MACHINE LEARNING METHODS FOR DETECTING AND CORRECTING DATA ERRORS IN WATER LEVEL TELEMETRY SYSTEMS

Thakolpat Khampuengson

A thesis submitted to
the School of Computing Sciences of The University of East Anglia
in fulfilment of the requirements for the degree of
DOCTOR OF PHILOSOPHY.
SEPTEMBER, 2022

Abstract

Water level data from telemetry stations can be used for early warning to prevent risk situations, such as floods and droughts. However, there is a possibility that the equipment in the telemetry station may fail, which will lead to errors in the data, resulting in false alarms or no warning of true alarms. Manually examining data is time-consuming and require expertise. As a result, the automated system is required. There are several algorithms available for detecting and correcting anomalous data, but the question remains as to which algorithm would be most suitable for telemetry data. To investigate and identify such an algorithm, statistical models, machine learning models, deep learning models, and reinforcement learning models are implemented and evaluated.

For anomaly detection, we first evaluated statistical models using our modified sliding window algorithm called Only Normal Sliding Windows (ONSW) to assess their performance. We then proposed Deep Reinforcement Learning (DRL) models and compared them to Deep Learning models to determine their suitability for the task. Additionally, we developed a feature extraction approach that combines the saliency map and nearest neighbor extracted feature (SM+NNFE) to improve model performance. Various ensemble approaches were also implemented and compared to other competitive methods. For data imputation, we developed the Full Subsequence Matching (FSM) technique, which fills in missing values by imitating values from the most similar subsequence.

Based on the results, machine learning models with ONSW are the best option for identifying abnormalities in telemetry water level data. Additionally, a deep reinforcement learning model could be used to identify abnormalities in crucial stations requiring further attention. Regarding data imputation, our technique outperforms other competitive approaches when dealing with water level data influenced by tides. However, relying solely on a single or limited number of models may be risky, as their performance could deteriorate in the future without being realized. Therefore, building models using ensemble techniques is a viable option for reducing errors caused by this issue.

Acknowledgement

Firstly, I would like to express my deepest appreciation to my primary supervisor, Dr. Wenjia Wang, and my secondary supervisor, Dr. Anthony Bagnall, for the unlimited advice, guidance, and encouragement they have provided me throughout my PhD journey. I have been incredibly fortunate to have their supervision, as they have shown great care for my work and responded promptly to my questions and queries. This thesis would not have been possible without their guidance and persistent help, and it is a great honour for me to have conducted this research under their supervision.

I would like to thank the Office of the Civil Service Commission (OCSC) and Hydro-Informatics Institute in Thailand for providing me with a full scholarship to pursue my studies. I also extend my gratitude to my colleagues at HII for their contribution in labelling the data used in this research.

I want to express my deepest appreciation to my wife and son, who have been my unwavering mental and physical support throughout my PhD journey. They have patiently endured the long nights and irregular working hours.

Lastly, I want to thank my dear friends, Davis, Kris, Omar, Nabil, Lucas, and Tom, for providing me with wonderful times and endless laughter. Your support has been invaluable to me, and I cannot thank you all enough.

Table of Contents

Table of Contents iv

List of Tables vii

List of Figures x

1 Introduction 1
- 1.1 Background . . . 1
- 1.2 Motivation . . . 4
- 1.3 Anomaly Definition . . . 4
- 1.4 Research Questions . . . 5
- 1.5 Aims and Objectives . . . 5
- 1.6 Contributions of the Proposed Work . . . 6
- 1.7 Thesis Outline . . . 8

2 Telemetry Systems 10
- 2.1 Introduction . . . 10
- 2.2 Telemetry Station . . . 10
  - 2.2.1 Telemetry Station Components . . . 10
  - 2.2.2 Data Collection Methods . . . 12
- 2.3 Telemetry Datasets . . . 13
  - 2.3.1 Meteorological Datasets . . . 13
  - 2.3.2 Hydrological Datasets . . . 14
- 2.4 Telemetry Systems Overview. . . . 14
- 2.5 Causes and Type of Anomaly Data . . . 15
- 2.6 Data Quality . . . 19
- 2.7 Summary . . . 23

3 Technical Background and Related Work 24
- 3.1 Introduction . . . 24
- 3.2 Data Pre-processing Approaches . . . 24
- 3.3 Anomaly Detection Approaches . . . 26
  - 3.3.1 Statistical Approaches . . . 27
  - 3.3.2 Machine Learning Approaches . . . 29
  - 3.3.3 Neural Network Approaches . . . 33
- 3.4 Data Correction and Imputation Approaches . . . 38
  - 3.4.1 Statistical Approaches . . . 39
  - 3.4.2 Machine Learning Approaches . . . 40

3.4.3 Neural Network Approaches . . . 41
3.5 Ensemble Methods . . . 43
3.6 Evaluation . . . 45
3.6.1 Binary Class Performance Measures . . . 46
3.6.2 Critical Difference Diagram . . . 46
3.6.3 Evaluating imputation methods . . . 48
3.7 Related Work . . . 49
3.7.1 Related Work for Data Preprocessing . . . 49
3.7.2 Related Work for Anomaly Detection . . . 50
3.7.3 Related Work for Data Imputation . . . 55
3.7.4 Related Work for Ensemble Methods . . . 58
3.8 Summary . . . 60

4 Anomaly Detection with Statistical Models and Their Ensembles 61
4.1 Introduction . . . 61
4.2 Methods . . . 62
4.2.1 Statistical Models . . . 62
4.2.2 A Modified Sliding Window Algorithm . . . 64
4.2.3 Ensemble Methods . . . 64
4.3 Evaluation . . . 67
4.3.1 Datasets . . . 67
4.3.2 Evaluation Metrics . . . 69
4.4 Results . . . 69
4.4.1 Individual Model Results . . . 69
4.4.2 Simple Ensemble Results . . . 74
4.4.3 Complex Ensemble Results . . . 75
4.5 Discussion . . . 78
4.6 Summary . . . 79

5 Anomaly Detection with Reinforcement Learning and Their Ensemble 81
5.1 Introduction . . . 81
5.2 Methods . . . 82
5.2.1 Reinforcement Learning . . . 82
5.3 Evaluation . . . 87
5.3.1 Datasets . . . 87
5.3.2 Evaluation metrics and Comparison methods . . . 89
5.3.3 Parameter setting . . . 89
5.4 Results . . . 90
5.4.1 Accuracies of DRL models . . . 90
5.4.2 Performance on the Same Station . . . 94
5.4.3 Performance on the Different Station . . . 96
5.4.4 Ensemble Results . . . 100
5.5 Discussion . . . 108
5.6 Summary . . . 109

6 New Multiple Features Extraction Method for Anomaly Detection. 111
6.1 Introduction . . . 111
6.2 Methods . . . 112

6.2.1 Saliency Map . . . 112
6.2.2 Nearest Neighbour Features Extraction (NNFE) . . . 115
6.2.3 SM+NNFE . . . 116
6.3 Evaluation . . . 116
6.3.1 Datasets . . . 116
6.3.2 Evaluation Models . . . 117
6.3.3 Evaluation Metrics . . . 117
6.3.4 Experiment Setting . . . 117
6.4 Results . . . 118
6.5 Summary . . . 123

7 Subsequence Matching Approaches for Data Correction and Imputation 125
7.1 Introduction . . . 125
7.2 Methods . . . 126
7.2.1 Full Subsequence Matching (FSM) . . . 126
7.2.2 Partial Subsequence Matching (PSM) . . . 128
7.3 Evaluation . . . 130
7.3.1 Dataset . . . 130
7.3.2 Missing data generation . . . 133
7.3.3 Comparision imputation methods . . . 133
7.3.4 Experimental Setting . . . 133
7.3.5 Evaluating imputation methods . . . 134
7.4 Results . . . 135
7.4.1 Discussion . . . 143
7.5 Summary . . . 144

8 Evaluation 145
8.1 Introduction . . . 145
8.2 Overview of the Research . . . 145
8.3 Evaluation . . . 146
8.3.1 Evaluation of the Research Methods . . . 146
8.3.2 Comparison between our anomaly detection methods . . . 147
8.3.3 Comparison of our data imputation methods with external methods. . . . 160
8.4 Summary . . . 162

9 Conclusion and Further Work 163
9.1 Summary . . . 163
9.2 Conclusion . . . 166
9.2.1 Anomaly Detection . . . 166
9.2.2 Anomaly Correction . . . 167
9.3 Suggestions for Further Work . . . 168

Bibliography 170

List of Tables

2.1 The number of stations for each percentage of missing data in each year. 19
2.2 The average percentage of missing data in each year each river basin. . 20
2.3 The average percentage of missing data in each month each river basin. 22

3.1 Confusion Matrix of Performance Measures. 46

4.1 Summary of data from 17 telemetry stations. 68
4.2 The performance indexes of 7 methods on telemetry water level data. (Since some models cannot detect anomalies in particular datasets, computing those three metrics is meaningless.) 70
4.3 TSP score of individual model (best results are in bold). 72
4.4 The performance indexes of 13 ensemble methods on telemetry water level data. (Since some models cannot detect anomalies in particular datasets, computing those three metrics is meaningless.) 73
4.5 Total and average TSP scores of simple ensemble models. 75
4.6 The performance indexes of 6 complex ensemble models on telemetry water level data. (Since some models cannot detect anomalies in particular datasets, computing those three metrics is meaningless.) 76
4.7 Total and average TSP scores of complex ensembles. 77

5.1 Demographic summary of the water level data of 8 stations used in this research. 87
5.2 The performance of DRL when increasing the learning epochs (the best F1-score of each row shown in bold). 91
5.3 The mean F1-score and standard deviation of all DRL, MLP, and LSTM models when testing with the dataset from different stations (the best F1-score of each row is shown in bold). 95
5.4 The number of training epochs and the time spent on each epoch for each model. 96

5.5 The F1-score of the best DRL models when testing with the dataset from same station (show in the bracket) and different stations, while the average F1-score and standard deviations of each station were calculated without their own scores. 96
5.6 The F1-score of the MLP models when testing with the dataset from the same station (shown in the bracket) and different stations. 98
5.7 The F1-score of the LSTM models when testing with the dataset from the same station (shown in the bracket) and different stations. 99
5.8 The performance of ensemble models (the best F1-score of each row is shown in bold). 101
5.9 The mean F1-score and standard deviations of all of the DRL, MLP, LSTM, and ensemble of DRL-based models when testing with the dataset from different stations (the best F1-score of each station is shown in bold). 103
5.10 The mean F1-score of the $WEDRL_3$ models when testing with the dataset from the same station (shown in the bracket) and different stations. 104
5.11 The performance of the ensemble models built by combining DRL and candidate models (the best F1-score of each row is shown in bold). . . . 105
5.12 The mean F1-score and standard deviation of all models when testing with the dataset from different stations (the best F1-score of each station is shown in bold). 106
5.13 The F1-score of the E3 models when testing with the dataset from the same station (shown in the bracket) and different stations. 107

6.1 Average performance scores of five machine learning models, trained with three different sets of features: Saliency Map (SM), our new feature set, SM+NNFE, and the baseline feature set, TSFRESH. 119
6.2 Average performance score with ensemble models. 121

7.1 The average imputation performance indexes of 14 methods on telemetry water level data with tidal influence. 135
7.2 The average imputation performance indexes of 14 methods on telemetry water level data with irrigation influence. 138
7.3 The average imputation performance indexes of 14 methods on telemetry water level data with rain influence. 141

8.1 Summary of data from 33 telemetry stations. 147
8.2 The F1-score of each model on univariate telemetry water level data with different characteristics. 148

8.3 F1-score of each machine learning models with multivariate telemetry water level datasets. 149
8.4 The f1-score of machine learning model with univariate data when used and not used sliding windows techniques. 151
8.5 The f1-score of machine learning model with multivariate data when used and not used sliding windows techniques. 152
8.6 The F1-score of different models on the Yahoo dataset. 158
8.7 The average RMSE of eight methods on telemetry water level data with tidal behaviour on different missing gap size. 161

List of Figures

1.1 Location of HII telemetry stations in Thailand. 2
1.2 The reported telemetry water level data on the webpage. 3

2.1 Components of a telemetry station: the green square represents the sensor unit, the blue square represents the control and transmission unit, and the red square represents the power unit. 11
2.2 An example of external factors that influence on water level data characteristics 15
2.3 The overview of telemetry systems. 16
2.4 Example of data that has been affected by the environment in summer, and not affected in rainy season. The instrument gave unreasonable readings as water levels should not be oscillating abruptly by physics. . 17
2.5 An example of anomaly in water data 18
2.6 An example of expert-labelled data from CPY012 station, May 2016. . 21

3.1 A simple perceptron neural network. 33
3.2 An example form of multiLayer perceptron neural network. 34
3.3 An unrolled RNN 35
3.4 An example form of LSTM networks. 35
3.5 An example form of convolutional neural networks. 36
3.6 An example form of residual neural network 37
3.7 An example form of autoencoder model. 38
3.8 SVR concept illustration. 41
3.9 GAN concept illustration. 43
3.10 An example Critical Difference (CD) diagram demonstrating how to interpret the results from a pairwise comparison of five models over multiple datasets. 48
3.11 Most similar subsequence that has same dynamic but difference position 57

4.1 Critical difference diagram for different statistical models. 72
4.2 Critical difference diagram for 13 different simple ensemble models. . . 75

4.3 Critical difference diagram for complex ensemble models. 77
4.4 Critical difference diagram for all models. 78

5.1 Overall process of DRL. 85
5.2 Water level data from eight stations: CPY011, CPY012, CPY013, CPY014, CPY015, CPY016, CPY017, and YOM009 (**a**-**h**). The different colours show the partitions of the data for training (blue), validation (orange) and testing (green). The anomalies are indicated by red crosses, x. . . 88
5.3 A critical difference diagram for 5 different DRL models on different datasets of telemetry water level data. 92
5.4 F1-score when increasing the learning epochs at each station. 93
5.5 Anomaly detection from the best DRL model of each station. (**a**) CPY011 with DRL; (**b**) CPY012 with DRL_{Valid}; (**c**) CPY013 with DRL_{F1}; (**d**) CPY014 with DRL_{Rwd}; (**e**) CPY015 with DRL_{F1}; (**f**) CPY016 with DRL_{F1}; (**g**) CPY017 with DRL; (h) YOM009 with DRL_{Acc}. 94
5.6 A critical difference diagram of each model. 95
5.7 Bar charts of average F1-score of the DRL, MLP, and LSTM when tested with the data collected from different stations. 99
5.8 F1-score of ensemble model when increasing the learning epochs at CPY011, CPY012, CPY013, CPY014, CPY015, CPY016, CPY017, and YOM009 (**a**-**h**). 102
5.9 Critical difference diagram for DRL, MLP, LSTM, and ensemble of DRL-based models when testing with the dataset from different stations. . . 103
5.10 A critical difference diagram of all models when testing with the dataset from different stations. 106

6.1 An Example of the Saliency Map 114
6.2 The critical difference diagram for 3 different feature extraction techniques against the existing feature extraction technique (TSFRESH) and the telemetry water level data before applying any feature extraction technique (WL) to the telemetry water level. 121
6.3 The critical difference diagram for four different ensemble models versus five individual models on telemetry water level. Ensemble generated the highest ranked. 122

7.1 Water level data with tidal influences. 131
7.2 Water level data with irrigation influences. 132
7.3 Water level data with rain influences. 132

7.4 A critical difference diagram for 14 different imputation techniques on tidal influence datasets of telemetry water level data. 136
7.5 The performance for imputing telemetry water level data with tidal influence for various missing gap size. 137
7.6 The performance for imputing telemetry water level data with irrigation influence for various missing gap size. 139
7.7 A critical difference diagram for 14 different imputation techniques on irrigation influence datasets of telemetry water level data. 140
7.8 A critical difference diagram for 14 different imputation techniques on rain influence datasets of telemetry water level data. 141
7.9 The performance for imputing telemetry water level data with rain influence for various missing gap size. 142

8.1 Bar charts of average F1-score different models with univariate and multivariate telemetry water level data. 150
8.2 Bar charts of average F1-score of different machine learning models with and without the sliding windows techniques. 153
8.3 A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW). 153
8.4 A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW) on irrigation data behaviours. 154
8.5 A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW) on rain data behaviours. 154
8.6 A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW) on tidal data behaviours. 155
8.7 The accuracy of each model on different windows size. 156
8.8 Example data from Yahoo datasets with an anomaly on the orange cross, x (**a**-**d**). 157
8.9 Overall average F1-Score from different anomaly detection techniques on A1-A4 datasets. 159
8.10 A critical difference diagram of all models with Yahoo time-series dataset. 160
8.11 A critical difference diagram of our approach and candidate imputation models on tidal influence dataset of telemetry water level data. 161

8.12 The RMSE for imputing telemetry water level data with tidal influence for various missing gap sizes. 162

9.1 System Overview 169

Chapter 1

Introduction

1.1 Background

The Hydro Informatics Institute, or HII, has been studying, building, and deploying telemetry water level stations around Thailand since 2004 (as shown in Figure 1.1). HII telemetry systems are comprised of sensors that collect hydrological and meteorological data such as temperature, humidity, air pressure, precipitation, and water level. Every ten minutes, the station transmits the measured data to the HII data centre through cellular or satellite networks. The data is available online at "www.thaiwater.net" and in the "ThaiWater" mobile applications, which are available for free download on Google Play and the App Store.

Floods and droughts occur more frequently and more severely in many parts of the world as an effect of climate change. This results in significant damage to infrastructure and fatalities. Floods continue to be one of the most destructive natural catastrophes in Thailand, causing up to £34.2 billion in economic damage and losses each year (Bank, 2012). The use of telemetry data can help mitigate risks associated with these natural disasters. For example, in the case of floods, telemetry data can help predict the severity and location of flooding, allowing individuals and emergency services to prepare and evacuate those in high-risk areas. The risks associated with floods can include damage to property, infrastructure, and the environment, as well as loss of life and business disruption. Effective flood management can reduce the number of fatalities per flood event, despite the significant increase in the number of floods (Parker et al., 2007).

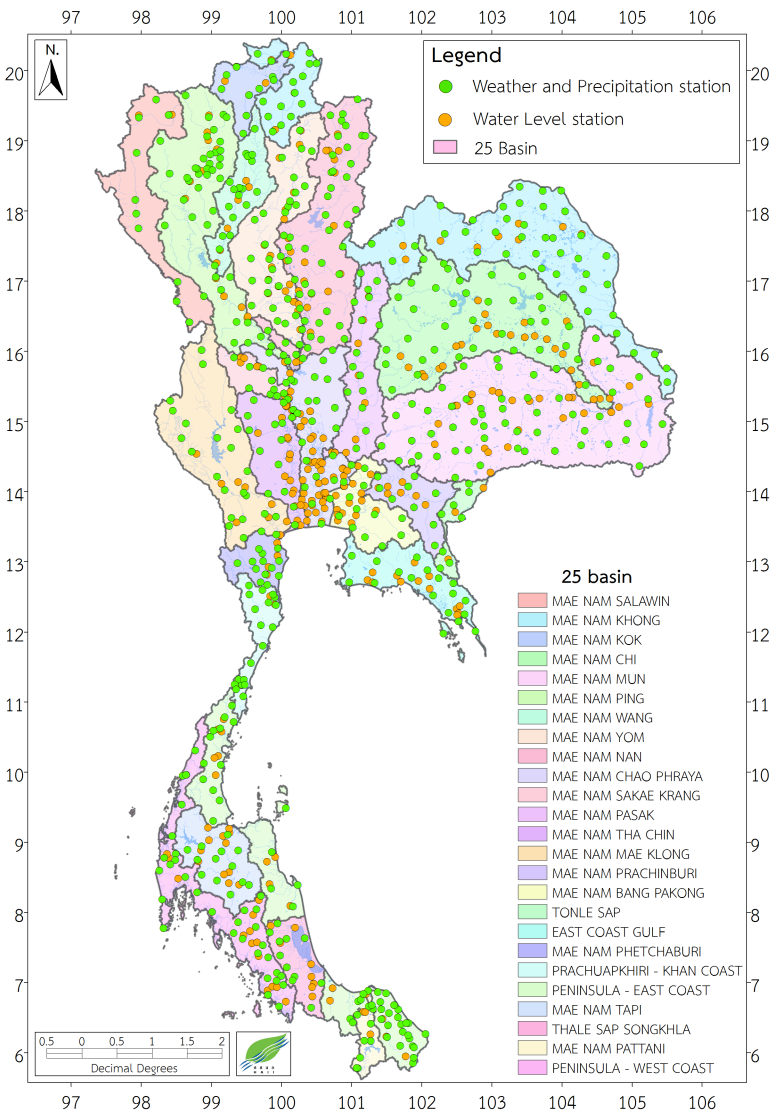


Figure 1.1: Location of HII telemetry stations in Thailand.

Similarly, in the case of droughts, telemetry data can assist in predicting the severity and duration of the drought, allowing for more effective planning and conservation efforts to be implemented. The risks associated with droughts can include agricultural losses, water shortages, environmental damage, and economic disruption. As a result, information about water levels will show how drought directly affects people and animals' lives (Senay et al., 2015).

To minimise damage and prevent loss of life, early warnings must be accurate and reliable. However, the collected data from the sensors at telemetry stations can be wrong due to some factors such as floating objects around the sensors, human or animal activities, or malfunctioning devices. Any errors in the data, generally referred to as anomalies, might lead to inaccurate decisions, such as false alarms or missed true

alarms.

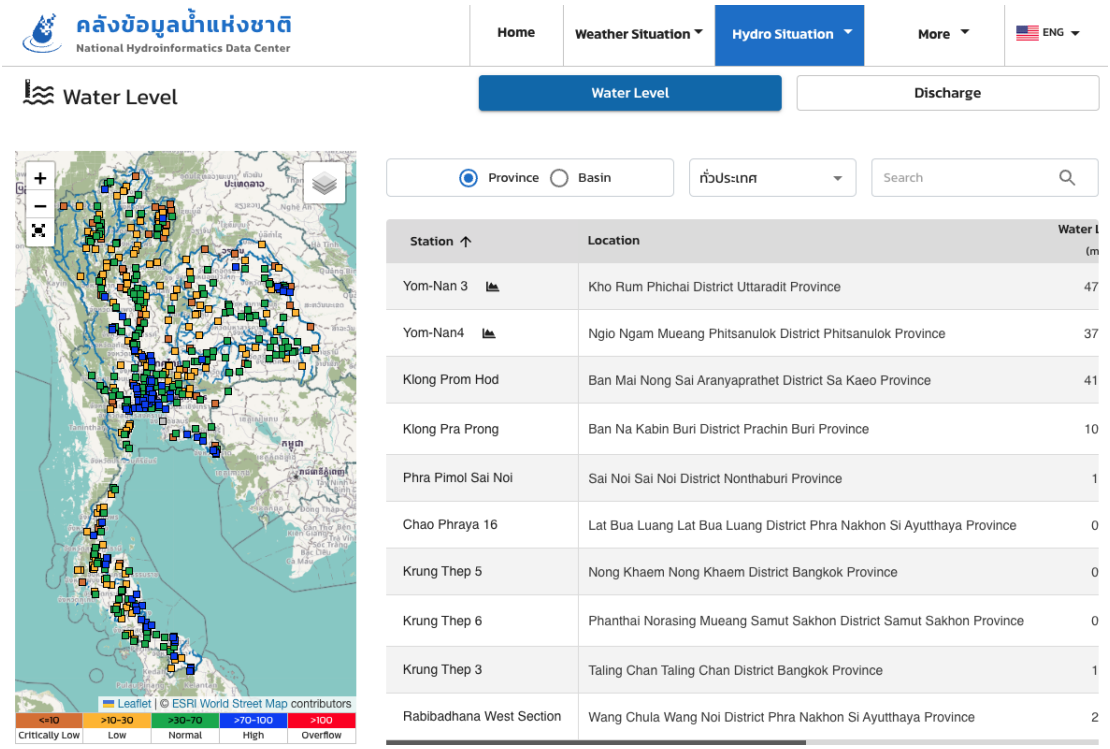


Figure 1.2: The reported telemetry water level data on the webpage.

The data from telemetry stations has been shown on the map, as depicted in Figure 1.2, representing the situations in the monitored area, which are classified based on the values of the measured data. The current approach for determining anomaly data begins with analysing data from stations identified as overflow/crisis. The data is then compared to neighbouring stations or earlier data from its own station. The data will be deleted if it is confirmed to be a false alarm. However, this procedure was carried out by people, and the correctness of this judgement depends on the experience and expertise. In other words, the information will be verified whenever there is a notice posted on the website.

Although the data can be manually checked before being disseminated for further analysis, the process requires knowledge experts to examine the data from each station and then make judgements on potential anomalies. This manual process is considerably labour-intensive and too slow to meet the requirements to deal with all the data from several hundred monitoring stations to send out early warnings in time.

The aim of this work is to apply machine learning methods to detect and correct errors in telemetry water level data with the highest possible accuracy and in the

shortest amount of time feasible. In order to determine the best approaches, we explore statistical models, machine learning models, deep learning models, and reinforcement learning models. Also, to fully develop our method, we investigate ensemble techniques, which is the intuitive way in machine learning to combine the results from different techniques to achieve the best results. The performance of different approaches is compared using a confusion matrix and appropriate statistical tests.

1.2 Motivation

Because the accuracy and reliability of early warning systems depend on the quality of the data, it is very important to be able to quickly detect and correct data errors in telemetry water level data. However, these anomalies require expert judgments to determine if they are anomalies, which is labor-intensive and too slow to deal with data from hundreds of monitoring stations to send early warnings in time. With the performance of machine learning methods, which are widely used in a variety of fields of work including the detection and correction of anomalies. Even though machine learning has been used in many studies to detect and correct anomalies in hydrological data (Yu et al., 2014; Pratama et al., 2016; Ma et al., 2017; Yang et al., 2017; Gao et al., 2018; van de Wiel et al., 2020), each water level dataset has different behaviours if it was measured from a different location or sensor. Consequently, successful approaches may not perform well when applied to other water level datasets. This challenge motivated us to apply machine learning to detect and correct data errors in the water level telemetry systems of HII.

