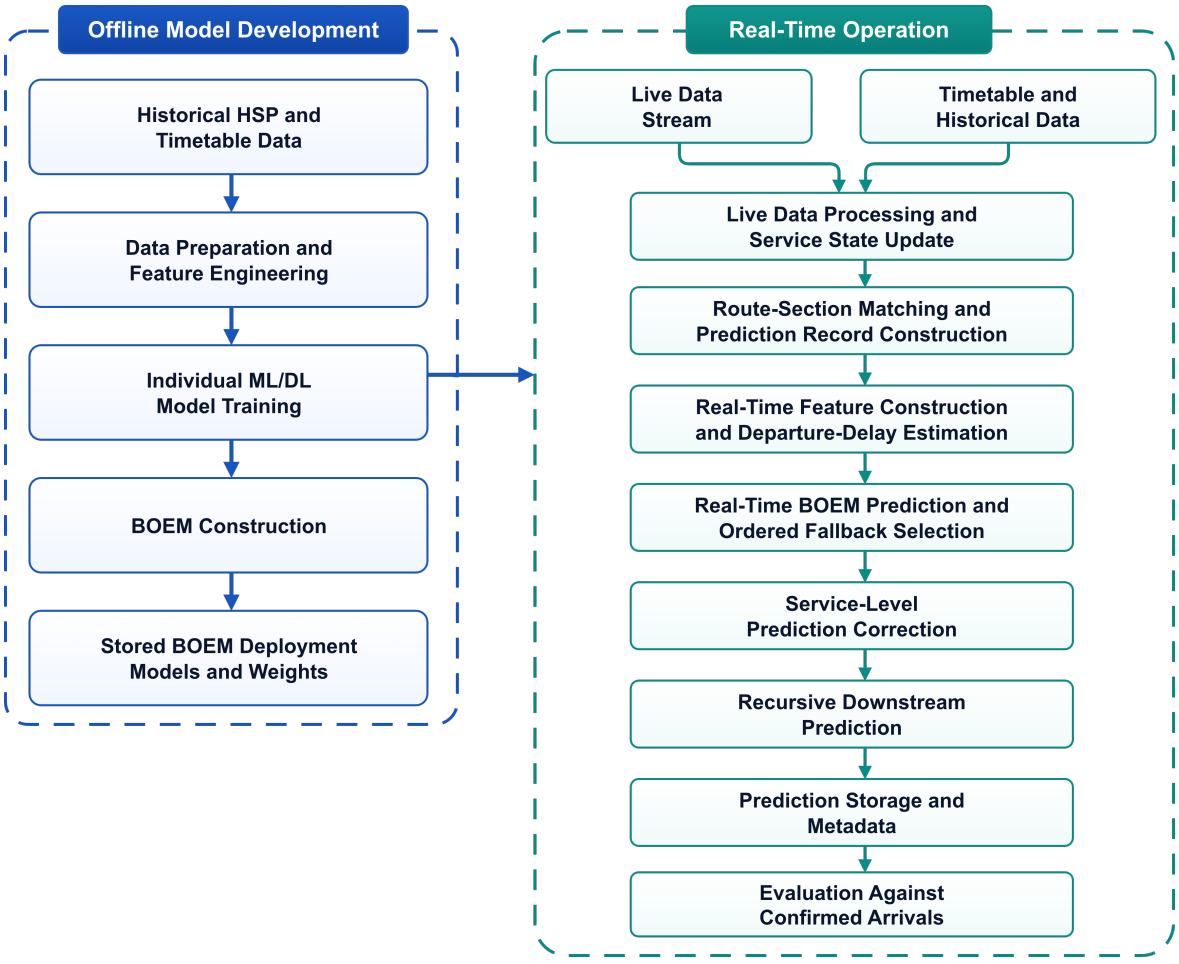


Figure 6.3: System architecture of the proposed real-time train delay prediction framework.

6.3.1 Offline Modelling Environment

The offline modelling environment is responsible for preparing the predictive components used by the real-time system. Historical railway data are first processed into the route-section structure required by the models. The individual models are then trained, and the ensemble models and their corresponding weights are obtained for each route section.

The trained models, ensemble weights, feature definitions, and other information required during inference are stored as model artefacts. These artefacts are prepared before the live system begins operation and are subsequently loaded when a prediction is required. Therefore, model training and ensemble-weight optimisation are not performed whenever a new live update is received.

Separating model development from live prediction is important because training and optimisation can require considerable computational resources. In contrast, the real-time system must produce predictions quickly and repeatedly for active train services. Preparing the models offline allows the online environment to focus on data processing and inference while still using the delay patterns learned from historical railway operations.

6.3.2 Real-Time Prediction Environment

The real-time prediction environment performs the main operational stages introduced in the previous section. It receives live railway messages and organises the extracted information by train service. Since these messages differ from the tabular records used during offline model training, they cannot be passed directly to the trained models.

A transformation process is therefore required between the live data stream and the prediction models. This process extracts the relevant service and timing information, resolves the station identifiers used by different railway data sources, identifies the corresponding route and station-to-station section, and constructs a structured prediction record. The required online features are then generated using the latest available information about the train service.

Once the feature vector has been prepared, the system selects the trained ensemble model associated with the relevant route section. The selected model and its stored

ensemble weights are applied to generate an arrival delay prediction for the next station. Where required, the prediction can also be extended to later sections of the journey. The online environment may additionally apply estimation and fallback mechanisms when some live information or model artefacts are unavailable. These mechanisms are described in the following sections.

6.3.3 Operational Knowledge Base and State Management

The operational knowledge base provides persistent storage and state management for the real-time prediction framework. Rather than serving only as a repository for final prediction outputs, the database maintains the information required to process successive updates for the same train service. It therefore acts as a working operational knowledge base that connects live data processing, feature generation, prediction, adjustment, and downstream propagation.

Each stored record is linked to a train service, service date, route, and station-to-station section. The database retains the latest available timing information, including planned, estimated, and confirmed movement records, together with the feature values and prediction metadata used during inference. This structure allows the system to distinguish between predictions generated from confirmed information and those based on estimated or fallback values.

State management allows the framework to maintain a consistent view of each active train service as new information becomes available. Live railway messages may provide several updates for the same service, and an estimated value may later be replaced by a confirmed observation. The storage layer therefore updates the current service context using newer or more reliable information while retaining earlier prediction records for traceability. The stored information also supports later stages of the real-time process. Confirmed observations and previous predictions from the same journey can be retrieved when estimating unavailable inputs, applying service-level prediction adjustment, or generating predictions for downstream sections. In this way, the database supports information exchange between different stages of the framework rather than acting only as an output archive. Prediction outputs are stored together with their associated route section, input-information source, selected model strategy, predicted delay, update time,

and other relevant metadata. This provides a traceable record of how each prediction was produced and allows the outputs to be accessed by the prototype dashboard and evaluated against confirmed arrival observations. The following sections describe the practical implementation of the framework, beginning with the preparation of live data for real-time prediction.

6.4 Live Data Preparation for Real-Time Prediction

As discussed in Chapter 4, real-time railway information available in operational environments is not provided in the model-ready tabular format used during offline model development. Instead, it is received as a continuous sequence of messages containing planned, estimated, and confirmed timing information for different train services and calling points. These updates may be incomplete, revised over time, received out of station order, or represented using identifiers that differ from those used in the historical datasets and trained model artefacts. Preparing this information for real-time prediction therefore requires several connected stages. The incoming messages must first be received, parsed, and restricted to the routes and services considered in this research. Successive updates must then be organised to maintain the latest state of each active train journey. This service-level information is subsequently transformed into station-to-station records, from which the operational, temporal, and historical features required by the trained models are constructed. Where confirmed information is not yet available, suitable estimation and fallback procedures are applied to complete the required model inputs.

The following subsections describe these preparation stages. The section begins with the reception and filtering of live railway updates, followed by service-state management and route-section record construction. It then explains online feature construction and the estimation of values that are unavailable at the prediction time.

6.4.1 Live Railway Data Acquisition and Filtering

The framework receives operational updates from the Darwin PushPort data stream through a continuous STOMP connection (National Rail Enquiries, 2023). Once the connection is established, newly published messages are received, decompressed, and

parsed to extract the available service, location, and timing information. Depending on the current state of a journey, these messages may contain planned, estimated, or actual arrival and departure times for one or more calling points. The live information is used together with timetable and station-reference data. Timetable records provide the planned service pattern, including the origin, destination, calling sequence, and scheduled timings. Station-reference data are used to align the identifiers found in the live stream with those used in the route definitions, historical records, and trained model artefacts.

The Darwin stream covers services across the wider railway network, while the framework considers only the routes and directions selected for this research. The received updates are therefore checked against the timetable and predefined route information. This filtering considers the service date, origin and destination, direction of travel, and relevant calling points. Messages relating to unrelated services, unsupported directions, or stations outside the selected routes are excluded. The remaining updates are forwarded to the service-state management stage. At this point, each message has been parsed and matched to a relevant train service, but successive updates for the same journey have not yet been combined. This process is described in the following subsection. An example of a raw message received through the STOMP connection is provided in Appendix 8.5.

6.4.2 Service-State Management

Live railway information is received as a sequence of updates rather than as a complete record for each train journey. The framework therefore maintains an updated service state for every relevant active train service. Messages are associated with a service using its real-time identifier and service date, allowing information received at different times to be linked to the same journey.

Several updates may be received for the same service and calling point. An early message may contain an estimated arrival or departure time, while a later message may revise that estimate or replace it with a confirmed actual value. The framework updates the stored service state using the most recent and reliable information available. Confirmed observations are given priority over estimated values, while older or repeated

updates are prevented from replacing newer records.

The order in which messages are received may also differ from the actual station sequence of the journey. For example, information for a later calling point may arrive before an update for an earlier station. The framework does not require all messages to arrive in route order. Instead, each update is matched to its service and calling point and incorporated into the current service state. When additional information becomes available, the corresponding service record is updated accordingly.

This process allows the framework to maintain the latest known arrival, departure, and timing status for each active journey without treating every message as an independent observation. The resulting service-level representation provides the operational context required to identify the relevant station-to-station movements and construct route-section prediction records in the next stage.

6.4.3 Route-Section Record Construction

The trained models used in this research operate at route-section level, whereas the maintained service state represents the wider sequence of calling points within a train journey. The service-level information must therefore be transformed into individual station-to-station records before the required model features can be generated. Each constructed record represents the movement from a current station to the next station on one of the predefined routes.

For each relevant train service, the framework compares the calling sequence obtained from the live and timetable information with the route definitions used during offline model development. Consecutive calling points are identified and matched to the corresponding route section. This process determines the first and second stations of the section, its direction of travel, and the route-specific section identifier required to locate the appropriate model artefacts.

Station-identifier alignment is also performed at this stage. The live stream, timetable records, historical data, and model artefacts may use different identifiers for the same station. These identifiers are therefore resolved to a consistent station representation before a station pair is accepted. This prevents a valid movement from being incorrectly excluded or matched to an unrelated route section because of differences in station

coding.

Once the route section has been identified, the framework extracts the main information required to describe the movement. This includes the train service identifier, service date, route and section identifiers, first and second stations, planned departure time from the first station, planned arrival time at the second station, and the latest available operational timing information. The record also indicates whether the departure information is confirmed, estimated, or currently unavailable.

A route-section record is passed to the real-time feature-construction stage only when the service direction and station pair can be matched to a supported route section and the required timetable context is available. When confirmed departure information is missing, the record is estimated using the procedures described later in this section. The output of this stage is a structured route-section record that links the latest service state to the station-to-station formulation used during model development. This record forms the basis for constructing the operational, temporal, and historical features required by the deployed prediction models.

6.4.4 Real-Time Feature Construction

After a route-section record has been constructed, the framework converts its operational and contextual information into the feature representation required by the deployed prediction models. This stage connects the live service state with the section-level feature structure used during offline model development.

A key requirement of real-time feature construction is consistency with the training data. The deployed models expect the same feature definitions, data types, ordering, and section-level interpretation used during model training. The framework therefore applies the same general transformation rules to the live records before prediction. This ensures that each input variable has the same operational meaning in both the offline and real-time environments.

The online feature vector combines information from several sources. Current operational values are obtained from the latest service state and include the available delay and timing information for the current station and route section. Temporal variables are derived from the service date and planned or observed timing information.

Historical context information is obtained from stored operational records and describe typical delay behaviour for the relevant station, section, and temporal context.

Not all required values are necessarily confirmed when a prediction is produced. The framework therefore constructs each feature using the strongest information available at that time. Confirmed operational values are used whenever possible. When these values are unavailable, estimated or derived information from the current journey may be used, followed by historical evidence or planned timetable values where necessary. The source of such values is retained so that predictions based on confirmed and estimated information can be identified separately.

The output of this stage is a model-ready feature vector associated with a specific train service and route section. It represents the latest available operational condition of the train together with its temporal and historical context. Confirmed values remain the preferred source, while estimated, historical, and planned values are used only where necessary. The most important instance of this process concerns departure information and is described separately in the following section.

- **Operational Features and Information** Operational features represent the latest available condition of the train at the beginning of the route section. These features are constructed using the current service state, planned timetable information, and confirmed or estimated movement times associated with the train journey. The departure delay at the first station is a central operational feature because it describes the delay carried by the train as it enters the section. When a confirmed departure time is available, the departure delay is calculated directly from the difference between the actual and planned departure times. When it is unavailable, the framework estimates the required value using the hierarchical procedure described later in this section. Other required operational values, such as dwell delay, are also calculated from confirmed information where possible or estimated using the available journey, timetable, and historical information. These values provide context about the current running condition of the train before it enters the route section.

- **Temporal Features** Temporal features are generated from the service date and

the relevant timetable or operational time. They represent factors such as the hour of operation and the day of the week, which allow the deployed models to account for variations associated with different operating periods. These variables are constructed using the same definitions and formats applied during offline model development. Together, the operational and temporal features provide a representation of the train's current running condition and the time context in which the prediction is made.

- **Historical Information for Estimation and Correction** Historical information supports the construction of operational features when live values are unavailable, incomplete, or not yet confirmed. It is not treated as a separate group of historical model features such as last checkpoint delay or mean previous delay. Instead, stored operational records are used to estimate or correct required inputs, particularly departure delay and dwell delay. The historical records are selected according to the available operational context, such as the route, station, route section, day type, and time of operation. This allows the estimated values to reflect conditions that are relevant to the current train service rather than using a single general value for all services. Historical information is used only when stronger service-specific information is unavailable. Confirmed operational observations remain the preferred source, followed by information derived from the current journey. Historical estimates and planned timetable values are then used where necessary to complete the required feature vector. The resulting operational values are combined with the temporal variables to form the complete online feature vector. This vector represents the latest available state of the train, while historical information supports the estimation and correction of operational inputs that cannot be obtained directly from the live data.

6.5 Departure Delay Estimation for Real-Time Prediction

The route-section models used in this research estimate the arrival delay at the next station using the operational state of the train at the current station. The departure delay from the current station is therefore an important input because it represents the

latest delay carried by the train as it enters the predicted section.

In historical datasets, the actual departure time is available after the train service has operated. During live operation, however, the departure event may not yet have been confirmed when a prediction is required. A train may already have arrived at the current station while its departure time remains unavailable, or the departure update may be received later than the corresponding arrival update. Waiting for the confirmed departure would delay the next-station prediction, while assuming zero delay could incorrectly represent the current condition of the train.

The framework therefore estimates the required departure information using the strongest evidence available at the time of prediction. Information from the current train journey is prioritised, followed by historical evidence and, only when no stronger source is available, the planned timetable. This is treated as an operational estimation process rather than as general missing-value imputation, because the estimated value represents the developing state of an active train service. The estimation procedure follows an ordered hierarchy and records the source used for each value. This allows predictions based on confirmed departure information to be distinguished from those based on arrival evidence, same-service propagation, historical behaviour, or planned-time fallback.

6.5.1 Rationale for Estimating Departure Delay

The rationale for estimating departure delay comes from the practical nature of real-time railway prediction. The system must often generate a prediction while the service is still moving through the network. At that point, not all timing information is available. This is particularly important in station-to-station prediction, because the framework needs to estimate the arrival delay at the next station as early as possible, not only after the train has already departed from the current station.

The departure delay at the first station of a section is one of the most informative indicators of the train's current condition. A train that departs late from the current station is more likely to arrive late at the next station, although the final delay may change depending on running time, dwell time, timetable recovery, and route characteristics. For this reason, departure delay forms a direct operational link between the current state of the train and the predicted arrival delay. However, in real-time

operation, the departure delay is not always directly observable. This does not mean that the current state of the train is completely unknown. For example, the train may already have an actual arrival time at the first station. Since the first station of the current section is also the end point of the previous section, the arrival delay at this station provides valuable evidence about the train's current delay state. Therefore, the framework first checks whether actual arrival information is available at the current station before moving to weaker estimation sources.

This is an important design choice. A missing departure time should not immediately be treated as zero delay or as a purely historical average. If the train has already arrived late at the current station, this information should influence the estimated departure delay for the next section. Similarly, if the system has already produced or stored information for the same train service at an upstream section, this evidence can be used to infer the current delay state. The proposed approach therefore uses the available operational evidence in an ordered and meaningful way. Departure delay estimation also improves consistency between offline and online prediction. The offline models are trained using features such as departure delay and dwell delay values. During live operation, these features must be reconstructed as accurately as possible from the information available at the time of prediction. The estimation stage provides a transparent and structured way to construct these inputs without assuming that all actual values are already known.

6.5.2 Hierarchical Estimation Strategy

The proposed departure delay estimation follows a hierarchical strategy. The hierarchy is based on a simple principle: use the most direct and reliable source first, and only move to less direct evidence when the stronger source is unavailable. This makes the estimation process practical, interpretable, and suitable for real-time operation. Figure 6.4 illustrates the hierarchical departure delay estimation strategy. The framework prioritises actual departure information, then actual arrival evidence at the first station, followed by same-service propagation, historical station-level behaviour, and finally a planned-time fallback.

The first level of the hierarchy uses the confirmed actual departure time from the

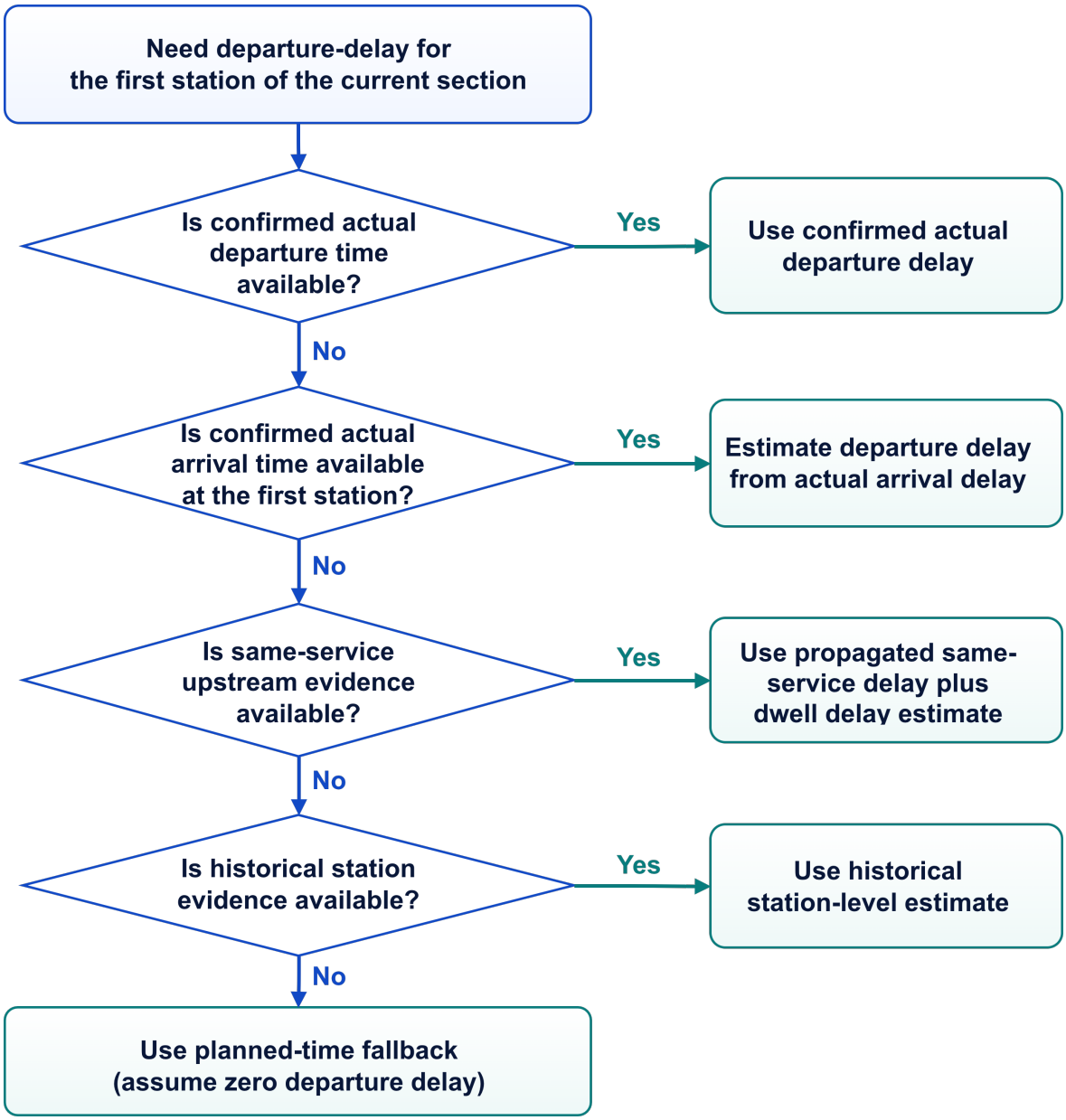


Figure 6.4: Hierarchical departure delay estimation strategy used when actual departure information is unavailable.

first station of the section. When this value is available, no estimation is required. The departure delay is calculated by comparing the actual departure time with the planned departure time. This is the most reliable case because it directly describes the train leaving the station. If the actual departure time is not available, the second level checks whether actual arrival information is available at the first station. This is a key part of the proposed framework. The first station of the current section is usually the station where the train has just arrived from the previous section. Therefore, the actual arrival delay at this station gives a strong indication of the current delay state of the train. If the train has arrived late, the departure delay for the next section should not be treated as missing or automatically set to zero. The framework uses this arrival-based evidence to estimate the departure delay required for prediction.

If arrival-based evidence at the first station is not available, the third level uses same-service propagation. In this case, the system uses information from previous sections of the same train journey. A previously predicted or observed delay from an earlier section of the same train journey can provide evidence about the likely delay at the current station. This propagated delay can be combined with dwell behaviour to estimate the departure delay from the first station of the current section. This reflects the operational fact that delays often move through a journey from one section to another. If same-service propagation is not available, the fourth level uses historical station-level behaviour. This means that the framework estimates the likely departure delay using previously observed services at the same station and in a similar temporal context. This provides a data-driven estimate when there is no reliable current-service evidence.

The final level is a planned-time fallback. If none of the previous sources can provide a usable value, the framework uses the planned departure time and assumes zero departure delay. This is the weakest source of evidence, but it keeps the real-time pipeline operational when the available data is very limited.

This ordered structure avoids treating all missing departure times in the same way. A missing departure time may still have strong supporting evidence from the actual arrival at the same station or from previous sections of the same service. The proposed hierarchy uses this evidence before relying on historical behaviour or fallback

assumptions.

The departure delay estimation hierarchy can be represented more formally as follows. For a station s_i, let t_i^{pd} denote the planned departure time, t_i^{ad} the actual departure time, t_i^{pa} the planned arrival time, and t_i^{aa} the actual arrival time. The estimated departure delay used by the real-time model is denoted by $\tilde{d}_i^d$. It is obtained from the strongest available source of evidence:

$$\tilde{d}_i^d = \begin{cases} t_i^{ad} - t_i^{pd}, & \text{if confirmed actual departure time is available,} \\ d_i^a + \tilde{d}_i^w, & \text{if confirmed actual arrival time at station } s_i \text{ is available,} \\ d_{i,\mathrm{RID}}^a + \tilde{d}_i^w, & \text{if current-service arrival evidence is available,} \\ \bar{d}_{i,\mathrm{hist}}^d, & \text{if historical station evidence is available,} \\ 0, & \text{otherwise.} \end{cases} \tag{6.5.1}$$

where $d_{i,\mathrm{RID}}^a$ is the latest observed or propagated arrival-delay evidence available for the same train service, $\tilde{d}_i^w$ is the estimated dwell-delay contribution, and $\bar{d}_{i,\mathrm{hist}}^d$ is the historical departure delay estimate for the relevant station and temporal context. This formulation shows that the framework does not treat a missing departure time as a simple missing value. Instead, it searches for progressively weaker sources of evidence before using the planned-time fallback.

6.5.3 Use of Actual and Propagated Values

The departure delay estimation stage uses several forms of evidence (available information sources), each with a different level of reliability. These values are not used randomly. They are applied in a controlled order so that the strongest available information is prioritised. Actual departure values are used whenever the confirmed departure time from the first station is available. This is the ideal case because the departure delay can be calculated directly from the difference between the planned and actual departure times. The resulting value represents the true operational state of the train as it enters the section being predicted.

Actual arrival values at the first station are used when the departure time is not yet available. This is particularly useful in real-time prediction because the train may

have arrived at the station, but the departure event may not yet have been confirmed. The actual arrival delay provides direct evidence about the train's current state at the boundary between the previous section and the current section. This value can therefore be used to estimate the departure delay more realistically than a simple default value. Propagated values are used when direct actual evidence at the first station is not available. In this case, the framework uses information from the same train service at an upstream section. The logic is that a train's delay state is not independent across sections. A delay observed or predicted upstream can influence the likely delay at the current station. By propagating this information forward and combining it with dwell behaviour, the framework can estimate a plausible departure delay for the current section.

Overall, the departure delay estimation strategy improves the practical value of the real-time prediction framework. It allows the system to generate predictions even when actual departure data is unavailable, while still using the best available evidence about the train's current state. This makes the framework more suitable for live railway operation, where predictions must often be made before all movement events have been confirmed.

6.6 Preparation and Integration of BOEM into the Real-Time Prototype

The previous sections explained how live railway information is transformed into route-section records and model-ready feature vectors. The next stage is to apply the section-level BOEM ensembles developed in Chapter 5 to these live inputs.

The BOEM weighting method, candidate-weight normalisation, optimisation objectives, and final ensemble training process are defined in Sections 5.8 and 5.10. These procedures are not repeated in full in this chapter. Instead, this section explains how BOEM is prepared using chronologically ordered data, stored for integration into the research prototype, matched to a live route section, and applied during real-time prediction.

All model training, preprocessing fitting, hyperparameter selection, and ensemble-weight optimisation are completed offline. During live operation, the

framework loads the stored models, preprocessing components, feature definitions, and ensemble weights. These components are not retrained or re-optimised when a new live update is received.

6.6.1 Time-Aware Preparation of BOEM for Real-Time Use

The BOEM ensembles are prepared for real-time use while preserving the chronological order of the historical observations. In a live prediction setting, only information that has already become available can be used. Future observations must therefore not influence earlier model training, validation, or ensemble-weight optimisation.

For each route section, the historical data are first arranged in chronological order and divided into a development period and a later unseen test period. The development period is used for model training and ensemble weight optimisation. The test period remains separate until all model and weight decisions have been completed.

Within the development period, expanding-window validation is used. Each validation round contains an earlier training block followed by a later validation block. As the rounds progress, the training block expands to include additional historical observations, while the next chronological period is used for validation. This structure follows the past-to-future direction of real-time prediction and reduces the risk of future-data leakage (Roberts et al., 2017).

Let D_{dev} denote the development data available for one route section. The chronological validation structure is written as:

$$\mathcal{V}_{\text{time}} = \left\{ \left(T_i^D, V_i^D\right) \right\}_{i=1}^{f}, \tag{6.6.1}$$

where T_i^D and V_i^D are the internal training and validation subsets in validation round i. They satisfy:

$$\max\left(\text{time}(T_i^D)\right) < \min\left(\text{time}(V_i^D)\right), \qquad i = 1, \ldots, f. \tag{6.6.2}$$

For example, if the development period contains observations from January to April, the first validation round may use January for training and February for validation. The second validation round may use January and February for training and March for validation, while the third validation round may use January to March for training and

April for validation. In this way, each validation period occurs after its corresponding training period. The candidate models are trained within each validation round, and their predictions on the corresponding validation block are retained.

Within each chronological validation round, BOEM follows the method defined in Section 5.8. The candidate models produce validation predictions, Bayesian optimisation evaluates complete raw candidate weight vectors, and one predefined evaluation metric is used in each validation round. Each raw candidate weight vector is normalised before its objective value is calculated, and the best normalised candidate found within the candidate-evaluation budget is retained. The round-specific best weight vectors are then averaged to obtain the final BOEM weight vector $\mathcal{W}^*$.

The chronological preparation therefore changes the order of the training and validation observations. It does not change the main BOEM weighting method defined in Chapter 5. After the final BOEM weights have been obtained, fresh instances of the required models are trained on the complete development period. This follows the final model-training procedure described in Section 5.10. The ensemble is then evaluated once using the unseen future test period. This follows the same procedure as Algorithm 2. The only difference is that it uses chronological validation rounds. Its output has the same form as the output of Algorithm 2, but the resulting weights and models are used in the real-time prediction framework.

6.6.2 Deployment Artefacts

A separate deployment artefact is prepared for each supported route section. The notation used for the model set and weight vector follows Section 5.10. For route section r, the model set is:

$$\mathcal{A}_r = [l_{r,1}, l_{r,2}, \ldots, l_{r,m_r}], \tag{6.6.3}$$

where m_r is the number of models stored for route section r. The corresponding weight vector is:

$$\mathcal{W}_r = [\omega_{r,1}, \omega_{r,2}, \ldots, \omega_{r,m_r}]. \tag{6.6.4}$$

The deployment artefact is represented as:

$$\mathcal{R}_r^{\mathrm{RT}} = \{\mathcal{A}_r, \mathcal{W}_r, \mathcal{P}_r, \mathcal{F}_r, \}\,, \tag{6.6.5}$$

where $\mathcal{P}_r$ contains the fitted preprocessing components and $\mathcal{F}_r$ contains the expected feature names, data types, and feature order. The route-section identifier is also stored with the artefact.

6.6.3 Real-Time Ensemble Inference

During live operation, the framework identifies the current route section and loads the selected deployment artefact. Let x_r denote the model-ready live input for route section r. For model $l_{r,j}$, the prediction is:

$$\hat{y}_{r,j} = l_{r,j}\left(\mathcal{P}_{r,j}(x_r)\right), \tag{6.6.6}$$

where $\mathcal{P}_{r,j}$ is the fitted preprocessing component stored for model $l_{r,j}$, and $\hat{y}_{r,j}$ is its predicted next-station arrival delay. The final real-time ensemble prediction is:

$$\hat{y}_r^{\mathcal{E}} = \sum_{j=1}^{m_r} \omega_{r,j}\hat{y}_{r,j}, \tag{6.6.7}$$

subject to:

$$\omega_{r,j} \geq 0, \qquad \sum_{j=1}^{m_r} \omega_{r,j} = 1. \tag{6.6.8}$$

Equation 6.6.7 is the route-section real-time form of Equation 5.10.4. It uses the same final weighted prediction defined in Chapter 5. It does not introduce a different ensemble calculation.

The prediction is stored together with the train-service identifier, service date, route section, input-information source, selected prediction strategy, prediction time, and model artefact identifier. These values allow each prediction to be traced to the live information and model components used to produce it.

6.6.4 Section Matching and Model Selection

Before prediction, the live route section must be matched to a stored model artefact. The live data, timetable records, route definitions, and trained models may use different

station identifiers. The framework therefore converts these identifiers into a consistent station representation before model selection.

The preferred case is an exact match. If the route, direction, and ordered station pair match a stored artefact, the corresponding BOEM ensemble is used directly. An exact artefact may not always be available. This may occur when the framework is extended to a route section that was not included during the main model-development stage. In this case, the framework may use a mapped source model. The mapping process is completed offline before live operation. For example, suppose that trained artefacts are available for route sections (A-B), (B-C), and (C-D), while no exact artefact is available for a target section (X-Y). The available source artefacts can be applied to historical development records from (X-Y). Their predictions are then compared with the observed arrival delays for this target section. The source artefact with the lowest predefined mapping error is selected and stored as the mapped artefact for (X-Y). The final test data and later live observations are not used during this selection.

Let r denote the target section and let $\mathcal{C}_r$ denote a set of candidate source artefacts. Let:

$$D_r^{\text{map}} = [X_r^{\text{map}}, Y_r^{\text{map}}] \tag{6.6.9}$$

denote the historical development data used to assess the candidate source artefacts for the target section. For candidate source $c \in \mathcal{C}_r$, let $\hat{Y}_r^{(c,\text{map})}$ denote its predictions on X_r^{map}. The selected source is:

$$c_r^* = \arg\min_{c \in \mathcal{C}_r} \mathcal{L}_{\text{map}} \left(Y_r^{\text{map}}, \hat{Y}_r^{(c,\text{map})} \right), \tag{6.6.10}$$

where $\mathcal{L}_{\text{map}}$ is the predefined mapping error measure. The same measure is used for all candidate source artefacts.

In simple terms, the framework tests each available source model using earlier data from the target section and selects the one with the lowest prediction error. The selected model is then used for later real-time predictions for that section. The mapping data are kept separate from the final test period and from later live observations. This prevents final evaluation data from influencing the selected source model. The selected

source is stored before live operation. Therefore, the live framework does not compare several source models every time a new update is received.

6.6.5 Prediction Strategy and Fallback Handling

The framework follows an ordered model selection strategy. It first searches for the most section-specific model source and then moves to weaker fallback options when the stronger source is unavailable.

- **Exact Section Artefact**

  The first option is the exact BOEM artefact stored for the same route, direction, and ordered station pair. This is the preferred source because the ensemble was developed specifically for the current route section.

- **Mapped Section Artefact**

  As described in previous part, if no exact artefact is available, the framework uses the mapped source artefact selected using Equation 6.6.10. The mapped source must have been assessed on historical development data from the target section before live operation.

- **Same Ordered Station-Pair Artefact**

  If no stored mapped source is available, the framework searches for an artefact developed for the same ordered station pair under another compatible route definition. For example, suppose Route 1 and Route 2 both include the ordered section (B-C). If an artefact was trained for section (B-C) on Route 1, the same artefact may be used for section (B-C) on Route 2 when the route definitions are compatible.

- **Reverse Station-Pair Artefact**

  If no suitable ordered-pair artefact is available, the framework may consider an artefact developed for the reverse station pair. This is a weaker option because operating conditions may differ by direction. For example, if no artefact is available for section (A-B), the framework may use an artefact trained for the reverse section (B-A).

- **Aggregated Model Fallback**

  If no exact, mapped, ordered-pair, or reverse-pair artefact is available, the framework may combine the predictions of the available trained models using their mean or median. This is the least section-specific option, but it allows the real-time system to continue producing predictions for new or data-sparse sections when no more suitable model can be used.

The ordered model selection process is illustrated in Figure 6.5.

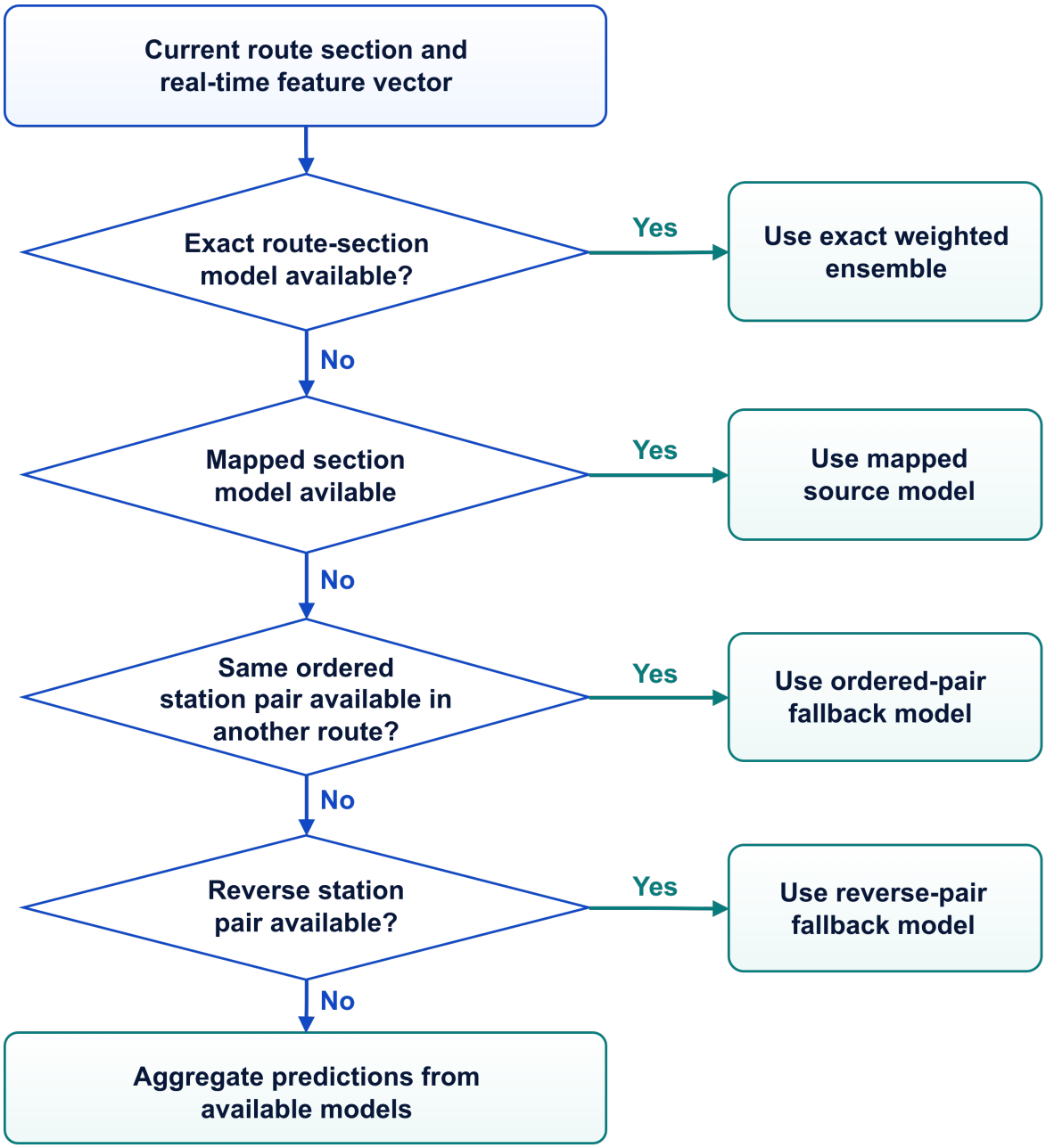


Figure 6.5: Ordered model selection and fallback strategy for real-time ensemble prediction.

6.7 Service-Level Prediction Correction Using Confirmed Observations

The deployed ensemble models generate predictions using delay patterns learned from historical data. However, the behaviour of an active train service may temporarily differ from these general patterns. For example, the model may repeatedly under-predict or over-predict the delay of one service as it moves along the route.

To respond to this service-specific behaviour, the framework applies a lightweight online correction. The purpose is not to retrain the ensemble or change its stored weights. Instead, confirmed arrival observations from the same journey are used to estimate a bias term that can be applied to later predictions for that service.

6.7.1 Service-Level Bias Estimation

Let u denote an active train service and let k denote a station at which both an earlier prediction and a confirmed arrival delay are available. The prediction error is calculated using the unadjusted model prediction:

$$e_{u,k} = d^{a}_{u,k} - \hat{d}^{a,\mathrm{raw}}_{u,k}, \tag{6.7.1}$$

where $d^{a}_{u,k}$ is the confirmed arrival delay and $\hat{d}^{a,\mathrm{raw}}_{u,k}$ is the corresponding BOEM prediction before service-level correction. A positive error indicates that the model under-predicted the delay, while a negative error indicates that it over-predicted the delay. The service-level bias is updated using an exponential moving average:

$$\tilde{b}_{u,k} = (1 - \lambda) b_{u,k-1} + \lambda e_{u,k}, \tag{6.7.2}$$

where $b_{u,k-1}$ is the previous bias value for service u, and λ controls the influence of the latest confirmed prediction error. The bias is initialised as:

$$b_{u,0} = 0. \tag{6.7.3}$$

λ allows the correction to respond gradually to recent errors without allowing one observation to fully determine the next prediction. To prevent an unusual prediction error from producing an excessive correction, the updated bias is limited to a predefined

range:

$$b_{u,k} = \min\left(b_{\max}, \max\left(b_{\min}, \tilde{b}_{u,k}\right)\right), \tag{6.7.4}$$

where $b_{\min} = -30$ minutes and $b_{\max} = 30$ minutes in the implemented framework. These limits are implementation safeguards rather than values learned by the model. They prevent one unusual confirmed error from producing an unrealistic service-level correction.