1.3 Anomaly Definition

The crucial thing is to provide a formal definition for the idea of abnormality. This is critical because various definitions of abnormalities need different detection strategies. As a result, the definition must identify the major features of anomalies and indicate their limits. However, there are various definition of anomaly and outlier such as Aggarwal (2017) stated that “Outliers are also referred to as abnormalties, discordants,

deviants, or anomalies in the data mining and statistics literature". Others considered a data outlier is a corrupted value, while anomalies are irregular points with a distinct pattern Günnemann et al. (2014). According to Braei and Wagner (2020), an anomaly is an observation or a series of observations that deviates significantly from the overall distribution of data. As we have shown, there is no clarity in the literature about the distinction between anomalies and outliers. Therefore, in this study, we use the words "outlier" and "anomaly" interchangeably and define anomalies as follows:

Definition "An anomaly is a data point or sequence of data points that differs significantly from a very small part of the dataset or whole dataset. "

1.4 Research Questions

The research question we aim to answer are:

- How efficient are the statistical models in detecting anomalies in telemetry water level data?
- Is deep reinforcement learning (DRL) applicable and effective for identifying abnormalities in water level data?
- Can multi features data improve the performance of anomaly detection models?
- How efficient is pattern-matching-based data imputation for imputing data errors in telemetry water level data?
- Can ensemble methods improve the performance of single models?

1.5 Aims and Objectives

There are many algorithms that have been used to detect and correct anomalous data, but which one is appropriate with our water level telemetry systems? Our aim is to study and determine which algorithms are the most suitable. We selected the statistical models, machine learning models, deep learning models, and reinforcement learning models to deal with. Moreover, the model that works well with these stations may not work well with others. We then applied the ensemble methods to combine different

individual models to achieve the best results. Given the fact that the data labelling procedure for use as ground-truth takes time because it requires hydrological expert judgement and the completion of data that relies on the maintenance interval of each year, we then utilise data from certain stations and some time periods to conduct each experiment. However, we believe that experimenting with the data from many stations may be worth doing as future work.

Our specific objectives to achieve the overall aim are as follows.

1. Choose datasets with different characteristics of water level to provide a suitable test for our approach.
2. Apply a number of statistical models and observe their performance.
3. Propose a deep reinforcement learning algorithm and compare it with a number of deep learning algorithm models and observe their performance.
4. Apply a number of anomaly detection algorithms and observe their performance when there are more features.
5. Propose a method to combine the individual models using ensemble techniques.
6. Apply the number of imputation models and observe their performance.
7. Evaluate and compare the proposed methods to establish the most appropriate machine learning algorithms in the context of detecting and correcting anomalies in water level data.

1.6 Contributions of the Proposed Work

This research provides several contributions including development a modified sliding window algorithm; an algorithm to saving the DRL models with different criteria of performance; a feature extraction algorithm that uses the least amount of data points in sliding windows; data imputation based on pattern searching; ensemble models for data detection; data imputation with pattern searching algorithm; the combination of different models with ensemble methods.

The following summarises the contributions of this research:

1. We have proposed statistical models with sliding windows to find anomalies in each window. But when the window is moved forward to the next step, the detected anomaly data will be included in the window and will affect the prediction of the next possible anomaly value along the time series. So we then applied the sliding windows algorithm by which only the normal values can be moved into the windows. This work was published in the International Conference on Innovative Techniques and Applications of Artificial Intelligence, 2020 (Khampuengson et al., 2020)

2. We have developed deep reinforcement learning models that were generated based on different performance criteria to address the time spent to find the optimum training epoch. The proposed deep reinforcement learning ensemble for detecting anomaly in telemetry water level data was published in Water journal, 2022 (Khampuengson and Wang, 2022)

3. We have devised a Nearest Neighbour algorithmic method, in conjunction with a Saliency Map, to extract more features for improving the performance of anomaly detection models.

4. We have proposed a technique for imputing telemetry water level data with subsequence matching. Instead of splitting the subsequence in two, the missing data are temporarily replaced with some constant values to produce a dummy full subsequence, and then a sliding window is used to search for the most similar historical data subsequence. The identified subsequence will be adapted to fit the missing part based on their similarity. The proposed novel methods for imputing missing values in water level monitoring data was published in Water Resources Management journal, 2022 (Khampuengson and Wang, 2023)

5. We have proposed the ensemble to combine the anomaly detection algorithm as follows.

    - Our first proposed methods are simple and complex ensembles with different statistical models.

- Our second is DRL ensemble models with weighted and majority voting.
- Finally, we also build the ensemble model from multivariate variables from the models that were trained with the extracted features from our feature extraction.

6. Data imputation in water level data with a pattern searching-based algorithm.

1.7 Thesis Outline

The remainder of this thesis is organised as follows:

- **Chapter 2: Telemetry Systems.** The details of telemetry systems and the description of datasets used in all the experiments are described in this chapter.
- **Chapter 3: Technical Background and Related Work.** This chapter describes the technical background on the data and methodologies developed in the literature for data pre-processing, error detection, and error correction. Also including the evaluation methods and measures that are used for comparing the performance of each method.
- **Chapter 4: Anomaly Detection with Statistical Models and Their Ensembles.** We presents empirical investigation of applying statistical-based models for detecting anomalies. In addition, it presents two ensemble methods, Simple and Complex models.
- **Chapter 5: Anomaly Detection with Reinforcement Learning and Their Ensembles.** In this chapter, we present an investigation of applying reinforcement learning for detecting anomalies and enhancing their performance utilising ensemble techniques.
- **Chapter 6 New Multiple Features Extraction Methods for Anomaly Detection.**: We present a feature extraction algorithm that was developed to extract important features from telemetry water level data in order to improve the performance of anomaly detection models.

- **Chapter 7: Subsequence Matching Approaches for Data Correction and Imputation.** This chapter presents an empirical investigation of applying pattern searching algorithms for data correction and imputation of missing and anomalous values.
- **Chapter 8: Evaluation.** In this chapter we present an overall evaluation and discussion of the work undertaken for this thesis and the results obtained.
- **Chapter 9: Conclusion and Further Work.** The final chapter presents some concluding remarks on each chapter, along with possible ideas and suggestions for future work.

Chapter 2

Telemetry Systems

2.1 Introduction

Telemetry is an automatic system for monitoring environments in a remote or inaccessible area and transmitting data via many kinds of media such as radio, infrared, ultrasonic, mobile network, satellite or cable, depending on the objective of project and limitation of each area. Since 2012, HII has been researching, developing, and installing telemetry water level stations around Thailand for measuring water levels in the river and other water sources. As of 2018, there are 342 water level stations installed across Thailand.

2.2 Telemetry Station

In this section we will describe a details of telemetry stations including component, data collection methods.

2.2.1 Telemetry Station Components

The telemetry stations are composed of three major components, as follows:

Sensor Unit

It is composed of a variety of sensors capable of transmitting a signal in the 0 – 5 V or 4 – 20 mA standard format. The HII's telemetry system supports the installation of any type of measurement kit that complies with that specification. Currently, measuring

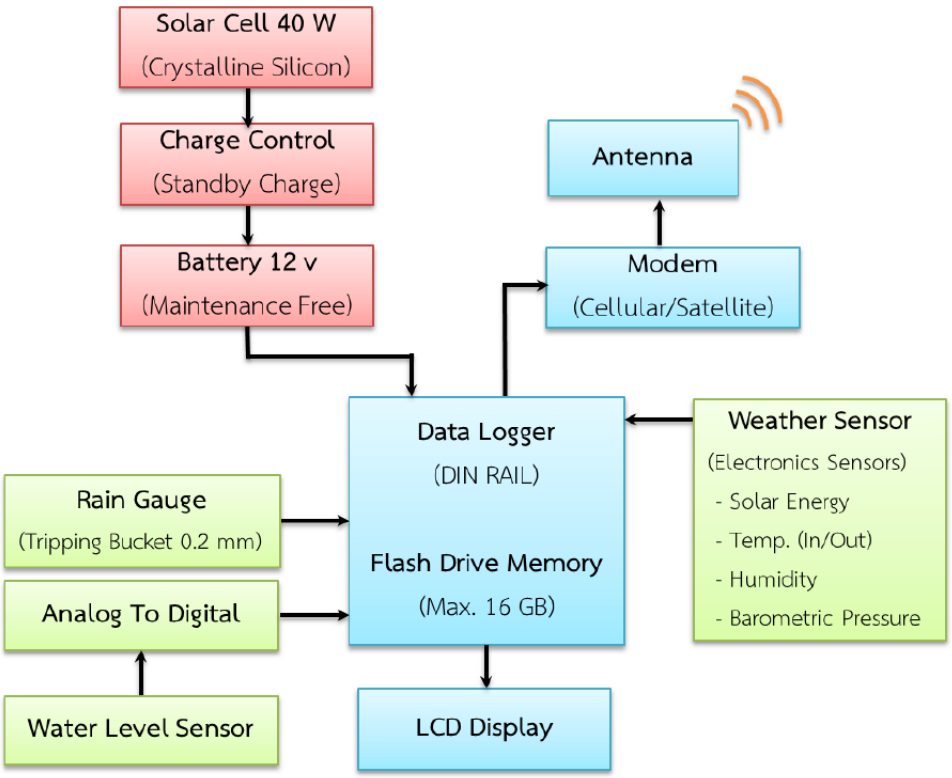


Figure 2.1: Components of a telemetry station: the green square represents the sensor unit, the blue square represents the control and transmission unit, and the red square represents the power unit.

kits for precipitation, temperature, humidity, air pressure, light intensity, and water level are being used, with varying degrees of precision and accuracy depending on the objective of use.

Control and Transmission Data Unit

This is the heart of the telemetry station. The researchers at HII have developed software for their own use, allowing them to customise and modify the system as needed. There is a time calibration system from the global positioning system (GPS) satellites to record and operate as scheduled. The measured data has been transmitted to the centre via message queuing telemetry transport (MQTT) protocol (Yuan, 2017) to be processed and published on the website.

Power Unit

The telemetry stations are powered by a 12 V automobile battery that is recharged through a solar panel. Additionally, since the telemetering system's equipment is small

and enclosed in a weatherproof box, it may be installed outdoors for simplicity of installation or inside for ease of maintenance.

2.2.2 Data Collection Methods

Telemetry water level stations are installed across Thailand to collect meteorological data and hydrological data for use in water management. They monitor air temperature, relative humidity , air pressure, precipitation, and water level. The equipment used to measure each data have been detailed as follows.

Air Temperature, Relative Humidity, and Air Pressure

These sensors are installed at a height of 1.5 metres in a radiation shield. The accuracy of the temperature sensor is ± 3 °C, and it can measure temperatures from -40 °C to 123.8 °C. The relative humidity is accurate to within ±2% RH across the working range of 0 - 100% RH. Pressure accuracy is ± 0.02 kPa for the working range of 30 to 110 kPa.

Precipitation

HAII telemetry system uses a tipping bucket sensor as a rain gauge. When rainwater fills in the bucket and reaches 0.2 mm, the (temporary) collected rainwater will be tipped-off and counted. Another empty bucket is ready for collecting further rainwater. An electrical signal will be sent for each counted tip using counting circuits. The amount of rainfall should be equal to the counted tips multiplied by 0.2 mm.

Water Level

The water level is measured with a radar-based level sensor which located at a fixed point above the water surface. This sensor sends high-frequency waves to the water surface. When waves touch the water surface, they will be reflected and received by the sensor. So, we can calculate the distance between the sensor and the water surface with the time intervals between transmission and reception. This can be converted into a water level. The main advantage of this method is that it is easy to maintain and install because we can install this sensor without contact with the water. The accuracy

is ±2 mm., the case is constructed of polybutylene terephthalate (PBT) plastic that is resistant to heat and sunshine, waterproof to IP66/67.

The HII telemetry water level station uses radar technology to monitor the water level in a river of interest due to its easy installation and high accuracy. Furthermore, due to the tiny size and light weight of the sensor, it was installed on the bridge railing. As a consequence, it is simple to move, install, and maintain.

2.3 Telemetry Datasets

HII telemetry systems are comprised of sensors that collect hydrological and meteorological data. The details of each datasets as follow:

2.3.1 Meteorological Datasets

The reported meteorological data reflects the weather conditions at the time of the report such as the temperature at 12:00 a.m. refer to the temperature on that time. Rainfall data, on the other hand, reflects the quantity of rain that fell in the past at the time of the report. Hourly rainfall at 10:00 a.m., for example, refers to rain that occurred between 09:01 a.m. and 10:00 a.m.

However, telemetry water level stations have been installed on a publicly accessible bridge, and the installed weather sensor is intended to be used for research and development of the HII's weather sensor. As a result, the quality of weather sensors and the accuracy of those under development are unsuitable for use in this research. Furthermore, many meteorological sensors have been damaged or malfunctioned due to external factors, such as when a car drives over a bridge, the vibration causes the sensor on the tipping bucket to tilt, resulting in false reported rainfall. Alternatively, if the rain bucket is stolen, the measured rainfall will not reflect the real amount of precipitation. As a result, meteorological datasets were discarded from our databases.

2.3.2 Hydrological Datasets

Water level data is a hydrological dataset collected from the HII's telemetry station. This data provides the water level at installed telemetry stations at the time the data was collected, which is every ten minutes.

Water level data is seasonal data, therefore it is usually a similar pattern in each season. However, there are external factors that influence the pattern of water level, which can be divided into three main factors:

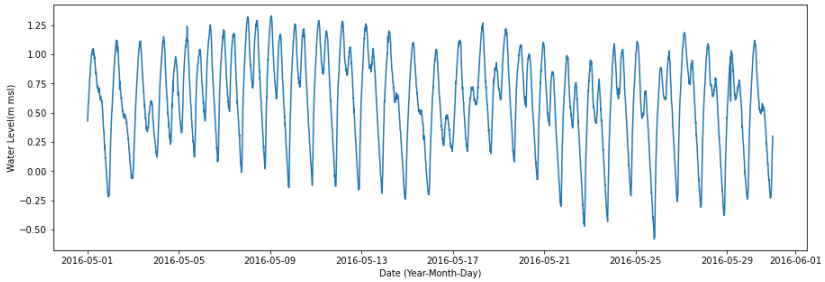
i) *Tidal*: The stations installed near the mouth of the river connecting to the sea will have a strong periodic pattern due to the effect from tidal, as shows in Figure 2.2a.

ii) *Rain*: When there is rainfall in those or upper areas, the water level is affected. As a result, the water level will vary in a variety of ways, both in upward and downward patterns, as depicted in Figure 2.2b.

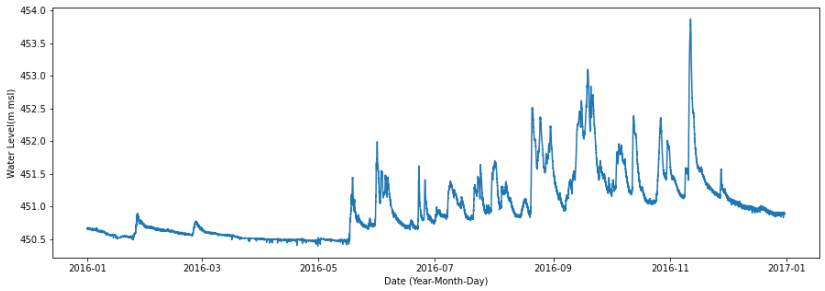
iii) *Irrigation*: The irrigation operation will have an effect on the water level in the canal. Because this canal was constructed to convey water from the main canal, which may be controlled by floodgates, in order to conduct irrigation in the distant areas or to prevent flooding. As a result, the water level will vary depending on the event. The example of data with irrigation effect as show in Figure 2.2c. Additionally, when the gate is in operation, such as closing or opening, the water level data from the nearby station changes rapidly. The measured water level in the canal away from the floodgate, on the other hand, has a few changes.

2.4 Telemetry Systems Overview.

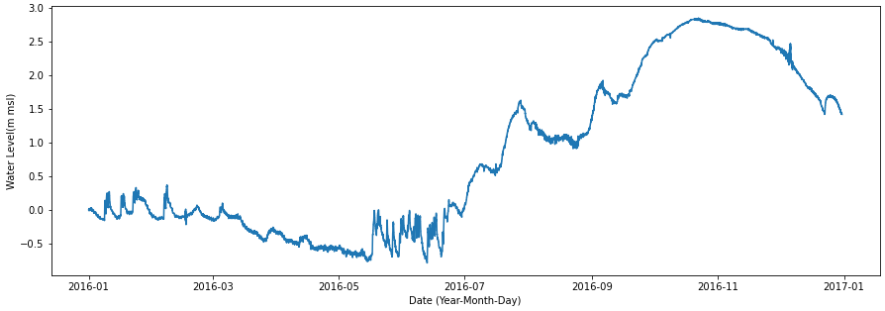
The overview of the telemetry system has been described in Figure 2.3. The measured data from telemetry stations has been transmitted and then the data has been stored in a database system on the cloud . The information that was collected is made available to the public through different data services such as websites, mobile applications, online services, etc. Meanwhile, any information that meets the criteria for a warning will be sent to the relevant agency for use in the process of decision-making in advance



(a) Tidal influences on data characteristics.



(b) Rain influences on data characteristics.



(c) Irrigation operation influences on data characteristics.

Figure 2.2: An example of external factors that influence on water level data characteristics

of any probable crisis. It was not only notified to the appropriate agency, but it was also displayed on the website to allow others to become aware of the current situation (as shown in Figure 1.2). There is also a process in place to have warning data checked by specialists to confirm an alarm and improve the accuracy of data from telemetry stations.

2.5 Causes and Type of Anomaly Data

Telemetry water level stations transmit the collected data to data center every 10 minutes. During the data collection and transmission, some errors can be introduced to the data. The main cause of anomaly values in telemetry time series data can be divided into 2 types as follow

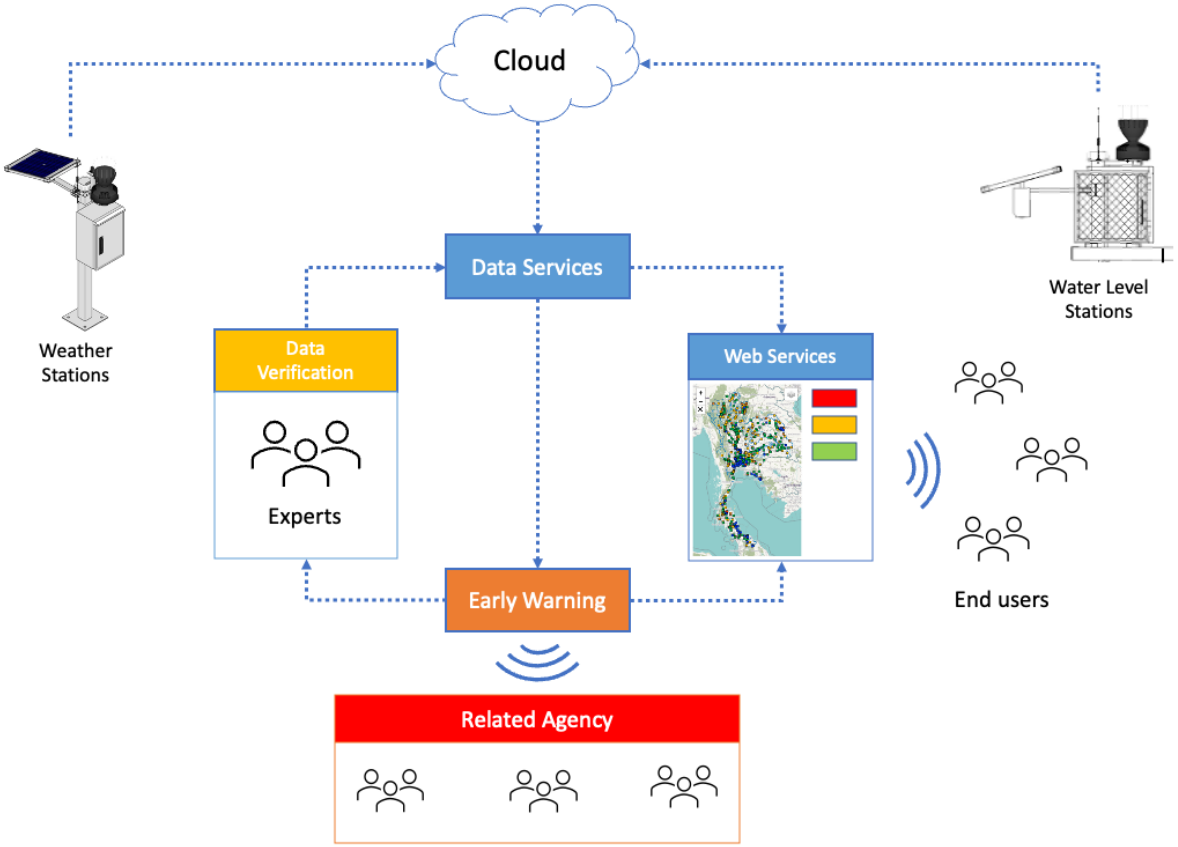


Figure 2.3: The overview of telemetry systems.

i) ***Instrument Error***. It occurs by malfunctions from any parts of the sensor, which may be caused by

- *Software Error:* It is caused by some problems or bugs in a program. These types of errors can be deterministic when the same bugs were encountered again and again. They can be fixed by developing new software to prevent this type of errors.

- *Hardware Error:* The cause of the anomaly could be due to a malfunction in the device used for measuring the water level. This malfunction can manifest in two ways: it could either result in an error in the measurement or make it seem like the device is working properly, but in reality, it is not able to measure the water level accurately or at all.It can be modified by replacing a new sensor or changing only the malfunction parts. The values from the station that anomaly usually either over or lower than measurement ability from instrument specification.

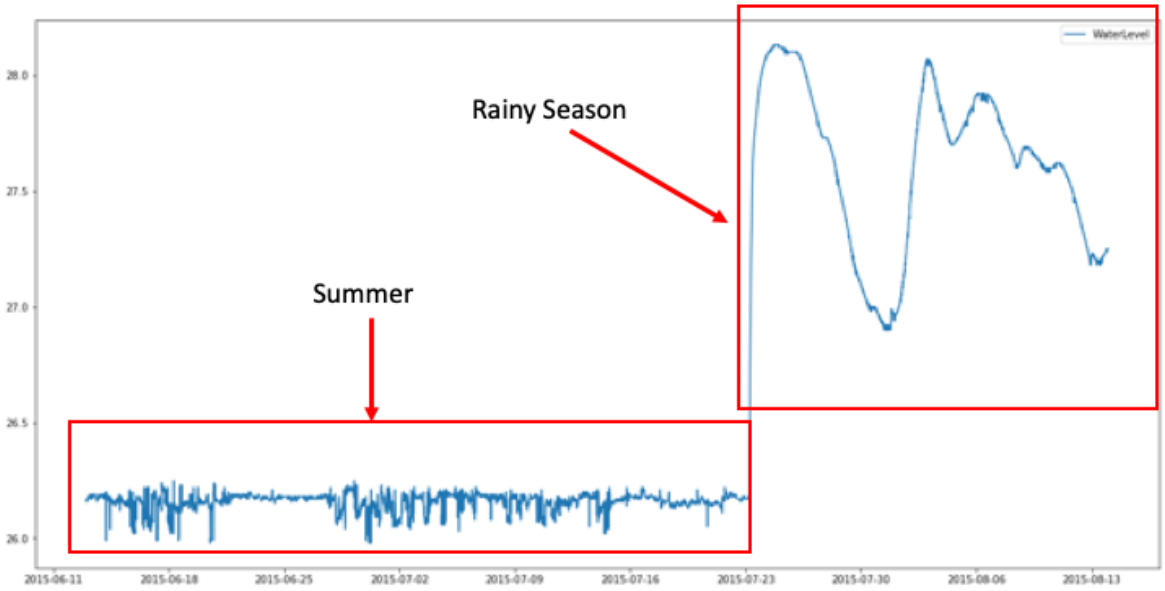


Figure 2.4: Example of data that has been affected by the environment in summer, and not affected in rainy season. The instrument gave unreasonable readings as water levels should not be oscillating abruptly by physics.

ii) ***Interference Error:*** It is usually caused by external factors besides the sensor such as environment, or human. Although the value from a station is in the range of instrument specification, the data is inconsistent when compared to other values. Usually, it is affected by the environment surrounding that has been changed temporarily or permanently, it is very difficult to identify this type of error and needs expert's analysis.

- *Temporary changed* surrounding occurs when the sensor has been affected by something suddenly or in short interval such as there is a boat under the water level sensor or in drought season there is no water left in the river then the sensor will detect another object(rock or grass) in this case the value will not be normal as presented in Figure 2.4.

- *Permanently changed* usually occurs when there is a new construction around the installed area such as new building or building extension that affects the measurement accuracy for each sensor.

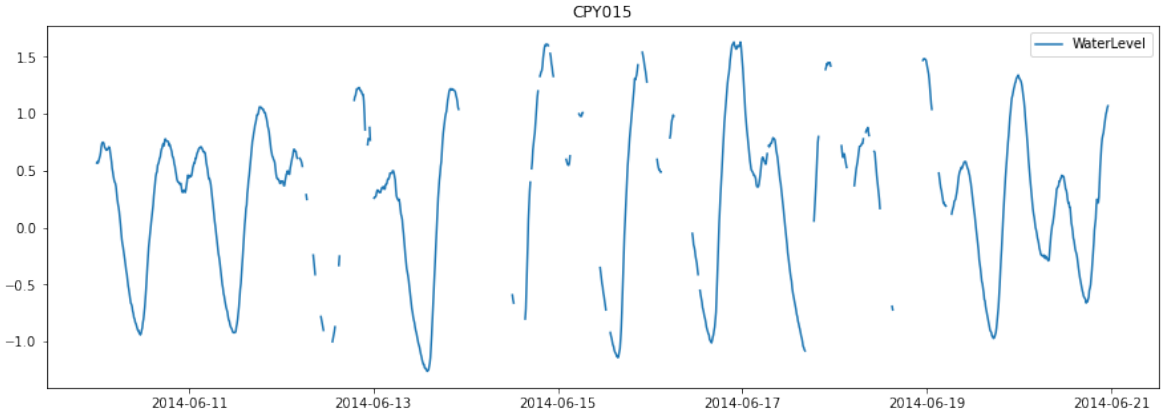
The types of anomaly data that is affected from above mentioned causes can be divided into the following categories.

i) ***Missing values***: This is because telemetry data cannot be sent to a server for the reasons mentioned above. Thus, no data has been kept in a database because of this.