6.7.2 Correction of Later Predictions

After the bias has been updated, it can be applied to a later prediction for the same train service. For a later route section ending at station s_{i+1}, the corrected prediction is:

$$\hat{d}^{a,\mathrm{corr}}_{u,i+1} = \hat{d}^{a,\mathrm{raw}}_{u,i+1} + b_{u,k}, \tag{6.7.5}$$

where $\hat{d}^{a,\mathrm{raw}}_{u,i+1}$ is the unadjusted BOEM prediction and $\hat{d}^{a,\mathrm{corr}}_{u,i+1}$ is the prediction after service-level correction.

The bias is updated only when a confirmed arrival observation can be matched to an earlier unadjusted prediction for the same train service and route section. Estimated arrival values are not used to update the bias. The updated bias is also applied only to later predictions for the same journey. It does not change the stored models, BOEM weights, or predictions for other train services.

A separate bias state is maintained for each combination of service identifier and service date. When a new train journey begins, its bias is initialised to zero. This prevents prediction errors from one journey from affecting another journey. The raw prediction, corrected prediction, confirmed arrival delay, prediction error, and updated bias are stored separately.

6.8 Recursive Downstream Delay Prediction

The section-level ensemble described earlier produces an arrival-delay prediction for the next station. However, during an active journey, it is also useful to estimate the likely delay at stations farther along the route. The framework therefore extends

the next-station prediction across the remaining supported route sections. In the implemented framework, recursive downstream prediction begins only from a prediction based on confirmed actual departure information. This avoids starting a multi-step prediction chain from an input that has already been estimated. The output of each recursive step is stored separately from the direct next-station prediction.

The process starts with the latest confirmed state of the train and the arrival-delay prediction for the next station. This predicted arrival delay is used to construct the departure-delay input for the following section. The relevant deployed ensemble is then used to generate the next downstream prediction. The process is repeated until the end of the supported route is reached or the required information cannot be constructed.

The notation follows the station-to-station journey representation defined in Section 3.3. As explained in Section 3.4, d_i^a, d_i^d, and d_i^w denote the arrival delay, departure delay, and dwell delay at station s_i, respectively. For the current route section (s_i, s_{i+1}), let$\hat{y}_{i,j}$ denote the prediction produced by base model $l_{i,j}$. For section (s_i, s_{i+1}), the direct BOEM prediction is calculated using Equation 6.6.7. The resulting ensemble output represents the predicted arrival delay at station s_{i+1}:

$$\hat{d}_{i+1}^a = \hat{y}_i^{\mathcal{E}} \tag{6.8.1}$$

Equation 6.6.7 applies the same weighted ensemble calculation defined in Equation 5.10.4. It does not introduce a different BOEM weighting method. To generate a prediction for the following route section (s_{i+1}, s_{i+2}), the framework first estimates the departure delay from station s_{i+1}:

$$\hat{d}_{i+1}^d = \alpha_{i+1}\hat{d}_{i+1}^a + \hat{d}_{i+1}^w, \tag{6.8.2}$$

where $\hat{d}_{i+1}^a$ is the predicted arrival delay, $\hat{d}_{i+1}^w$ is the estimated dwell-delay contribution, and α_{i+1} is the delay-retention factor. The delay-retention factor represents the proportion of delay that is normally carried across a route section. It is estimated from historical observations for the relevant section and temporal context. Let x_h denote the observed departure delay at the first station and let y_h denote the observed arrival delay at the second station for historical observation h. The factor is estimated as:

$$\alpha_i = \frac{\sum_{h=1}^{n} x_h y_h}{\sum_{h=1}^{n} x_h^2}. \tag{6.8.3}$$

The implementation requires at least several suitable historical observations before a data-based value is used. When sufficient historical evidence is not available, the framework uses:

$$\alpha_i = 1. \tag{6.8.4}$$

This default assumes that the latest predicted arrival delay is carried forward without additional recovery or growth before the dwell contribution is considered. The estimated departure delay is then included in the recursively constructed feature vector for the following section:

$$X_{i+1}^{\text{rec}} = \mathcal{G}\left(\hat{d}_{i+1}^{d}, Z_{i+1}\right), \tag{6.8.5}$$

where $\mathcal{G}(\cdot)$ is the online feature-construction process and Z_{i+1} contains the remaining temporal, timetable, and contextual variables required by the deployed section model. The downstream arrival-delay prediction is then:

$$\hat{d}_{i+2}^{a} = f_{i+1}\left(X_{i+1}^{\text{rec}}\right), \tag{6.8.6}$$

where f_{i+1} is the selected ensemble or fallback strategy for section (s_{i+1}, s_{i+2}). The recursive relationship can therefore be summarised as:

$$\hat{d}_{i+1}^{a} \rightarrow \hat{d}_{i+1}^{d} \rightarrow X_{i+1}^{\text{rec}} \rightarrow \hat{d}_{i+2}^{a}. \tag{6.8.7}$$

The same process is repeated for the later supported route sections. Each prediction is stored with its recursive step so that direct and downstream predictions can be identified separately.

The predicted arrival delay at a station cannot always be used directly as the departure delay for the following section. A train may recover some delay during the station stop, or it may gain additional delay when the actual dwell is longer than

planned. The framework therefore uses the estimated dwell-delay contribution when constructing the departure-delay input for the next section. This allows the recursive feature vector to represent possible changes in delay at an intermediate station. The dwell contribution is added only when the departure-delay input is constructed in Equation 6.8.2. It is not added again to the output of the following section model. This prevents the same dwell effect from being counted twice during recursive prediction.

6.9 Implementation Considerations for Live Operation

Using the framework with live railway data introduces requirements that are not present during offline experiments. The system must process new messages, update train-service states, construct features, select models, generate predictions, and store the results while train journeys are still active. The implementation therefore considers runtime efficiency, message volume, and operational reliability. These considerations do not change the trained models or the BOEM weighting method. They support the practical execution of the framework.

6.9.1 Runtime Efficiency

All model training, preprocessing fitting, hyperparameter selection, and ensemble-weight optimisation are completed offline. During live operation, the framework loads the stored artefacts, constructs the required feature vector, executes the selected models, and combines their outputs using the stored weights.

No model retraining or weight optimisation is performed when a new live message is received. This reduces the amount of computation required during prediction. Models with an exact zero ensemble weight do not need to be executed because they make no contribution to the final prediction. The implementation also separates live-message processing, service-state management, feature construction, model inference, prediction storage, and dashboard access. This modular structure reduces unnecessary repetition and allows problems in one stage to be identified without stopping all parts of the framework.

6.9.2 Operational Reliability

Live railway messages may be incomplete, delayed, duplicated, received out of sequence, or revised as a journey progresses. The framework therefore records the source, status, and update time of the information used for each prediction. Newer confirmed observations are given priority over earlier estimates. Repeated or older messages are prevented from replacing more recent information. The service-state process also allows information received out of station order to be linked to the correct service and calling point.

When confirmed departure information is unavailable, the hierarchical departure-delay estimation procedure described in Section 6.5 is used. When an exact section artefact is unavailable, the model selection and fallback procedures described in Section 6.6.5 are applied. The original model output is also retained when service-level correction is applied. Therefore, the raw prediction, corrected prediction, information source, selected model strategy, and confirmed observation can be compared during later evaluation. These mechanisms do not guarantee that every prediction has the same level of reliability. Instead, they allow the framework to continue operating under incomplete conditions while keeping a clear record of how each prediction was produced.

6.10 Practical Challenges and Solutions

Moving from offline model evaluation to an effective real-time prediction framework introduced several practical difficulties that were not present when working with prepared historical datasets. In this setting, the challenge was not only to generate accurate predictions, but also to produce them within a suitable time and ensure that the most recent and reliable information could be collected, interpreted, transformed, and supplied to the appropriate model while the train journey was still in progress. Addressing these requirements involved substantial work in data integration, cleaning, filtering, estimation, storage, and runtime management. The main challenges encountered during the development of the framework are summarised below.

- Integrating Multiple Railway Data Sources.

As discussed in Chapter 4, the information required by the real-time framework is distributed across several sources rather than being available in a single complete dataset. In the offline prediction framework, the required data could be prepared and stored in a structured knowledge base before model training and evaluation. In contrast, the real-time framework must combine live operational updates with timetable data, historical train movement records, station references, route definitions, trained models, and ensemble weights. These sources use different structures, identifiers, and naming conventions. They therefore had to be connected, aligned, and transformed into a consistent representation based on train services, stations, routes, and station-to-station sections. This representation was designed to match the section-level inputs used during model training, as described in Chapter 5.

- Processing and filtering live railway data.

  The real-time framework receives a large and continuous volume of railway information, while only a small proportion of these records is relevant to the selected routes and station-to-station sections. The incoming stream may contain unrelated train services, movements in the opposite direction, repeated updates, outdated records, and incomplete messages. Therefore, the received information must be checked and filtered before it can be used for prediction. This process also requires station identifiers and route information from different data sources to be aligned with the route sections, trained models, and ensemble weights used by the framework.

- Managing incomplete and changing service information.

  Unlike the prepared datasets used in offline experiments, live railway information is not received as one complete and fixed record. Information about different stations may arrive at different times, and an early estimate may later be revised or replaced by a confirmed actual value. Updates may also be received in a different order from the actual sequence of the journey. The framework therefore maintains the latest available state of each active train service. Newer and more reliable information is used to update earlier records, while repeated or outdated

values are prevented from creating inconsistent predictions.

- Handling unavailable departure information.

  A confirmed departure time is not always available when a real-time prediction is required. However, departure delay is one of the important inputs used by the trained models. To address this problem, the framework uses a hierarchical estimation process. A confirmed actual departure is used whenever it is available. Otherwise, the departure information is estimated using the latest operational information, previous observations or predictions for the same train service, historical station behaviour, and finally the planned departure time as a fallback. The source of the departure value is also retained so that predictions based on actual and estimated departure information can be identified and evaluated separately.

- Maintaining prediction coverage and downstream stability.

  The live record must be matched to the correct route, station pair, trained model, and ensemble weights before a prediction can be produced. However, differences between station identifiers or incomplete model coverage may prevent an exact match in some cases. The framework therefore uses controlled mapping and fallback strategies to select the most suitable available model. A further challenge arises when the prediction is extended to later stations. In this case, the output from one section becomes part of the input to the next section, which can increase uncertainty as the prediction moves farther along the route. Updated service observations, estimated dwell behaviour, historical delay-retention patterns, and controlled fallback values are therefore used to reduce unrealistic accumulation of delay across downstream sections.

- Managing storage, runtime resources, and system monitoring.

  The framework must receive live information, update active train services, generate predictions, and store the results at the same time. The operational database was therefore designed to retain prediction values, timing information, selected model strategies, downstream outputs, and related metadata for later evaluation. At the

same time, the active in-memory state must remain limited so that old or inactive services do not continue to consume resources during long-running operation. The prototype dashboard supports this process by showing the current services, actual and estimated departure information, selected prediction strategies, and downstream predictions. It was also used during development to inspect the complete prediction process and identify inconsistencies or missing information.

Addressing these challenges required work beyond the training and evaluation of the prediction models. The resulting framework combines data integration, filtering, service-state management, missing-information estimation, model selection, downstream prediction, storage, and monitoring within a single real-time process. These mechanisms allow the proposed ensemble models to operate under the incomplete, changing, and high-volume conditions of a live railway environment.

6.11 Time-Aware Model Evaluation for Real-Time Framework

This section presents the results of the time-aware model evaluation conducted for the real-time prediction framework. The main goal is to evaluate whether the proposed modelling approach remains effective when the models are developed and evaluated using the dataset prepared for real-time prediction, before they are used in the operational prediction pipeline. These results therefore assess model performance under a time-aware validation setting, but they do not represent the final live operational evaluation. The operational real-time prediction outputs and their comparison with railway-provided estimates are presented in the next section. The results examine overall predictive accuracy, route-level behaviour, section-level stability, and inference feasibility. This provides evidence that the selected models, especially BOEM, are suitable for use in the later real-time prediction workflow. These experiments were conducted offline, and only the models trained at this stage were subsequently used in the real-time framework.

The results in this time-aware setting should not be compared directly with the historical results in Section 5.11, because the two evaluations use different feature sets and test designs. First, the real-time feature set excludes previous-delay variables that

are not available at the time of prediction, including last checkpoint delay and mean previous delays, as explained in Section 6.4.4. The SHAP analysis in Section 5.11.5 showed that these variables were among the most informative predictors in the historical setting. Their removal therefore makes the real-time prediction task more difficult. Second, the historical evaluation summarised performance across five cross-validation folds, whereas the time-aware evaluation uses a single later unseen test period after chronological model development, as described in Section 6.6.1. This design better represents the chronological conditions of real-time prediction, but the results are based on fewer and more time-specific test observations. The measured differences between BOEM and the individual models may therefore be smaller or more variable. Lower performance and smaller improvements in this section are consequently reasonable under the more restricted and realistic real-time setting.

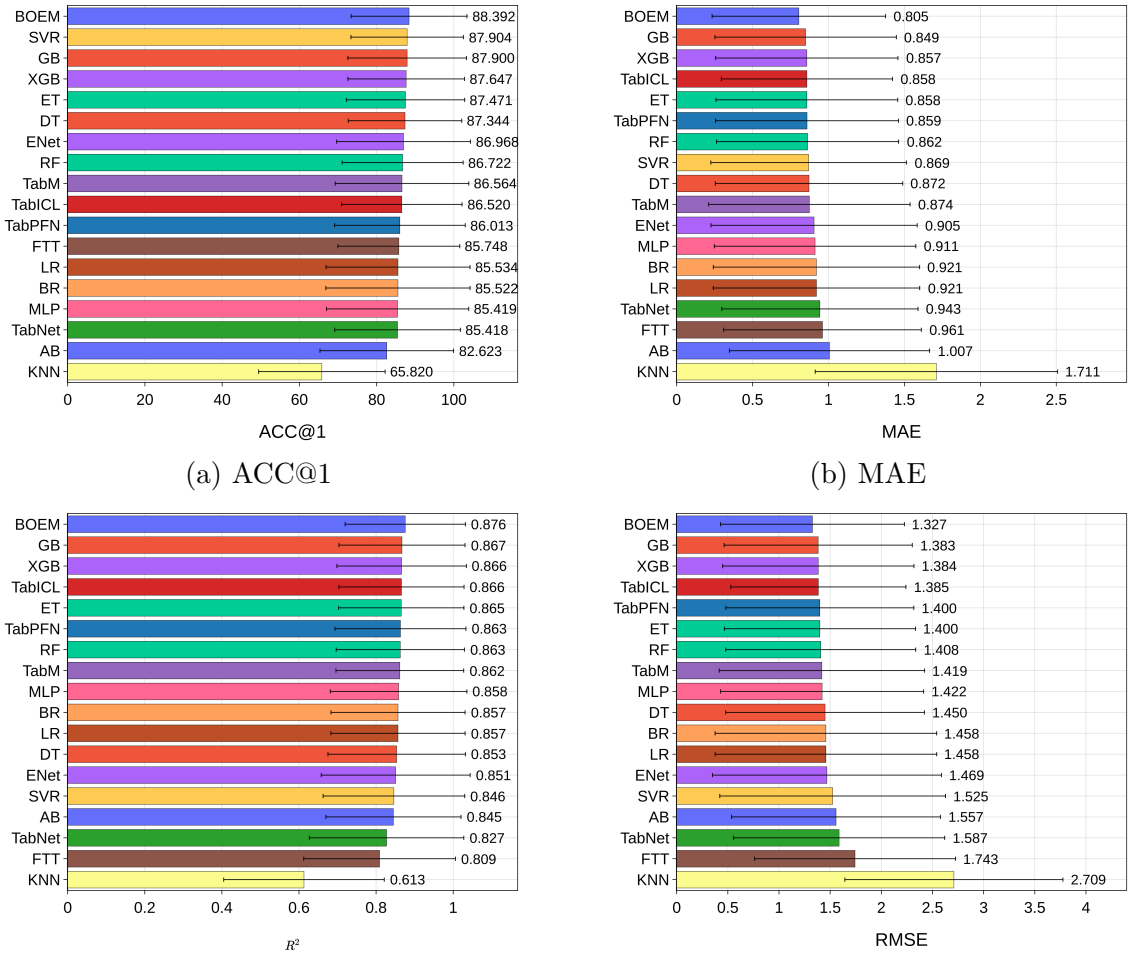
6.11.1 Overall Time-Aware Evaluation Performance

Figure 6.6 shows the overall time-aware evaluation performance of the models prepared for the real-time prediction framework. The values are averaged across all station-to-station sections and both routes. The error bars show section-level variation, so wider bars indicate that model performance changes noticeably across different parts of the railway network. This is expected because some sections have more regular delay behaviour, while others are affected by stronger delay propagation, station dwell variation, or operational complexity that have been examined in Chapter 4.

BOEM achieved the strongest overall performance across the reported metrics. It obtained the highest ACC@1 and R^2, and the lowest MAE and RMSE. The closest competing models were mainly tree-based and tabular models, including GB, XGB, ET, TabICL, and TabPFN. However, BOEM remained ahead of these models, showing that the ensemble weighting strategy provides a more balanced prediction profile across the full set of sections.

The improvement is most visible for the error-based metrics. BOEM reduced MAE by more than 5% compared with the closest individual competitor and reduced RMSE by about 4% compared with the closest individual competitor. These improvements were observed in the mean results across the heterogeneous route sections rather than

in only one section. The error bars also show that all models experience noticeable section-level variation, but BOEM maintains the best mean performance while remaining competitive across the route network. These results support the use of BOEM in the later operational prediction pipeline.



(a) ACC@1 (b) MAE

(c) R^2 (d) RMSE

Figure 6.6: Overall time-aware evaluation performance of the models prepared for the real-time prediction framework.

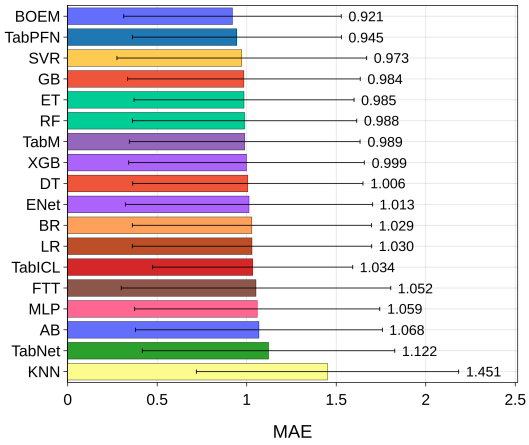
6.11.2 Route-Level Time-Aware Evaluation Results

The route-level results examine whether the models prepared for real-time prediction perform consistently across the two routes. Figure 6.7 reports MAE and RMSE separately for the NL route and the WW route. As in the previous figure, the error bars show the standard deviation across the sections within each route.

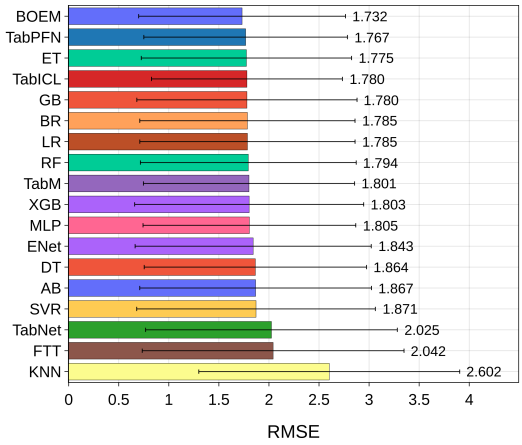
BOEM achieved the lowest MAE and RMSE on both routes. On the NL route, BOEM remained ahead of the closest competing models, including TabPFN, SVR, GB, and ET. The improvement was observed for both MAE and RMSE, indicating

lower average errors and fewer large prediction deviations. The error bars also show that section-level variation exists within the route, which is expected because different station-to-station sections have different delay characteristics.

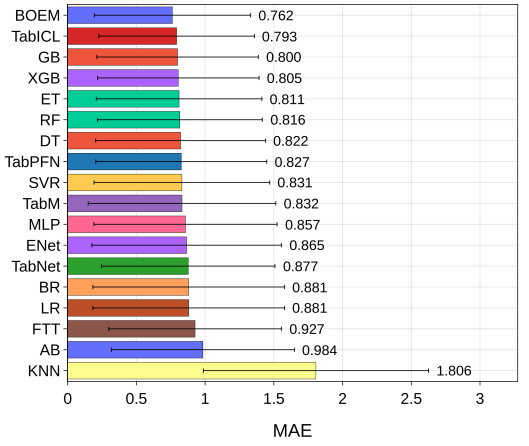
The WW route showed lower mean error values overall than the NL route, indicating that the models achieved better predictive performance on this route. BOEM again produced the lowest MAE and RMSE. The advantage was clearer on this route, especially for RMSE, where BOEM reduced error by more than 4% compared with the closest individual competitor. This indicates that the proposed ensemble is useful not only for improving average error, but also for reducing larger prediction deviations.



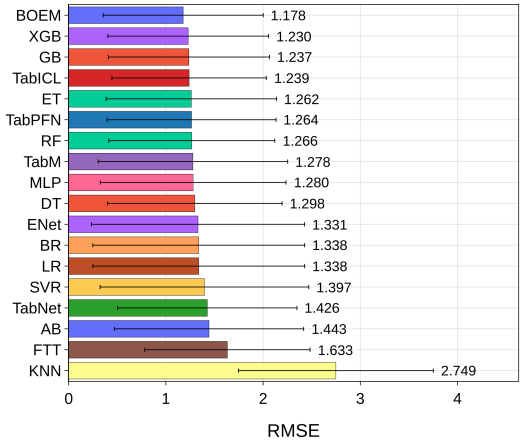
(a) MAE across NL route sections



(b) RMSE across NL route sections



(c) MAE across WW route sections



(d) RMSE across WW route sections

Figure 6.7: Route-level time-aware evaluation performance of the real-time prediction models. Error bars show the standard deviation across sections within each route.

The route-level comparison confirms that BOEM does not only perform well in the overall average, but also remains the strongest model on both routes. The difference between NL and WW also highlights why route-level evaluation is needed, because an overall result alone can hide route-specific differences in prediction difficulty.

6.11.3 Section-Level Time-Aware Evaluation Results

The aim of the section-level analysis is to examine whether the performance advantage of BOEM is observed across the route sections rather than being driven by only a small number of favourable sections.

The evaluation uses the real-time modelling dataset and the time-aware procedure described in Section 6.6.1. This procedure preserves temporal order so that models are evaluated using later observations without access to future data.

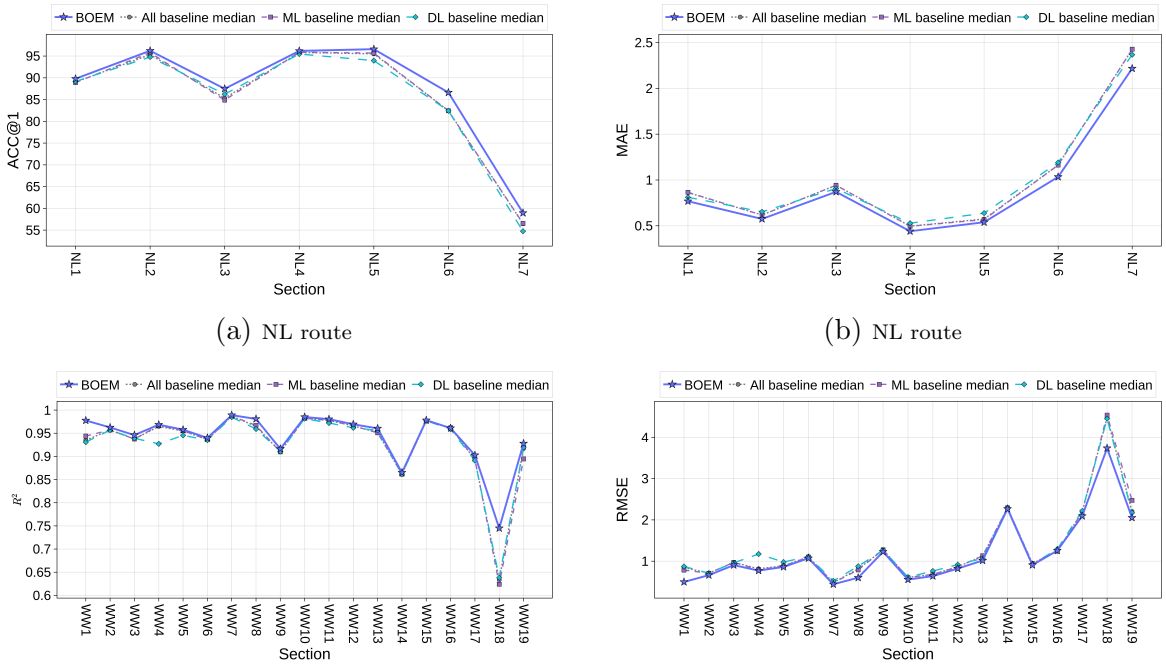
The evaluated models include conventional ML regressors, tabular DL models, and the proposed BOEM. To make the section-level comparison more stable, BOEM is compared with three median-based summary references: the median performance of all non-BOEM models, the median performance of the ML models, and the median performance of the DL models. Here, the median refers to the central performance value of a predefined group of baseline models, rather than the median of the delay target values. For each section and evaluation metric, the performance values within each predefined model group were ordered and their median was calculated. The median provides a summary of typical group performance and is less sensitive than the mean to unusually strong or weak individual results (Huber and Ronchetti, 2009).

Figure 6.8 shows selected section-level evaluation metrics for the two routes. For the NL route, ACC@1 and MAE are reported. BOEM remains above the three baseline medians for ACC@1 across all seven NL sections, and its advantage becomes clearer in sections where the baseline medians show weaker performance, especially NL6 and NL7. The same pattern appears for MAE, where BOEM remains below the three medians throughout the route. This shows that, on the NL route, the ensemble provides not only better average accuracy but also more stable local behaviour across sections.

For the WW route, R^2 and RMSE are reported. BOEM again remains consistently stronger than the three median references across the full set of WW sections. In many sections, the gap is modest because the median baselines already perform well. However, the benefit becomes more visible in the more difficult sections, particularly where the baseline curves deteriorate more strongly. This is especially clear around WW18, where BOEM preserves a noticeably higher R^2 and a lower RMSE than the three median baselines. A similar pattern can also be seen in several other sections with elevated

error levels, such as WW17 and WW19.

Overall, the section-level results show that the advantage of BOEM is not limited to a small number of favourable cases. The differences are generally smaller where the median references already perform well and become larger in some sections where their performance deteriorates. This supports the view that the proposed ensemble improves robustness across heterogeneous station-to-station conditions.



(a) NL route (b) NL route

(c) WW route (d) WW route

Figure 6.8: Section-level time-aware evaluation results comparing BOEM with the median-based summary references

6.11.4 Inference Time Results

Inference time was examined to assess whether the models were suitable for real-time use. Table 6.1 reports the mean and standard deviation of the inference time per prediction across all final-test samples. The experiments were conducted using the software and computing environment described in Section 3.10. The values are presented in milliseconds and ordered from the fastest to the slowest model. All models were loaded into memory before timing. Training, hyperparameter tuning, and weight optimisation were excluded. Before timing, three warm-up runs were performed to reduce initial setup effects. The inference process was then repeated 20 times, and the recorded times were used to calculate the mean and standard deviation.

Table 6.1: Mean and standard deviation of inference time per prediction.

Model	Mean (ms)	SD (ms)
DT	0.0028	0.0011
LR	0.0028	0.0011
BR	0.0028	0.0011
ENet	0.0028	0.0012
MLP	0.0035	0.0016
GB	0.0047	0.0013
XGB	0.0053	0.0019
TabM	0.0085	0.0014
AB	0.0140	0.0076
RF	0.0268	0.0095
KNN	0.0316	0.0091
ET	0.0351	0.0138
TabNet	0.0509	0.0036
SVR	0.2168	0.1049
FTT	0.3692	0.0085
TabICL	0.5505	0.0623
TabPFN	0.5855	0.0653
BOEM	0.7665	0.1577

The simple models, including DT, LR, BR, and ENet, had the lowest inference times. TabICL and TabPFN were the slowest individual models, with mean inference times of 0.5505 ms and 0.5855 ms, respectively. However, all individual models remained below one millisecond per prediction. BOEM required 0.7665 ± 0.1577 ms per prediction. Its inference time was affected mainly by the slowest models with non-zero weights rather than by the sum of all model inference times. This is because the selected models were executed in parallel. TabPFN and TabICL were among the strongest individual models and were included in many BOEM configurations. Their higher inference times therefore contributed directly to the BOEM runtime. The final weighted aggregation added only a small additional cost. Although BOEM combined several models, its mean inference time remained below one millisecond per prediction. This indicates that the additional computational cost of the ensemble was small and supports the feasibility of using BOEM for real-time train delay prediction.

6.12 Real-Time Prediction Results

This section evaluates the predictions generated by the real-time framework described earlier in this chapter during live railway operation. The previous section presented the time-aware evaluation of the models prepared for real-time framework. This section

focuses on the live inference stage, where predictions generated by the implemented pipeline are later matched with confirmed arrival outcomes. The aim is to assess whether the BOEM-based real-time framework provides more accurate arrival-delay predictions than the available railway-provided estimates within the collected live-data sample.

The real-time evaluation includes three main parts. First, direct BOEM predictions are compared with railway-provided estimates. The deployment and application of the trained BOEM ensembles are described in Section 6.6. The results are then reported separately according to whether actual or estimated departure information was used, as explained in Section 6.5. Finally, recursive downstream predictions are compared with simple baselines and, where available, with railway next-station estimates. The recursive prediction process is described in Section 6.8. The results are presented at the overall, route, and chain-horizon levels, mainly using MAE and RMSE.

6.12.1 Real-Time Evaluation Data

The real-time framework was operated for 40 days in 2026. During this period, live railway updates were received and processed, and the resulting direct and recursive predictions were stored for evaluation. The stored predictions were later matched with confirmed arrival-delay outcomes derived from the HSP actual-arrival records. For the comparison with railway next-station estimates, only records containing a recursive BOEM prediction, a corresponding railway estimate, and a confirmed arrival outcome were included.

Table 6.2 summarises the final samples used in the real-time evaluation. The direct prediction sample is divided according to whether confirmed actual departure information or framework-estimated departure information was used. The recursive sample contains downstream predictions generated across different prediction horizons. All delay values in the table are reported in minutes. Mean positive arrival delay is calculated using only records with an actual arrival delay greater than zero. Each direct prediction record represents one prediction for a service at a specific station-to-station section. Each recursive record represents one downstream prediction at a particular chain horizon. The eight route directions contain 80 directional station-to-station

sections.

Table 6.2: Summary of the samples used for evaluating predictions generated during the 40-day live-data collection period.

Evaluation scenario	Sample size	Routes directions	Directional sections	Mean arrival delay	Mean positive arrival delay
Direct prediction: actual departure	43,407	8	80	1.37 (4.03)	4.39 (4.34)
Direct prediction: estimated departure	107,298	8	80	7.39 (9.84)	10.75 (9.79)
Direct prediction: total	150,705	8	80	5.66 (9.01)	9.47 (9.32)
Recursive prediction vs. railway next estimate	171,938	8	80	2.08 (4.44)	5.03 (4.40)

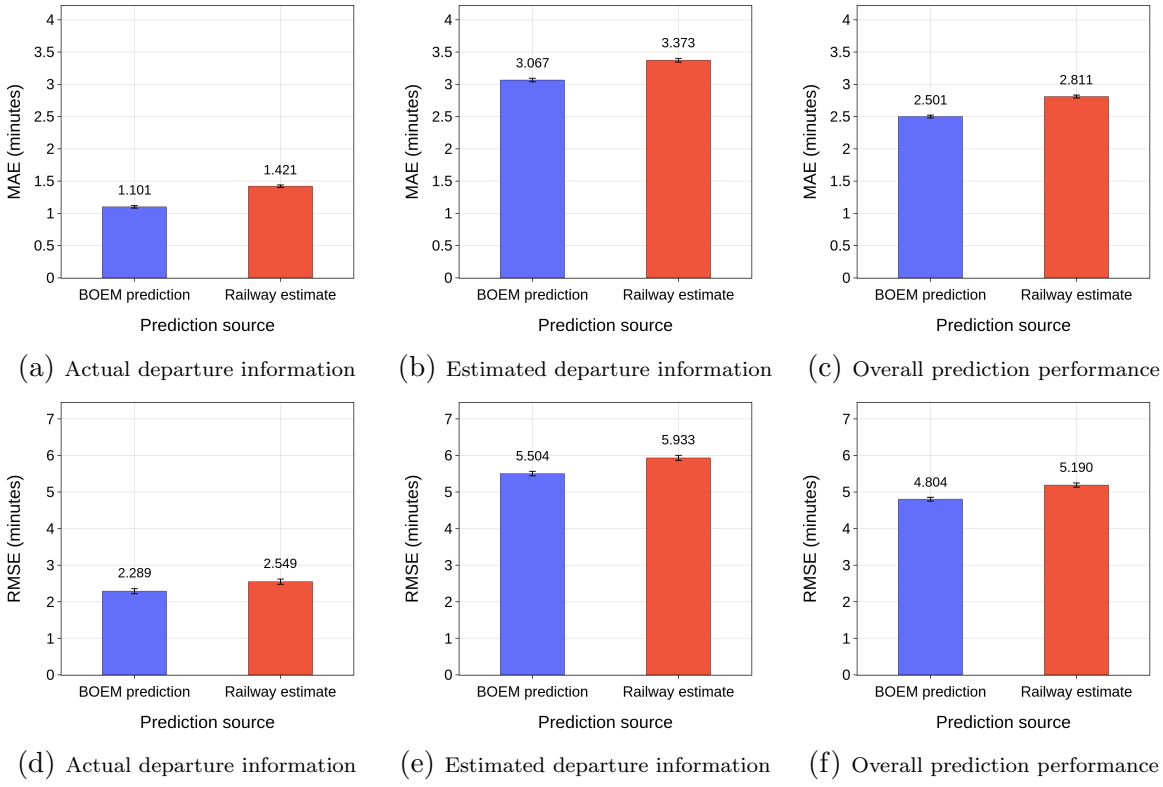
The estimated-departure sample contained larger and more variable confirmed arrival delays than the actual-departure sample, indicating that the two groups represented different operational conditions and may be happened under more difficult operational conditions. The recursive evaluation contained more predictions. The recursive sample contained more prediction records because one prediction chain can generate separate downstream predictions for several later stations. The larger sample size therefore does not represent the same number of independent train services, because several recursive records can belong to the same service and prediction chain.

6.12.2 Overall Direct Prediction Performance

The first real-time experiment evaluates the direct station-to-station predictions generated by the BOEM-based framework during the live-data collection period. The goal is to compare BOEM predictions with railway-provided estimates using confirmed arrival delays derived from the HSP actual-arrival records as the common reference. This comparison examines whether the BOEM predictions are closer to the confirmed arrival outcomes than the corresponding railway estimates.

Figure 6.9 reports MAE and RMSE for three reporting groups: records using actual departure information, records using estimated departure information, and all prediction records combined. The distinction between actual and estimated departure information follows the departure-time estimation process described in Section 6.5. The reason for

separating these cases is that predictions based on actual departure information are made with more certain input data, while predictions based on estimated departure information reflect a more difficult operational case.



(a) Actual departure information (b) Estimated departure information (c) Overall prediction performance

(d) Actual departure information (e) Estimated departure information (f) Overall prediction performance

Figure 6.9: Overall direct real-time prediction performance comparing BOEM predictions with railway estimates. Error bars indicate variation around the mean estimates.

For MAE, BOEM achieved lower error than the railway estimate in all three settings. The improvement was largest when actual departure information was available, where BOEM reduced MAE by approximately 23%. In the estimated-departure setting, the reduction was smaller but still meaningful, at approximately 9%. Across all predictions combined, BOEM reduced MAE by approximately 11%. This shows that BOEM achieved lower direct arrival-delay prediction errors when confirmed departure information is available and when the system must rely on estimated departure information. BOEM also achieved lower RMSE than the railway estimate in the actual-departure, estimated-departure, and combined groups.

The error bars in Figure 6.9 indicate variation around the reported averages. They are relatively small compared with the difference between BOEM and the railway estimate in most cases, particularly for MAE. This supports the interpretation that the observed improvement is not only a visual difference between two bars, but a consistent

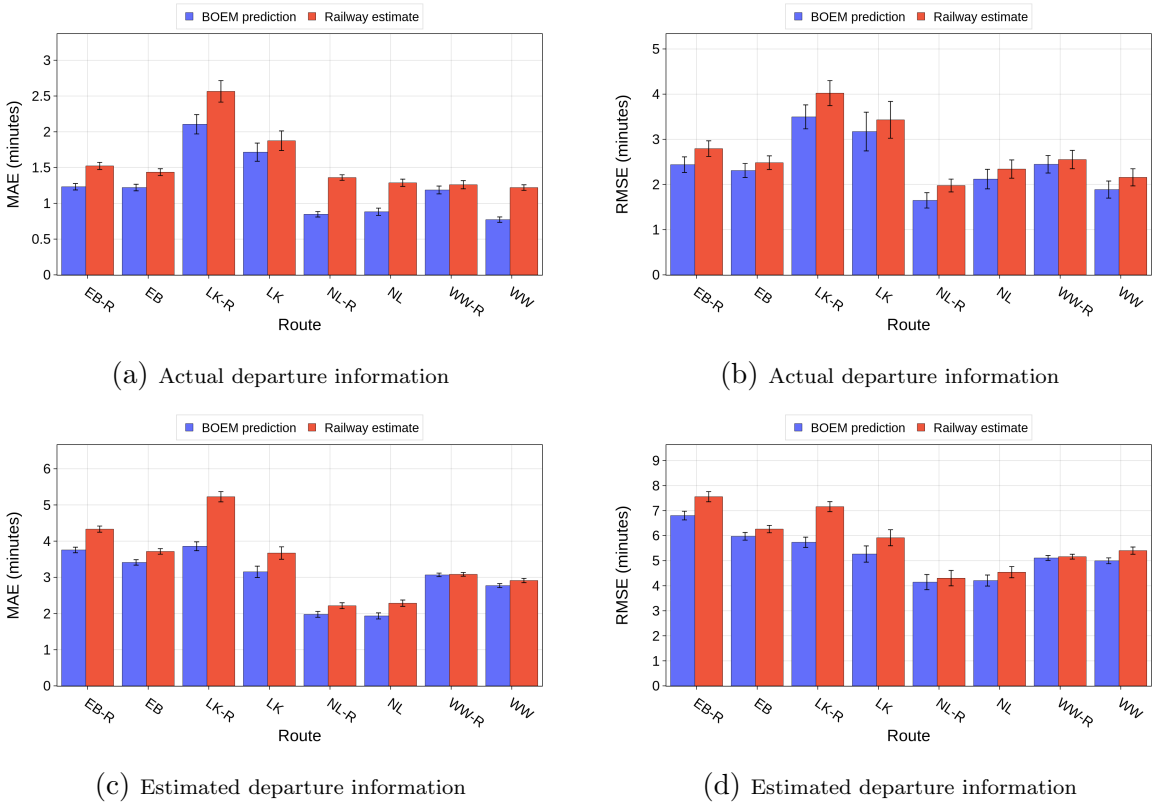
pattern across the evaluated prediction records. These reductions refer to prediction error and do not indicate any reduction in the actual train delays. On average, BOEM predictions were closer to the confirmed arrival outcomes than the railway-provided estimates in each reported group. The lower MAE shows that the average arrival-delay error is reduced, while the lower RMSE suggests better control of larger prediction errors. The better results when actual departure information is available are expected because the input data are more reliable. BOEM also achieved lower errors than the railway estimate in the estimated-departure group, showing that the approach remains useful when confirmed departure information is not yet available.