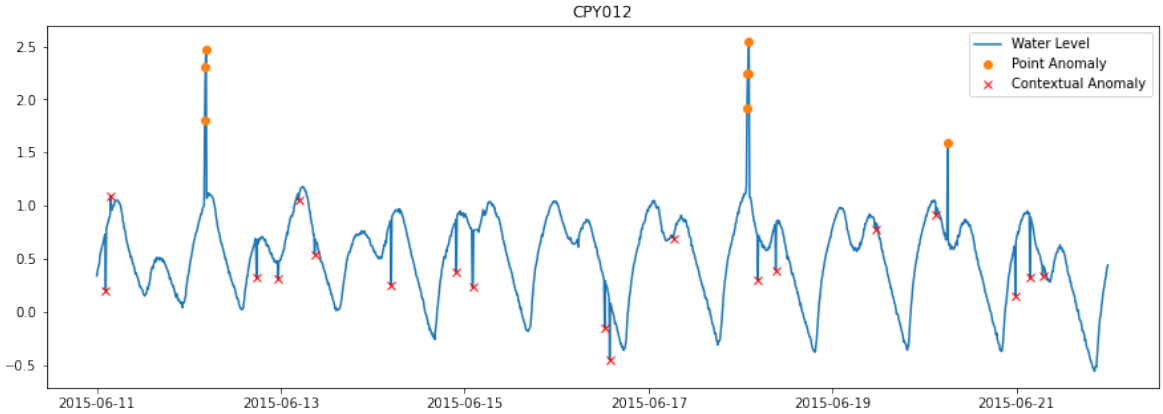
ii) ***Point anomaly***: If a value deviates significantly from previous values , it is considered an anomaly. These anomalies could be caused by faulty sensors, or they could be a short-term event of interest to system operators, like a flash flood.

iii) ***Contextual anomaly***: Some values may be considered normal in one context, whereas in another situation, the same values may be considered anomalous.

Figure 2.5 illustrated the three type of anomalies that often occurred on telemetry water level data.



(a) Missing data



(b) Point and contextual anomalous

Figure 2.5: An example of anomaly in water data

2.6 Data Quality

HII provided time series water level data between 2013 and 2018, gathered from telemetry stations across Thailand. There are 243 stations which automatically measure the water level at a frequency of once every 10 minutes. So, each station has 52,560 data points per year.

We begin by examining the missing data from each station for each year and classifying them according to the percentage of missing data, as given in Table 2.1. As we can see, the majority of telemetry stations were missing data by around 10% per year. In 2013, there were 40 sites with more than 90% missing data; this number has dropped year after year to just six stations in 2018. In contrast, the number of stations with less than 10% missing data has increased from 75 in 2013 to 119 in 2018. Because the telemetry stations were installed in remote areas and only maintained once a year. Thus, a station that fails to operate after a few days or weeks of maintenance will resume operation after the following year of maintenance. However, HII recognised this issue and has worked to enhance the quality and performance of the telemetry station each year. As a consequence, the telemetry station can function as the number of stations with missing data over than 90% has decreased each year.

Table 2.1: The number of stations for each percentage of missing data in each year.

Missing(%)	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**Average**
1 - 10	75	64	109	98	118	119	97
11 - 20	62	61	24	38	27	39	42
21 - 30	17	32	12	23	20	12	19
31 - 40	12	20	10	29	21	9	17
41 - 50	11	10	21	20	18	13	16
51 - 60	10	8	14	1	5	14	9
61 - 70	5	1	6	5	13	15	8
71 - 80	8	7	8	3	12	9	8
81 - 90	3	4	1	4	7	7	4
91 - 100	40	36	38	22	2	6	24

Telemetry water level stations have been installed in various locations with the aim of monitoring the water behaviour of each river basin, which is critical information for water management. The river basin in Thailand have been divided into 25 basins. The average annual percentage of missing data for each basin, as well as the number of installed telemetry stations (NS) for each basin, are presented in Table 2.2.

Table 2.2: The average percentage of missing data in each year each river basin.

Basin (No. of Stations)	**2013**	**2014**	**2015**	**2016**	**2017**	**2018**	**Avg.**
Bang Pakong(12)	17.19	13.62	10.99	24.41	26.44	10.69	17.22
Chi(28)	30.45	36.59	33.77	27.79	28.52	31.18	31.38
Chao Phraya(35)	14.65	17.71	23.38	21.25	19.4	29.73	21.02
East Coast(12)	80	70.4	69.86	10.85	19.92	5.7	42.79
West Coast(3)	70.52	71.61	67.09	67.06	44.57	6.45	54.55
Kok(5)	16.49	14.11	21.46	18.64	24.5	10.34	17.59
Khong(12)	30.09	25.17	25.9	17.35	16.86	18.28	22.28
Mae Klong(7)	40.32	33.07	26.09	29.83	37.52	47.32	35.69
Mun(22)	24.2	32.33	23.42	24.58	26.18	20.46	25.20
Nan(19)	27.23	20.26	9.86	21.06	24.05	23.56	21.00
Salawin(2)	15.42	5.01	25.51	33.6	10.3	2.02	15.31
Pasak(6)	23.86	17.14	25.13	18.7	20.32	35.8	23.49
Pattani(4)	90.22	95.73	86.13	67.51	59.14	40.98	73.29
Phetchaburi(5)	31.46	27.09	30.37	41.04	16.59	29.18	29.29
Ping(13)	38.43	24.49	25.74	19.38	17.33	15.85	23.54
Prachin Buri(8)	45.09	46.29	31.85	17.37	19.63	17.95	29.70
South East Coast(11)	25.43	32.91	14.88	19.88	10.92	27.55	21.93
Sakae Krang(5)	84.64	83.24	90.48	80.38	48.56	11.54	66.47
Songkhla Lake(7)	22.15	38.54	13.05	32.15	23.97	26.73	26.10
South West Coast(16)	54.32	61.33	65.3	52.26	47.59	40.2	53.50
Tha Chin(12)	10.17	18.78	41.12	22.39	12.89	11.63	19.50
Tonle Sap(3)	70.79	68.57	62.8	8.63	1.7	21.05	38.92
Tapi(11)	50.66	47.05	47.93	45.17	29.15	14.41	39.06
Wang(5)	6.43	14.14	5.86	13.34	9.55	37.68	14.50
Yom(18)	29.16	25.09	19.22	17.86	16.24	30.12	22.95

The Chao Phraya basin is the largest river basin in Thailand, with 35 installed telemetry stations. In this river basin, the average missing data from installed telemetry stations is around 21%. The highest rate of missing data from telemetry was found in the Pattani basin at 73.29%. Meanwhile, the Wang basin, the northern area, had the lowest average percentage of missing data at 14%. However, the average rate of missing data in this basin increased to 37.68 percent in 2018, an increase of more than 20% over the previous year.

Furthermore, since telemetry stations are placed in remote places throughout the country, travel to maintain just one station may result in unnecessary maintenance expenditures. As a result, HII decided to maintain every station with a single trip once a year in order to save money, which they usually do before the rainy season that starts in June. And the maintenance procedure begins with the northern station, then moves to the north-east, central, and finally the southern station. Table 2.3 shows the average

percentage of missing data for each river basin in each month. We can observe that the data missing rate decreased between May and June for northern stations, June and July for north-east stations, July and August for central stations, and after August for southern stations.

The water level data from telemetry stations is unlabelled in the first place. To evaluate the accuracy of our models in this study, we asked some experts at the HII to look at the data of each station and then make judgements on possible anomalies. Given the fact that, due to human limitations and the amount of data, these processes may not be able to find all of the anomalies in all datasets. An example of expert-labelled data is shown in Figure 2.6. As a consequence, we will select the appropriate datasets from some stations to do the experiments on. However, we believe that experimenting with more datasets may be worth doing as future work.

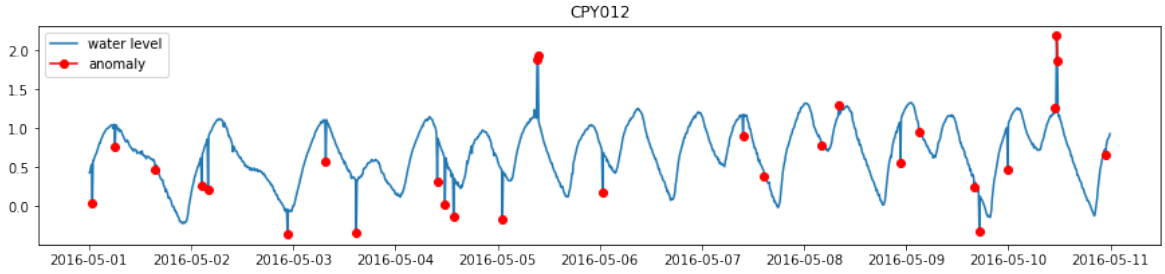


Figure 2.6: An example of expert-labelled data from CPY012 station, May 2016.

Table 2.3: The average percentage of missing data in each month each river basin.

Region	Basin (No. of Stations)	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Northern	Kok(5)	42.54	35.57	22.08	20.84	14.45	10.16	7.67	6.71	9.81	4.68	14.76	23.00
	Nan(19)	34.14	32.12	22.54	25.02	20.57	18.48	18.50	15.99	18.74	12.72	14.64	19.38
	Ping(13)	30.61	28.28	25.38	27.57	23.74	26.04	27.47	20.55	22.56	17.81	15.94	16.85
	Salawin(2)	34.76	28.48	18.49	19.17	6.34	4.72	13.65	12.02	8.48	8.59	12.62	17.18
	Wang(5)	16.00	14.22	16.43	19.62	16.65	14.79	16.16	12.05	13.00	8.05	13.38	13.70
	Yom(18)	28.53	28.16	27.56	32.15	32.44	27.72	22.91	17.40	15.59	10.43	14.16	18.72
NorthEast	Chi(28)	39.46	42.29	40.97	41.89	43.75	31.28	28.36	18.28	18.60	17.83	23.86	30.69
	Khong(12)	27.79	31.09	27.02	26.96	27.38	22.93	24.77	16.54	14.36	11.88	15.96	21.13
	Mun(22)	36.43	32.38	30.75	34.60	29.86	27.18	24.91	24.39	17.81	12.56	13.34	18.52
Central	Bang Pakong(12)	24.81	18.36	13.43	18.23	17.94	16.79	19.57	15.34	17.02	13.55	20.03	11.86
	Chao Phraya(35)	24.59	23.64	19.57	20.80	22.40	24.46	25.54	19.16	18.68	15.83	20.53	17.29
	Gulf River(12)	45.66	46.27	44.11	42.72	43.34	42.63	41.91	43.55	46.70	44.18	41.76	30.87
	Gulf River(3)	62.11	57.04	56.20	56.82	58.68	61.73	56.49	53.67	51.60	48.03	46.99	45.51
	Mae Klong(7)	42.96	41.83	36.46	33.48	37.83	37.57	37.21	37.07	35.85	30.43	28.47	29.41
	Pasak(6)	25.09	28.20	37.43	45.94	36.38	22.50	16.87	11.67	6.82	8.69	18.73	23.99
	Phetchaburi(5)	40.99	39.05	34.36	35.74	30.07	32.42	31.40	20.17	17.40	17.03	22.98	30.55
	Prachin Buri(8)	36.78	39.75	39.11	36.32	34.42	29.58	27.26	30.02	25.45	20.22	19.68	18.36
	Tha Chin(12)	23.56	21.91	19.52	20.83	20.30	19.82	18.60	16.66	17.68	19.14	19.50	16.69
	Sakae Krang(5)	70.05	71.95	73.51	73.79	74.58	76.45	75.65	61.70	56.55	54.19	54.39	55.28
Southern	Pattani(4)	76.38	76.58	76.17	73.98	74.35	77.10	77.19	75.68	73.86	69.25	62.34	66.67
	South East Coast(11)	19.01	22.72	20.81	20.17	21.94	23.65	25.97	24.39	29.19	21.44	17.01	16.95
	Songkhla Lake(7)	26.25	28.39	26.79	29.01	25.44	28.28	30.29	25.50	31.35	30.76	15.41	15.96
	South West Coast(16)	53.62	54.36	54.99	56.90	59.27	57.91	58.51	56.43	55.60	47.82	42.04	44.56
	Tonle Sap(3)	45.05	39.67	40.57	40.65	40.34	43.47	40.43	40.77	40.62	35.72	34.11	25.66
	Tapi(11)	40.89	40.90	41.65	41.88	43.60	41.68	40.99	41.65	38.50	30.34	31.74	35.04
	Average	37.92	36.93	34.64	35.80	34.24	32.77	32.33	28.69	28.07	24.45	25.37	26.55

2.7 Summary

This chapter describes the details of the telemetry station and datasets that were used in this research. The telemetry stations are composed of three major components: 1) the sensor unit; 2) the control and transmission unit; and 3) the power unit. Telemetry stations have been utilised for a variety of purposes, including data collection for hydrological study, sensor studies, and early warning. However, with the limitation and objective of installing the weather sensor, we intended to use only water level data in this research. Missing values, point anomalies, and contextual anomalies are the three primary categories of anomalies in water level data. Although there are datasets from over 200 telemetry water level stations, the data quality of each station differs for a variety of reasons. In addition, we are unable to label data at all stations. As a result, appropriate data selection in each experiment is essential.

The methodology that have been used in this research has been describe in the next chapter.

Chapter 3

Technical Background and Related Work

3.1 Introduction

Telemetry data is the backbone of various applications, and it plays a crucial role in decision-making processes. However, real-world data is often incomplete, inconsistent, and contains errors. Therefore, effective pre-processing, error detection, and error correction techniques are essential to ensure data accuracy and reliability. This chapter aims to provide an overview of the technical background and methodologies for data pre-processing, error detection, and data correction.

3.2 Data Pre-processing Approaches

Data pre-processing is an important stage in the machine learning pipeline that involves cleaning, transforming, and preparing raw data for modeling. The following is a general process for data pre-processing:

- *Data collection:* Raw data is collected from various sources, such as databases, APIs, or files.
- *Data cleaning:* This step involves identifying and handling missing values, outliers, and inconsistencies in the data. Duplicate data points are also removed.
- *Data transformation:* Data is transformed into a format that is suitable for modeling. This can include scaling, normalization, feature extraction, and feature

selection.

- *Splitting the dataset:* The data is split into training, validation, and test sets. This ensures that the model is trained on one set of data and evaluated on a different set of data.

In real-world datasets like telemetry water level data, missing values and abnormalities are commonly noticed as a result of faulty sensors or unexpected events. So, data cleaning is the most important aspect of data pre-processing for telemetry water level datasets. When missing data is present in a dataset, improper handling may lead to biassed or inaccurate results. This process can be optimised to ensure that the data is accurate, reliable, and consistent for subsequent analysis. This can improve the accuracy and performance of machine learning models or other analytical techniques applied to the data. Thus, the missing data must be addressed before proceeding with the analysis. These are some popular strategies for missing data pre-processing:

- *Deletion:* This technique involves removing the rows or columns that contain missing values. It is the simplest technique but can result in loss of important information.

- *Imputation:* This technique involves filling in the missing values with estimated values. There are various methods for imputation, such as mean imputation, median imputation, mode imputation, regression imputation, and k-nearest neighbor imputation.

- *Interpolation:* This technique involves estimating the missing values based on the values of other data points. There are various methods for interpolation, such as linear interpolation, spline interpolation, and polynomial interpolation.

- *Multiple imputation:* This technique involves creating multiple imputed datasets by estimating the missing values multiple times using different methods. The results from the multiple imputed datasets are then combined to provide more accurate estimates.

As our research focuses on detecting and correcting anomalies, we must keep all errors and label them to improve our models' learning. However, to avoid making unnecessary

changes to the data, we excluded stations with a high percentage of missing values. Moreover, if there were gaps in the data, we used an interpolation method with a polynomial degree 2 to fill them.

Telemetry water level datasets has only one feature which may have not enough information for training models to achieved the best accuracy. Feature extraction is a technique that can be used to extract more features. The process involves using mathematical or statistical methods to derive new features that capture additional information from the original feature. One common approach is to use a transformation function to create new features based on the original feature. For example, if the original feature is a measurement of temperature, a transformation function could be used to derive features such as mean, variance, and standard deviation. Another approach is to use feature engineering, which involves manually creating new features based on domain knowledge or intuition. For example, in image processing, new features can be derived by extracting edges, corners, and other high-level features from the raw pixel data. The goal of feature extraction is to create a set of features that can provide more meaningful and relevant information to machine learning algorithms, ultimately leading to better model performance. Additionally, through effective feature extraction techniques, we can gather valuable insights from the telemetry water level data, leading to improved model performance. We demonstrated a suitable technique for feature extraction in Chapter 6.

3.3 Anomaly Detection Approaches

Anomaly detection aims to identify data that is different from the majority of the data, which results from an error (Zimek and Schubert, 2017), and which will often result in some kind of issue, such as strange patterns in network traffic that could signal a hack. Anomaly detection is often used in preprocessing to remove anomalous data from the dataset, which frequently results in a significant increase in statistical accuracy. There are various strategies for detecting anomalies, but this study focuses on techniques for detecting errors in time series data. Anomaly detection methods on time-series data can be divided in three main categories (Braei and Wagner, 2020):

3.3.1 Statistical Approaches

The easiest method for spotting data anomalies is to find data points that differ from the distribution's common statistical features, such as mean, median, mode, and quantiles. Some of the most common statistical anomaly detections are:

Z-Score

The Z-score is a statistical measure that indicates how many standard deviations an observation is from the mean. In this method, the time series data is first pre-processed to remove any trends or seasonal components using techniques such as differencing or seasonal decomposition. Then, the mean and standard deviation are used to figure out the z-score for each observation in the time series. Any observation that has a z-score beyond a certain threshold (e.g., 3 standard deviations) is considered an anomaly. An example of research that applies Z-Score is the research to detect anomalies in real-time temperature time series by Liu et al. (2020).

Interquartile Range (IQR)

IQR method is a statistical technique used to identify outliers or anomalies in a dataset. It involves calculating the difference between the upper and lower quartiles of the data distribution, which is the range containing the middle 50% of the observations. To apply the IQR method, the data is sorted, and the first and third quartiles are calculated. The IQR is then computed as the difference between these quartiles, and the lower and upper boundaries are determined as Q1 - 1.5*IQR and Q3 + 1.5*IQR, respectively. Any observation that falls outside the lower or upper boundaries is considered an outlier or anomaly. The IQR method is useful for detecting anomalies in skewed distributions, but may not be suitable for datasets with a small number of observations or Gaussian distributions. Vinutha et al. (2018) applied the IQR technique to identify outliers in the NSLKDD'99 dataset, which can produce false positive alarms and reduce the efficiency of intrusion detection systems.

Sigma Rule of Thumb (K-Sigma)

The data set that has most of the data distributed around the mean values in a symmetric shape is called *"normally distributed"*. Consequently, when a data set has a normal distribution, then the mean and the median are the same (in the case of a perfectly bell-shape) or almost equal, 99.7% of all data falls between $[mean \pm 3 * \sigma]$. This rule is also known as "three-sigma rule of thumb". We defined values outside that interval as anomalous. K-Sigma was used by Ostroumov and Kuzmenko (2021) to find outliers in unmanned aerial system (UAS) data, and it proved to be the most successful approach.

Autoregressive Model (AR)

AR is a linear model when current value X_t is based on a finite set of previous values of length p and error values ϵ, also called AR process of order p or AR(p), as represent as follow:

$$X_t = c + \sum_{i=1}^{p} a_i \cdot X_{t-i} + \epsilon_t$$

where a and c are the coefficients values which can be approximated by using the training data.

Moving Average (MA)

The current value may be calculated using the average of prior data. And if the current value is significantly different from the average, it may be labelled as an anomaly. A moving average is calculated by averaging the values over a specific number of previous values, called the window size. It is simple and commonly used in time series analysis and forecasting, as shown by Bernacki and Kołaczek (2015) which can apply for anomaly detection.

Autoregressive Moving Average Model (ARMA)

ARMA is the combination of AR and MA, which is often used for a univariate time-series. A time series of the ARMA(p,q), where p is the order of autoregression, and q

is the order of the moving average. The equation is given by:

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-1} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t$$

where a is the autoregressive model's parameters, b is the moving average model's parameters, c is a constant, and ϵ is an errors (or white noise). The main challenge when applying this method is to select appropriate values for p and q and the data need to be stationary (Braei and Wagner, 2020).

Auto-Regressive Integrated Moving Average (ARIMA)

One of the most significant issues with time-series datasets is that they might be non-stationary. Typically, we utilise the differencing or integrated technique to convert the series into a stationary time series by subtracting the t-1 value from the time series' t values. If, after applying the first differencing, we are still unable to get a stationary time series, we apply the second-order differencing or more orders until it is stationary. ARIMA models use this approach to handle ARMA model issues that need stationary data, by combining a number of differences previously applied to the model in order to make it stationary, the number of previous lags, and residual errors in order to forecast future values.

3.3.2 Machine Learning Approaches

Machine learning methodologies have gained in popularity over the last several decades as processing power has increased exponentially, especially for data science applications. As a consequence, a growing number of academics have begun to use machine learning approaches for abnormality detection. Below is a brief overview of popular machine learning-based techniques for anomaly detection.

Density-Based Anomaly Detection

- *K-nearest neighbour*

  It operates on the concept that every data point that is close to another belongs

to the same class. In other words, a new data point is more likely to have the same class label as its k-nearest neighbours than the data points farther out (Peterson, 2009). As a result, anomaly data tends to be located farther away from the class of similar data.

- *Local Outlier Factor (LOF)*

  Breunig et al. (2000) proposed the LOF algorithm for finding anomalous data points based on the concept of local density. By comparing the local density of a given data point to the local densities of its neighbors, the point that has a substantially lower density than its neighbours is considered an anomaly.

- *Density-Based Spatial Clustering of Applications with Noise(DBSCAN)*

  It was suggested by Ester et al. (1996) with the idea of finding the core points in the dataset that have at least *minPoints* around them within an *epsilon* distance and then making clusters from these samples. Following that, all of the points that are close enough (within epsilon distance) to any sample in a cluster are added to the cluster. Then, it does the same thing over and over again for the new samples that are added to the cluster. DBSCAN will decide the cluster number on its own, and outlier samples will be given the number -1.

Clustering-Based Anomaly Detection

Clustering is one of the most common ideas in the field of unsupervised learning. The data points that are similar are likely to be in the same groups or clusters, based on how far they are from the local centroids that they belong to. K-means is a common way to group things together. It makes groups of data points that are "k" similar. Data that does not fit into one of these groups could be marked as an anomaly.

Support Vector Machine-Based Anomaly Detection

Boser et al. (1992) presented a supervised classification method, which evolved into what are now known as support vector machines (SVMs). It is a traditional machine learning approach that can be used to detect anomalies in datasets. It is especially

beneficial for multidomain applications running in a big data environment. The objective of the SVM algorithm is to identify and formulate a hyperplane that separates data points in an N-dimensional space. The data points that lie on either side of the hyperplane can be assigned different classes. Furthermore, the dimension of the hyperplane is determined by the number of features. For example, a hyperplane is just a line when the number of input features is two, and becomes a two-dimensional plane when the number of input features is three. Support vectors are data points that are closer to the hyperplane and have an impact on the hyperplane's location and orientation.

Gaussian Naive Bayes (GNB)

This is a popular algorithm used for classification tasks in machine learning. It is a variant of Naive Bayes that assumes the features are independent of each other and normally distributed. This means that the algorithm calculates the probability of each class based on the likelihood of the input data given the class and the prior probability of the class.

To train the model, GNB estimates the mean and variance of each feature for each class in the training data. During inference, the algorithm calculates the probability of each class based on the likelihood of the input feature vector given each class, using Bayes' theorem.

Despite its assumption of feature independence, GNB has been found to work well in many practical scenarios and is commonly used in text classification and spam filtering, as well as in medical and biological applications. However, GNB may not perform well when the features are strongly correlated or the class distribution is imbalanced.

Decision Tree (DT)

This algorithm used for classification, regression, and anomaly detection tasks. The algorithm creates a tree-like model of decisions and their possible consequences or outcomes.

In anomaly detection using decision trees, the algorithm learns the patterns of normal behavior from the training data and identifies anomalies as data points that do not

follow the learned patterns. The algorithm partitions the feature space into regions and assigns a decision rule to each region based on the feature values.

To detect anomalies, the decision tree algorithm classifies new instances as either normal or anomalous based on their position in the feature space relative to the decision boundaries learned from the training data. Data points that fall outside of the decision boundaries are considered anomalous.

Decision trees are easy to interpret and visualize, which makes them a popular choice for anomaly detection tasks in many domains. However, decision trees can be prone to overfitting the training data and may not generalize well to new data if the tree is too complex. Therefore, it is important to tune the hyperparameters of the decision tree, such as the maximum tree depth, minimum number of instances per node, and minimum improvement in separation, to avoid overfitting and ensure good performance on new data.

Isolation Forest

The Isolation Forest was initially developed by Liu et al. (2008). It isolates data from a collection by randomly selecting a feature and then processing the randomly selected subsample in a tree structure. Anomalies are more likely to be detected on the shorter branches of the tree, since the tree finds it easier to identify them from the other data.

Extreme Gradient Boosting (XGBoost)

XGBoost was introduced by Chen and Guestrin (2016). It is a gradient boosted decision tree (GBDT) implementation optimised for speed and performance. It is built on top of the following technologies: supervised machine learning, decision trees, ensemble learning, and gradient boosting. XGBoost is a scalable and highly accurate gradient boosting solution that pushes the limits of boosted tree algorithm processing power. It was created in particular to boost the performance and computational speed of machine learning models. With XGBoost, trees are generated in parallel rather than sequentially as with GBDT. It looks at gradient values to figure out how good each possible split in the training set is.

3.3.3 Neural Network Approaches

During the last decade, deep learning methods have made remarkable progress in computer vision. Researchers were inspired by this achievement to use similar techniques to identify abnormalities by using neural networks to make predictions. Following that, the error of prediction is used as a threshold to classify data points as normal or abnormal.

In this subsection, we have chosen the most popular methods that have been used for anomaly detection in time series data in recent years.

Multiple Layer Perceptron (MLP)

An MLP is a network of primitive neurons known as perceptrons. The perceptron produces a single output y from a large number of inputs x by constructing a linear combination based on the weights (w) and bias(b) of the inputs and then passing the result through some nonlinear activation function to create the nonlinear decision boundary, as depicted in Figure 3.1. This may be mathematically expressed as

$$y = \varphi(w \cdot x + b)$$

where φ is an activation function.

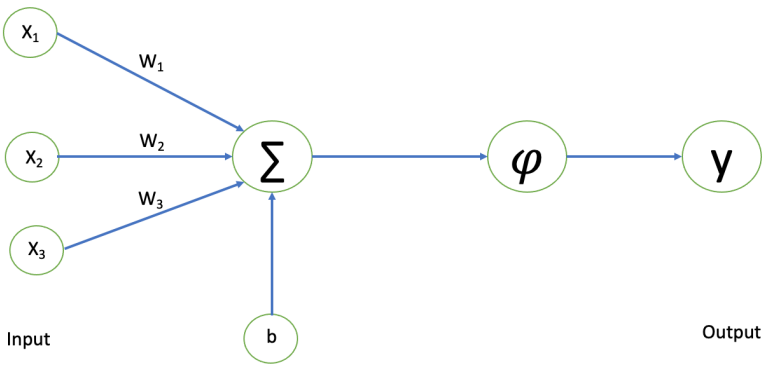


Figure 3.1: A simple perceptron neural network.

To make a single perceptron work better, we can group several neurons together in layers and create a multilayer perceptron (MLP). The input signal travels through each layer of the network one after another, which helps improve its overall performance.

The signal-flow of such a network with two hidden layers of five neurons each and one output, as depicted in Figure 3.2.

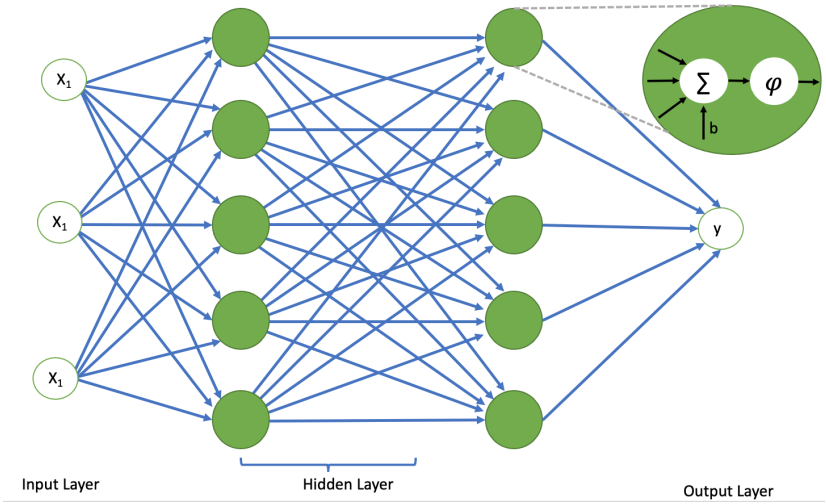


Figure 3.2: An example form of multiLayer perceptron neural network.

Recurrent Neural Network (RNN)

The input data of a typical neural network is organised in such a way that one input has no impact on the other. This is not the case while looking at time series or textual data, since values coming later in the sequence are often impacted by prior values. As a result, we must consider the whole sequence rather than just the last data point.

RNN was made to solve this problem by making standard neural networks more powerful with the concept of "memory" which enables them to store the states or information associated with prior inputs in order to generate the sequence's next output. As seen in Figure 3.3, RNN employs cyclical hidden states to make each of these steps dependent on the previous. The output of a recurrent neural network is the following:

$$y_t = \varphi(w_x \cdot x_t + y_{t-1} \cdot w_y + b)$$

where w_x is the weight for x inputs, w_y is the weight for layer that connect to the output layer.

RNN is extensively utilised in a variety of fields, including natural language modeling, voice and speech recognition, image and video caption generation, and also time series data.

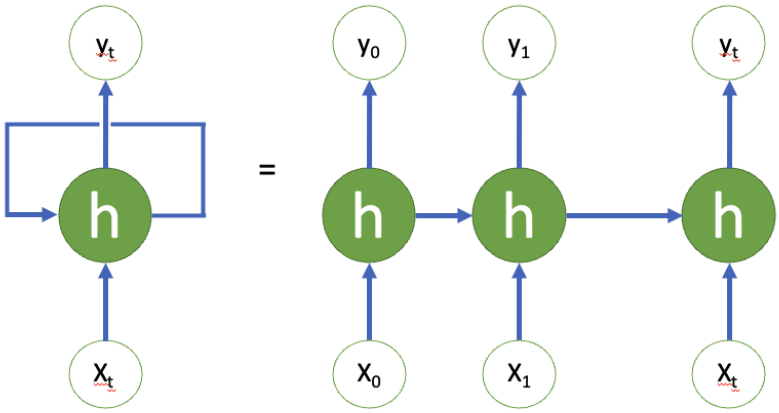


Figure 3.3: An unrolled RNN

Long Short Term Memory (LSTM)

When RNNs are trained over very long sequences of data, the gradients tend to become either extremely large or so extremely tiny that they vanish to almost zero. LSTM is designed to avoid the long-term sequence problem by adding gates, or memory cells, that allow it to control whether to store or delete previous information. Weights were employed to give priority to the data, which the algorithm also learns. This simply means that it learns over time what information is important and what is not. The LSTM has three gates: an input gate, a forget gate, and an output gate. Input gates decide whether or not to accept new input. The forget gate is in charge of deleting unnecessary information. The output gate selects and displays valuable information from the current cell state. Figure 3.4 shows the structure of an LSTM network.

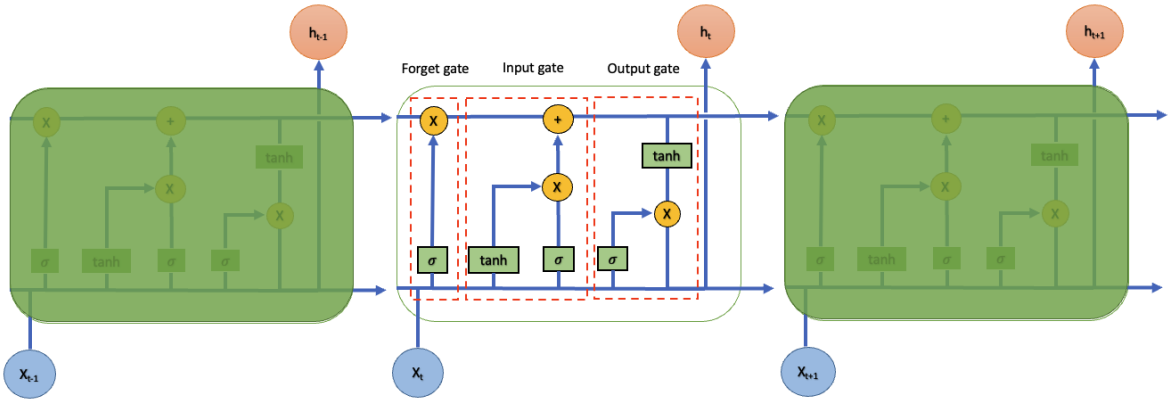


Figure 3.4: An example form of LSTM networks.

Gated Recurrent Unit (GRU)

The GRU seeks to address the problem of vanishing gradients that occurs with regular recurrent neural networks. To overcome this issue, GRU uses an update and reset gate to keep information from a long time ago and choose which information should be transmitted to the output, rather than deleting information that is irrelevant to the prediction like LSTM does.

Convolutional Neural Networks (CNN)

CNNs are most often employed in computer vision to perform tasks such as object detection, classification, and segmentation. They have three main types of layers, as depicted in Figure 3.5, which are as follows: 1) The convolutional layer is the first layer of a convolutional network and serves as the brain and major processing centre in a CNN. 2) Pooling layers, also known as downsampling, reduce the dimensionality of input data by reducing the number of parameters. 3) Fully-connected layer, where the neurons in this layer have full connections to all activations in the previous layer, this layer performs the task of classification based on the features that were found in the previous layers and the different filters that they used. CNNs are widely used for deep learning for three important reasons: first, they eliminate the need for manual feature extraction; second, they produce very accurate recognition results; and third, they can be retrained for new recognition tasks, which lets you build on an existing network.

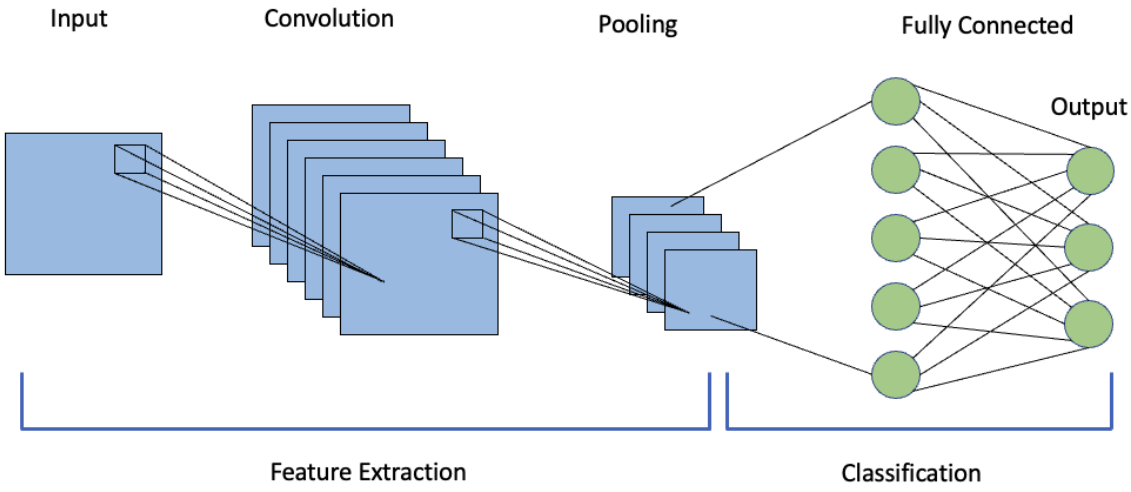


Figure 3.5: An example form of convolutional neural networks.

Residual Neural Network (ResNet)

A residual neural network (ResNet) is an artificial neural network (ANN) that uses skip connections, or shortcuts, to bypass some layers. In typical ResNet models, two or three layers that contain batch normalisation and an activation function (usually ReLU) are often skipped.

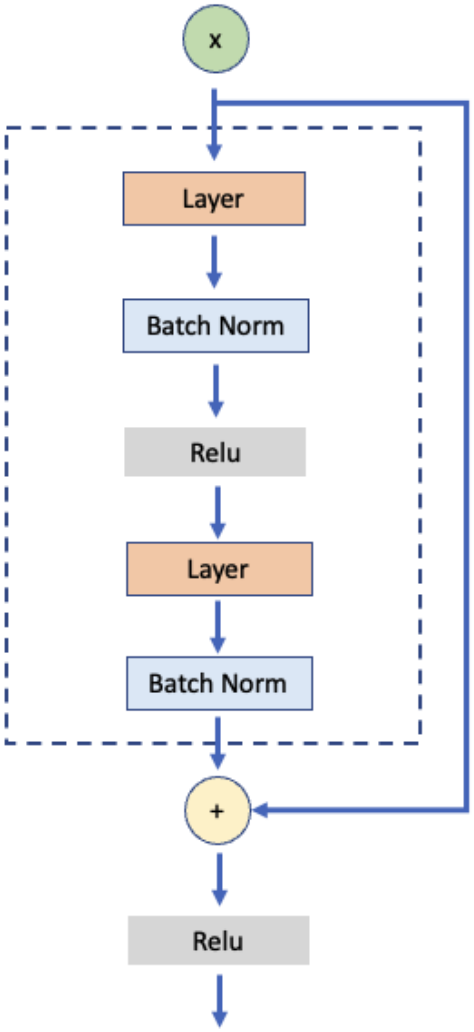


Figure 3.6: An example form of residual neural network

One of the issues that ResNets address is the well-known vanishing gradient. This is due to the fact that when the network is excessively deep, the gradients from which the loss function is derived simply drop to zero after numerous applications of the chain rule. As a consequence, the weights never update their values, and hence no learning occurs. However, in the case of ResNets, gradients can flow straight through via the skip connections from later layers to early filters.

Autoencoder

An autoencoder is a special kind of multi-layer neural network that reduces the number of dimensions of data in a way that is both hierarchical and nonlinear. The output layer typically has the same number of nodes as the input layer, and the design is layered and symmetric. An autoencoder's purpose is to produce the output as nearly as possible to the input. An autoencoder has two connected networks: 1) *encoder*, which takes an input and turns it into a compressed representation of knowledge in the bottleneck layer; and 2) *decoder*, which turns the compressed representation back into the original input. An example form of autoencoder as illustrates in Figure 3.7. Because the autoencoder generates a simplified representation of the data, it is a natural method for detecting outliers. The basic idea is that outliers are significantly more difficult to express correctly than normal data. As a result, the error in reconstructing an outlier will be high.

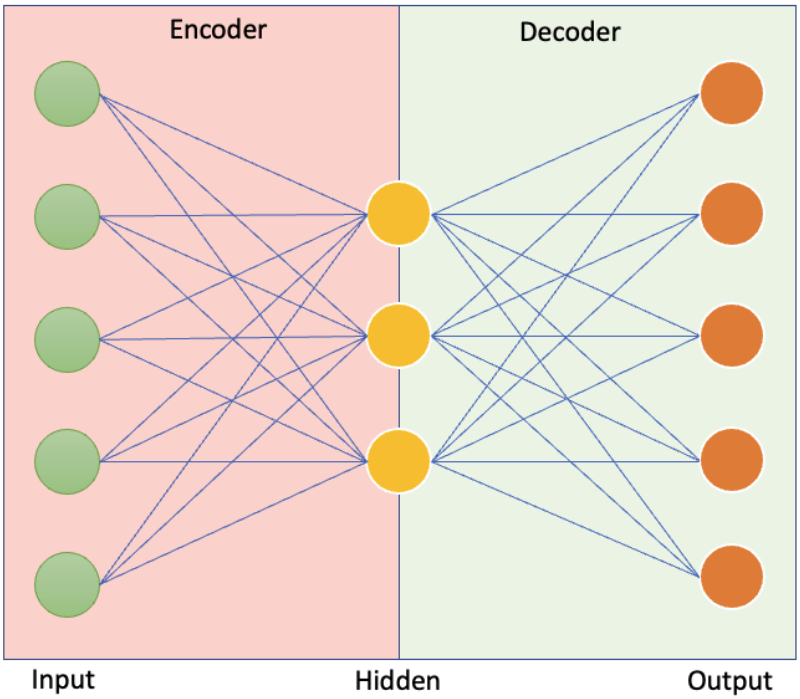


Figure 3.7: An example form of autoencoder model.

3.4 Data Correction and Imputation Approaches

Discovered anomalies and missing data are often removed or excluded from a series of data, which is problematic because many statistical analyses require complete data sets

to avoid biassed or erroneous analysis results. As a consequence, effective methods for estimating missing values from available data are required. Missing value imputation algorithms commonly employ the following techniques:

3.4.1 Statistical Approaches

A simple and widely used technique of data imputation is to use statistical methods to estimate a value from the values that are present. The mean and mode are the most straightforward imputation techniques for numerical and categorical values. The mean technique fills in missing data by averaging the value across all present data. On the other hand, the mode technique uses the most frequently occurring value in all the present data to fill in the missing attribute values.

Using a regression model to make a prediction about the value of the anomalous data point based on the values of other variables in the data set is yet another statistical method for imputed anomalies that have been discovered. Estimating the relationship between the variables and predicting the value of an anomalous data point based on the values of the other variables are both things that may be accomplished with the use of the regression model. This strategy has the potential to be effective in circumstances in which there is a strong association between the variables that are included in the data collection.

Interpolation is another widely used approach for data imputation. The most frequently used methods for interpolation include linear interpolation, polynomial interpolation, spline interpolation, and multiple imputation. The selection of an appropriate method depends on the complexity of the data and the specific issue that needs to be resolved. Linear interpolation is best suited for simple data structures, while spline interpolation is more appropriate for complex structures. Interpolation methods can be used to predict missing data or fix inaccurate measurements. The accuracy of the results depends on the quality of the available data and the assumptions made about the data. Interpolation is a widely used technique in many fields, including survey research, clinical trials, and longitudinal studies.

3.4.2 Machine Learning Approaches

Traditional approaches for data correction and imputation rely on statistical techniques and expert knowledge, which can be time-consuming and subjective. With the recent advances in machine learning, there has been growing interest in using machine learning approaches for data correction and imputation. In this subsection, we have selected the most widely used techniques for anomaly correction and imputation in time series data over the past few years.

k-Nearest Neighbours (KNN)

It works on the assumption that neighbouring data points belong to the same class. In other words, a new data point is more likely to have the same class label as its k-nearest neighbours than distant data points (Peterson, 2009). k-NN identifies the neighboring points through a measure of distance and the missing values can be estimated using completed values of neighboring observations.

Decision Tree

A decision tree is a machine learning system that shows all possible outcomes and their connected routes in the form of a tree. This approach imputes missing values by first constructing decision trees to identify missing values for each variable, and then using the associated trees to fill in the missing values for each variable (Twala, 2009).

Random Forest

A random forest is an ensemble of decision tree algorithms. It is a development of the bagging technique, an effective ensemble approach to decision trees in which each decision tree is fitted to a slightly different training dataset. As a result, the predictions and errors made by each tree are more distinct from each other. When the predictions from these less correlated trees are combined to make a prediction, the result can be better than if they were all bagged together (Breiman, 2001).

MissForest (MF):

It is another machine learning-based data imputation technique that based on the Random Forest (RF) algorithm which has been created by Stekhoven and Bühlmann (2012). It can be divided into 3 main steps. Firstly, replace the missing values with the mean (for continuous variables) or the most frequent class (for categorical variables). Secondly, the observed observations are served as the training set and the missing observations are served as the prediction set. The training sets and the prediction sets are fed into a RF model. Then, the RF model's predictions are put in place of the prediction set, creating a transformed dataset. Finally, one imputation loop is complete when all missing variables are imputed. Imputations are repeated.

Support Vector Regression (SVR)

SVR is a technique similar to support vector machine (SVM), except it is used to solve regression issues. The goal of regression analysis is to use a training set to identify a function (hyperplane) that can map an input domain to a target domain within the decision boundary and have the least error rate. So, we can use this function to predict the values that are missing in our dataset.

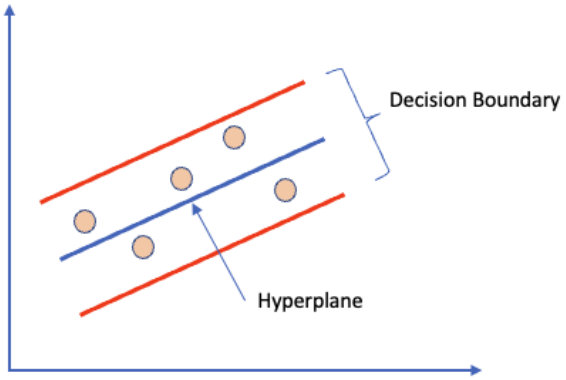


Figure 3.8: SVR concept illustration.

3.4.3 Neural Network Approaches

Neural networks have been used to estimate missing values by training them as regression models using only the remaining complete features from incomplete datasets as inputs. This allows the model to learn from the incomplete datasets. In the last

decade, there has been much research using neural networks for data imputation, as reviewed by Fang and Wang (2020); Saad et al. (2020). However, the type of neural network that is widely used was developed based on RNN, LSTM, and GAN.

RNN-Based

Beginning with the RNN-Based method, delaying the output of RNNs by a particular number of time frames has been used to incorporate future knowledge into present predictions. But a bi-direction RNN (BRNN) takes into account all input sequences in both the past and the future when predicting the output vector Schuster and Paliwal (1997). In order to do this, one RNN processes the sequence forward in time. Another RNN processes the sequence in the reverse direction of time, from end to beginning, with no interaction between them. This concept has been applied for data imputation by Cao et al. (2018). They developed bidirectional recurrent imputation for time series (BRITS). Instead of using complete data for training a model, they use a sequence with missing data as an input. Since RNN uses the information from prior inputs to generate the future values, they replace the current actual values with the obtained imputation and validate them with the future observed values. But the estimation error cannot be obtained immediately when there is some missing data. So, they used BRNN to figure out the missing value by getting the estimation error from both directions, forward and backward.

LSTM-Based

It is possible to boost the capacity of BRNNs by stacking hidden layers of LSTM cells, a technique known as bidirectional LSTM (Bi-LSTM) (Ma et al., 2020; Kulanuwat et al., 2021). It is more powerful than LSTM networks that are just used in one direction. The Bi-LSTM model exploits the advantages of the BRNN while also addressing the issue of vanishing gradients (Salehinejad et al., 2017). This technique, however, has a disadvantage in that it increases computing complexity as compared to LSTM, which is because of the forward and backward learning processes.

GAN-Based

Goodfellow et al. (2014) was the first to define the generative adversarial networks (GAN) architecture. The GAN model's architecture is made up of two sub-models: a generator model, which is in charge of making new examples, and a discriminator model, which is in charge of figuring out if the new examples are real examples from the domain or fake examples made by the generator model, as illustrated in Figure 3.9.

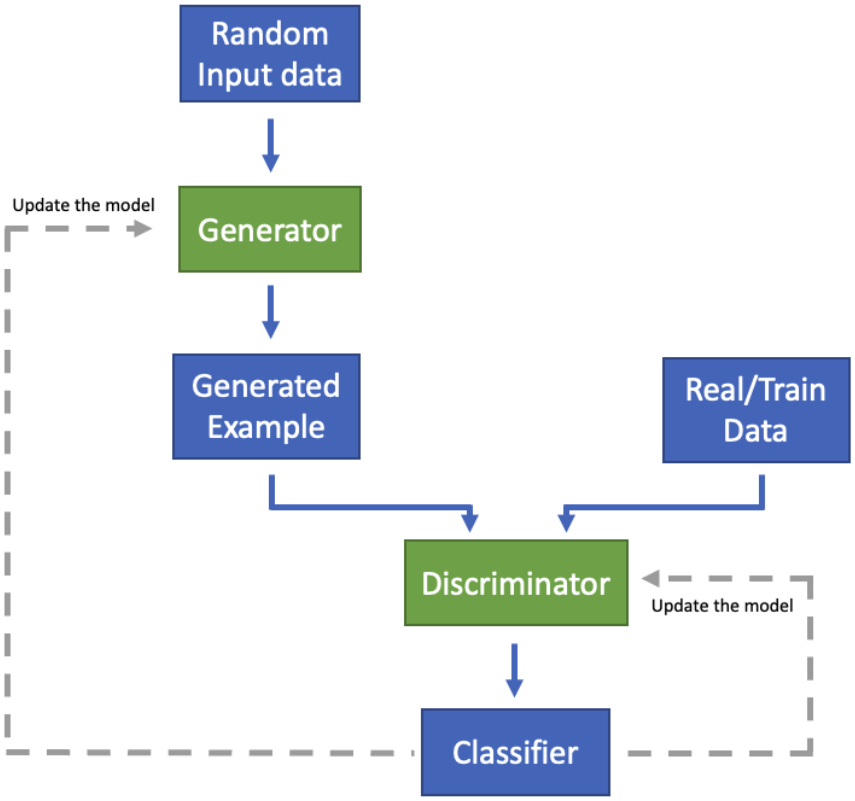


Figure 3.9: GAN concept illustration.

3.5 Ensemble Methods

The ensemble technique is another alternative that has been regularly employed to increase model performance. The basic idea behind this strategy consists of producing results using numerous models, and then integrating multiple outcomes into a consistency function to obtain final results.

Based on the underlying machine learning techniques, there are two basic forms of ensembles: *homogeneous* and *heterogeneous*. A homogeneous ensemble is constructed from learners of the same type, e.g., a set of decision trees. A heterogeneous ensemble is

constructed from several kinds of learners, such as a support vector machine, k-nearest neighbour, or neural networks.

To generate the ensemble models consists of three main steps as follows:

i) Data preparing: A dataset, which contains a number of records in which each instance represent a set of input features along with the target class, used to train the base models.

ii) Model generator: various types of model have been generated by training with the prepared data from the previous step and are placed into a pool as the member candidate for building ensemble.

iii) Combiner: This is the step for making the final decision from among the selected models in the model pool.

In the combiner step, the results of each model can be combined by voting or averaging schemes. Voting is used for classification, which is the main focus of our work, and averaging is used for regression. Our research is a classification problem, so we will focus on voting schemes. Voting classification can be divided into two main different schemes:

- *Majority Voting*: The predictions of each model in an ensemble have to be aggregated, and the final prediction is the class that gets the most votes. To prevent the possibility of a tie in voting, we will construct each of our ensembles using an odd number of classifiers. This means that the number of classifiers in the ensemble will not be evenly divisible by two, which reduces the likelihood of an equal number of votes for different classes. However, it is important to note that having an odd number of classifiers does not completely eliminate the possibility of ties if there are the same number of votes for different options.

- *Weighted Voting*: When multiple models with varying levels of performance are involved in decision-making processes, treating all models equally can lead to unsatisfactory results. This is because different models may have different strengths and weaknesses, and giving equal weight to all predictions may not be

the most effective approach. So, we devised a weighted voting mechanism to take this difference into consideration when making a final decision in an ensemble. With the weighted voting method, the contribution from a model is weighed by its performance. For a model m_i, after it has been trained with the training data, its weight score w_i is derived by using its F1-score that is calculated on the given validation dataset, so we will have a set of F1-scores of each model, $F_1m = \{F_1m_1, F_1m_2, ..., F_1m_M\}$. Then these F1-scores will be ranked to find the maximum and minimum scores. Finally, calculate the normalised weighting score w_i for module m_i using the equation below:

$$w_i = \frac{F_1m_i - min(F_1m)}{max(F_1m) - min(F_1m)}, \ \forall \ i = 1, ..., M \tag{3.5.1}$$

The output of an ensemble, $\Phi(x)$, is calculated by multiplying the weight with the output of an individual module and taking the argument of maxima as follows:

$$\Phi(x) = argmax \sum_{i=1}^{M} w_i m_i(x) \tag{3.5.2}$$

Where M is the number of models in an ensemble, $m_i(x)$ is the predicted class of model i.