An additional analysis compared the final bias-adjusted direct predictions with the corresponding raw BOEM predictions. The RID-based adjustment affected approximately 3% of the direct predictions. This relatively low coverage was expected because the adjustment could only be applied after a confirmed actual arrival had been received from an earlier station for the same train service. Predictions generated before the first confirmed arrival therefore retained their raw BOEM values. The adjustment slightly improved prediction accuracy for some records and routes. For example, for direct predictions using confirmed actual departure information, it reduced MAE by approximately 0.04%. These findings show that the RID-based adjustment was correctly applied and had a positive effect in some operational cases, although its overall contribution to prediction accuracy was limited and may benefit from further investigation and refinement.

6.12.3 Route-Level Direct Prediction Performance

The route-level analysis examines whether the advantage of BOEM is consistent across the eight route directions in the live evaluation. Here, each route direction is represented by an operational route key, including EB, LK, NL, and WW and their reverse-direction counterparts, as described in Section 4.4. These route directions represent different service and operating conditions within the collected sample. Figure 6.10 compares BOEM predictions with railway-provided estimates using MAE and RMSE, separated by actual and framework-estimated departure information. This provides a more detailed view of whether the direct prediction performance remains consistent across different

parts of the operational network, rather than only in the overall average.



(a) Actual departure information

(b) Actual departure information

(c) Estimated departure information

(d) Estimated departure information

Figure 6.10: Route-level direct real-time prediction performance under actual and estimated departure information. Error bars indicate route-level variation.

The results show that BOEM achieved lower MAE and RMSE than the railway estimate across all route groups in both departure-time settings. When actual departure information was available, BOEM produced a stronger reduction in error. When estimated departure information was used, the prediction task became more difficult, but BOEM still remained better than the railway estimate across all routes. The smaller gains when confirmed departure information was unavailable show that uncertainty in the departure-time input reduces prediction accuracy, but the proposed framework still provides a consistent advantage over the railway estimate.

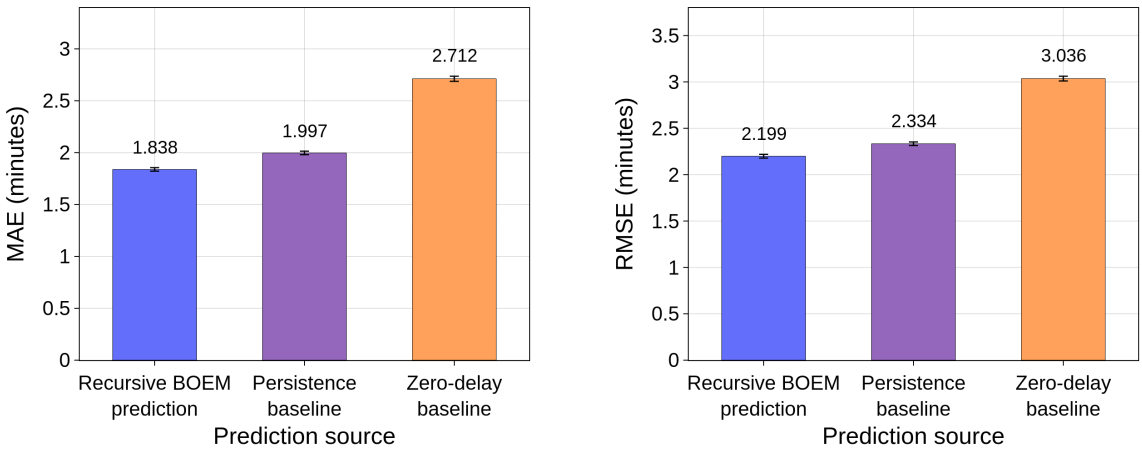
The error bars show variation around the route-level averages. In some routes, especially those with higher error levels, the bars are wider, indicating greater variation in prediction difficulty. Even with this variation, the BOEM bars generally remain lower than the railway estimate bars, supporting the robustness of the direct prediction results across routes.

6.12.4 Overall Recursive Downstream Prediction Performance

This experiment evaluates the recursive downstream predictions generated by the real-time framework described in Section 6.8. The purpose is to examine whether the framework can predict delays beyond the next station-to-station movement. In this setting, each BOEM prediction is used to update the input state for the next downstream section, allowing delays to be predicted recursively across several later route sections.

Recursive BOEM is compared with two simple baselines. The persistence baseline carries the latest available delay value forward unchanged to all later sections in the same prediction chain. For example, if the latest available delay is 3 minutes, the persistence baseline predicts a delay of 3 minutes for each later section. The zero-delay baseline assigns a delay of zero minutes to every downstream section. These baselines represent two simple assumptions: delay continuation and full delay recovery.

Figure 6.11 shows the chain-run mean MAE and RMSE. MAE and RMSE were first calculated separately for each recursive prediction chain and were then averaged across the evaluated chains. So each recursive chain has a balanced influence on the reported result. This is suitable for evaluating downstream prediction because different services may have different numbers of future prediction steps.



(a) MAE (b) RMSE

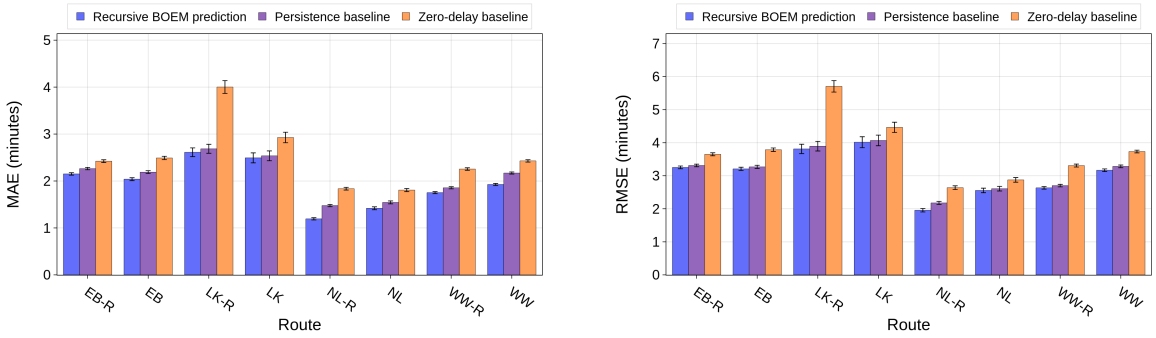
Figure 6.11: Overall recursive downstream prediction performance based on mean chain-level errors.

Recursive BOEM achieved the lowest error for both MAE and RMSE. Compared with the persistence baseline, it reduced MAE by approximately 8% and RMSE by

approximately 6%. The improvement was larger compared with the zero-delay baseline, with reductions of approximately 32% for MAE and 28% for RMSE. These results indicate that recursive BOEM estimates downstream delay progression more accurately than the two simple assumptions that delay either remains unchanged or returns to zero.

6.12.5 Route-Level Recursive Prediction Performance

Figure 6.12 compares recursive BOEM with the persistence and zero-delay baselines across the eight route directions. Recursive BOEM achieved lower MAE and RMSE than both simple baselines across all eight route directions. The advantage is clearer on routes where delay propagation is more variable, because simple baselines cannot adapt to route-specific delay changes.



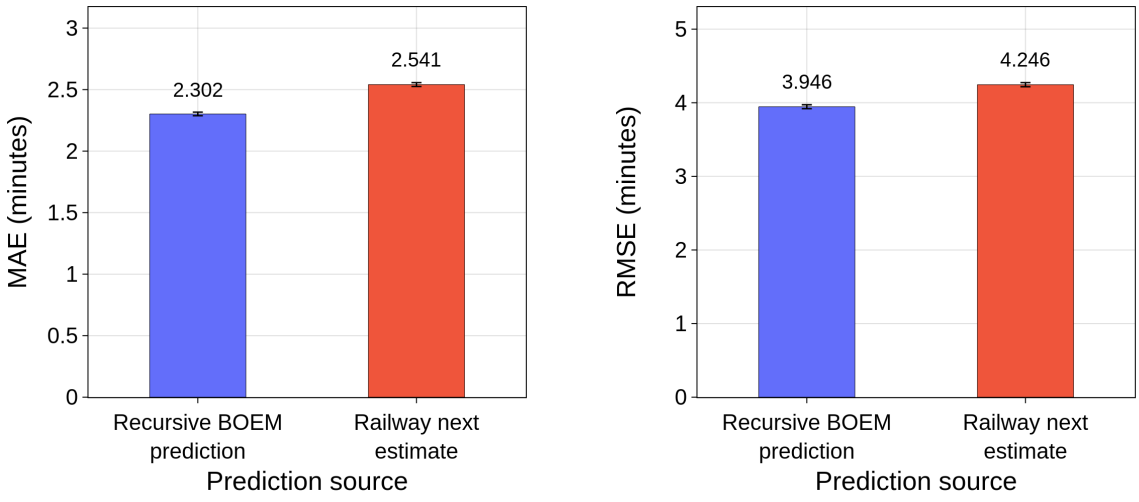
(a) MAE (b) RMSE

Figure 6.12: Route-level recursive prediction performance compared with simple baselines.

The route-level results also show that the size of the improvement is not uniform. In some route groups, the persistence baseline is close to recursive BOEM, which suggests that the latest delay state already provides a strong indication of downstream behaviour for those services. In other route groups, particularly where the errors are larger, recursive BOEM provides a clearer reduction in both MAE and RMSE. This indicates that BOEM is better able to follow route-specific delay progression than fixed assumptions that delays either remain unchanged or return to zero.

6.12.6 Recursive Prediction Compared with Railway Next-Station Estimates

This experiment compares recursive BOEM predictions with the corresponding railway next-station estimates. Unlike the previous recursive comparison, which used simple baselines, this comparison evaluates BOEM against an operational railway reference for records where a corresponding railway next-station estimate was available. The aim is to examine whether recursive BOEM achieves lower downstream prediction errors than the available railway estimates. Figure 6.13 shows the overall MAE and RMSE results. Recursive BOEM achieved lower error than the railway next estimate for both metrics. Relative to the railway next-station estimate, recursive BOEM reduced the pooled MAE by approximately 9% and the pooled RMSE by approximately 7%. This shows that the lower recursive BOEM errors were also observed against an operational railway estimate, rather than only against the two simple baselines.

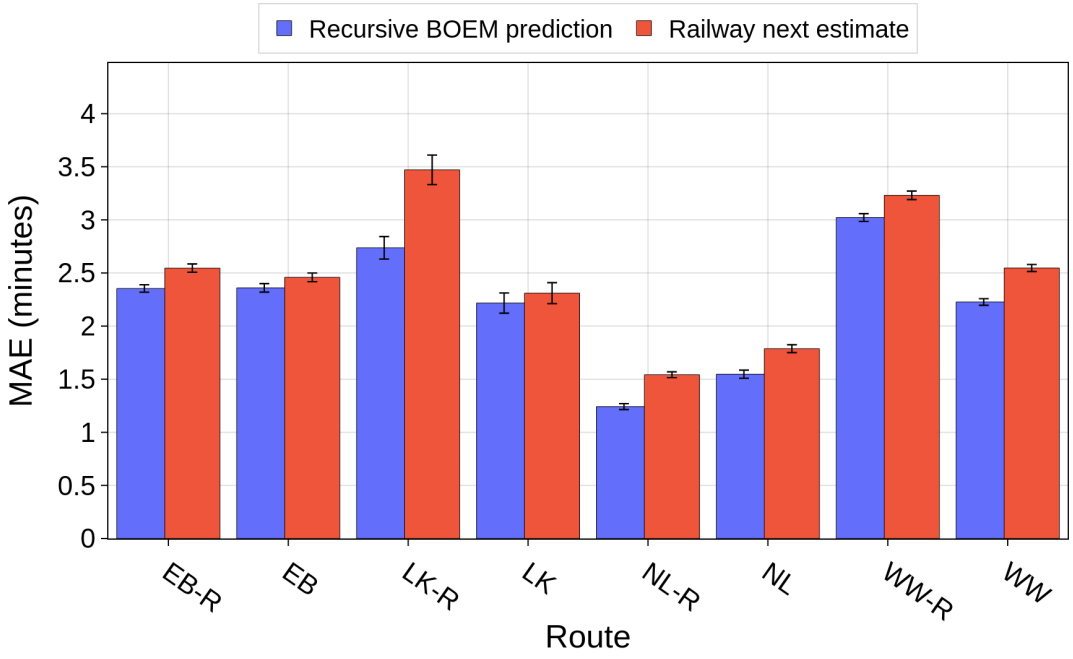


(a) MAE (b) RMSE

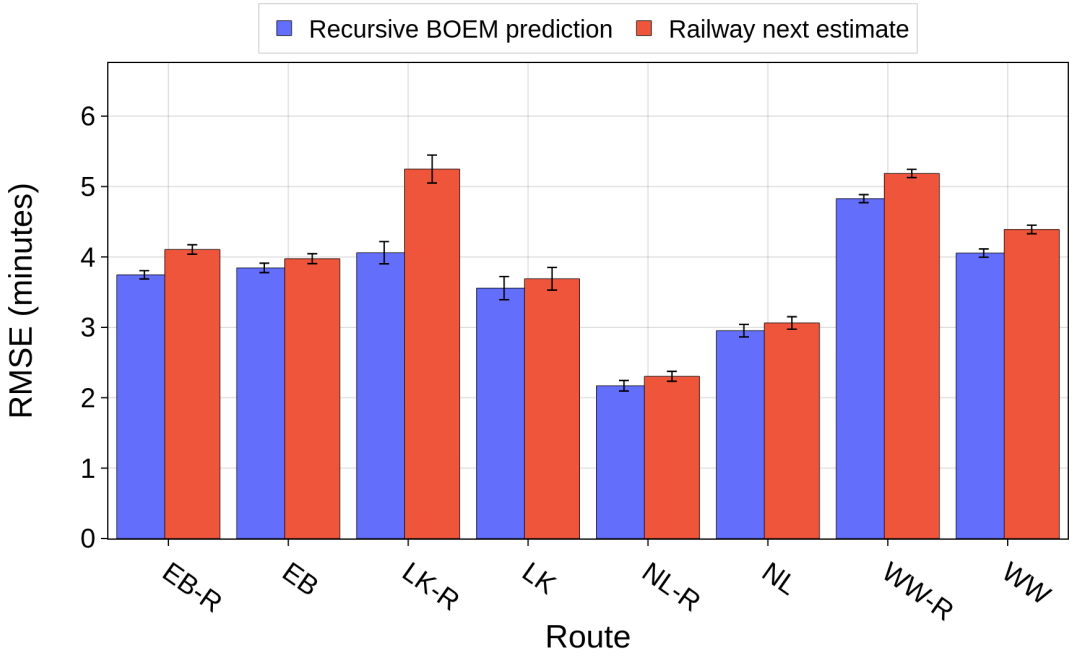
Figure 6.13: Overall recursive downstream prediction performance compared with railway next-station estimates.

Figure 6.14 reports the same comparison across routes. Recursive BOEM produced lower MAE and RMSE than the railway next-station estimate across all eight route directions. The size of the improvement varied between routes, showing that the relative advantage of BOEM was affected by the characteristics of each route. On routes where the two bars are close, the railway estimate is already competitive, but BOEM still remains at least comparable.

Overall, these results provide further evidence for the predictive performance of



(a) MAE



(b) RMSE

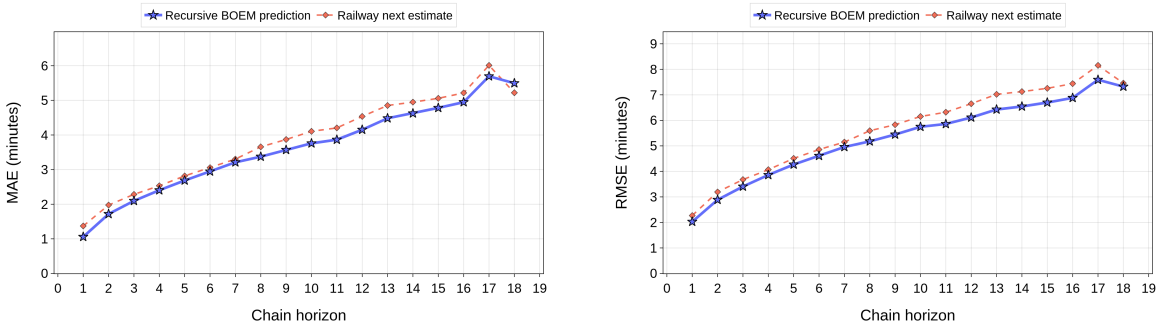
Figure 6.14: Route-level recursive prediction performance compared with railway next-station estimates.

the recursive component. Recursive BOEM improves over simple baselines and also performs better than the railway next-station estimate in the overall comparison. This indicates that the proposed approach can support downstream delay prediction beyond the immediate next movement.

6.12.7 Recursive Prediction Performance Across Chain Horizons

This subsection examines how recursive prediction error changes as the prediction is extended farther along the route. Here, the chain horizon represents the number of recursive station-to-station prediction steps ahead of the latest available service state. Horizon 1 represents the first downstream prediction, while larger horizon values represent predictions propagated across additional route sections. Longer horizons rely on more recursively generated inputs and less recent confirmed information. For example, consider five consecutive stations, s_1, s_2, s_3, s_4, and s_5. Suppose that the latest confirmed service state is available at s_2. Horizon 1 is the first downstream prediction and estimates the delay for the section from s_2 to s_3. The predicted state at s_3 is then used as an input for Horizon 2, which estimates the delay for the section from s_3 to s_4. Similarly, Horizon 3 uses the recursively predicted state at s_4 to estimate the delay for the section from s_4 to s_5. Therefore, a larger horizon represents a prediction farther from the latest confirmed station and involves more recursively generated inputs.

Figure 6.15 shows MAE and RMSE across increasing chain horizons. The MAE and RMSE of both methods generally tend to increase at longer chain horizons. This indicates that downstream prediction naturally becomes more uncertain when more future sections are predicted recursively.



(a) MAE across chain horizons

(b) RMSE across chain horizons

Figure 6.15: Recursive prediction performance across downstream chain horizons.

At each chain horizon, recursive BOEM and the railway estimate were evaluated using the same matched prediction records and confirmed arrival outcomes. Recursive BOEM remains lower than the railway next-station estimate across most horizons for both MAE and RMSE.

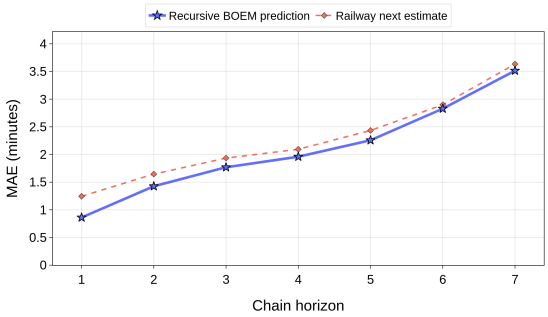
Overall, the horizon results show that recursive BOEM provides its clearest advantage at short and medium prediction horizons. It also maintains lower RMSE throughout the full evaluated range. The results at the longest horizons are less certain because fewer prediction chains reach these positions. Therefore, the long-horizon findings should be interpreted together with the number of available observations. This supports the practical value of BOEM in real-time railway operation, where early downstream warnings are often the most useful for service management.

6.12.8 Route-Specific Recursive Prediction Across Chain Horizons

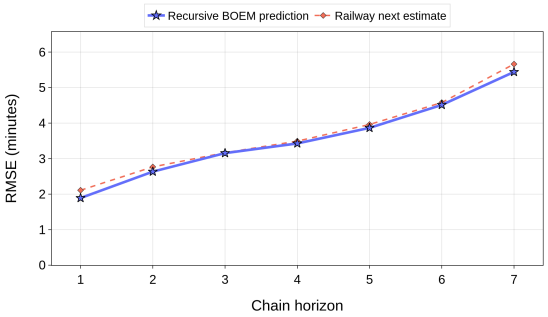
To examine route-specific recursive performance, Figure 6.16 presents the results for the NL and WW routes across chain horizons. Larger horizon values represent predictions propagated across more downstream route sections and therefore rely on more recursively generated inputs.

A similar general pattern is observed on both routes. Recursive BOEM prediction remains better than the railway next estimate across almost all chain horizons for both MAE and RMSE. Prediction errors generally increased at longer chain horizons, since longer recursive propagation is more challenging. However, BOEM keeps a consistent advantage over the railway next estimate, which indicates that the recursive framework remains effective even when the prediction is extended further along the route.

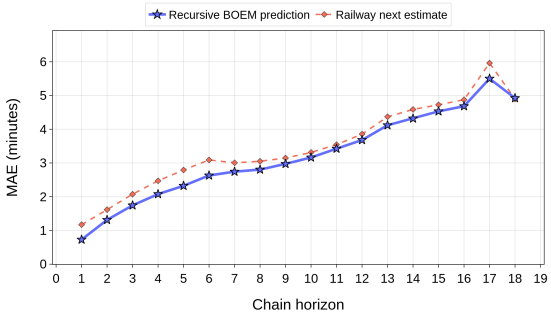
On the NL route, the improvement is visible from the first horizon and remains present across the full chain. The gap is more noticeable at the shorter and medium horizons, while the two methods become closer at the longest horizon. On the WW route, the same general trend appears over a longer chain. Errors rise more gradually across horizons, but BOEM still maintains lower MAE and RMSE than the railway next estimate for most of the route. These results suggest that the proposed recursive approach is not only effective overall, but also remains reliable when applied across



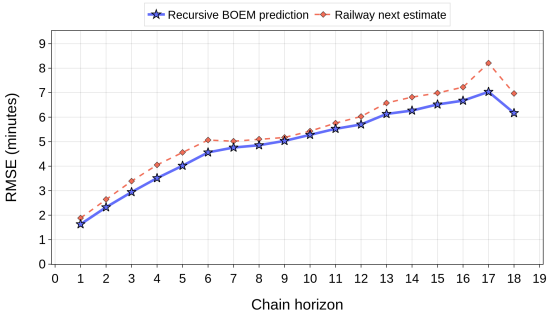
(a) NL route



(b) NL route



(c) WW route



(d) WW route

Figure 6.16: Route-specific recursive chain prediction performance across downstream horizons.

different route structures and different chain lengths.

6.13 Discussion of the Real-Time Evaluation

6.13.1 Offline and Real-Time Evaluation

The offline and real-time experiments examined different parts of the prediction problem. The offline experiments provided a controlled setting for comparing many models across routes, route sections, evaluation measures, and additional test conditions. These experiments were useful for identifying the strongest models and examining their behaviour in detail.

The real-time experiments provided a more difficult evaluation. In this setting, predictions had to be produced when some input information was incomplete or had to be estimated. The results therefore depended not only on the trained model, but also on the quality and timing of the available operational information. The real-time framework used models that were trained offline. Their parameters were not updated after each new live record. The framework therefore performed real-time inference rather than online or incremental learning.

Taken together, the offline, time-aware, and real-time results provide a wider evaluation of BOEM. The offline results show its predictive strength under controlled conditions, while the real-time results provide evidence that the framework remains useful when information is less complete and changes during a train journey.

6.13.2 Real-Time and Recursive Prediction

The real-time experiments examined whether the proposed framework could produce useful predictions when a train was operating. This setting was more difficult than the historical experiments because new information arrived during the journey, some input values were not always available, and predictions had to be produced before the final arrival times were known.

The first part of the evaluation examined model development for the real-time framework. A walk-forward expanding validation method was used to keep the observations in time order. Models were trained using earlier data and evaluated using later data. This provided a more realistic test than a random data split because a real prediction system must use past information to predict future train movements.

BOEM achieved the strongest overall performance during the real-time model

development experiments. It produced the lowest overall MAE and RMSE and maintained the strongest mean performance across the evaluated route sections. The route-level and section-level results also showed that its performance was not based only on one route or one route section. Hybrid BOEM remained strong on both the NL and WW routes. These results supported the selection of BOEM for the prototype real-time prediction framework. However, strong validation performance alone does not show that a model will provide useful live predictions. The framework was therefore also evaluated using prediction records created from live railway updates and matched with observed arrival times.

The direct prediction results showed that BOEM improved on the railway-provided estimates in the evaluated records. When confirmed departure information was available, BOEM reduced MAE by approximately 23%. This was the strongest improvement because confirmed departure information provided a more reliable description of the current train state. When confirmed departure information was not available, the framework used estimated departure information. Under this condition, BOEM reduced MAE by approximately 9% compared with the railway-provided estimates. The improvement was smaller than under the confirmed departure condition because an estimated input introduces additional uncertainty into the prediction. Across all evaluated direct predictions, BOEM reduced MAE by approximately 11% compared with the railway-provided estimates. This shows that the framework provided useful predictions under both departure-information conditions. However, it also shows that the quality of the input information had a clear effect on prediction performance.

This difference has an important practical meaning. A model cannot fully remove uncertainty that is already present in its inputs. When the departure state is confirmed, the model can use a more accurate starting point. When the departure state must first be estimated, any error in that estimate may also affect the arrival-delay prediction. Improving the departure estimation process could therefore improve the later predictions.

This finding is consistent with earlier research on real-time railway prediction. Previous studies have noted that models tested on complete historical data may face greater difficulty when live information is missing, delayed, or uncertain (Spanninger et al., 2022; Tiong et al., 2023a). They have also shown that recent delay information

is an important input for predicting later train delays (Lapamonpinyo et al., 2022; Taleongpong et al., 2022). The difference between the confirmed and estimated departure results in this research supports these observations.

The recursive prediction experiment extended the framework beyond the next route section. The predicted state from one section was used as input for the next section, allowing predictions to be generated for later parts of the journey. This is useful because railway staff and passengers may need information about more than the next station.

Recursive BOEM reduced MAE by approximately 8% and RMSE by approximately 6% compared with the persistence baseline. It also reduced MAE by approximately 32% and RMSE by approximately 28% compared with the zero-delay baseline. These results show that the model learned more than a simple rule that assumes the current delay will remain unchanged or return immediately to zero.

The comparison with railway next-station estimates provided a more practical reference. Recursive BOEM reduced MAE by approximately 9% and RMSE by approximately 7% compared with these estimates. This shows that the proposed framework added useful information beyond the simple baselines and the available railway estimate in the evaluated records. The improvement was clearer at short and medium horizons but became smaller farther downstream as uncertainty increased through recursive prediction. Therefore, predictions at longer horizons should be interpreted with greater caution. The inference-time experiment also showed that BOEM produced predictions very quickly, with a speed comparable to the slowest tabular DL model. This provides further evidence that BOEM is suitable for real-time inference.

The prototype dashboard showed how the direct and recursive predictions could be presented and compared with railway estimates. The dashboard was not evaluated as a fully deployed railway decision-support system. However, it demonstrated that the prediction pipeline and model outputs could be connected within one live monitoring interface.

Overall, the real-time results showed that the proposed framework could process live updates and generate direct and recursive arrival-delay predictions. BOEM improved on railway-provided estimates in the evaluated records, especially when confirmed departure

information was available. The recursive results also showed useful performance for later route sections, although uncertainty increased with prediction horizon. These findings provide the main evidence related to the fifth research question.

6.13.3 Implications for Railway Operations and Decision Support

The findings of this research have several implications for railway operations. First, BOEM can support more reliable arrival delay prediction at the station-to-station level. More accurate predictions can improve passenger information, especially when a service is already delayed and passengers need realistic information about arrival times and possible connections. In the asymmetric cost experiment, BOEM achieved the lowest calculated cost when underprediction was given a higher penalty than overprediction.

As reported in Section 5.11.1, the proposed ensemble reduced MAE by approximately 9% compared with the next three best-performing base models. This was equal to an average reduction of approximately 0.06 minutes, or 3.6 seconds, for each route-section prediction. To illustrate the possible cumulative scale, approximately 7.5 million passenger trains were planned in Great Britain between April 2025 and March 2026 (Office of Rail and Road, 2026). If each train had an average of 10 predicted route sections and the same improvement was maintained, this would represent approximately 4.5 million fewer accumulated absolute prediction-error minutes, equivalent to around 75,000 hours. This is only a hypothetical illustration and not an estimate of national performance. The value represents the total reduction in error across repeated section-level predictions, not a reduction in actual train delays, passenger journey time, or railway operating hours. However, more accurate predictions may provide more reliable information for train regulation, platform planning, passenger connections, and passenger information.

Second, the recursive prediction results show that BOEM can support downstream delay awareness. Controllers and information systems may need to understand not only the expected delay at the next station, but also how the delay may change across later sections of the route. The recursive component provides a structured way to extend predictions along the route and compare them with railway next-station estimates. This

can support early warnings for later stations, especially at short and medium horizons where the model improvement was clearest.

Third, the historical transfer experiments and live evaluation provide complementary evidence of the framework's applicability across railway settings. The proposed method retained useful predictive performance in both transfer directions between NL and WW, while the live framework generated and evaluated predictions across NL, WW, LK and EB and their reverse directions. These eight route directions extend the real-time evaluation beyond the original two historical directions and provide additional evidence across the selected services and directional conditions. The historical experiments examine transfer between route datasets, whereas the live evaluation demonstrates prediction performance during active journeys; together, they strengthen the evidence for the proposed approach. Although these settings do not represent the full diversity of railway networks, the findings support further application and evaluation on additional routes. Route-specific validation should examine differences in infrastructure, service patterns, traffic density, timetables, station spacing and delay propagation before extending the performance claims to other operators or network configurations.

Fourth, the computational characteristics of the framework support its possible use in real-time applications. BOEM introduces additional training cost because of Bayesian optimisation, but this cost occurs during offline model training. During prediction, the trained base models produce their outputs and BOEM combines them using the learned weights. This structure may support applications such as dynamic timetabling, dispatch decision support, passenger information systems, and early-warning tools (Wen et al., 2020).

The real-time pipeline and prototype dashboard also show how live railway updates can be processed and how direct and recursive predictions can be displayed. The dashboard allows the proposed model outputs to be inspected and compared with railway-provided estimates. It therefore connects the modelling work with a practical monitoring environment.

Finally, the results show that railway delay prediction methods should not be evaluated using average error alone. A model with a slightly lower MAE may still be less useful if it performs poorly when input data are incomplete, if its performance

changes across routes, or if it frequently underpredicts important delays. The evaluation framework used in this research provides a wider basis for selecting models for practical railway prediction.

Overall, the proposed framework for delay prediction could support passenger information, downstream delay monitoring, and operational decision support. However, its wider use would require further evaluation on additional routes and under longer real-time operating conditions.

6.14 Answer to Research Question 5

The time-aware and live evaluations provide the following answer to the fifth research question introduced in Chapter 1.

RQ5. Can the proposed ensemble framework predict train arrival delays in real time?

Yes, the proposed framework successfully generated real-time train arrival-delay predictions under the routes and operational conditions evaluated in this research. The framework processed live railway updates and produced direct and recursive arrival-delay predictions across the selected routes. It used offline-trained section-specific models, constructed the required features from live information, and estimated departure information when confirmed values were unavailable. Therefore, the framework performed real-time inference rather than online or incremental learning. For direct prediction, BOEM reduced MAE by approximately 23% when confirmed departure information was available and by approximately 9% when departure information had to be estimated. For recursive downstream prediction, BOEM reduced MAE by approximately 9% and RMSE by approximately 7% compared with the available railway next-station estimates, while also outperforming the persistence and zero-delay baselines. The prototype dashboard demonstrated how the predictions could be displayed and compared with railway estimates in a live monitoring interface. These findings show that the proposed framework can support real-time train arrival-delay prediction within the evaluated routes and operational conditions. The supporting evidence is presented in Sections 6.11.4, 6.12.2, 6.12.4 and 6.12.6.

6.15 Summary

This chapter presented the development and evaluation of the research prototype for real-time train arrival delay prediction. It explained how live railway messages are filtered, organised by train service and transformed into model-ready route-section records. It also described departure-delay estimation, model selection, service-level correction, recursive downstream prediction, storage and prototype monitoring.

BOEM was prepared for real-time use through chronological model development and evaluation. The time-aware results showed that the ensemble remained competitive under the feature restrictions and past-to-future evaluation conditions used for the real-time framework. The inference-time measurements also supported the feasibility of model-level prediction within the tested computing environment.

The live evaluation showed that BOEM achieved lower direct prediction errors than the available railway estimates under both confirmed and estimated departure information. The improvement was greater when confirmed departure information was available. Recursive prediction also improved on the persistence and zero-delay baselines and on the available railway next-station estimates. Prediction error generally increased at longer recursive horizons, where more inputs had to be generated from earlier predictions.

These findings addressed RQ5 by demonstrating that the framework could generate and evaluate direct and downstream arrival-delay predictions using live railway information. This conclusion applies to the routes, monitoring period and operational conditions evaluated. The framework and dashboard remain research prototypes, and further evaluation is required before operational adoption.

Chapter 7 brings together the findings, their contributions to knowledge, the overall limitations and directions for future work.

Chapter 7

Conclusion and Further Work

This chapter brings together the findings from the development and historical evaluation of the ensemble methods in Chapter 5 and the real-time investigation in Chapter 6. It summarises the principal findings, explains their relationship to the research questions and contributions to knowledge, and considers the overall scope and limitations of the evidence. It then identifies directions for further development and evaluation.

7.1 Main Findings

This research investigated train arrival delay prediction using machine learning, tabular deep learning, and heterogeneous ensemble methods. The aim was to improve prediction accuracy and examine model performance under different railway data and operating conditions. The work focused on station-to-station and real-time prediction scenarios. The principal contributions to knowledge concern the findings on multi-metric weighting, Bayesian ensemble-weight search, hybrid model combinations, reliability and real-time predictive performance obtained through the MWEM and BOEM investigations.

The historical experiments showed that the performance of individual models varied across routes, route sections, and evaluation metrics. This supported the use of heterogeneous ensembles rather than relying on a single predictor. MWEM combined models using multi-metric validation weighting, while BOEM extended this approach by using Bayesian optimisation to search jointly for more effective ensemble weights.

Across the offline experiments, BOEM achieved the strongest overall performance among the individual models, ensemble configurations, and reconstructed published methods. It achieved an R^2 of 0.914, an MAE of 0.594 minutes, an RMSE of 1.055

minutes, and an ACC@1 of 92.819%. Compared with MWEM, it reduced MAE by approximately 7% and RMSE by approximately 6%. It also reduced MAE by approximately 8% and RMSE by approximately 9% compared with the strongest individual models for these measures. Additional experiments assessed the reliable performance of the proposed method under missing-value and cross-dataset conditions. It achieved lower average error and variation than the individual models, and its performance deteriorated more slowly as the level of missing values increased.

In addition to the historical offline evaluation, the proposed method was tested in a live real-time setting. BOEM reduced MAE by approximately 23% when confirmed departure information was available and by approximately 9% when departure information had to be estimated. For recursive downstream prediction, it reduced MAE by approximately 9% and RMSE by approximately 7% compared with the available railway estimates. These results show that the framework can support real-time next-station and downstream arrival-delay prediction.

Overall, the findings provide evidence that multi-metric ensemble weighting and hybrid model combination can improve train delay prediction within the datasets and evaluation settings examined in this research.

7.2 Synthesis of Findings and Contributions to Knowledge

The findings connect the research questions to the methodological developments and evidence presented in the preceding chapters. The detailed answers to RQ1–RQ4 are provided in Section 5.13, while the answer to RQ5 is provided in Section 6.14. The contribution of each investigation is summarised below.

For RQ1, MWEM provides a procedure for selecting and weighting heterogeneous models using validation evidence from several performance metrics. The comparisons with individual models and equal-weight averaging show the value of this procedure under the evaluated historical conditions. The controlled BOEM weighting experiment provides additional evidence about the use of several metrics rather than one validation criterion.

For RQ2, BOEM extends direct validation-based weighting by searching complete combinations of model weights. The contribution to knowledge is the evidence obtained about the predictive effectiveness of the proposed Bayesian weighting configuration in the evaluated train-delay settings. The observed improvements are interpreted alongside the additional offline computational requirements and the statistical results of the respective comparisons.

For RQ3, the comparison of BOEM-ML, BOEM-DL and hybrid BOEM provides evidence about the value of combining the two predefined model pools. The hybrid configuration achieved stronger overall results in the evaluated historical setting. This supports their combined use in this research, rather than establishing that hybrid ensembles necessarily outperform either model pool in every prediction problem.

For RQ4, the section-level, missing-departure and cross-dataset analyses extend the evaluation beyond overall mean accuracy. They provide evidence about performance consistency, sensitivity to estimated inputs and transfer between the selected routes. Their contribution is an assessment of reliability under specified conditions; wider railway generalisation remains to be established.

For RQ5, the research prototype connects offline-trained predictors with live-data preparation, unavailable-input estimation and direct and recursive prediction. Its evaluation demonstrates predictive feasibility within the collected live-data sample. This contribution is distinguished from industrial deployment or validated operational decision support.

The route-section prediction formulation and integrated data pipeline provide the common foundation for these investigations. Their roles are explained in Chapters 3, 4, 5 and 6.

Table 7.1 summarises how each research question was investigated, the evidence obtained and the resulting contribution to knowledge. The detailed answers are provided in Sections 5.13 and 6.14.

Table 7.1: Mapping between research questions, investigations, evidence and contributions to knowledge.

RQ / contribution	**Investigation**	**Evidence obtained**	**Knowledge established**
RQ1 / C1	MWEM compared with equal-weight averaging; controlled comparison of multi-metric and single-metric BOEM-ML weighting.	MWEM improved on AE; multi-metric BOEM-ML achieved stronger overall results across the four reported measures. Sections 5.11.1 and 5.11.2.	Evidence that complementary validation metrics can improve ensemble weighting under the evaluated conditions.
RQ2 / C2	Bayesian ensemble-weight search evaluated against MWEM and other ensemble configurations.	BOEM achieved the lowest overall test errors among the evaluated configurations. Table 5.4.	Evidence of the predictive effectiveness of the proposed Bayesian weight-search configuration for the investigated train-delay settings.
RQ3 / C3	Comparison of ML-only, DL-only and hybrid model pools using the same BOEM weighting procedure.	Hybrid BOEM achieved stronger overall results across all four main evaluation measures. Table 5.4.	Empirical evidence of a benefit from combining these two predefined model pools under the common historical evaluation design.
RQ4 / C4	Section-level analysis, simulated missing departure information and transfer between the NL and WW routes.	BOEM-ML showed lower error and variation, slower deterioration under missing departure information, and useful transfer performance. Sections 5.11.3, 5.11.9 and 5.11.10.	Evidence about performance consistency, sensitivity to reconstructed inputs and transfer between the selected routes.
RQ5 / C5	Live direct and recursive prediction using confirmed and estimated departure information.	BOEM achieved lower errors than the corresponding railway estimates in the evaluated comparisons. Sections 6.12.2 and 6.12.6.	Evidence of predictive value from offline-trained ensembles under the evaluated live-data and incomplete-information conditions.

7.3 Scope and Limitations of the Findings

The main limitations of this research are summarised as follows:

- The main historical experiments were based on two UK railway routes, and the real-time evaluation covered selected routes and monitoring periods. Further evaluation on more routes, operators, countries, years, and disruption conditions is needed before wider generalisability can be confirmed.

- The missing values experiment were created by masking values and did not represent all possible real-time data problems, such as delayed updates, incorrect values, or the complete loss of a data source.

- The framework used tabular station-to-station data and did not directly represent wider network factors, such as rolling stock circulation, crew constraints, platform conflicts, and interactions between several train services. The available data also did not contain the ordered event sequences needed by sequence-based models (Huang et al., 2020b; Li et al., 2021a), or the complete network structure required by graph-based models (Tang et al., 2025; Li et al., 2024a,b).

- The interpretation of the final BOEM ensemble remains limited. SHAP analysis was applied to the GB model and showed the main relationships between its input features and predictions. However, this analysis does not directly explain individual BOEM predictions.

- The real-time framework and dashboard were evaluated as research prototypes rather than as a fully deployed railway decision-support system. The study did not include long-term operational evaluation or formal user testing with railway controllers, planners, or passengers.

7.4 Further Work

The findings of this research provide several opportunities for further development and evaluation.

First, future work should evaluate the proposed methods using larger and more diverse railway datasets, including additional UK routes and railway networks in other countries. This would provide stronger evidence about cross-route performance and help identify which route characteristics affect prediction accuracy.

Another direction is to improve the feature representation. The current framework uses tabular station-to-station data. Future work could examine graph-based, sequence-based, or combined spatio-temporal models when more detailed train movement and network data are available. This could help represent how delays spread between train services. Future studies could also examine whether weather, infrastructure conditions, and major events improve prediction.

The framework could also be extended from individual routes to connected routes or a wider railway network. This could help the models capture interactions between services and how delays spread from one route to another. Future work could also include delay causes and distinguish between primary and reactionary delays, provided that suitable network-level data are available.

The current real-time framework uses models trained offline. Future work could explore online or incremental learning, allowing model weights or base learners to update as new data become available. This may help the framework adapt to timetable changes, engineering work, seasonal patterns, rolling-stock changes, and other operational changes.