Ensemble methods can provide several benefits over individual models, including improved accuracy, stability, and robustness. It can also provide insights into the underlying patterns in the data. However, ensemble methods can be computationally expensive and may require significant computational resources.

3.6 Evaluation

This section describes the existing methods and measures that will be used to carry out the experiments and evaluate the results.

3.6.1 Binary Class Performance Measures

As this is essentially a binary classification problem, we chose *Recall(R), Precision(P)* and *F1* measures. They are calculated with the following equations based on a confusion matrix as shown in Table 3.1.

Table 3.1: Confusion Matrix of Performance Measures.

		Predicted	
		Anomaly	Normal
Actual	Anomaly	TP	FN
	Normal	FP	TN

Recall : The number of correctly classified anomalies is divided by the total of successfully classified anomalies and mistakenly classified normal.

$$Recall = \frac{TP}{TP + FN} \tag{3.6.1}$$

Precision : The number of correctly classified anomalies is divided by the total of successfully classified anomalies and mistakenly classified anomalies.

$$Precision = \frac{TP}{TP + FP} \tag{3.6.2}$$

$F1$: The harmonic mean of precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.6.3}$$

Where TP, FP, FN, and TN denote the number of True Positive, i.e correct classification for anomaly data, False Positive - the number of incorrect classification for anomaly data, False Negative - the number of incorrect classification for normal data, and True Negative - the number of correct classification for normal data, respectively.

3.6.2 Critical Difference Diagram

To make statistical comparisons, we implemented a statistically rigorous test for multiple classifiers across many datasets. This approach was initially described by Demšar

(2006) and is intended to examine the statistical significance of classifiers. This technique takes the strategy of testing the null hypothesis against the alternative hypothesis. The null hypothesis states that no difference exists between the average rankings of k algorithms on N datasets. The alternative hypothesis is that at least one algorithm's average rank differs.

In the first place, the k methods are ranked according to their performance over the N datasets, then the average ranking of each algorithm is calculated. To test the null hypothesis, the Friedman test is calculated using Equation 3.6.4.

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{3.6.4}$$

Where R_j is the rank of the jth of k algorithms on N datasets and the statistic is estimate using a Chi-squared distribution with $k-1$ degrees of freedom.

If the null hypothesis is rejected at the selected significance level α, the post-hoc Nemenyi test is used to compare all classifiers to each other. The Nemenyi test is similar to the Turkey test for ANOVA, and uses a critical difference (CD), which is presented in Equation 3.6.5

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{3.6.5}$$

Where q_α is calculated by the difference in the range of standard deviations from the smallest valued sample and the largest valued sample. The results of these tests are often visualised using a CD diagram. Classifiers are shown on a number line based on their average rank across all datasets, and bold CD lines are used to connect classifiers that are not significantly different. In Figure 3.10 we present an example critical different diagram.

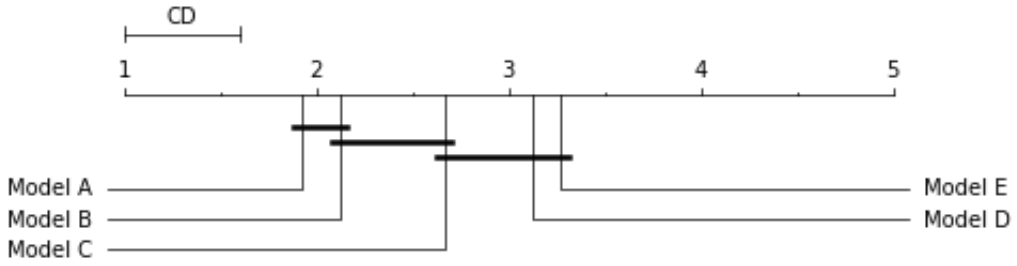


Figure 3.10: An example Critical Difference (CD) diagram demonstrating how to interpret the results from a pairwise comparison of five models over multiple datasets.

To interpret the results of the critical difference is straightforward and demonstrates their utility. Models A and B are in the same CD line and are not significantly different. However, model A is significantly different from models C, D, and E. Model B is not significantly worse than models A or C, but is significantly better than models E and D. Model C is significantly worse than model A, but is not significantly better than model D or model E. Finally, models D and E are not significantly worse than each other or model C, but are significantly worse than models A and B. This diagram provides a good breakdown of the rankings of each classifier and how they compare to one another. The bold CD lines give an understanding of where classifiers are similar in performance and where one or more classifiers may be better than others.

3.6.3 Evaluating imputation methods

To verify the performance of our proposed methods, we compared them to existing imputation methods using three distinct measures, including root mean square error (RMSE), mean absolute error (MAE), and similarity (Sim). They are detailed as follows:

- *RMSE:* The average squared difference between the imputed value $\hat{y}$ and the respective genuine value y is referred to RMSE. This metric is very useful for determining overall correctness. The technique with the lowest RMSE would be the most effective.

$$RMSE(\hat{y}, y) = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(\hat{y}_i - y_i)^2}$$

Where T is the number of missing values.

- *MAE:* This is compute as average of the absolute difference between imputed values $\hat{y}$ and actual values y, which calculated by:

$$MAE(\hat{y}, y) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

The method that is more effective will have a lower MAE.

- *Sim:* $\mathrm{Sim}(\hat{y}, y)$ defines the similar percentage between the imputed value ($\hat{y}$) and the actual data (y). It is calculated by:

$$Sim(\hat{y}, y) = \frac{1}{T}\sum_{i=1}^{T}\frac{1}{1 + \frac{|\hat{y}_i - y_i|}{max(y) - min(y)}}$$

A higher similarity indicates a better ability to infill missing values.

3.7 Related Work

In this section we will present some work which is directly related to this thesis. This section will consist of subsections including data preprocessing, anomaly detection, anomaly correction and imputation, and ensemble methods.

3.7.1 Related Work for Data Preprocessing

Time series analysis has become increasingly important in many fields, including finance, economics, and engineering. However, before conducting any analysis, it is essential to preprocess the data to ensure that the time series is suitable for analysis. This literature review will discuss some common data preprocessing techniques used in time series analysis.

The first step in data preprocessing for time series is to check for missing values. Missing values can occur due to various reasons, such as data collection errors or equipment malfunctions. The presence of missing values can affect the accuracy of the analysis, so

it is essential to impute the missing values before proceeding with any analysis. Several imputation methods can be used (Rashid and Gupta, 2021). For example, Raja and Thangavel (2020) proposed a new method for handling missing values in data mining using unsupervised machine learning techniques, called rough K-means centroid-based imputation. It combines soft computation approaches with clustering techniques to overcome the problems of inconsistency. The method is compared with other imputation algorithms and is found to perform well, particularly in handling uncertainty and inconsistencies in datasets. The results obtained by the proposed method are promising for the prediction of missing values. Another study from Zhang et al. (2021b) suggested using a generative adversarial network (GAN) with an encoder network to fill in missing values in multivariate time series data. The proposed model outperforms existing state-of-the-art methods in imputation tasks and downstream applications such as classification and regression. However, the choice of method depends on the nature of the data and the amount of missing data.

Data normalization is also an important preprocessing step in time series analysis as disscussed in Asesh (2022). Normalization refers to the process of transforming the data to a common scale, enabling comparisons between different time series. The normalization techniques include min-max scaling, z-score normalization, and log transformation. The choice of normalization technique depends on the distribution of the data and the intended analysis.

In conclusion, data preprocessing is a crucial step in time series analysis (Alasadi and Bhaya, 2017; Ramrez-Gallego et al., 2017; Zelaya, 2019). It ensures that the data is suitable for analysis and eliminates any errors that could affect the results.

3.7.2 Related Work for Anomaly Detection

The data generated from a water level monitoring station is a time series. There are many methods for detecting anomalies in time series data. One basic approach is to use statistics-based methods, as reviewed in Rousseeuw and Hubert (2011); Zimek and Filzmoser (2018). For example, simple and exponential smoothing techniques were used to identify anomalies in a continuous data stream of temperature in an industrial

steam turbine (Kumar et al., 2012). But in general whilst they provided a baseline, they have a disadvantage in handling trends and periotics, e.g., the water level will dramatically rise before the flood, which differs considerably from the other data points and may lead to an increased false alarm rate. In addition, they can be affected by the types of anomaly and some work well for a certain type of problem. For example, for missing and outlier values, when the data is normally distributed, the K-means clustering method (Lin et al., 2018) is usually used, as it is simple and relatively effective. Kulanuwat et al. (2021) used the median and the median absolute deviation (MAD) to detect anomalies in telemetry water level. The findings showed that their approaches produced good results for non-cyclical data behaviors. The two-sided median method and the one-sided median method were used to detect unexpected jump values in time series (Basu and Meckesheimer, 2007). However, there is unfortunately no general guideline for choosing a method for a given problem.

Most of their models have been computed based on a sliding window (Golab, 2004; Ding and Fei, 2013; Jiang et al., 2011). A time window is specified with n continuous input data points, and a model has been generated from the data in this window. Then the window is shifted by a given step size along the time series and the model is recomputed on the next window. This has two drawbacks: the computed values are limited to a specific window and it is time-consuming.

Some machine learning methods were used to detect anomalies. But as almost all the applied methods use supervised learning algorithms to learn from the labelled historical data, there are several issues with them. 1) There is not enough labelled data for a learning algorithm to learn well to generate good enough models. 2) Because time-series data is continuous or streaming in real time, models learned from historical data may need to be retrained as new data is received (Hill et al., 2007). 3) Each model is limited by the data it has been trained with, so it may be suitable to detect a particular type of error, but not other anomalies.

For hydrological data, Yu et al. (2014) developed a time series outlier detection method by comparing the observed data with the result from a forecasting model that learns from the historical data enclosed in a sliding window. Similar techniques were used by

Ma et al. (2017) to identify outliers in meteorological data. However, the optimization of window size is time-consuming for different hydrological elements and user requirements. van de Wiel et al. (2020) used regression-based approaches to locate outliers in water level time series with univariate and multivariate models. Their research reveals that multivariate outlier models outperform univariate outlier detection algorithms. However, optimisation of threshold values when using regression-based models may require a certain amount of time and complexity for various hydrological elements and user requirements.

Change point detection (CPD) is an important method for time series analysis. It indicates an unexpected and significant change in the analysed time series stream data and has been studied in many fields, as surveyed by Aminikhanghahi and Cook (2017) and Truong et al. (2020). However, the CPD has no ability to detect anomalies since not all detected change points are abnormalities. Many studies are being conducted to solve this problem by integrating CPD with other models to increase anomaly detection effectiveness. For example, Apostol et al. (2021) presented new techniques, called rule-based decision systems, that combine the results of anomaly detection algorithms with CPD algorithms to produce a confidence score for determining whether or not a data item is indeed anomalous. They tested their suggested method using multivariate water consumption data collected from smart metres, and the findings demonstrated that anomaly detection can be improved. Moreover, it has been proposed to detect anomalies in file transfer by using the CPD to detect the current bandwidth status from the server, then using this to calculate the expected file transfer time. The server administrator has been notified when observed file transfers take longer than expected, which may mean it may have something wrong (Dao et al., 2018). Siris and Papagalou (2004) investigated the cumulative sum (CUSUM) algorithm for change point detection to detect SYN flood attacks. The results demonstrated that the proposed algorithm provided robust performance with both high and low intensity attacks. Although change point detection performed well in many domains, the majority of them focused on changes in the behaviour of time series data (sequence anomaly) rather than point anomaly, which is my primary research emphasis. Furthermore, water level data at certain stations is strongly periodic with tidal effects, resulting in numerous data

points changing from high tides to low tides each day, which is typical behaviour.

To perform further in-depth analysis, the time domain can be converted to different domains. For example, Ren et al. (2019) converted data to the spatial domain by using Saliency Map (SM) (Hou and Zhang, 2007) to find outliers in real-time services and a deep Convolutional Neural Network (CNN) to improve performance. Additionally, Pan et al. (2020) also used saliency maps to extract feature information from multivariate time series, which improved the accuracy of prediction.

Ouyang et al. (2017) demonstrated a method for performing anomaly detection by extracting hierarchical time series features. But the challenge is how to find the smallest size of a window in datasets. While Yang et al. (2021) applied a time series features extractor, TSFRESH, and a genetic algorithm-based feature selection method to extract dominant features from stream data. They also trained an XGBoost model to record the pattern of stream data to detect the appropriate anomaly detection service in real time.

Although extracted features can improve the performance of models, most existing feature extraction methods divide data into windows for calculating new features. However, inappropriate window size can lead to time-consuming and extracted non-useful features, which lead to the poor performance of models.

In recent decades, machine learning methods, including deep neural networks (DNNs), have been satisfactorily implemented in various hydrological issues such as outlier detection (Yu et al., 2014; van de Wiel et al., 2020), water level prediction (Yang et al., 2017; Park et al., 2022), data imputation (Vu et al., 2021), flood forecasting (Chang et al., 2018), streamflow estimation (Liu et al., 2022), etc. For example, Zhou et al. (2019) proposed the R-ANFIS (GL) method for modelling multistep-ahead flood forecasts of the Three Gorges Reservoir (TGR) in China, which was developed by combining the recurrent adaptive-network-based fuzzy inference system (R-ANFIS) with the genetic algorithm and the least square estimator (GL). Chang et al. (2020) presented a flood prediction by comparing the expected typhoon tracking and the historical trajectory of typhoons in Taiwan in order to predict hydrographs from rainfall projections impacted by typhoons. The PCA-SOM-NARX approach was developed by Chang et al.

(2022) to forecast urban floods, combining the advantages of three models. Principal component analysis (PCA) was used to derive the geographical distributions of urban floods. To construct a topological feature map, high-dimensional inundation recordings were grouped using a self-organizing map (SOM). To build 10-minute-ahead, multistep flood prediction models, nonlinear autoregressive with exogenous inputs (NARX) was utilised. The results showed that not only did the PCA-SOM-NARX approach produce more stable and accurate multistep-ahead flood inundation depth forecasts, but it was also more indicative of the geographical distribution of inundation caused by heavy rain events. Even though we can use forecasting methods to find anomalies by using prediction error as a threshold to classify data points as normal or not, it may take time to find the suitable threshold for each station.

Reinforcement Learning (RL) is an algorithm that imitates the human learning process. It is based on the self-learning process in which an agent learns by interacting with the environment without any assumptions or rules. With the advantage of being able to learn on their own, it can identify unknown anomalies (Pang et al., 2020), which gives it an edge over other models. RL has been applied to a variety of applications such as games (Mnih et al., 2013; Silver et al., 2017), robotics (Kormushev et al., 2013; Polydoros and Nalpantidis, 2017), natural language processing (Sharma and Kaushik, 2017; Luketina et al., 2019), computer vision (Le et al., 2021), etc. It has also been used in some studies to detect anomalies in data, such as an experiment by Huang et al. (2018) that shows the use of the deep Q-function network (DQN) algorithm to detect anomalies in time series. Network intrusion detection systems (NIDS) are developed by Hsu and Matsuoka (2020), based on deep reinforcement learning. They utilised it to identify anomalous traffic on the campus network with a combination of flexible switching, learning, and detection modes. When the detection model performs below the threshold, the model is retrained. In the comparison against three traditional machine learning approaches, their model outperformed on two benchmark datasets, NSL-KDD and UNSW-NB15. A binary imbalanced classification model based on DRL was introduced by Lin et al. (2020). They developed the reward function by setting the rewards for the minority class to be greater than the rewards for the majority class, which made DRL paying more attention to the minority class. They compared it to

seven imbalanced learning methods and found that it outperformed other models in text datasets and extremely imbalanced data sets. Although deep learning and RL methods have achieved excellent results in time series, one common issue is that their performance varies and it is hard to predict when they do better and when they perform relatively poor.

An Autoencoder is an unsupervised learning neural network, it comprised two parts, an *encoder* and a *decoder*. They are effectively used for solving many applied problem, from face recognition (Gao et al., 2015; Xu et al., 2017), anomaly detection (Pereira and Silveira, 2018; Zhou and Paffenroth, 2017; Zong et al., 2018; Gong et al., 2019) to noise reduction (Jiang et al., 2017; Maas et al., 2012; Chiang et al., 2019). In hydrological domain, Kao et al. (2021) presented the SAE-RNN model, which combines the stacked autoencoder (SAE) with a recurrent neural network (RNN) for multistep-ahead flood inundation forecasting. Starting with SAE to encode the high dimensionality of input datasets (flood inundation depths), and then utilising an LSTM-based RNN model to predict multistep-ahead flood characteristics based on regional rainfall patterns, and then decoding the output by SAE into regional flood inundation depths. They conducted experiments on datasets of flood inundation depths gathered in Yilan County, Taiwan, and the findings demonstrated that SAE-RNN can reliably estimate regional inundation depths in practical applications.

3.7.3 Related Work for Data Imputation

Time series imputation methods are classified into two types depending on the input data. The first technique, which is the primary focus of our study, is univariate algorithms, which depend on a single variable to impute missing values. The second is multivariate approaches, which estimate missing data by examining the relationship between each variable.

Several fundamental approaches to filling missing data in univariate, including mean, median, LOCF (last observation carried forward), and interpolation techniques, are frequently utilised. When only one or a few consecutive missing data points are present, those methods provide acceptable results. However, when the missing gaps are large,

the results are bad.

Some studies have been conducted in the recent decade with the goal of imputing missing values in meteorological and hydrological data. For example, the study by Yang et al. (2017) used the mean of nearby points imputation methods for imputing the missing data of the integrated data set from the water level and atmospheric data. Lai and Kuok (2019) suggested a method known as bayesian principal component analysis (BPCA) to impute missing values in rainfall data; the findings showed that BPCA outperformed KNN when dealing with large continuous missing gaps. while the research by Gao et al. (2018); Pratama et al. (2016) provided a review of applied statistical and machine learning methodologies for imputed anomalies and missing data.

With the significant contribution of ML in many domains, it has also been applied to imputation tasks. Kim et al. (2015) compared ML, artificial neural network, and physical-based model for recovering streamflow data. The result revealed that ML were generally better than other at capturing high flows. Dwivedi et al. (2022) used random forest to impute the continuous gaps and extreme of sub-hourly ground water. Li et al. (2022) improve the availability of IoT multivariate data and ability of anomaly detection by applied the imputation techniques to impute the detected anomalies data. Others research that using a ML model to estimate the missing values are averaging the prediction from two directions (forward and backward) as the final imputed values (Akouemo and Povinelli, 2014; Bokde et al., 2018; Phan, 2020). Two forecasting models for estimating the missing value based on the forecaster and the backcaster of time series were proposed by Moahmed et al. (2014).

All of the approaches that use subsequences before and after missing gaps for searching or training models are based on the assumption that current values of time series have an influence from past or previous values. But in the case of water level data, only using values from the past or future may not correspond to the current situation since the behaviour of water levels has altered due to climate change.

Replacing the missing gap by using the values from their most similar subsequence is extensively used in many domains. Dynamic time warping (DTW) is an excellent

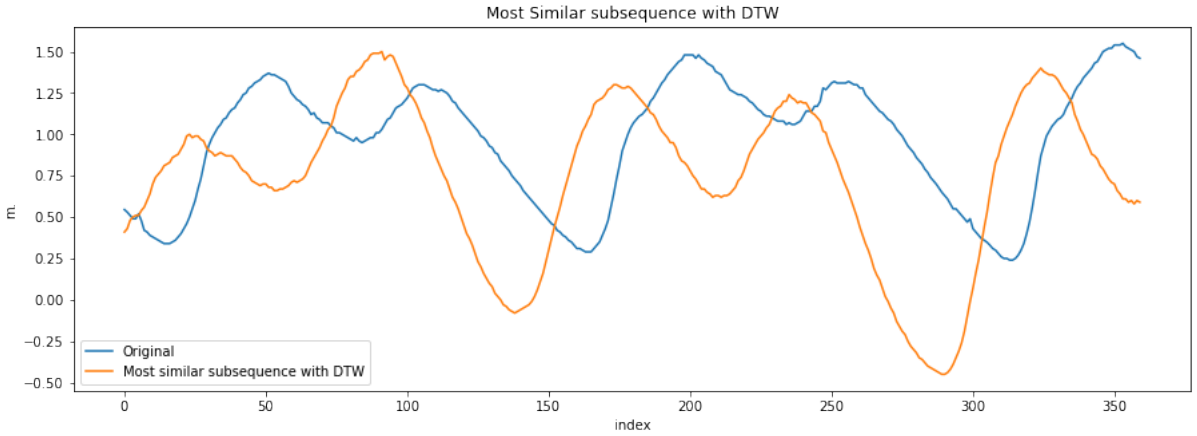


Figure 3.11: Most similar subsequence that has same dynamic but difference position

technique of this kind, which is extensively used in many domains, such as Tormene et al. (2009) used DTW to find the most similar incomplete time series from the stored reference of an arm movement sensor. Or, the research by Caillault et al. (2020), the imputation of missing data for univariate time series was proposed, which used derivative dynamic time warping (DDTW) developed by Keogh and Pazzani (2001) to search for subsequences before the missing gap, and then to repair the gaps by replacing them with the next subsequence that is most similar. The disadvantage of using dynamic time warping is that it takes time. They addressed this issue by extracting sequence features in sliding windows using a shape-feature extraction algorithm (Caillault et al., 2016), then calculating DDTW only if the correlation between the shape-feature of this window and the subsequences before the missing gap is very high. The results demonstrate that their method produces superior outcomes when dealing with time series with high correlation and strong seasonality.

Although DTW can find the most similar pattern that has similar dynamics, it may warp the shape by expanding or compressing, so the position of missing gaps may not be the same position as the original pattern that we are looking for as depicted in Figure 3.11.

The another way that uses for searching the most similar subsequence is to find two subsequences that have the lowest Euclidean distance as an indication of similarity. Since we need to calculate the distance of every pairwise in time series, the time needed to search for matches in a large time series dataset can be long and hence is considered

as a disadvantage of this method. To address this issue Yeh et al. (2016) developed the techniques called matrix profile (MP), to speed up the process. An MP gives the distances between all subsequences and their nearest neighbours and thus can be used to efficiently extract some patterns characterised by a time series, such as motifs and discords. Very similar subsequences in a time series are called motifs, whereas very differing subsequences are called discords.

In the last two decades, deep learning techniques have been widely applied to various problems in time series analysis. Various studies have exploited deep learning to impute the missing data. Zhang and Thorburn (2021) proposed a dual-head sequence-to-sequence imputation model (Dual-SSIM) for water quality data imputation. By averaging the prediction from gated recurrent units (GRU) with the information before and after the missing gaps. In this proposal, the model imputes missing data more accurately than 5 benchmarks. Kulanuwat et al. (2021) reported work that used three approaches for imputing missing data, including linear interpolation, spline interpolation, and bidirectional LSTM for data imputation on telemetry water level data. Spline interpolation performed better on non-cyclical data, while bidirectional long short-term memory (BiLSTM) beat other interpolation approaches on a particular tidal data pattern. But, one common disadvantage that all deep learning neural networks have is very time consuming, which makes them less practical in real time applications, such as water level analysis and flood forecasting.

Although neural networks offer great performance and accuracy in many domains, it is not possible to design a single universal network architecture that works in all instances. Also, the models need to be retrained to improve how well they work, which makes it hard for users to know when they need to be retrained.

3.7.4 Related Work for Ensemble Methods

Some of the previous research has shown that it is possible to combine various individually trained models to produce a model that is more accurate than any of the single models (Opitz and Maclin, 1999). This combination of multiple models to work together is called the ensemble method. It has been demonstrated to be effective in

a wide range of real-life problems, such as weather forecasting (Gneiting and Raftery, 2005), detecting anomalies in cellular networks (Ciocarlie et al., 2013), wireless sensor networks detection (Curiac and Volosencu, 2012), gene expression data for cancer classification (Tan and Gilbert, 2003).

Time series based on ensemble modelling have recently attracted attention. In a study by Yu et al. (2017), they introduced the method EN-RTON2, which is an ensemble model with real-time updating using online learning and a submodel for real-time water level forecasts. However, they experimented with fewer datasets, a smaller number of records, and lower data frequency than our datasets. Furthermore, the authors offer no indication of the time necessary for training models and forecasting, which may be inadequate in our case given the number of stations and frequency of data transmission. The ensemble models were proposed by Iftikhar et al. (2020), which applied the sliding window based ensemble method to find the anomaly pattern in sensor data for preventing machine failure. They used a combination of classical clustering algorithms and the principle of biclustering to construct clusters representing different types of structure. Then they used these structures in a one-class classifier to detect outliers. The accuracy of these methods was tested on a time series of real-world datasets from the production of industry. The results have verified the accuracy and the validity of the proposed methods.

Kieu et al. (2019) proposed two autoencoder ensemble frameworks for unsupervised outlier identification in time series data based on sparsely connected recurrent neural networks, which address the issues from Chen et al. (2017) that given the poor results when using autoencoder with time series data. In one of the frameworks called the Independent Framework, multiple autoencoders are trained in a manner that is independent of one another, whereas in the other framework, the Shared Framework, multiple autoencoders are trained jointly in a manner that is multi-task learning. They experimented by using univariate and multivariate real-world datasets. Experiment results reveal that the suggested autoencoder ensembles with a shared framework outperform baselines and state-of-the-art approaches. However, a disadvantage of this method is its high memory consumption when training many autoencoders together. The time

factor is a challenge for real-time flood forecasting; typical physically based numerical models are often too slow for use in real-time forecasting systems. To address this issue, Berkhahn et al. (2019) proposed an ensemble of neural network models for the prediction of maximum water levels during a flash flood event. According to the results of this study, even if the outputs of the ensemble do not beat physically based models, they may be considered suitable for real-time forecasting. Additionally, it offers alternatives to enhance the accuracy of predictions, which may beat the other physical models.

3.8 Summary

In this chapter, the literature on the technical background of the data and methodologies developed for detecting and correcting anomalies in data is reviewed. We start by introducing data pre-processing, then review a number of algorithms for detecting and correcting anomalies, as we will use those for our experimentation. Furthermore, we provide a description of the ensemble technique that involves a review of the most common ensemble methods and the different ways of combining models. Finally, a survey of the related studies that covered anomaly detection, data imputation, and ensemble methods with hydrological data is provided.

The work that was done on anomaly detection using statistical models is going to be detailed in the next chapter.

Chapter 4

Anomaly Detection with Statistical Models and Their Ensembles

Contributing Publications

- Khampuengson, T., Bagnall, A., and Wang, W. (2020). Developing ensemble methods for detecting anomalies in water level data. In *International Conference on Innova- tive Techniques and Applications of Artificial Intelligence*, pages 145–151. Springer.

4.1 Introduction

Several machine learning techniques were employed to identify anomalies, but most of them utilised supervised learning algorithms to learn from labelled historical data, which led to various challenges. Firstly, there is a scarcity of labelled data, making it difficult for the learning algorithm to produce satisfactory models. Secondly, since time is a continuous or streaming variable, the models developed from past data may require retraining with newly arriving data (Hill et al., 2007). Finally, each model is restricted to the data it has been trained on, making it effective at detecting specific types of errors but not all anomalies.

While statistical models do not require labelled data. Furthermore, they are widely used to identify anomalies since they are not only simple to implement but also have a low computational cost, making them an attractive option for tasks where time-consuming processing is a concern. Using the advantages of such approaches, we intend

to create statistical models to detect anomalies in water level data from telemetry stations as fast and as accurately as possible.

4.2 Methods

In this section we described the details of chosen statistical model, and development of sliding windows algorithm.

4.2.1 Statistical Models

Seven basic models were selected as the member candidates for building an ensemble, which are *Auto Regression (AR), Differenced Based(DB), Interquartile Range (IQR), Sigma rules of thumb (K-Sigma), Moving Average Smoothing(MAS), Slope as an Angle (SA), Z-Score*. They were chosen because they are simple and use no or few data for training. As a result, they take much shorter time to calculate, thus are suitable for detecting anomaly in near real-time situations. Each of these 7 models is briefly summarised below.

- *Autoregression (AR)*

  AR methods are a class of time-series models used for anomaly detection. These methods involve modelling the time series data using previous values of the same series and using the difference between the actual and predicted values to identify anomalies. Once the model is fitted, the difference between the actual and predicted values is calculated, and observations that fall outside a certain threshold or confidence interval are flagged as anomalies.

- *Difference Based (DB)*

  In time series data, anomalies can be detected by identifying values that deviate significantly from the average difference between normal values. By computing the difference between consecutive observations and setting a threshold, we can identify and flag as anomalies those values that exceed it.

- *Interquartile Range (IQR)*

An outlier or extreme value can be detected by using median as opposed to the mean, which is often summarized by the difference between the first and the third quartiles, called the *Interquartile Range(IQR)*, the values that are not in this range will be defined as an anomaly.

- *Sigma Rule of Thumb (K-Sigma)*

  The K-Sigma method is a statistical technique that involves defining boundaries for normality as k times the standard deviation away from the mean and identifying any observation that falls outside the boundaries as an anomaly, making it useful for detecting anomalies in datasets.

- *Moving Average Smoothing (MAS)*

  This technique used to remove the noise from the data. It is simple and commonly used in time series analysis and forecasting. User has to define the number of raw observation data, called windows size. The value at time period t is calculated by finding the average value of the raw observations inside the windows. The anomaly has been defined by compare the expected values and a root mean square error (RMSE) is calculated.

- *Slope as Angle (SA)*

  The anomaly data is the point that is usually significantly different from the previous data point. Consequently, the value that could be anomaly will have a slope angle close to 90°. In this research, we defined a point as an anomaly if there is angle slope more than 45°.

- *Z-Score*

  Z-score is the difference between the value and the mean expressed as the number of standard deviations. The observation values that has a Z-score lower than -2.5 and greater than 2.5, it will be considered as anomalous.

4.2.2 A Modified Sliding Window Algorithm

Every model, except SA, requires to employ a *sliding window* algorithm to find the threshold θ for the current window and uses it to identify an anomaly. But it has a drawback, that is, when the window is moved forward to the next step, the detected anomaly data will be included in the window and because this anomaly data has not been removed or corrected, it will add some bias to the threshold and then as a consequence it will affect the prediction of next possible anomaly value along the time series.

We then modified the sliding window algorithm by updating the sequence in the window once the next member is detected as a normal value, so-called *only normal sliding windows*. The algorithm of new sliding windows as show in algorithm 1.

Algorithm 1 Only Normal Sliding Windows

```
procedure NEW SLIDING WINDOW(Model, Data)
    start = 0
    end = windowSize
    anomaly = array()
    while end ≤ count(Data) do
        input = Data[start:end]
        nextVal = data[end+1]
        threshold = Model(Data[start:end])        ▷ Findind the threshold values
        if nextVal > threshold then                ▷ Detected as anomaly data
            Data[end+1].remove
            anomaly.append[actual]                 ▷ Store anomaly values in an array
        else                                       ▷ Detected as normal data
            start = start + 1
            end = end + 1
```

4.2.3 Ensemble Methods

We have identified 7 classic methods for detecting anomalies. In general they are simple and fast so have been used in many. But each model has its limits, only suitable for detecting some particular types of errors. Nevertheless, they can be constructively combined into form an ensemble to work together so that they can compensate each other's weakness and then perform better than individual models working separately. However, the fundamental issues of ensemble methods and emphasised that a successful ensemble can be built with some appropriate models selected by using suitable criteria,

otherwise an ensemble may not improve at all (Wang, 2008).

So, in this experiment, we developed two types of ensembles. The first type is a *simple ensemble*, the second type is the *complex ensemble* which has a compound structure of ensemble of ensembles. This section describes how these ensembles are constructed in detail.

Simple ensemble

A simple ensemble is built with some models selected from 7 basic models briefly mentioned earlier. But a key question is what criterion we should use to select a model as a member of an ensemble. We devised a new scoring function (see below) to calculate the goodness score of a model and then use this score to determine if a model is good enough to be selected. Once an ensemble is formed, a decision-making function is applied to work out the final output of the ensemble. In this study, a simple majority voting method is used. So, a simple ensemble operates in 2 main stages: *Model Selection* and *Decision Making*.

- *Model Selection*: It is done in three steps:

  (1) Evaluating the accuracies of models.

  We firstly use three different measures TP, FP and FN as criteria to asses the performance of a model.

  (2) Ranking the models with different criteria.

  With those 3 measures, we produce 3 rankings R_1, R_2, and R_3 respectively. Then for each ranking of the models, we calculate their score in ranking R_j as follows:

$$S_{(m_i,R_j)} = \frac{N + 1 - r_{(m_i,R_j)}}{N}, \in [\frac{1}{N}, 1] \tag{4.2.1}$$

Where, where: $S_{(m_i,R_j)}$ = Score of model m_i in ranking R_j.
$r_{(m_i,R_j)}$ = ranking position of m_i in R_j.
N = number of models in a ranking.
i = index of models: 1, 2, ..., N.
j = index of Rankings: 1, 2, 3.

Then we devised a new measure called total score of performance (TSP) that combines the three scores from each model (TSP_{m_i}) by the following equation.

$$TSP_{m_i} = \frac{1}{N}\sum_{j=1}^{3} S_{(m_i,R_j)}, \in [\frac{1}{N}, 1] \tag{4.2.2}$$

Then all the models are ranked again by their TSP score in a descending order, the models with higher TSP values are ranked higher. In doing so, we produced 4 rankings for each model.

(3) Selecting models for building ensembles.

In this stage, we need to decide what models and how many models should be selected to build an ensemble. A general consideration is to select a certain number of suitable models that will maximize the performance of an ensemble model. To avoid a tie-situation in decision making, we set the number of member models to be an odd number as there are only 7 basic models in total in this experiment, we set three different sizes for ensembles: 7, 5 and 3, to investigate whether the size of an ensemble has any influence on its performance. So we chose top 7, 5 and 3 models from each ranking to build simple ensembles respectively. In this way, we built 9 simple ensembles and they are coded with their size and the used measure, e.g. Top5TP represents an ensemble built with top 5 of TP score models. In summary, we have 4 ensembles with top 3 models from each rankings, i.e.Top3TP, Top3FN, Top3FP and Top3TSP, 4 with top 5 models from each rankings: Top5TP, Top5FN, Top5FP and Top5TSP, and one ensemble with all the 7 models, called Ensem7.

In addition, we also used another pair of measures - *Sensitivity* and $Specificity$ to select the same numbers of models to build ensembles for comparison. So, we have 4 more simple ensembles, Top5Sen, Top3Sen, Top5Spec and Top3Spec.

In total, we constructed 13 simple ensembles based on 6 different performance measures and 2 different sizes.

- *Decision Making*: Although there are several decision-making strategies to combine the outputs of the models in an ensemble, in this research we chose the simple majority voting approach for its simplicity and efficiency, which is particularly essential for our anomaly detection system to work fast enough in real-time with streaming data. A data point will be classified as an anomaly by an ensemble if more than half of its member models predict it as anomaly, otherwise, normal.

These simple ensembles were tested on the testing data and the results will be presented in the next section. Our initial experimental results show that simple ensembles are better than the individual models but we want to improve further. So we developed a method for building complex ensembles.

Complex ensemble

A complex ensemble is built by using selected simple ensembles as its member models. It can simply be viewed as an ensemble of ensembles, so donated as *EoE*. An EoE still uses the majority voting among the selected simple ensembles to determine its final result. From the 13 simple ensembles built earlier, we can construct 5 complex ensembles by selecting top 3, 5, 7, 9 and 11 simple ensembles based on their TSP score, and another one with all the 13 simple ensembles. They are donated as EoE3, EoE5, EoE7, EoE9, EoE11 and EoE13, respectively.

These complex ensembles were in the same ways as for the individual models, and simple ensembles with the same dataset.

4.3 Evaluation

4.3.1 Datasets

To demonstrate and compare the efficiency of each anomaly detection model that will be developed in this research, we chose 17 telemetry stations from Yom Basin that installed in the same year, which represents the typical geological, meteorological and hydrological features in Thailand. All the stations installed VEGA PULS WL 61/62 instruments to measure water level every 10 minutes during the years of 2013 to 2018.

Table 4.1: Summary of data from 17 telemetry stations.

Station	No. of Records	No. of Anomaly	% of Anomaly Data
DIV002	167708	24106	14.37
DIV004	215330	121	0.06
DIV006	216541	20849	9.63
NAN008	113833	1	0.00
VLGE13	191276	57	0.03
YOM001	177323	0	0.00
YOM002	179756	0	0.00
YOM003	192959	0	0.00
YOM004	163131	92	0.06
YOM005	168782	2902	1.72
YOM006	149321	4	0.00
YOM007	170142	0	0.00
YOM008	162099	2	0.00
YOM009	178241	2626	1.47
YOM010	202398	3080	1.52
YOM011	134031	638	0.48
YOM012	224404	1	0.00

Telemetry water level data is unlabelled in the first place. To evaluate the accuracy of our models, we asked some expert at the HII to look at the data to identify and label the anomalies. The details of the data are summarised in Table 4.1. We can see that not every station has anomaly data and some stations only have one anomaly. Because of HII telemetry stations installed radar water level gauge located at a fixed place above the water surface, the object under the sensors such as weeds, boats, or things that float along the water can affect sensors' reading of water level. In a dry season, the sensor may not detect the water surface properly, instead, it may detect waterbed or grass land. Figure 2.4 gives an example of such, where the water level sensor gave abnormal or incorrect readings on the water levels in Summer but produced reasonable water levels during the rainy season. For these reasons, the data from these stations are very difficult for human to label normal or abnormal as ground-truth. Although we can avoid the stations that have a low number of anomaly from this experiment, we want to keep it to test our models to see if they can misidentify normal data, which in turn can help our experts to check the labelled anomaly data for validating the ground-truth.

4.3.2 Evaluation Metrics

As this is essentially a binary classification problem, we chose the values from the confusion matrix which is a widely used technique for summarizing the performance of a classification algorithm. As our purposes are to detect the anomaly and reduce the false alarm, $Precision, Recall$ and $F1 - score$ are suitable to use as criteria.

4.4 Results

The 7 classic anomaly detection methods have been trained with the training data of the chosen stations by moving the windows over the entire duration to find their decision threshold. Then those cut-off values are applied to the testing data to evaluate their performance with Recall, Precision, F1-score and TSP score, by comparing predicted values and their corresponding ground-truth. Then these basic individual models were used to build simple ensembles and the simple ensembles were used to build complex ensembles. All the ensembles were tested in the same manners as for the individual models. Their testing results are presented below separately.

4.4.1 Individual Model Results

The results of each model when detect anomaly in telemetry water level data as shows in Table 4.2. Even though the AR model's recall score was satisfactory, averaging approximately 0.6190, its precision and F1-score were inadequate, averaging 0.0298 and 0.0532, respectively. This indicates that the AR model produced a large number of false alarms. Additionally, the AR model is ineffective at detecting anomalies in stations with few anomalies. On the other hand, the DB model had a lower average recall score than the AR model, but its precision and F1-score were better. Moreover, it could detect abnormalities in as few as seven stations. The DB model showed an impressive F1-score for identifying anomalies in the DIV006 and YOM010 datasets, exceeding 0.40. In terms of recall and F1-score, the IQR model outperformed the AR and DB models, but its accuracy was inferior to that of the DB model.

Table 4.2: The performance indexes of 7 methods on telemetry water level data. (Since some models cannot detect anomalies in particular datasets, computing those three metrics is meaningless.)

Method	AR			DB			IQR			K-Sigma		
Station	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DIV002	0.8254	0.1350	0.2321	0.0006	0.4412	0.0012	1.0000	0.9637	0.9815	0.0378	0.4745	0.0701
DIV004	0.1488	0.0001	0.0002	0.9587	0.0216	0.0423	1.0000	0.0115	0.0227	0.5041	0.0311	0.0586
DIV006	0.8801	0.0966	0.1740	0.3096	0.6755	0.4246	0.9108	0.7926	0.8476	0.0390	0.1325	0.0603
NAN008	-	-	-	-	-	-	1.0000	0.0024	0.0048	1.0000	0.0009	0.0019
VLGE13	0.9649	0.0004	0.0007	-	-	-	1.0000	0.0254	0.0495	0.0351	0.0007	0.0014
YOM001	-	-	-	-	-	-	-	-	-	-	-	-
YOM002	-	-	-	-	-	-	-	-	-	-	-	-
YOM003	-	-	-	-	-	-	-	-	-	-	-	-
YOM004	0.3804	0.0003	0.0005	-	-	-	1.0000	0.1311	0.2317	0.0326	0.0017	0.0032
YOM005	0.6323	0.0123	0.0241	0.0003	0.0345	0.0007	1.0000	0.5333	0.6956	0.1937	0.0844	0.1175
YOM006	-	-	-	1.0000	0.1053	0.1905	1.0000	0.0023	0.0047	0.25	0.0007	0.0014
YOM007	-	-	-	-	-	-	-	-	-	-	-	-
YOM008	-	-	-	-	-	-	1.0000	0.0087	0.0172	1.0000	0.0009	0.0018
YOM009	0.7403	0.0122	0.0241	0.0274	0.3495	0.0508	0.9992	0.2134	0.3517	0.0880	0.0759	0.0815
YOM010	0.5259	0.0092	0.0181	0.9040	0.2685	0.4141	0.9673	0.2295	0.3709	0.1125	0.0752	0.0901
YOM011	0.4732	0.0025	0.0049	-	-	-	1.0000	0.2247	0.3669	0.1528	0.0497	0.0750
YOM012	-	-	-	-	-	-	1.0000	0.0006	0.0012	1.0000	0.0006	0.0012
Average	0.6190	0.0298	0.0532	0.4572	0.2709	0.1606	0.9906	0.2415	0.3035	0.3420	0.0714	0.0434
Std	0.2630	0.0499	0.0867	0.4778	0.2396	0.1879	0.0256	0.3228	0.3441	0.3961	0.1286	0.0426

Method	MAS			SA			Z-Score		
Station	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DIV002	0.9992	0.9998	0.9995	-	-	-	1.0000	0.5910	0.7429
DIV004	0.9917	0.0144	0.0284	-	-	-	1.0000	0.0033	0.0065
DIV006	0.4107	0.8037	0.5436	0.0013	0.9643	0.0026	0.9114	0.2455	0.3868
NAN008	1	0.0029	0.0058	-	-	-	1.000	0.0001	0.0002
VLGE13	1.0000	0.0068	0.0135	-	-	-	1.0000	0.0020	0.0041
YOM001	-	-	-	-	-	-	-	-	-
YOM002	-	-	-	-	-	-	-	-	-
YOM003	-	-	-	-	-	-	-	-	-
YOM004	1.0000	1.0000	1.0000	-	-	-	1.0000	0.0066	0.0131
YOM005	0.5955	0.1769	0.2727	0.0003	0.3333	0.0007	1.0000	0.1082	0.1953
YOM006	1.0000	0.0177	0.0348	1.0000	0.5714	0.7273	1.0000	.0003	0.0005
YOM007	-	-	-	-	-	-	-	-	-
YOM008	1	0.0047	0.0093	-	-	-	1.0000	0.0001	0.0002
YOM009	0.3747	0.9929	0.5441	-	-	-	1.0000	0.0934	0.1709
YOM010	0.9069	0.1972	0.3240	-	-	-	0.9656	0.1917	0.3199
YOM011	0.9854	0.2722	0.4266	-	-	-	1.0000	0.0491	0.0936
YOM012	1.0000	0.0027	0.0054	-	-	-	1.0000	-	0.0001
Average	0.8665	0.3455	0.3237	0.3339	0.6230	0.2435	0.9905	0.1076	0.1488
Std	0.2379	0.4305	0.3643	0.5769	0.3186	0.4190	0.0256	0.1733	0.2222

The IQR model performed exceptionally well in detecting anomalies in stations with over 10% anomalies, such as DIV002 and DIV006, with an F1-score of 0.9815 and 0.8476, respectively. The K-Sigma model's performance was poor, with low recall (0.3420), accuracy (0.0714), and F1-score (0.0426) scores. However, it could identify abnormalities in more stations than the AR and DB models. The MAS model had a higher accuracy and F1-score than the IQR model, but its recall score was significantly lower. Despite this, the MAS model's performance was perfect, with a score of 1.00 in each performance metric, when detecting anomalies in YOM004 files. The SA model could only detect anomalies in three datasets (DIV006, YOM005, and YOM006). Its recall score was poor, but its accuracy was excellent, with an average precision of 0.6230. Lastly, Z-Score had an ideal recall score in ten datasets, but its accuracy score was very low, resulting in a high false-alarm rate.

Overall, the IQR and Z-Score models demonstrated the highest performance in terms of the recall index, with an average of over 0.99, while the SA model showed the worst performance with an average of 0.3339. The SA model provided the greatest average precision score at 0.6230, followed by MAS at 0.3455, and IQR at 0.2415. Meanwhile, MAS had the highest average F1-score at 0.3237, followed by IQR at 0.3035.

Despite the SA model's best performance in the average precision index, it is not suitable for detecting anomalies in water level data due to its poor performance on the average recall and F1 indexes, and its ability to identify only a few anomalies from three stations. While IQR and Z-score have the highest recall scores in 11 and 12 stations, respectively, their precision scores are relatively low in some stations. For instance, IQR has a recall score of 1.0000 but a precision score of just 0.0115 in DIV004. Similarly, MAS effectively identified anomalies in YOM004 without any detection errors but displayed poor precision when analyzing data in certain stations like DIV004 and VLGE013.

Even if a model has a high average score, it does not always mean it is the best at recognising all types of errors. As a result, we compute TSP from each model, which is shown in Table 4.3. The AR model has the lowest performance with an average TSP score of only 0.46. The IQR model is the best model because it has not only

the highest score from 8 stations but also the highest total (14.29) and average scores (0.84) over all the stations.

Table 4.3: TSP score of individual model (best results are in bold).

Station	AR	DB	IQR	KSigma	MAS	SA	ZScore
DIV002	0.43	0.43	**0.86**	0.43	0.76	0.43	0.76
DIV004	0.24	0.62	**0.81**	0.57	0.67	0.43	0.76
DIV006	0.52	0.52	**0.76**	0.33	0.67	0.43	0.76
NAN008	0.33	0.57	0.86	0.81	**0.90**	0.62	0.76
VLGE13	0.43	0.48	**0.90**	0.48	0.81	0.52	0.76
YOM001	0.71	0.95	0.86	0.90	0.76	**1.00**	0.81
YOM002	0.71	0.90	0.86	**1.00**	0.76	0.95	0.81
YOM003	0.71	0.95	0.86	0.90	0.76	**1.00**	0.81
YOM004	0.43	0.43	0.86	0.43	**1.00**	0.48	0.76
YOM005	0.52	0.48	**0.90**	0.48	0.52	0.52	0.76
YOM006	0.14	0.95	0.81	0.38	0.90	**1.00**	0.76
YOM007	0.71	0.95	0.86	0.90	0.81	**1.00**	0.76
YOM008	0.33	0.57	**0.90**	0.81	0.86	0.62	0.76
YOM009	0.52	0.43	0.71	0.48	0.67	0.43	**0.76**
YOM010	0.33	0.62	**0.86**	0.48	0.62	0.43	0.67
YOM011	0.43	0.48	**0.81**	0.48	0.71	0.52	0.76
YOM012	0.33	0.57	0.81	0.86	**0.90**	0.62	0.76
Total	7.86	10.90	**14.29**	10.71	13.10	11.00	13.00
Average	0.46	0.64	**0.84**	0.63	0.77	0.65	0.76
Std.	0.17	0.21	0.05	0.23	0.12	0.24	0.03

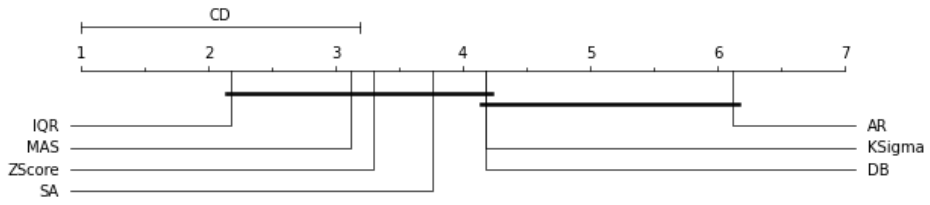


Figure 4.1: Critical difference diagram for different statistical models.

Figure 4.1 shows the comparison of the critical difference between the different statistical models. The number associated with each algorithm is the average rank of the models on each type of datasets, and solid bars represent groups of classifiers with no significantly difference. We can observe that IQR has the highest ranked than other. AR not only has the lowest ranked but also significantly difference from IQR, MAS, Z-Score, and SA.

Table 4.4: The performance indexes of 13 ensemble methods on telemetry water level data. (Since some models cannot detect anomalies in particular datasets, computing those three metrics is meaningless.)