Finally, future work could examine how the predictions could support decision-making by railway operators. For example, when a long delay is expected, the predictions could be combined with operational information to help operators consider alternative actions and provide earlier information to passengers. The current framework predicts delays but does not recommend actions, so this would require further development and evaluation.

7.5 Final Remarks

This research shows the value of moving beyond single models and single-metric evaluation for train arrival delay prediction. The proposed BOEM framework combined

ML and tabular DL models using multi-metric Bayesian-optimised weighting and achieved strong results across the historical and real-time evaluation settings used in this research.

Further evaluation on additional routes, over longer live monitoring periods, and with operational users is required before operational adoption by railway organisations. Within the scope of this research, the findings provide evidence that multi-metric heterogeneous ensemble learning can improve route-section-level and real-time train arrival delay prediction.

Chapter 8

Appendix

8.1 Use of Generative Artificial Intelligence

OpenAI (ChatGPT V4 and V5) was occasionally used during the preparation and revision of this thesis to check and improve grammar, clarity, and sentence structure (OpenAI, 2023, 2025). All research, implementation, experiments, analysis, interpretation, and final thesis were completed by myself. All suggestions produced by ChatGPT were reviewed and edited before use.

8.2 Introduction

This appendix provides supplementary information supporting the data analysis, modelling, implementation, and evaluation presented in the main chapters of this research. The additional material is included here to provide further detail and improve reproducibility without interrupting the main flow of the research. These materials should therefore be read alongside the corresponding sections referenced throughout the appendix.

8.3 Description of Final Dataset Features

This appendix provides a summary of the main features created for the final modelling dataset, which were introduced in Section 4.7. Table 8.1 describes each feature in terms of its meaning, variable type, expected range or categories, and its role in the modelling process.

Table 8.1: Description of the main features used in the final modelling dataset.

Feature name	**Description**	**Type**	**Range or categories**	**Use / note**
Current station	The station at the start of the station-to-station section.	Categorical	Stations on the selected route	Used to define the section and route context.
Next station	The station at the end of the station-to-station section.	Categorical	Stations on the selected route	Used to define the prediction location.
Route direction	The direction of travel for the selected route.	Categorical	Route-direction labels	Used for route matching and section separation.
Departure delay	The delay observed when the train departs from the current station.	Numerical	Negative, zero, or positive delay values	Used as a main operational input.
Dwell delay	The difference between observed and planned dwell time at the current station.	Numerical	Negative, zero, or positive delay values	Used to describe station stopping behaviour.
Last checkpoint arrival delay	The most recent known arrival delay from a previous checkpoint.	Numerical	Negative, zero, or positive delay values	Used in offline modelling when available; excluded from real-time modelling due to limited live availability.

Feature name	Description	Type	Range or categories	Use / note
Mean previous arrival delay	The average arrival delay from previous sections of the same service.	Numerical	Negative, zero, or positive delay values	Captures recent service-level delay history; excluded from real-time modelling due to limited live availability.
Last checkpoint departure delay	The most recent known departure delay from a previous checkpoint.	Numerical	Negative, zero, or positive delay values	Used in offline modelling when available; excluded from real-time modelling due to limited live availability.
Mean previous departure delay	The average departure delay from previous sections of the same service.	Numerical	Negative, zero, or positive delay values	Captures recent departure-delay behaviour; excluded from real-time modelling due to limited live availability.
Last checkpoint dwell delay	The most recent known dwell delay from a previous checkpoint.	Numerical	Negative, zero, or positive delay values	Used in offline modelling when available; excluded from real-time modelling due to limited live availability.

Feature name	Description	Type	Range or categories	Use / note
Mean previous dwell delay	The average dwell delay from previous stops of the same service.	Numerical	Negative, zero, or positive delay values	Captures recent dwell-time behaviour; excluded from real-time modelling due to limited live availability.
Day of week	The weekday on which the service operated.	Categorical	Monday–Sunday	Represents weekly operating patterns.
Day of month	The calendar day of the month.	Ordinal	1–31	Represents within-month time variation.
Hour of day	The hour in which the section movement occurred.	Ordinal	0–23	Represents time-of-day effects.
Month	The month in which the service operated.	Ordinal	1–12	Represents monthly variation.
Season	The season in which the service operated.	Categorical	Spring, summer, autumn, winter	Represents seasonal operating conditions.
Weekend status	Indicates whether the service operated during a weekend.	Binary	Weekend or weekday	Represents differences between weekday and weekend operation.
Holiday status	Indicates whether the service operated during a public holiday period.	Binary	Holiday or non-holiday	Represents holiday-related operating conditions.

Feature name	Description	Type	Range or categories	Use / note
Peak-period status	Indicates whether the service operated during a peak period.	Binary	Peak or off-peak	Represents demand-related time conditions.
Arrival delay	The observed arrival delay at the next station.	Numerical target	Negative, zero, or positive delay values	Prediction target for station-to-station modelling.

8.4 Base Model Descriptions and Hyperparameter Settings

This appendix provides additional information about the base models used in the ensemble framework introduced in Chapter 5. Since the baseline models are well-established methods, the main chapter focuses on their role within the proposed MWEM and BOEM frameworks rather than explaining each model in detail. Therefore, compact descriptions of the individual base models and their main hyperparameter settings are provided here to support reproducibility without interrupting the main flow of the chapter.

The base models were selected to provide different predictive behaviours. This was important because train delay patterns may include linear relationships, non-linear interactions, local similarity effects, and complex feature dependencies. The model pool therefore included conventional machine learning models and tabular deep learning models. Depending on the ensemble configuration, these models were used separately or together within the proposed ensemble framework. The models were first evaluated as individual predictors and then used as base learners in the proposed ensemble configurations. The conventional machine learning models formed the ML model pool, the tabular deep learning models formed the DL model pool, and the combined set formed the hybrid model pool used by the final BOEM configuration.

8.4.1 Conventional Machine Learning Models

The conventional machine learning model pool included linear, tree-based, kernel-based, instance-based, and neural network regression models. These models were included because they represent different assumptions about the relationship between the input features and the next-station arrival delay. Some models are simple and interpretable, while others are better suited to non-linear feature interactions. This diversity was important for the proposed ensemble framework because MWEM and BOEM combine predictions from different model families.

Linear Regression (LR). Linear Regression is a simple linear model that estimates arrival delay as a weighted combination of the input features. It assumes an

approximately linear relationship between the predictors and the target variable. In this thesis, LR was used as a simple and interpretable baseline for assessing whether more complex models provide additional predictive value.

Bayesian Ridge (BR). Bayesian Ridge is a regularised linear regression model that estimates model coefficients using a Bayesian formulation. It introduces prior assumptions over the coefficients, which can make the model more stable when features are correlated or when the dataset contains noise. In this thesis, BR was used as a more robust linear baseline than ordinary linear regression.

Elastic Net (ENet). Elastic Net is a regularised linear model that combines L1 and L2 penalties. The L1 component can reduce the influence of less useful features, while the L2 component improves stability by shrinking coefficient values. ENet was included to test whether a sparse and regularised linear structure could provide useful predictions for station-to-station delay modelling.

Support Vector Regression (SVR). Support Vector Regression is a kernel-based regression model that attempts to fit a prediction function while allowing small errors within a defined margin. By using kernel functions, SVR can model non-linear relationships between input features and arrival delay. It was included because railway delay behaviour may contain non-linear patterns that cannot be captured by simple linear models.

k-Nearest Neighbours (KNN). k-Nearest Neighbours predicts arrival delay by comparing a new record with similar records in the training data. The prediction is based on the delays observed for the nearest neighbouring examples. KNN was included to represent instance-based learning, where prediction depends on local similarity rather than an explicit fitted equation or tree structure.

Multi-Layer Perceptron (MLP). The Multi-Layer Perceptron is a feed-forward neural network consisting of one or more hidden layers. It learns non-linear relationships by applying weighted transformations and activation functions to the input features. In this thesis, MLP was used as a conventional neural network baseline for structured tabular delay prediction.

Decision Tree (DT). Decision Tree regression splits the input space into a set of decision rules and predicts arrival delay from the average target value in the resulting

leaf nodes. It can capture non-linear relationships and feature interactions in an interpretable way. However, a single decision tree can be sensitive to small changes in the data, so it was mainly used as a simple tree-based baseline.

Random Forest (RF). Random Forest is an ensemble of decision trees trained using bootstrap samples and random feature selection. The final prediction is obtained by averaging the predictions of many individual trees. RF was included because it is usually more stable than a single decision tree and can capture non-linear feature interactions in tabular data.

AdaBoost (AB). AdaBoost is a boosting method that combines a sequence of weak learners. Each new learner gives more attention to observations that were predicted poorly by earlier learners. In this thesis, AB was included as a boosting-based baseline to test whether sequential error correction could improve arrival delay prediction.

Extra Trees (ET). Extra Trees is a tree-based ensemble method similar to Random Forest, but it introduces additional randomness when selecting split points. This can reduce variance and improve generalisation in some tabular prediction problems. ET was included to provide another tree-ensemble baseline with a different source of model diversity.

Gradient Boosting (GB). Gradient Boosting builds an ensemble of weak learners sequentially, where each new tree attempts to correct the errors made by the previous trees. It is well suited to modelling complex non-linear relationships and feature interactions. In this thesis, GB was one of the main tree-based baseline models and was also used for supporting interpretability analysis.

Extreme Gradient Boosting (XGB). Extreme Gradient Boosting is an efficient and regularised implementation of gradient boosting. It includes mechanisms such as shrinkage, tree regularisation, and controlled tree growth, which can improve predictive performance and reduce overfitting. XGB was included as a strong boosting baseline for structured train delay prediction.

8.4.2 Tabular Deep Learning Models

The tabular deep learning model pool included neural architectures designed specifically for structured tabular data. These models were used to examine whether modern deep

learning approaches could provide complementary predictive behaviour to conventional machine learning models. Unlike image or sequence deep learning architectures, these models are designed to learn from mixed numerical and categorical tabular features.

FT-Transformer (FTT). FT-Transformer is a transformer-based model for tabular data (Gorishniy et al., 2021). It applies attention mechanisms to transformed feature representations, allowing the model to learn interactions between input variables. In this thesis, FTT was included to represent attention-based deep learning for structured railway delay data.

TabNet. TabNet is a deep learning model that uses sequential attention to select relevant features at different decision steps (Arik and Pfister, 2021). This allows the model to focus on different parts of the input feature set when making predictions. TabNet was included because it provides an interpretable attention-based approach for tabular prediction tasks.

TabPFN. TabPFN is a tabular prediction model based on prior-data fitted networks (Hollmann et al., 2025). It is designed to make predictions on tabular datasets by using knowledge learned from many synthetic tasks during pretraining. In this thesis, TabPFN was included as a modern tabular deep learning baseline that can provide strong performance without extensive task-specific tuning.

TabM. TabM is a recent tabular deep learning model designed to improve prediction performance on structured data through efficient neural modelling (Gorishniy et al., 2025). It was included to assess whether newer tabular neural architectures could provide useful complementary predictions within the ensemble framework.

TabICL. TabICL is an in-context learning model for tabular data (Qu et al., 2026). It aims to make predictions by using the structure of the available training examples as context, rather than relying only on conventional parameter fitting for a single dataset. In this thesis, TabICL was included to represent recent in-context learning approaches for tabular regression.

8.4.3 Hyperparameters Considered for the Base Models

Hyperparameter tuning was applied to the candidate base models before the final model comparison and ensemble construction. Since the experiments in this thesis were

conducted across 26 sections, the final selected hyperparameter values could differ from one section to another. In addition, several base models were considered, and each model included a different number of tunable parameters. Reporting the exact selected values for every model and every section would therefore require a very large number of tables. For example, reporting section-specific hyperparameters for 26 sections and 12 base models would require hundreds of model–section configurations. Many of these configurations would also contain several hyperparameters per model. Such a detailed listing would substantially increase the length of the appendix, exceed the practical page limit, and reduce readability. More importantly, the exact selected hyperparameter values are outputs of the tuning process, not fixed methodological assumptions. The more important methodological information is the search space within which each model was allowed to vary during optimisation. Therefore, this appendix reports the hyperparameters considered for each base model and the corresponding search ranges or candidate values. This provides a compact and reproducible description of the tuning process without listing every section-specific final configuration. Where the same hyperparameter was tuned using slightly different candidate values in different experimental scripts, the table reports the combined overall search range. For example, for tree depth, the minimum and maximum values considered across the relevant tree-based models are reported rather than listing every individual candidate value. Table 8.2 summarises the hyperparameters considered for the base models, their search ranges or candidate values, and their role in controlling model behaviour.

Table 8.2: Hyperparameters considered for the base models.

Model	Hyperparameter	Range / values considered	Purpose
LR – Linear Regression			
	No tuned hyperparameter	Default configuration	Ordinary least-squares linear regression was used without hyperparameter tuning.
KNN – K-Nearest Neighbours			
	`n_neighbors`	3–31	Controls the number of neighbouring samples used for prediction.
	`weights`	`uniform`, `distance`	Determines whether all neighbours have equal influence or closer neighbours have higher influence.
	`p`	1, 2	Defines the distance metric, corresponding to Manhattan and Euclidean distance.
	`leaf_size`	20–40	Controls the tree structure used for efficient neighbour search.
ENet – Elastic Net			
	`alpha`	10^{-4}–1.0	Controls the overall regularisation strength.
	`l1_ratio`	0.05–0.95	Controls the balance between Lasso and Ridge penalties.

Continued on next page

Table 8.2: Hyperparameters considered for the base models (continued).

Model	Hyperparameter	Range / values considered	Purpose
	`max_iter`	2000–10000	Sets the maximum number of optimisation iterations.
	`selection`	`cyclic`, `random`	Defines the coefficient update strategy during optimisation.
DT – Decision Tree			
	`max_depth`	`None`; 3–30	Controls the maximum depth of the tree.
	`min_samples_split`	2–50	Sets the minimum number of samples required to split an internal node.
	`min_samples_leaf`	1–25	Sets the minimum number of samples required at a leaf node.
	`max_features`	`None`, `auto`, `sqrt`, `log2`, 1.0	Controls the number of features considered when finding the best split.
	`ccp_alpha`	0–0.01	Controls cost-complexity pruning to reduce overfitting.
RF – Random Forest			
	`n_estimators`	100–500	Controls the number of trees in the forest.
	`max_depth`	`None`; 3–30	Controls the maximum depth of each tree.
	`min_samples_split`	2–30	Sets the minimum number of samples required to split an internal node.

Continued on next page

Table 8.2: Hyperparameters considered for the base models (continued).

Model	Hyperparameter	Range / values considered	Purpose
	`min_samples_leaf`	1–15	Sets the minimum number of samples required at a leaf node.
	`max_features`	`auto`, `sqrt`, `log2`, 1.0	Controls the number of features considered at each split.
	`bootstrap`	`True`, `False`	Determines whether bootstrap samples are used when building trees.
SVR – Support Vector Regression			
	`C`	0.1–50.0	Controls the penalty applied to prediction errors.
	`epsilon`	0.01–1.0	Defines the width of the insensitive error tube.
	`gamma`	`scale`, `auto`; 0.01–0.1	Controls the kernel coefficient for non-linear mapping.
MLP – Multi-Layer Perceptron			
	`hidden_layer_sizes`	1–3 hidden layers; 32–128 neurons per layer	Defines the depth and width of the neural network architecture.
	`activation`	`relu`, `tanh`	Selects the activation function used in the hidden layers.
	`solver`	`adam`	Defines the optimisation algorithm used for training.

Continued on next page

Table 8.2: Hyperparameters considered for the base models (continued).

Model	Hyperparameter	Range / values considered	Purpose
	`alpha`	10^{-5}–10^{-2}	Controls L2 regularisation strength.
	`learning_rate_init`	10^{-4}–10^{-2}	Sets the initial learning rate.
	`max_iter`	500–1000	Sets the maximum number of training iterations.
	`early_stopping`	`True`	Stops training when validation performance does not improve.
BR – Bayesian Ridge Regression			
	`alpha_1`	10^{-7}–10^{-4}	Controls the prior shape parameter for the noise precision.
	`alpha_2`	10^{-7}–10^{-4}	Controls the prior inverse-scale parameter for the noise precision.
	`lambda_1`	10^{-7}–10^{-4}	Controls the prior shape parameter for the weight precision.
	`lambda_2`	10^{-7}–10^{-4}	Controls the prior inverse-scale parameter for the weight precision.
	`max_iter`	300–1000	Sets the maximum number of optimisation iterations.
	`tol`	10^{-4}–10^{-2}	Defines the convergence tolerance.
AB – AdaBoost			

Continued on next page

Table 8.2: Hyperparameters considered for the base models (continued).

Model	Hyperparameter	Range / values considered	Purpose
	`n_estimators`	50–1000	Controls the number of weak learners in the boosting process.
	`learning_rate`	0.01–1.0	Controls the contribution of each weak learner.
	`loss`	`linear`, `square`, `exponential`	Selects the loss function used during boosting.
GB – Gradient Boosting			
	`n_estimators`	50–1000	Controls the number of boosting stages.
	`learning_rate`	0.01–0.2	Controls the contribution of each tree.
	`max_depth`	2–30	Controls the maximum depth of each regression tree.
	`min_samples_split`	2–30	Sets the minimum number of samples required to split an internal node.
	`min_samples_leaf`	1–20	Sets the minimum number of samples required at a leaf node.
	`subsample`	0.6–1.0	Controls the fraction of samples used for each boosting stage.
	`max_features`	`None`, `auto`, `sqrt`, `log2`	Controls the number of features considered when splitting.
ET – Extra Trees			

Continued on next page

Table 8.2: Hyperparameters considered for the base models (continued).

Model	Hyperparameter	Range / values considered	Purpose
	`n_estimators`	50–1000	Controls the number of extremely randomised trees.
	`max_depth`	`None`; 3–30	Controls the maximum depth of each tree.
	`min_samples_split`	2–30	Sets the minimum number of samples required to split an internal node.
	`min_samples_leaf`	1–15	Sets the minimum number of samples required at a leaf node.
	`max_features`	`auto`, `sqrt`, `log2`, 1.0	Controls the number of features considered at each split.
	`bootstrap`	`True`, `False`	Determines whether bootstrap sampling is used.
XGB – Extreme Gradient Boosting			
	`n_estimators`	50–1000	Controls the number of boosting rounds.
	`learning_rate`	0.01–0.2	Controls the contribution of each boosting step.
	`max_depth`	2–30	Controls the maximum depth of each boosted tree.
	`min_child_weight`	1–20	Controls the minimum child-node weight required for further partitioning.
	`subsample`	0.6–1.0	Controls the fraction of training samples used for each tree.

Continued on next page

Table 8.2: Hyperparameters considered for the base models (continued).

Model	Hyperparameter	Range / values considered	Purpose
	`colsample_bytree`	0.6–1.0	Controls the fraction of features used for each tree.
	`gamma`	0–0.2	Controls the minimum loss reduction required to make a split.
	`reg_alpha`	0–1	Controls L1 regularisation.
	`reg_lambda`	0.1–10	Controls L2 regularisation.

8.5 Example of a Raw STOMP Message

Figure 8.1 presents an example of a raw railway message received from the Darwin PushPort data stream through the STOMP connection. The message is shown before route filtering, service-state processing, and conversion into the section-level format used by the prediction models.

The raw message contains service and location information together with planned, estimated, or actual timing updates for one or more calling points. Depending on the message type and the current state of the train journey, some timing fields may be unavailable or may later be revised by a subsequent update. The real-time processing pipeline extracts the required fields, matches the service to the timetable and route definitions, and converts the relevant information into a structured route-section prediction record.

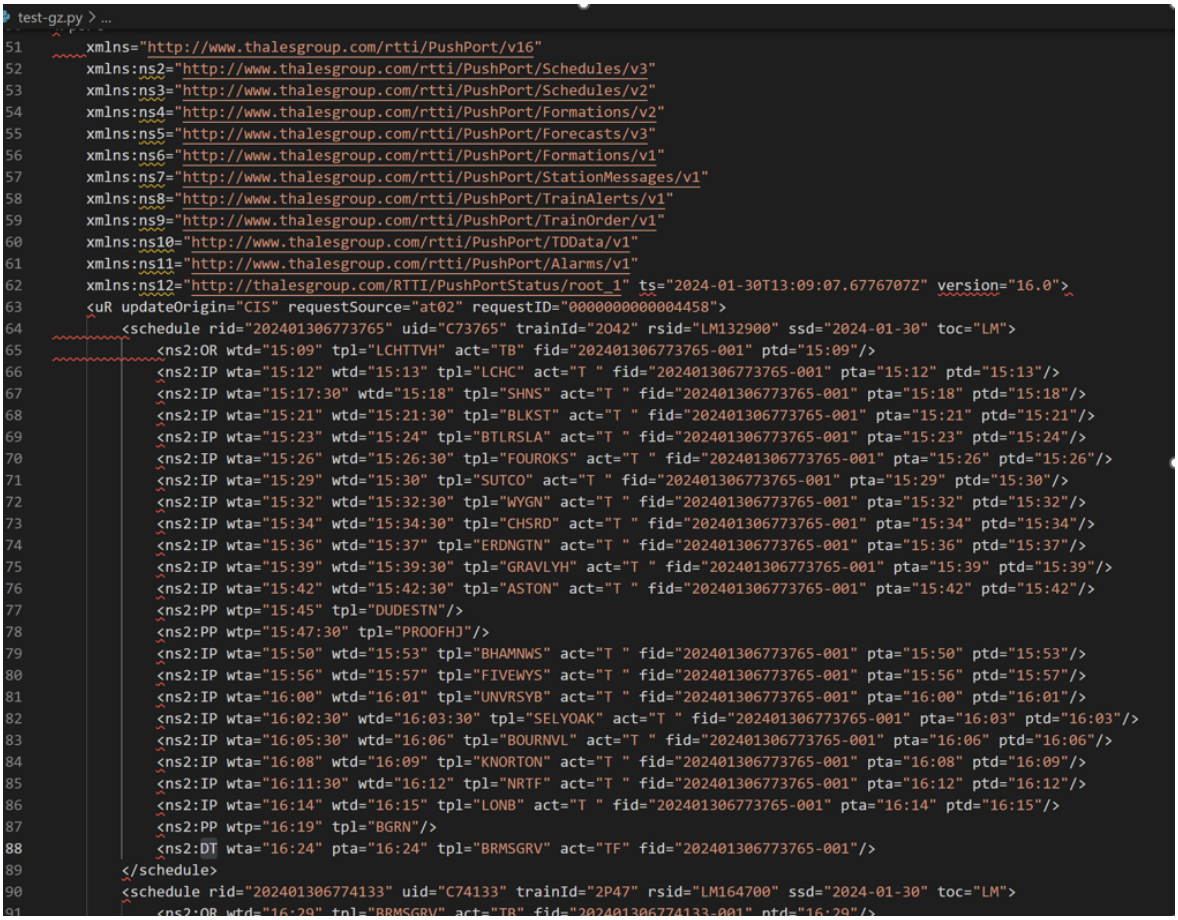


Figure 8.1: Example of a raw railway message received through the STOMP connection before filtering and transformation.

8.6 Operational Dashboard Prediction Fields

The lower part of the operational dashboard presents the prediction outputs in tabular form. Table 8.3 summarises the main fields displayed in the dashboard prediction tables and explains their operational meaning.

Table 8.3: Description of the fields displayed in the operational dashboard prediction tables.

Displayed field	**Description**
Route	The monitored route associated with the train service.
Train service identifier	The unique real-time identifier used to track a specific train service.
Route section	The station-to-station section for which the prediction is generated.
Station pair	The full names of the first and second stations of the predicted route section.
Departure information	Indicates whether the prediction is based on a confirmed actual departure or an estimated departure.
Prediction strategy	Identifies the BOEM model or fallback strategy used to generate the prediction.
Departure delay	The departure-delay value used as an input to the real-time prediction process.
Predicted arrival delay	The arrival delay predicted by the real-time BOEM framework.
Railway arrival estimate	The arrival-delay estimate provided by the live railway operational feed and retained as a comparison value.
Last update time	The most recent time at which the corresponding prediction record was updated.

These fields provide traceability between the live operational information received for each service, the prediction strategy applied by the framework, and the resulting arrival-delay prediction.

8.7 Summary

This appendix provided supplementary information on the final dataset features, base models and hyperparameter settings, and the real-time implementation. It also presented an example of a raw STOMP message and described the main prediction fields displayed in the operational dashboard.

Bibliography

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Al Ghamdi, M., Parr, G., and Wang, W. (2020). Weighted ensemble methods for predicting train delays. In *International Conference on Computational Science and Its Applications*, pages 586–600. Springer.

Al Ghamdi, M., Parr, G., and Wang, W. (2024). Heterogeneous machine learning ensembles for predicting train delays. *IEEE Transactions on Intelligent Transportation Systems*, 25(6):5138–5153.

Ardıç, Ö. (2024). Forecasting train arrival delays on the ankara–eskişehir high-speed line in turkey. *Journal of Rail Transport Planning & Management*, 30:100448.

Arik, S. Ö. and Pfister, T. (2021). Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, pages 6679–6687.

Arthaud, F., Lecoeur, G., and Pierre, A. (2024). Transformers à grande vitesse: Massively parallel real-time predictions of train delay propagation. *Journal of Rail Transport Planning & Management*, 29:100418.

Bao, X., Li, Y., Li, J., Shi, R., and Ding, X. (2021). Prediction of train arrival delay using hybrid elm-pso approach. *Journal of Advanced Transportation*, 2021(1):7763126.

Barbour, W., Mori, J. C. M., Kuppa, S., and Work, D. B. (2018). Prediction of arrival times of freight traffic on us railroads using support vector regression. *Transportation Research Part C: Emerging Technologies*, 93:211–227.

Bifet, A., Gavalda, R., Holmes, G., and Pfahringer, B. (2023). *Machine learning for data streams: with practical examples in MOA*. MIT press.

Boateng, V. and Yang, B. (2023a). Ensemble stacking with the multi-layer perceptron neural network meta-learner for passenger train delay prediction. In *2023 IEEE Conference on Artificial Intelligence (CAI)*, pages 21–22. IEEE.

Boateng, V. A. and Yang, B. (2023b). A global modeling pruning ensemble stacking with deep learning and neural network meta-learner for passenger train delay prediction. *IEEE Access*, 11:62605–62615.

Borin, V. P., Sant'Ana, J. M. d. S., Raheel, U., and Mahmood, N. H. (2026). Integrating meteorological and operational data: A novel approach to understanding railway delays in finland. *arXiv preprint arXiv:2601.16592*.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., and Wagenaar, J. (2014). An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37.

Cerqueira, V., Torgo, L., and Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11):1997–2028.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., Chen, K., Mitchell, R., Cano, I., Zhou, T., et al. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4.

Corman, F. and Kecman, P. (2018). Stochastic prediction of train delays in real-time using bayesian networks. *Transportation Research Part C: Emerging Technologies*, 95:599–615.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Ding, X., Xu, X., Li, J., and Shi, R. (2021). A train delays prediction model under different causes based on mtgnn approach. In *2021 IEEE international intelligent transportation systems conference (ITSC)*, pages 2387–2392. IEEE.

Dou, X. (2020). Flight arrival delay prediction and analysis using ensemble learning. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 836–840. IEEE.

Dutta, H. (2009). Measuring diversity in regression ensembles. In *IICAI*, volume 9, page 17p.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37.

Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., and Suganthan, P. N. (2022). Ensemble deep learning: A review. *Engineering applications of artificial intelligence*, 115:105151.

Gao, B., Zhang, L., Ou, D., and Dong, D. (2023). A novel deep learning model for short-term train delay prediction. *Information Sciences*, 645:119270.

Gao, T., Chen, J., and Xu, H. (2024). Data-driven train delay prediction incorporating dispatching commands: An xgboost-metaheuristic framework. *IET Intelligent Transport Systems*, 18(10):1777–1796.

Gashler, M., Giraud-Carrier, C., and Martinez, T. (2008). Decision tree ensemble: Small heterogeneous is better than large homogeneous. In *2008 Seventh international conference on machine learning and applications*, pages 900–905. IEEE.

Gaurav, R. and Srivastava, B. (2018). Estimating train delays in a large rail network using a zero shot markov model. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1221–1226. IEEE.

Golchehreh, M., Parr, G., and Wang, W. (2026). Multi-metric weighted ensemble method for train delay prediction. In *2026 11th International Conference on Machine Learning Technologies (ICMLT)*, pages 511–517. IEEE.

Gorishniy, Y., Kotelnikov, A., and Babenko, A. (2025). Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *International Conference on Learning Representations*, volume 2025, pages 77899–77935.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. (2022). Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520.

Hallowell, S. F. and Harker, P. T. (1996). Predicting on-time line-haul performance in scheduled railroad operations. *Transportation Science*, 30(4):364–378.

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *nature*, 585(7825):357–362.

Hauck, F. and Kliewer, N. (2020). Data analytics in railway operations: Using machine learning to predict train delays. In Neufeld, J. S., Buscher, U., Lasch, R., Möst, D., and Schönberger, J., editors, *Operations Research Proceedings 2019*, pages 741–747. Springer.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. (2025). Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326.

Huang, P., Lessan, J., Wen, C., Peng, Q., Fu, L., Li, L., and Xu, X. (2020a). A bayesian network model to predict the effects of interruptions on train operations. *Transportation Research Part C: Emerging Technologies*, 114:338–358.

Huang, P., Wen, C., Fu, L., Lessan, J., Jiang, C., Peng, Q., and Xu, X. (2020b). Modeling train operation as sequences: A study of delay prediction with operation and weather data. *Transportation research part E: logistics and transportation review*, 141:102022.

Huang, P., Wen, C., Fu, L., Peng, Q., and Tang, Y. (2020c). A deep learning approach for multi-attribute data: A study of train delay prediction in railway systems. *Information Sciences*, 516:234–253.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer.

Huber, P. J. and Ronchetti, E. M. (2009). *Robust Statistics*. John Wiley & Sons.

Kecman, P. and Goverde, R. M. (2015). Predictive modelling of running and dwell times in railway traffic. *Public Transport*, pages 295–319.

Kuipers, R. (2024). *Dwell time delays for commuter trains: An analysis of the influence of passengers on dwell time delays*. Number 331. Lund University.

Laifa, H., Ghezalaa, H. H. B., et al. (2021). Train delay prediction in tunisian railway through lightgbm model. *Procedia Computer Science*, 192:981–990.

Lapamonpinyo, P., Derrible, S., and Corman, F. (2022). Real-time passenger train delay prediction using machine learning: A case study with amtrak passenger train routes. *IEEE Open Journal of Intelligent Transportation Systems*, 3:539–550.

Leng, N. and Corman, F. (2022). Communicating delays and adjusted disposition timetables: Modelling and evaluating the impact of incomplete information to passengers. *Expert Systems with Applications*, 191:116265.

Li, J., Xu, X., Ding, X., Liu, J., and Ran, B. (2024a). Bayesian spatio-temporal graph convolutional network for railway train delay prediction. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):8193–8208.

Li, J., Xu, X., Shi, R., and Ding, X. (2021a). Train arrival delay prediction based on spatial-temporal graph convolutional network to sequence model. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2399–2404. IEEE.

Li, Z., Dong, W., Ji, Y., Luo, J., and Huang, P. (2026). Integrating graph and neural relational inference for network-wide train delay prediction: An equilibrium between accuracy and interpretability. *Transportation Research Part C: Emerging Technologies*, 182:105418.

Li, Z., Huang, P., Wen, C., Dong, W., Ji, Y., and Rodrigues, F. (2024b). Railway network delay evolution: A heterogeneous graph neural network approach. *Applied Soft Computing*, 159:111640.

Li, Z., Wen, C., Hu, R., Xu, C., Huang, P., and Jiang, X. (2021b). Near-term train delay prediction in the dutch railways network. *International Journal of Rail Transportation*, 9(6):520–539.

Ling, Q.-H., Song, Y.-Q., Han, F., Yang, D., and Huang, D.-S. (2016). An improved ensemble of random vector functional link networks based on particle swarm optimization with double optimization strategy. *Plos one*, 11(11):e0165803.

Lu, Z., Xia, J., Wang, M., Nie, Q., and Ou, J. (2020). Short-term traffic flow forecasting via multi-regime modeling and ensemble learning. *Applied Sciences*, 10(1):356.

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., and Lee, S.-I. (2020). From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Marković, N., Milinković, S., Tikhonov, K. S., and Schonfeld, P. (2015). Analyzing passenger train arrival delays with support vector regression. *Transportation Research Part C: Emerging Technologies*, 56:251–262.

McElfresh, D., Khandagale, S., Valverde, J., Prasad C, V., Ramakrishnan, G., Goldblum, M., and White, C. (2023). When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36:76336–76369.

McKinney, W. et al. (2011). pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9.

Mienye, I. D. and Sun, Y. (2022). A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *Ieee Access*, 10:99129–99149.

Mohammed, A. and Kora, R. (2023). A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University-Computer and Information Sciences*, 35(2):757–774.

Nair, R., Hoang, T. L., Laumanns, M., Chen, B., Cogill, R., Szabó, J., and Walter, T. (2019). An ensemble prediction model for train delays. *Transportation Research Part C: Emerging Technologies*, 104:196–209.

National Rail (2026). Darwin Data Feeds. Accessed: 27 June 2026.

National Rail Enquiries (2023). Darwin: Push port. https://wiki.openraildata.com/index.php/Darwin:Push_Port. Open Rail Data Wiki. Accessed: 27 February 2026.

Network Rail (2015). Wessex route study. Technical report, Network Rail. Accessed: 25 July 2026.

Network Rail (2016). Anglia route study. Technical report, Network Rail. Accessed: 25 July 2026.

Network Rail (2022). Railway performance. https://www.networkrail.co.uk/who-we-are/how-we-work/performance/railway-performance/. Accessed: 1 September 2022.

Network Rail (2026). Railway Performance. https://www.networkrail.co.uk/who-we-are/how-we-work/performance/railway-performance/. Accessed: 27 May 2026.

Office of Rail and Road (2026). How many trains arrive on time? Data Portal. [ORR].

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., Mazzino, N., and Anguita, D. (2016). Advanced analytics for train delay prediction systems by including exogenous weather data. In *2016 IEEE international conference on data science and advanced analytics (DSAA)*, pages 458–467. IEEE.

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., Mazzino, N., and Anguita, D. (2018). Train delay prediction systems: a big data analytics perspective. *Big data research*, 11:54–64.

Open Rail Data (2023). HSP – open rail data wiki. Accessed: 18-February-2025, https://wiki.openraildata.com/index.php/HSP.

Open Rail Data Wiki (2023). SCHEDULE. Accessed: 27 June 2026.

OpenAI (2023). GPT-4. https://openai.com/index/gpt-4/. OpenAI website. Accessed: 4 November 2026.

OpenAI (2025). Introducing GPT-5. https://openai.com/index/introducing-gpt-5/. OpenAI website. Accessed: 4 November 2026.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Pavlyuk, D. (2020). Towards ensemble learning of traffic flows' spatiotemporal structure. *Transportation Research Procedia*, 47:361–368.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Pongnumkul, S., Pechprasarn, T., Kunaseth, N., and Chaipah, K. (2014). Improving arrival time prediction of thailand's passenger trains using historical travel times. In *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 307–312. IEEE.

Pritchard, J., Sadler, J., Blainey, S., Waldock, I., and Austin, J. (2021). Predicting and mitigating small fluctuations in station dwell times. *Journal of Rail Transport Planning & Management*, 18:100249.

Qu, J., Holzmüller, D., Varoquaux, G., and Morvan, M. L. (2026). Tabiclv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*.

Rail Delivery Group (2026). National Rail Enquiries Data Feeds. Accessed: 27 June 2026.

Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schröder, B., Thuiller, W., et al. (2017). Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. *Ecography*, 40(8):913–929.

Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 8(4):e1249.

Sahoo, R., Pasayat, A. K., Bhowmick, B., Fernandes, K., and Tiwari, M. K. (2022). A hybrid ensemble learning-based prediction model to minimise delay in air cargo transport using bagging and stacking. *International Journal of Production Research*, 60(2):644–660.

Sarhani, M. and Voß, S. (2024). Prediction of rail transit delays with machine learning: How to exploit open data sources. *Multimodal Transportation*, 3(2):100120.

Shahhosseini, M., Hu, G., and Pham, H. (2022). Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Machine Learning with Applications*, 7:100251.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Shi, R., Xu, X., Li, J., and Li, Y. (2021). Prediction and analysis of train arrival delay based on xgboost and bayesian optimization. *Applied Soft Computing*, 109:107538.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.

Spanninger, T., Trivella, A., Büchel, B., and Corman, F. (2022). A review of train delay prediction approaches. *Journal of Rail Transport Planning & Management*, 22:100312.

Srinivas, C., Asritha, K., Legala, S. R., Katarpu, S., et al. (2025). Ensemble learning for accurate train delay prediction.

Taleongpong, P., Hu, S., Jiang, Z., Wu, C., Popo-Ola, S., and Han, K. (2022). Machine learning techniques to predict reactionary delays and other associated key performance indicators on british railway network. *Journal of Intelligent Transportation Systems*, 26(3):311–329.

Tang, R., Liu, R., and Lin, Z. (2025). Predicting primary delay of train services using graph-embedding based machine learning. *Journal of Rail Transport Planning & Management*, 34:100518.

Tashman, L. J. (2000). Out-of-sample tests of forecasting accuracy: an analysis and review. *International journal of forecasting*, 16(4):437–450.

Tiong, K. Y., Ma, Z., and Palmqvist, C.-W. (2023a). Analyzing factors contributing to real-time train arrival delays using seemingly unrelated regression models. *Transportation Research Part A: Policy and Practice*, 174:103751.

Tiong, K. Y., Ma, Z., and Palmqvist, C.-W. (2023b). A review of data-driven approaches to predict train delays. *Transportation Research Part C: Emerging Technologies*, 148:104027.

Usmani, M., Memon, Z. A., Danyaro, K. U., and Qureshi, R. (2024). Optimized multi-level multi-type ensemble (omme) forecasting model for univariate time series. *IEEE Access*, 12:35700–35715.

Wadsworth, S. and Niemi, J. (2025). Bayesian stacking via proper scoring rule optimization using a gibbs posterior. *arXiv preprint arXiv:2509.04203*.

Wang, M., Qin, Y., Wang, L., Wang, X., and Du, X. (2023a). Research on train delay forecast of high-speed railway based on data driving. In *International Conference on Electrical and Information Technologies for Rail Transportation*, pages 627–634. Springer.

Wang, W. (2008). Some fundamental issues in ensemble methods. In *IEEE World Congress on Computational Intelligence*, pages 2243–2250.

Wang, X., Hyndman, R. J., Li, F., and Kang, Y. (2023b). Forecast combinations: An over 50-year review. *International Journal of Forecasting*, 39(4):1518–1547.

Wen, C., Mou, W., Huang, P., and Li, Z. (2020). A predictive model of train delays on a railway line. *Journal of Forecasting*, 39(3):470–488.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, pages 79–82.

Wu, Y., Liu, L., Xie, Z., Chow, K.-H., and Wei, W. (2021). Boosting ensemble accuracy by revisiting ensemble diversity metrics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16469–16477.

Xu, J., Wang, W., Gao, Z., Luo, H., and Wu, Q. (2023). A novel markov model for near-term railway delay prediction. *Computers & Industrial Engineering*, 181:109302.

Yaghini, M., Khoshraftar, M. M., and Seyedabadi, M. (2013). Railway passenger train delay prediction via neural network model. *Journal of advanced transportation*, 47(3):355–368.

Yang, S., Guo, M., Wu, J., Chen, Y., Huang, G., and Zeng, B. (2024). Hybrid inverted transformer-cnn model for train primary-delay recovery time prediction. In *Proceedings of the 4th Asia-Pacific Artificial Intelligence and Big Data Forum*, pages 1103–1110.

Yong, T. K., Ma, Z., and Palmqvist, C.-W. (2025). Ap-grip evaluation framework for data-driven train delay prediction models: systematic literature review. *European Transport Research Review*, 17(1):13.

Zhang, Y., Liao, L., Yu, Q., Ma, W., and Li, K. (2021). Using the gradient boosting decision tree (gbdt) algorithm for a train delay prediction model considering the delay propagation feature. *Advances in Production Engineering & Management*, 16(3):285–296.

Zhou, Z.-H. (2025). *Ensemble methods: foundations and algorithms*. CRC press.

Zilko, A. A., Kurowicka, D., and Goverde, R. M. (2016). Modeling railway disruption lengths with copula bayesian networks. *Transportation Research Part C: Emerging Technologies*, 68:350–368.