Method	Ensem7			Top5TP			Top3TP			Top5FN			Top3FN			Top5FP			Top3FP		
Station	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DIV002	0.8602	0.9998	0.9247	0.9996	0.9938	0.9967	1.0000	0.9895	0.9947	0.9996	0.9938	0.9967	1.0000	0.9895	0.9947	0.8602	0.9999	0.9248	0.8597	0.9999	0.9245
DIV004	0.9752	0.1117	0.2005	0.9917	0.0772	0.1433	1.0000	0.0113	0.0223	0.9917	0.0772	0.1433	1.0000	0.0113	0.0223	0.9752	0.1733	0.2943	0.5041	0.1784	0.2635
DIV006	0.5645	0.9651	0.7124	0.8865	0.9621	0.9227	0.9110	0.2616	0.4065	0.8865	0.9621	0.9227	0.9110	0.2616	0.4065	0.5616	0.9723	0.7120	0.3768	0.9956	0.5466
NAN008	1.0000	0.0625	0.1176	1.0000	0.0078	0.0155	1.0000	0.0069	0.0138	1.0000	0.0078	0.0155	1.0000	0.0069	0.0138	1.0000	0.0625	0.1176	-	-	-
VLGE13	1.0000	0.5377	0.6994	1.0000	0.0718	0.1340	1.0000	0.0430	0.0825	1.0000	0.0718	0.1340	1.0000	0.0430	0.0825	0.0351	0.1000	0.0519	-	-	-
YOM001	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM002	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM003	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM004	0.4130	0.9744	0.5802	1.0000	0.7797	0.8762	1.0000	0.5935	0.7449	1.0000	0.7797	0.8762	1.0000	0.5935	0.7449	0.4130	0.9744	0.5802	0.4130	0.9744	0.5802
YOM005	0.4321	0.8101	0.5636	0.9683	0.6173	0.7540	1.0000	0.1188	0.2124	0.9683	0.6173	0.7540	1.0000	0.1188	0.2124	0.1809	0.6872	0.2864	0.0003	0.0769	0.0007
YOM006	1.0000	0.2667	0.4211	1.0000	0.1600	0.2759	1.0000	0.1081	0.1951	1.0000	0.1600	0.2759	1.0000	0.1081	0.1951	1.0000	0.2222	0.3636	1.0000	0.2667	0.4211
YOM007	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM008	1.0000	0.1111	0.2000	1.0000	0.0241	0.0471	1.0000	0.0182	0.0357	1.0000	0.0241	0.0471	1.0000	0.0182	0.0357	1.0000	0.1538	0.2667	-	-	-
YOM009	0.3907	0.9771	0.5582	0.7555	0.4890	0.5937	0.9992	0.0865	0.1592	0.7555	0.4890	0.5937	0.9992	0.0865	0.1592	0.0152	0.7018	0.0298	0.3747	0.9949	0.5444
YOM010	0.8862	0.6700	0.7631	0.9056	0.5955	0.7185	0.9737	0.3509	0.5159	0.9056	0.5955	0.7185	0.9737	0.3509	0.5159	0.8687	0.6669	0.7545	0.0879	0.2933	0.1352
YOM011	0.6130	0.9173	0.7349	0.9919	0.6595	0.7922	1.0000	0.2944	0.4549	0.9919	0.6595	0.7922	1.0000	0.2944	0.4549	0.1463	0.7438	0.2446	-	-	-
YOM012	1.0000	0.0333	0.0645	1.0000	0.0065	0.0130	1.0000	0.0063	0.0126	1.0000	0.0065	0.0130	1.0000	0.0063	0.0126	1.0000	0.0357	0.0690	-	-	-
Average	0.7796	0.5721	0.5031	0.9615	0.4188	0.4833	0.9911	0.2222	0.2962	0.9615	0.4188	0.4833	0.9911	0.2222	0.2962	0.6197	0.4995	0.3612	0.2782	0.3677	0.4270
Std	0.2548	0.3997	0.2777	0.0727	0.3751	0.3828	0.0251	0.2891	0.3109	0.0727	0.3751	0.3828	0.0251	0.2891	0.3109	0.4077	0.3799	0.2924	0.3474	0.4442	0.2910

Method	Top5Sen			Top3Sen			Top5Spec			Top3Spec			Top5TSP			Top3TSP		
Station	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DIV002	0.9996	0.9938	0.9967	1.0000	0.9895	0.9947	0.8602	0.9999	0.9248	0.8597	0.9999	0.9245	1.0000	0.9979	0.9989	1.0000	0.9895	0.9947
DIV004	0.9917	0.0772	0.1433	1.0000	0.0113	0.0223	0.9752	0.1733	0.2943	0.5041	0.1784	0.2635	0.9917	0.0610	0.1149	1.0000	0.0113	0.0223
DIV006	0.8865	0.9621	0.9227	0.9110	0.2616	0.4065	0.5616	0.9723	0.7120	0.3768	0.9956	0.5466	0.4194	0.9211	0.5764	0.9126	0.8753	0.8936
NAN008	1.0000	0.0078	0.0155	1.0000	0.0069	0.0138	1.0000	0.0625	0.1176	-	-	-	1.0000	0.0078	0.0155	1.0000	0.0033	0.0065
VLGE13	1.0000	0.0718	0.1340	1.0000	0.0430	0.0825	0.0351	0.1000	0.0519	-	-	-	1.0000	0.2651	0.4191	1.0000	0.0430	0.0825
YOM001	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM002	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM003	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM004	1.0000	0.7797	0.8762	1.0000	0.5935	0.7449	0.4130	0.9744	0.5802	0.4130	0.9744	0.5802	1.0000	0.9109	0.9534	1.0000	0.5897	0.7419
YOM005	0.9683	0.6173	0.7540	1.0000	0.1188	0.2124	0.1809	0.6872	0.2864	0.0003	0.0769	0.0007	0.6116	0.6153	0.6134	1.0000	0.6848	0.8129
YOM006	1.0000	0.1600	0.2759	1.0000	0.1081	0.1951	1.0000	0.2222	0.3636	1.0000	0.2667	0.4211	1.0000	0.1250	0.2222	1.0000	0.0136	0.0268
YOM007	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
YOM008	1.0000	0.0241	0.0471	1.0000	0.0182	0.0357	1.0000	0.1538	0.2667	-	-	-	1.0000	0.0247	0.0482	1.0000	0.0476	0.0909
YOM009	0.7555	0.4890	0.5937	0.9992	0.0865	0.1592	0.0152	0.7018	0.0298	0.3747	0.9949	0.5444	0.4619	0.8471	0.5978	0.9992	0.4838	0.6519
YOM010	0.9056	0.5955	0.7185	0.9737	0.3509	0.5159	0.8687	0.6669	0.7545	0.0879	0.2933	0.1352	0.8956	0.6494	0.7529	0.9504	0.4140	0.5768
YOM011	0.9919	0.6595	0.7922	1.0000	0.2944	0.4549	0.1463	0.7438	0.2446	-	-	-	0.9919	0.6816	0.8079	1.0000	0.3271	0.4930
YOM012	1.0000	0.0065	0.0130	1.0000	0.0063	0.0126	1.0000	0.0357	0.0690	-	-	-	1.0000	0.0065	0.0130	1.0000	0.0021	0.0043
Average	0.9615	0.4188	0.4833	0.9911	0.2222	0.2962	0.6197	0.4995	0.3612	0.2782	0.3677	0.4270	0.8748	0.4703	0.4718	0.9894	0.3450	0.4152
Std	0.0727	0.3751	0.3828	0.0251	0.2891	0.3109	0.4077	0.3799	0.2924	0.3474	0.4442	0.2910	0.2207	0.3949	0.3582	0.0268	0.3572	0.3846

4.4.2 Simple Ensemble Results

Table 4.4 displays the results of various ensemble models using different combination techniques. Ensem7 demonstrated the best performance in detecting anomalies in the DIV002 dataset, with an F1-score of 0.9247. Ensem7 also performed well in the DIV006, VLGE13, YOM010, and YOM011 datasets, with F1-scores greater than 0.70. However, identifying anomalies in the YOM012 dataset was the most challenging for Ensem7, with a precision of 0.0333 and an F1-score of 0.0645

Top5TP, Top5FN, and Top5Sen demonstrated better performance than Ensem7 on certain datasets such as DIV002, DIV06, YOM004, YOM09, and YOM011, but their average F1-score remained lower than that of Ensem7. Top3TP, Top3FN, and Top3Sen achieved a perfect recall of 1.00, an accuracy of 0.9895, and an F1-score of 0.9947 while detecting anomalies in the DIV002 dataset. However, identifying anomalies in DIV002, NAN008, VLGE13, and YOM012 proved to be challenging, as evidenced by their low accuracy and F1-scores. Top5FP and Top5Spec demonstrated the highest F1-score in DIV002, but their performance was weak in NAN008, VLGE13, YOM009, and YOM012, with F1-scores below 0.100. Although Top3FP and Top3Spec worked well on DIV002, they failed to detect any anomalies in NAN008, VLGE13, YOM008, YOM011, and YOM012.

Top5TP showed the best performance in detecting anomalies in the DIV002 dataset, with an F1-score of 0.9989. However, there were significant false alarms on several datasets, including DIV004, NAN008, YOM006, YOM08, and YOM012. Top3TSP achieved strong results on several datasets with F1-scores above 0.80 but performed poorly on others, such as the NAN008 dataset, where it had a recall of 1.000, a precision of 0.0033, and an F1-score of 0.0065.

Ensem7 had the highest average F1-score of 0.5031, followed by Top5TP, Top5FN, and Top5Sen, which had an average score of 0.4833. Top3Sen had the lowest average F1-score of 0.2962. While Ensem7 performed better than others in terms of F1-score, it generated a significant number of false positives on DIV004, NAN001, YOM006, and YOM012, as evidenced by high recall but very low precision scores.

Table 4.5: Total and average TSP scores of simple ensemble models.

Station	Ensem7	Top5TP	Top3TP	Top5FN	Top3FN	Top5FP	Top3FP	Top5Sen	Top3Sen	Top5Spec	Top3Spec	Top5TSP	Top3TSP
DIV002	0.49	0.59	**0.77**	0.59	**0.77**	0.59	0.44	0.59	**0.77**	0.59	0.44	0.67	**0.77**
DIV004	0.49	0.67	**0.77**	0.67	**0.77**	0.54	0.44	0.67	**0.77**	0.54	0.44	0.59	**0.77**
DIV006	0.54	0.67	0.69	0.67	0.69	0.54	0.44	0.67	0.69	0.54	0.44	0.28	**0.77**
NAN008	**0.95**	0.87	0.77	0.87	0.77	**0.95**	0.44	0.87	0.77	**0.95**	0.44	0.87	0.69
VLGE13	**0.90**	0.85	0.77	0.85	0.77	0.49	0.44	0.85	0.77	0.49	0.44	0.87	0.77
YOM001	0.95	0.82	0.74	0.82	0.74	0.95	**1.00**	0.82	0.74	0.95	**1.00**	0.87	0.85
YOM002	0.90	0.87	0.74	0.87	0.74	0.95	**1.00**	0.87	0.74	0.95	**1.00**	0.79	0.77
YOM003	0.90	0.82	0.74	0.82	0.74	0.95	**1.00**	0.82	0.74	0.95	**1.00**	0.87	0.85
YOM004	0.59	0.85	0.77	0.85	0.77	0.59	0.59	0.85	0.77	0.59	0.59	**0.87**	0.69
YOM005	0.49	0.62	0.74	0.62	0.74	0.49	0.44	0.62	0.74	0.49	0.44	0.51	**0.85**
YOM006	**1.00**	0.87	0.77	0.87	0.77	0.92	**1.00**	0.87	0.77	0.92	**1.00**	0.79	0.69
YOM007	**1.00**	0.85	0.74	0.85	0.74	**1.00**	**1.00**	0.85	0.74	**1.00**	**1.00**	0.87	0.77
YOM008	0.90	0.82	0.74	0.82	0.74	**1.00**	0.44	0.82	0.74	**1.00**	0.44	0.85	0.87
YOM009	0.49	0.64	0.74	0.64	0.74	0.38	0.54	0.64	0.74	0.38	0.54	0.51	**0.77**
YOM010	0.49	0.64	**0.74**	0.64	**0.74**	0.49	0.44	0.64	**0.74**	0.49	0.44	0.51	0.62
YOM011	0.49	0.64	0.74	0.64	0.74	0.49	0.44	0.64	0.74	0.49	0.44	0.67	**0.77**
YOM012	0.90	0.87	0.77	0.87	0.77	**0.95**	0.44	0.87	0.77	**0.95**	0.44	0.87	0.69
Total	12.44	**12.95**	12.77	**12.95**	12.77	12.26	10.49	**12.95**	12.77	12.26	10.49	12.28	**12.95**
Average	0.73	**0.76**	0.75	**0.76**	0.75	0.72	0.62	**0.76**	0.75	0.72	0.62	0.72	**0.76**
Std.	0.22	0.11	0.02	0.11	0.02	0.24	0.26	0.11	0.02	0.24	0.26	0.18	0.07

Table 4.5 presents the TSP score for each of the simple ensemble models, indicating that the simple ensembles generally perform better than the individual models for detecting anomalies. The Top5TP, Top5FN, Top5Sen, and Top3TSP models achieved the highest average TSP score of 0.76, while the Top3FP and Top2Spec models had the lowest average TSP score of 0.62. However, the results from the CD diagram in Figure 4.2 revealed that Top5TP, Top5FN, and Top3Sen had the highest performance rankings, with a significant difference from Top3Sen, Top3FN, Top3TP, Top3Spec, and Top3FP. Therefore, it can be concluded that the Top5TP, Top5FN, and Top3Sen models are the most effective in detecting anomalies among the simple ensemble models.

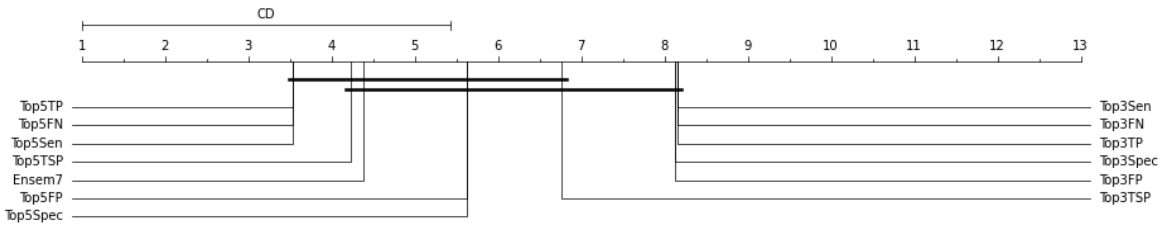


Figure 4.2: Critical difference diagram for 13 different simple ensemble models.

4.4.3 Complex Ensemble Results

Table 4.6 displays the performance results of several complex ensemble models. EoE13 achieved an F1-score greater than 0.75 on several datasets, but it struggled with detecting anomalies in the NAN008, YOM008, and YOM012 datasets. EoE11 performed similarly to EoE13, except for detecting anomalies in the YOM004 and YOM005 datasets,

where it performed poorly. On the other hand, EoE9 outperformed EoE11 by a significant margin on the YOM004 and YOM006 datasets, with its F1-score being 2-4 times better. EoE7 generated the lowest F1-score on the DIV004 dataset, but it outperformed EoE13, EoE11, and EoE7 in detecting anomalies in the YOM008 dataset. Finally, EoE5 and EoE3 struggled to detect anomalies in the YOM012 dataset, as evidenced by their high recall but low precision scores.

Table 4.6: The performance indexes of 6 complex ensemble models on telemetry water level data. (Since some models cannot detect anomalies in particular datasets, computing those three metrics is meaningless.)

Method	EoE13			EoE11			EoE9		
Station	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DIV002	0.9996	0.9938	0.9967	0.9996	0.9938	0.9967	1.0000	0.9926	0.9963
DIV004	0.9917	0.0772	0.1433	0.9917	0.0772	0.1433	0.9917	0.0693	0.1295
DIV006	0.8865	0.9621	0.9227	0.8865	0.9621	0.9227	0.8865	0.9621	0.9227
NAN008	1.0000	0.0078	0.0155	1.0000	0.0078	0.0155	1.0000	0.0078	0.0155
VLGE13	1.0000	0.0718	0.1340	1.0000	0.0718	0.1340	1.0000	0.0635	0.1195
YOM001	-	-	-	-	-	-	-	-	-
YOM002	-	-	-	-	-	-	-	-	-
YOM003	-	-	-	-	-	-	-	-	-
YOM004	1.0000	0.7731	0.8720	1.0000	0.0718	0.1340	0.4130	0.9744	0.5802
YOM005	0.9683	0.6862	0.8032	0.9683	0.6173	0.7540	0.9810	0.6185	0.7587
YOM006	1.0000	0.1600	0.2759	1.0000	0.1600	0.2759	1.0000	0.2667	0.4211
YOM007	-	-	-	-	-	-	-	-	-
YOM008	1.0000	0.0247	0.0482	1.0000	0.0247	0.0482	1.0000	0.0247	0.0482
YOM009	0.7555	0.4889	0.5937	0.7555	0.4890	0.5937	0.8275	0.4890	0.6147
YOM010	0.9001	0.6147	0.7305	0.9056	0.5955	0.7185	0.9219	0.5850	0.7158
YOM011	0.9919	0.6696	0.7995	0.9919	0.6595	0.7922	0.9984	0.5870	0.7393
YOM012	1.0000	0.0065	0.0130	1.0000	0.0065	0.0130	1.0000	0.0065	0.0130
Average	0.9578	0.3786	0.4460	0.9583	0.3119	0.3787	0.9183	0.3879	0.4232
Std	0.0753	0.3538	0.3709	0.0750	0.3317	0.3477	0.1689	0.3642	0.3388

Method	EoE7			EoE5			EoE3		
Station	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DIV002	1.0000	0.9895	0.9947	1.0000	0.9895	0.9947	1.0000	0.9895	0.9947
DIV004	1.0000	0.0113	0.0223	0.9917	0.0772	0.1433	0.9917	0.0772	0.1433
DIV006	0.8865	0.9621	0.9227	0.8865	0.9621	0.9227	0.8865	0.9621	0.9227
NAN008	1.0000	0.0078	0.0155	1.0000	0.0625	0.1176	1.0000	0.0625	0.1176
VLGE13	1.0000	0.0718	0.1340	1.0000	0.0718	0.1340	1.0000	0.5089	0.6746
YOM001	-	-	-	-	-	-	-	-	-
YOM002	-	-	-	-	-	-	-	-	-
YOM003	-	-	-	-	-	-	-	-	-
YOM004	1.0000	0.9892	0.9946	1.0000	0.7797	0.8762	1.0000	0.7797	0.8762
YOM005	1.0000	0.6163	0.7626	1.0000	0.1188	0.2124	1.0000	0.1188	0.2124
YOM006	1.0000	0.2667	0.4211	1.0000	0.2667	0.4211	1.0000	0.2667	0.4211
YOM007	-	-	-	-	-	-	-	-	-
YOM008	1.0000	0.1176	0.2105	1.0000	0.1250	0.2222	1.0000	0.1538	0.2667
YOM009	0.9992	0.4836	0.6518	0.7555	0.4890	0.5937	0.7555	0.4890	0.5937
YOM010	0.9737	0.4009	0.5680	0.9737	0.3509	0.5159	0.9737	0.3509	0.5159
YOM011	0.9984	0.5870	0.7393	0.9919	0.6595	0.7922	0.9984	0.5950	0.7456
YOM012	1.0000	0.0065	0.0130	1.0000	0.0333	0.0645	1.0000	0.0357	0.0690
Average	0.9882	0.3767	0.4546	0.9666	0.3330	0.4180	0.9672	0.3667	0.4632
Std	0.0329	0.3581	0.3664	0.0739	0.3186	0.3158	0.0741	0.3038	0.3025

Table 4.7: Total and average TSP scores of complex ensembles.

Station	EoE13	EoE11	EoE9	EoE7	EoE5	EoE3
DIV002	0.56	0.56	0.56	**0.83**	**0.83**	**0.83**
DIV004	**0.89**	**0.89**	0.67	0.72	**0.89**	**0.89**
DIV006	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
NAN008	0.89	0.89	0.89	0.89	**1.00**	**1.00**
VLGE13	0.94	0.94	0.72	0.94	0.94	**1.00**
YOM001	0.94	0.94	0.94	0.94	0.94	**1.00**
YOM002	0.83	0.83	0.83	0.94	0.94	**1.00**
YOM003	0.83	0.83	0.83	0.94	0.94	**1.00**
YOM004	0.72	**1.00**	0.44	**1.00**	0.83	0.83
YOM005	0.56	0.50	0.56	**0.83**	0.78	0.78
YOM006	0.78	0.78	**1.00**	**1.00**	**1.00**	**1.00**
YOM007	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
YOM008	0.83	0.83	0.83	0.89	0.94	**1.00**
YOM009	0.61	**0.78**	0.67	0.72	**0.78**	**0.78**
YOM010	0.44	0.50	0.56	**0.83**	0.78	0.78
YOM011	0.67	0.61	0.78	0.78	0.61	**0.83**
YOM012	0.89	0.89	0.89	0.89	0.94	**1.00**
Total	13.39	13.78	13.17	15.17	15.17	**15.72**
Average	0.79	0.81	0.77	0.89	0.89	**0.92**
Std.	0.17	0.17	0.18	0.09	0.11	0.10

The TSP scores of complex ensembles are presented in Table 4.7 revealing that EoE3 is the top-performing model with an average TSP score of 0.92. EoE5 and EoE7 are closely behind with the same average TSP score of 0.89. EoE3 outperformed other models in 14 out of 17 stations, achieving a full score in 10 of those 14 stations. This finding is supported by CD diagram in Figure 4.3, which shows that EoE3 had the highest rank, followed by EoE5 and EoE7. Additionally, EoE3 is not only the best-performing model, but it is also significantly different from EoE9 and EoE13.

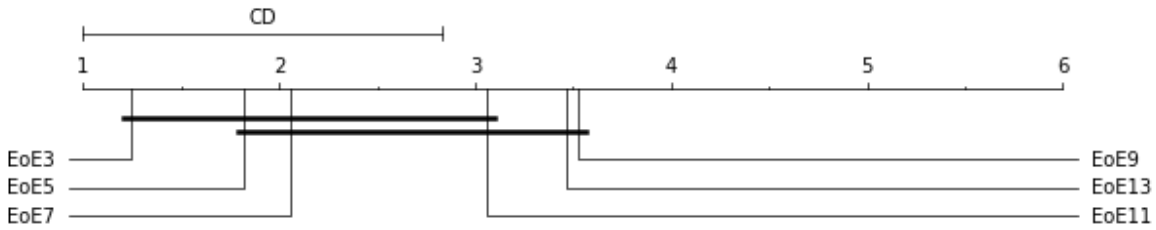


Figure 4.3: Critical difference diagram for complex ensemble models.

Finally, CD diagram was created using a set of 26 models, which included 7 statistical models, 13 simple ensemble models, and 6 complex ensemble models., as depicted in Figure 4.4. Based on the results, EoE3 is the best performing model, while AR is the least effective. The complex ensemble models performed better than the simple ensemble models and statistical models. Among the statistical models, IQR was not only the best but also outranked some of the simple ensemble models such as Top3Spec

and Top3FP. Additionally, EoE3, EoE7, and EoE5 showed significant performance differences compared to the other 7 statistical models.

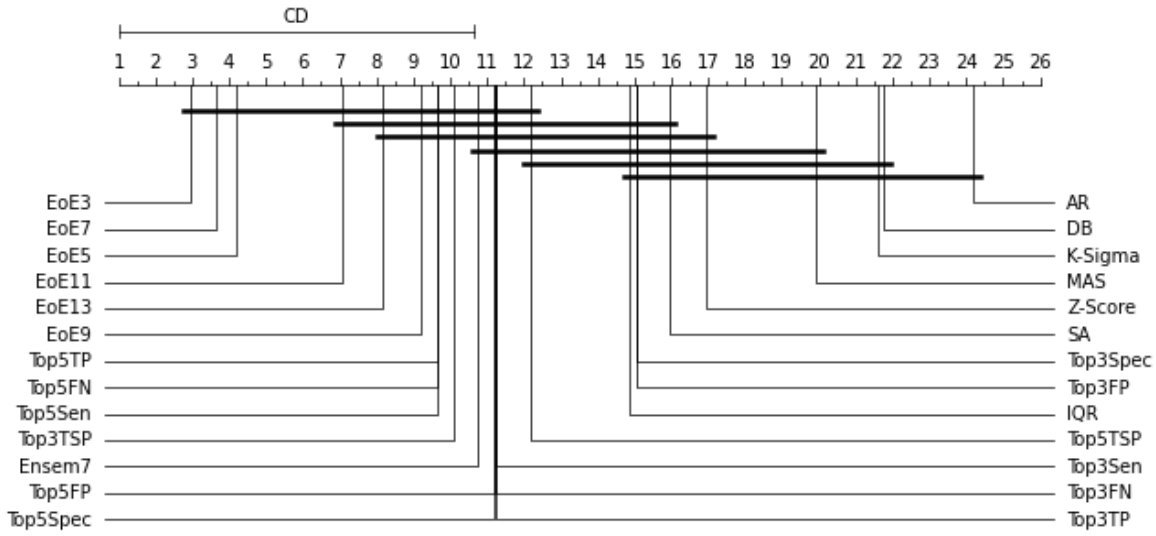


Figure 4.4: Critical difference diagram for all models.

4.5 Discussion

Regarding individual models, although several models had high recall scores, their precision was insufficient, resulting in a high rate of false alarms and a low F1-score. Further analysis revealed that data from several stations demonstrated high fluctuations or erratic patterns due to external factors such as a water shortage in a part of the river during the dry season, causing the sensor to detect the riverbed or any obstacle instead of the water. As a result, the water level measurements provided inaccurate measurements, making it difficult for humans to determine the veracity of the data, which in turn resulted in the models struggling to learn from and generalise from this data. This was reflected in the high false-negative values for stations DIV006, YOM009, and YOM010 across all models. Despite this, some models performed well in detecting normal data, which was useful for retaining relevant data in the datasets. However, at some stations, such as DIV004, NAN008, and YOM012, the normal data was greatly outnumbered by anomalies, resulting in poor accuracy and significant standard deviations.

On the other hand, the simple ensemble model outperformed individual models. It

is evident that the performance of an ensemble model is not based on the number of members but on the selection strategies used to construct the ensemble. For example, the seven-member ensemble (Ensem7) performed worse than the three-member ensemble (Top3TSP), while the five-member ensemble (Top5TP, Top5FN, and Top5Sen) performed better than the three-member ensemble (Top3TSP).

Moreover, the results showed that complex ensemble models outperformed both individual and simple ensemble models. Furthermore, the selection procedures used were found to be more critical than the total number of models used.

4.6 Summary

In this research, we applied statistical model for detecting anomaly in water level data from telemetry systems. We produced a modified a sliding window algorithm and devised a total scoring function (TSP) by combining 3 measures - TP, FP and FN, to assess the overall performance of a model.

The *classic models, simple ensembles* and *complex ensembles* were tested on the data from 17 stations, the results show that the classic model IQR is the best individual model at detecting anomalies but poor for classifying normal data. In general simple ensembles are more accurate and consistent than individual models. The best simple ensemble (Top5TP, Top5FN, and Top5Sen) outperformed the best individual model IQR by achieving the greater accuracy on detecting anomaly data and more accurate results for normal data than IQR. Further improvements were produced by our complex ensembles. It is clear that the complex ensemble EoE3, with only three member models, beats both the best individual model IQR and the best simple ensemble with clear margins in detecting anomalies and also normal data.

Even though statistical methods have been effective in many domains, they may not be sufficient for anomaly detection in telemetry water level data. Therefore, it is important to explore alternative solutions. One promising approach is to use reinforcement learning models, which have been shown to effectively learn from complex data sets and identify patterns that are difficult to detect using traditional statistical methods.

With the use of reinforcement learning models, we can improve the accuracy and reliability of anomaly detection in telemetry water level data. In the next chapter, we will develop and apply reinforcement learning models for detecting anomalies in the telemetry water level data.

Chapter 5

Anomaly Detection with Reinforcement Learning and Their Ensemble

Contributing Publications

- Khampuengson, T. and Wang, W. (2022). Deep reinforcement learning ensemble for detecting anomaly in telemetry water level data. *Water*, 14(16):2492.

5.1 Introduction

In Chapter 4, we first studied seven statistics-based models for detecting the anomalies. Although an individual model can be used to identify anomalies, it still produces too many false alarms since the water level will dramatically rise before the flood, and therefore the water level in such a scenario is notably different from the other data points. As a result, the majority of statistical models will identify such points as anomalous. We also created two ensemble methods to improve the performance of individual models. The first ensemble method is simple as it just combines some selected models with majority voting as its decision making function. However, the test results showed that the simple ensemble models did not work well enough, even though they were usually better than most of the basic individual models. So, we then developed a complex ensemble method. It basically builds an ensemble of some simple ensembles selected from the candidates with some criteria. The findings indicate that a complex ensemble can improve the performance of individual models in recognising

both abnormal and normal data.

Reinforcement Learning (RL) is an algorithm that imitates the human learning process. It is based on the self-learning process in which an agent learns by interacting with the environment without any assumptions or rules. With the advantage of being able to learn on their own, it can identify unknown anomalies (Pang et al., 2020), which gives it an edge over other models. RL has been used in some studies to detect anomalies in data (Huang et al., 2018; Hsu and Matsuoka, 2020; Lin et al., 2020). However, we discovered that none of the RL methods have been applied to identify anomalies in telemetry water level data. We wonder whether RL is applicable for identifying abnormalities in telemetry water level data. To answer this question, we conducted an intensive investigation by evaluating the accuracy of RL models. Furthermore, even if the final RL models perform well on training data, there is no guarantee that they will also perform well on testing data. Then we proposed a strategy to build some ensembles by selecting some suitable RL models.

5.2 Methods

This section describes firstly how reinforcement learning is constructed for detecting anomalies in water level telemetry data; and then how an ensemble can be built effectively by selecting suitable individual models to improve the accuracy of anomaly detection.

5.2.1 Reinforcement Learning

RL is a branch of machine learning and it is one of the most active areas of research in artificial intelligence (AI), which is growing rapidly with a wide variety of algorithms. It is goal-oriented learning. The learner, or agent, learns from the result, or rewards, of its actions without being taught what actions to take. The way in which the agent decides which action to perform depends on the policy, which can be in the form of a lookup table or a complex search process. So, a policy function defines the agent's behaviour in an environment.

Most techniques that are used to find the optimal policy for resolving the RL problem are based on the Markov decision process (MDP), whereby the probability of next state s' depends only on the current state s and action a. It is represented by five important variables (Atienza, 2018):

- A finite set of states (S), which may be discrete or continuous.
- A finite set of actions (A). The agent takes an action a from the action set A, $a \in A$.
- A transition probability ($T(s, a, s')$), which is the probability to get from state s to another state s' with action a.
- A reward probability ($R(s, a, s') \in \mathbb{R}$), which is the reward after going from state s to another state s' with action a.
- A discount factor(γ), which focuses on controls the important immediate and future rewards and lies within o to 1, $\gamma \in [0, 1]$.

The goal of learning is to maximise the expected cumulative reward in each episode. The agent should try to maximise the reward from any state s. The total reward R at state s as the sum of current rewards and the total discounted reward at the next state s', which can be represented as follow:

$$R(s) = R(s, a, s') + \gamma R(s')$$

The algorithm that has been widely used in RL is Q-learning. It tries to maximize the values from Q-function, as shown in Equation (5.2.1), which can be approximated using the Bellman equation, which represents how good it is for an agent to perform a particular action in a state s.

$$NewQ(s, a) = Q(s, a) + \alpha(r + \gamma \max Q'(s', a') - Q(s, a)) \tag{5.2.1}$$

where α is the learning rate, and $\max Q'(s', a')$ is the highest Q value between possible actions from the new state s'.

Deep Q-Learning Network (DQN)

Q-learning has a limitation: it does not perform well with many states and actions. Furthermore, going through all the actions in each state would be time-consuming. Therefore, DQN (Mnih et al., 2015) has been developed to solve those issues by using a neural network (NN). The Q-value is approximated by an NN with weights w, instead of finding the optimal Q-value through all possible state-action pairs, and errors are minimized through gradient descent.

An agent usually does not know what action is best at the beginning of training. It may select the greatest action that is the best based on history (exploitation) or may explore new possibilities that may be better or worse (exploration). However, when should an agent "exploit" rather than "explore"? This remains a challenge since if the chosen action results in a faulty selection, an agent may get stuck in incorrect learning for a time. The epsilon-greedy algorithm is a simple way to balance exploration and exploitation. It does this by randomly choosing between exploration and exploitation and using the hyperparameter ϵ to switch between random action and Q-values, as shown in Equation (5.2.2). The normal procedure is to begin with $\epsilon = 1.0$ and gradually lower it to a small value, such as 0.01.

$$a = \begin{cases} \text{select a random action } a & \text{with probability } \epsilon \\ \\ argmax_a Q(s,a) & \text{otherwise} \end{cases} \tag{5.2.2}$$

Moreover, we make a transition from one state s to the next state s' by performing some action a and receive a reward r as $T(s,a,s')$. So, neural networks may overfit with correlated experience from those transitions. So, we saved the transition information in a buffer called *replay memory* and trained the DQN with a random transition in replay memory instead of training with last transitions. It will reduce the correlated experience of learning each time, and then it will reduce the overfitting of the model.

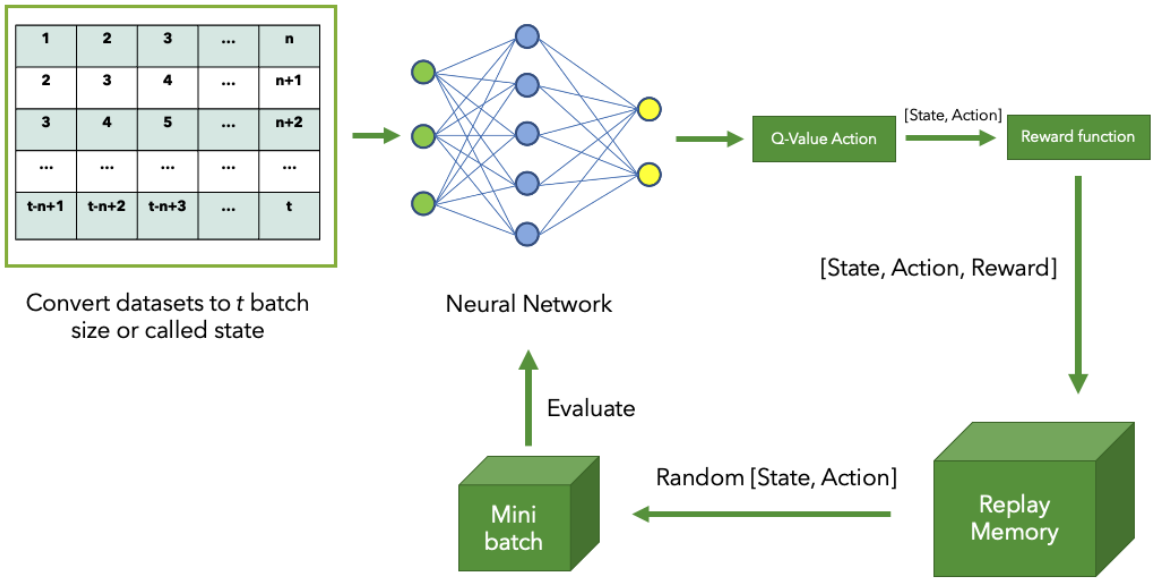


Figure 5.1: Overall process of DRL.

Deep Reinforcement Learning Model (DRL)

The action of the DRL agent is to determine whether or not a data point is an abnormality. We assigned a value of 1 to the anomaly class and a value of 0 to the normal class. DQN was chosen as our reinforcement learning strategy. When state s is received, an MLP is used as the RL agent's brain to generate Q-value, which is then followed by the Q function. The epsilon decay approach is used for exploration and exploitation. In order to explore the entire environment space, we use the greedy factor ϵ to determine whether our DRL agent should follow the Q function or randomly select an action. The overall process of DRL is depicted in Figure 5.1.

For each iteration, DQN receives the set of states S and predicts the label for training the DRL model. The transition is stored in replay memory. In each epoch, a mini batch of replay memory is sampled and used to train the model for loss minimization. Moreover, whether the model will learn well or not depends on the rewards function. The good reward function has an effect on the model's performance. If we offer a high reward for correctly identifying normal data in datasets, DRL may identify all data as normal in order to get the highest score. If, on the other hand, we give a high reward for finding outliers, DRL might label all data as outliers to get the best score.

Since our datasets are imbalanced, we will give the reward of the minority class higher than the majority class and give the penalty when our model misclassifies (Lin et al.,

2020). This will impact on the results in Q-values, then the model will select the best action to maximize the rewards. The reward function is defined below

$$rewards = \begin{cases} A & \text{predicted anomaly correct} \\ B & \text{predicted wrong} \\ C & \text{predicted normal correct} \end{cases} \tag{5.2.3}$$

A general issue in training neural networks is to determine how long they should be trained. Too few epochs may result in the model learning insufficiently, whereas too many epochs may result in the model overfitting. So, the performance of the model must be monitored during training by evaluating it on a validation data set at the end of each epoch and updating the model if the performance of the model on a validation is better than at the previous epoch. In our experiments, we selected 5 criteria as the conditions for generating the models: four performance metrics and the maximum number of epochs. The four measures are F1-score, the reward of each epoch, accuracy, and validation loss values. In the end, we will have five models: the finished training model (DRL), the models with the highest F1-score (DRL_{F1}), the models with the highest rewards (DRL_{Rwd}), the model with the highest accuracy (DRL_{Acc}), and the model with the lowest validation loss values (DRL_{Valid}).

Ensemble Methods

In general, the capacity of an individual model is limited and may have only learned some parts of the problem, and hence may make mistakes in the areas where it has not learned sufficiently. Therefore, it can be useful to combine some individual models to form an ensemble to allow them to work collectively to compensate for each other's weaknesses. Many studies (Wang, 2008; Wan et al., 2021; Casciaro et al., 2021; Marathe et al., 2021) have shown that if an ensemble is built with diverse models and appropriate decision-making functions, it can improve the accuracy of classification and also reliability. In our research, we created multiple ensembles by selecting suitable DRL models that had been generated from the previous experiments. We investigated two combining methods to aggregate the outputs from the member models of an ensemble:

simple majority voting and weighted voting algorithms.

5.3 Evaluation

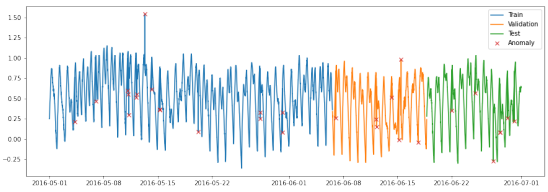
5.3.1 Datasets

Since the DRL algorithm takes a lot of time for training on the computing facilities that we had, we were limited to consider some relatively small datasets. After data preprocessing, the 8 stations from the HII telemetry water level station were chosen for use in this experiment, including CPY011, CPY012, CPY013, CPY014, CPY015, CPY016, CPY017, and YOM009. We chose the datasets from May and June for CPY011, CPY012, CPY013, CPY015, CPY016, and CPY017 in 2016 and similar months in 2015 for CPY014 and YOM009 because they have a low percentage of missing data. Figure 5.2 shows the water levels of these eight stations. It is visually clear that station YOM009 has very different behaviour from the others because it is located in a different region.

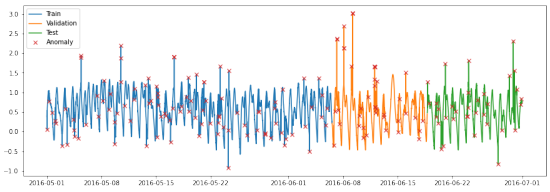
All the data are normalised and divided into 3 subsets, with the first 60% of a time series for training, the next 20% for validating , and the last 20% for testing, respectively. Table 5.1 shows the demographics of one partition of the data from each station. As can be seen, in general, the rates of anomalies are quite low for most stations, but the variances are considerably large. For example, they varied from 0.14% to 7.22% in the training data.

Table 5.1: Demographic summary of the water level data of 8 stations used in this research.

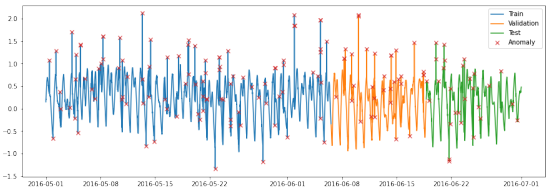
Code	Training		Validating		Testing		Total	
	Rec.	Anomaly	Rec.	Anomaly(%)	Rec.	Anomaly(%)	Rec.	Anomaly(%)
CPY011	5142	16(0.31%)	1713	7(0.41%)	1714	7(0.41%)	8569	30(0.37%)
CPY012	5142	101(1.96%)	1713	41(2.39%)	1714	34(1.98%)	8569	176(2.05%)
CPY013	5142	97(1.89%)	1713	33(1.93%)	1714	28(1.63%)	8569	158(1.84%)
CPY014	5142	49(0.95%)	1713	7(0.41%)	1714	4(0.23%)	8569	60(0.70%)
CPY015	5142	7(0.14%)	1713	15(0.88%)	1714	34(1.98%)	8569	56(0.65%)
CPY016	5142	367(7.14%)	1713	220(12.84%)	1714	107(6.24%)	8569	694(8.10%)
CPY017	5142	42(0.82%)	1713	2(0.12%)	1714	3(0.18%)	8569	47(0.55%)
YOM009	5142	417(7.22%)	1713	173(10.97%)	1714	81(3.79%)	8569	624(7.28%)
Avg.	5142	137(2.66%)	1713	62(3.63%)	1714	37(2.17%)	8569	231(2.69%)



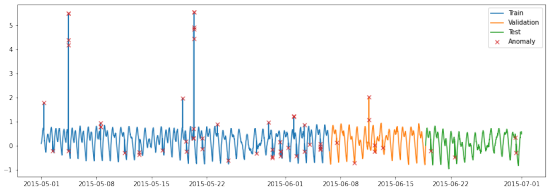
(a) CPY011



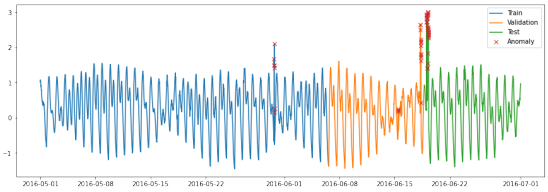
(b) CPY012



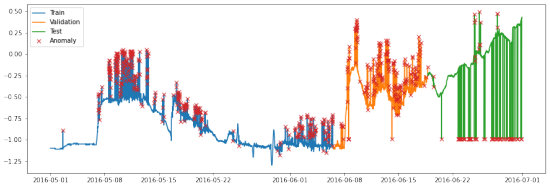
(c) CPY013



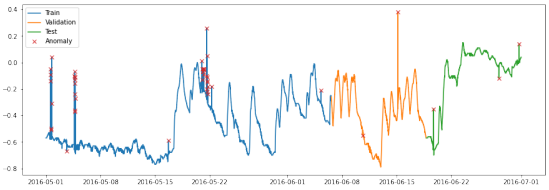
(d) CPY014



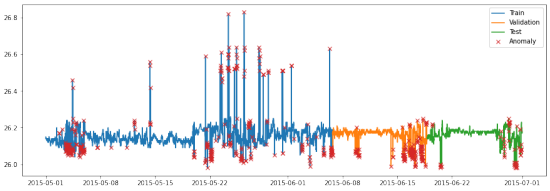
(e) CPY015



(f) CPY016



(g) CPY017



(h) YOM009

Figure 5.2: Water level data from eight stations: CPY011, CPY012, CPY013, CPY014, CPY015, CPY016, CPY017, and YOM009 (**a**-**h**). The different colours show the partitions of the data for training (blue), validation (orange) and testing (green). The anomalies are indicated by red crosses, x.

5.3.2 Evaluation metrics and Comparison methods

Since our main goal is to divide the telemetry water level data into "normal" and "abnormal" groups, we decided to use commonly used performance metrics, such as *Recall, Precision*, and *F1-score*, to measure how well our models worked. These measures allowed us to evaluate the accuracy of the models in detecting anomalies while minimising false positives and negatives. To enable statistical comparisons between the models, we utilised a CD diagram, which is a powerful tool for identifying significant differences between different models' performances. The CD diagram allowed us to determine which models performed significantly better than others and to make informed decisions about which models to select for the anomaly detection task.

5.3.3 Parameter setting

Four sets of experiments

We designed four sets of experiments to test DRL models and ensemble models. (1) to train various DRL models and test them with the different data sampled from the same water level monitoring stations; (2) to train various DRL models with the data from a station and then test them with the data from other stations; (3) to build several ensembles by selecting different numbers of the DRL models and test them with the testing data from the same stations; and (4) to test the ensembles with the data from different stations. The purpose of doing these cross-station testing is to check and evaluate the generalisation ability of the DLR models and the ensembles.

Parameter Setting

For the DRL model, a multilayer perceptron network was used in the Q-network with the following parameters: the number of input nodes in the input layer was 36, one hidden layer with 18 nodes, and 2 nodes in the output layer. Moreover, epsilon-greedy policy (ϵ) was used for exploration from 0.1 to 0.0001. The size of replay memory is 50,000, discount factor of intermediate rewards γ was 0.99. The Adam algorithm was used to optimise the parameters of Q-Network and the learning rate was 0.001. The batch size was 256, training with 100, 500, 1000, 5000, and 10,000 episodes.

The episode was over when the number of incorrectly identified anomalies was greater than the number of certain anomalies in the training set or had been trained on all the samples in the training set. We set the reward function parameters for A, B, and C to be 0.9, -0.1, and 0.1, respectively. Furthermore, the window size of 6 was chosen to save time during the training process.

For comparison, MLP and LSTM were used with the identical structures as we used in DRL. They were trained using 100 epochs with early stopping to avoid overfitting. For each setting, the experiments were repeated 10 times with variations, and then the means and standard deviations of the results are reported in the next section.

5.4 Results

5.4.1 Accuracies of DRL models

For each station, various DRL models were generated over a range of epochs from 100 to 10,000, with the intention of investigating how well our proposed DRL method learns at the different points of training. The results are shown in Table 5.2.

Using the CPY011 dataset, we observed that DRL and DRL_{Rwd} with 1000 training iterations not only earned the highest F1-score of 0.8333, 0.7143 recall, and 1.0000 precision but also provided the highest average F1-score of 0.7433. However, after 1000 epochs of training, the performance of all models, with the exception of DRL_{Valid} decreased and then rose when 10,000 epochs were used.

The top models to identify anomalies on the CPY012 dataset are DRL_{Valid}, with a maximum F1-score of 0.7826 after 10,000 training epochs. However, DRL_{Acc} obtained the greatest average F1-score with 0.7234. Meanwhile, 10,000 training epochs with DRL_{F1} and DRL_{Acc} delivered the highest F1-score for identifying anomalies in CPY013 data, at 0.8000 F1-score. Furthermore, DRL_{F1} provided the highest average F1-score of 0.6963.

Table 5.2: The performance of DRL when increasing the learning epochs (the best F1-score of each row shown in bold).

Station	Epochs	*DRL*			*DRL*$_{F1}$			*DRL*$_{Rwd}$			*DRL*$_{Acc}$			*DRL*$_{Valid}$		
		Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1
CPY011	100	0.8571	0.5455	0.6667	0.8571	0.7500	**0.8000**	0.8571	0.5455	0.6667	0.8571	0.7500	**0.8000**	0.7143	0.6250	0.6667
	500	0.8571	0.7500	**0.8000**	0.8572	0.5455	0.6667	0.8571	0.7500	**0.8000**	0.8571	0.5455	0.6667	0.8571	0.6667	0.7500
	1000	0.7143	1.0000	**0.8333**	0.8571	0.6667	0.7500	0.7143	1.0000	**0.8333**	0.8571	0.6667	0.7500	0.4286	0.2727	0.3333
	5000	0.7143	0.6250	0.6667	0.8571	0.5000	0.6316	0.7143	0.6250	0.6667	0.8571	0.5000	0.6316	0.7143	0.7143	**0.7143**
	10,000	0.8571	0.6667	**0.7500**	1.0000	0.5833	0.7368	0.8571	0.6667	**0.7500**	1.0000	0.5833	0.7368	0.8571	0.4000	0.5455
	Avg	0.8000	0.7174	**0.7433**	0.8857	0.6091	0.7170	0.8000	0.7174	**0.7433**	0.8857	0.6091	0.7170	0.7143	0.5357	0.6020
	Std	0.0782	0.1743	0.0760	0.0639	0.0997	0.0674	0.0782	0.1743	0.0760	0.0639	0.0997	0.0674	0.1749	0.1901	0.1689
CPY012	100	0.7059	0.4000	0.5106	0.7647	0.7027	**0.7324**	0.6764	0.3898	0.4946	0.7059	0.6857	0.6957	0.7647	0.7027	**0.7324**
	500	0.7647	0.6341	0.6933	0.7941	0.7297	**0.7606**	0.7353	0.6250	0.6757	0.7941	0.7297	**0.7606**	0.7941	0.7297	**0.7606**
	1000	0.6765	0.6571	0.6667	0.7647	0.6667	**0.7123**	0.6176	0.6000	0.6087	0.7647	0.6667	**0.7123**	0.7647	0.4062	0.5306
	5000	0.7059	0.7059	0.7059	0.6176	0.6176	0.6176	0.7059	0.7273	**0.7164**	0.6765	0.6970	0.6866	0.7059	0.7273	**0.7164**
	10,000	0.6471	0.7586	0.6984	0.7059	0.8000	0.7500	0.7059	0.7742	0.7385	0.7059	0.8276	0.7619	0.7941	0.7714	**0.7826**
	Avg	0.7000	0.6311	0.6550	0.7294	0.7033	0.7146	0.6882	0.6233	0.6468	0.7294	0.7213	**0.7234**	0.7647	0.6675	0.7045
	Std	0.0436	0.1378	0.0821	0.0702	0.0684	0.0572	0.0446	0.1489	0.0984	0.0483	0.0637	0.0357	0.0360	0.1481	0.1005
CPY013	100	0.8710	0.3506	0.5000	0.8710	0.6136	**0.7200**	0.9032	0.3836	0.5385	0.8710	0.6136	**0.7200**	0.6774	0.1214	0.2059
	500	0.6774	0.3684	0.4773	0.8065	0.5000	0.6173	0.7419	0.3966	0.5169	0.8065	0.5000	0.6173	0.8387	0.4906	**0.6190**
	1000	0.8065	0.5952	0.6849	0.8065	0.5682	0.6667	0.7097	0.6111	0.6567	0.8065	0.5682	0.6667	0.9677	0.5556	**0.7059**
	5000	0.7742	0.5714	0.6575	0.6774	0.6774	**0.6774**	0.8065	0.5682	0.6667	0.6774	0.6364	0.6562	0.6774	0.5250	0.5915
	10,000	0.7097	0.6667	0.6875	0.8387	0.7647	**0.8000**	0.7742	0.6857	0.7273	0.8387	0.7647	**0.8000**	0.7419	0.6571	0.6970
	Avg	0.7678	0.5105	0.6014	0.8000	0.6248	**0.6963**	0.7871	0.5290	0.6212	0.8000	0.6166	0.6920	0.7806	0.4699	0.5639
	Std	0.0770	0.1423	0.1039	0.0736	0.1015	0.0685	0.0743	0.1337	0.0899	0.0736	0.0978	0.0706	0.1237	0.2045	0.2061
CPY014	100	0.7500	0.7500	**0.7500**	0.7500	0.7500	**0.7500**	0.7500	0.7500	**0.7500**	0.7500	0.7500	**0.7500**	0.7500	0.7500	**0.7500**
	500	0.7500	0.3750	0.5000	0.7500	0.7500	0.7500	0.7500	1.0000	**0.8571**	0.7500	0.7500	0.7500	0.7500	1.0000	**0.8571**
	1000	0.7500	0.3750	0.5000	0.7500	0.5000	**0.6000**	0.7500	0.3750	0.5000	0.7500	0.5000	**0.6000**	0.7500	0.5000	**0.6000**
	5000	0.7500	0.3333	0.4615	0.7500	0.5000	**0.6000**	0.7500	0.3333	0.4615	0.7500	0.5000	**0.6000**	0.7500	0.4286	0.5455
	10,000	0.2500	0.1667	0.2000	0.7500	0.6000	**0.6667**	0.2500	0.1667	0.2000	0.7500	0.6000	**0.6667**	0.7500	0.4286	0.5455
	Avg	0.6500	0.4000	0.4823	0.7500	0.6200	**0.6733**	0.6500	0.5250	0.5537	0.7500	0.6200	**0.6733**	0.7500	0.6214	0.6596
	Std	0.2236	0.2137	0.1952	0.0000	0.1255	0.0751	0.2236	0.3405	0.2584	0.0000	0.1255	0.0751	0.0000	0.2495	0.1385
CPY015	100	0.2353	0.4706	0.3137	0.3235	0.5238	0.4000	0.2353	0.4706	0.3137	0.3235	0.5238	**0.4000**	0.1765	0.3529	0.2353
	500	0.2059	0.4667	0.2857	0.3235	0.5000	**0.3929**	0.2059	0.4667	0.2857	0.3235	0.5000	**0.3929**	0.1471	0.3846	0.2128
	1000	0.3824	0.4483	0.4127	0.3824	0.5000	**0.4333**	0.3824	0.4483	0.4127	0.3824	0.5000	**0.4333**	0.4118	0.4516	0.4308
	5000	0.2353	0.4706	0.3137	0.3235	0.5238	**0.4000**	0.2353	0.4706	0.3137	0.3235	0.5238	**0.4000**	0.2353	0.4706	0.3137
	10,000	0.3824	0.4483	0.4127	0.3824	0.5200	**0.4407**	0.3824	0.4483	0.4127	0.3824	0.5200	**0.4407**	0.3824	0.4483	0.4127
	Avg	0.2883	0.4609	0.3477	0.3471	0.5135	**0.4134**	0.2883	0.4609	0.3477	0.3471	0.5135	**0.4134**	0.2706	0.4216	0.3211
	Std	0.0868	0.0116	0.0604	0.0323	0.0124	0.0219	0.0868	0.0116	0.0604	0.0323	0.0124	0.0219	0.1202	0.0503	0.0995
CPY016	100	0.6636	0.3100	0.4226	0.5981	0.5203	**0.5565**	0.6916	0.2960	0.4146	0.5981	0.5203	**0.5565**	0.6168	0.4889	0.5455
	500	0.6636	0.2763	0.3901	0.6355	0.4048	0.4945	0.6449	0.2727	0.3833	0.5981	0.5161	**0.5541**	0.5047	0.5094	0.5070
	1000	0.6355	0.2547	0.3636	0.5981	0.5333	0.5639	0.6449	0.2644	0.3750	0.6168	0.5641	**0.5893**	0.6168	0.4342	0.5097
	5000	0.5888	0.2727	0.3728	0.5888	0.6238	**0.6058**	0.3084	0.2089	0.2491	0.5421	0.6105	0.5743	0.5234	0.2902	0.3733
	10,000	0.5794	0.2366	0.3360	0.6168	0.4177	0.4981	0.6355	0.2208	0.3277	0.6262	0.5447	**0.5826**	0.6168	0.4177	0.4981
	Avg	0.6262	0.2701	0.3770	0.6075	0.5000	0.5438	0.5851	0.2526	0.3499	0.5963	0.5511	**0.5714**	0.5757	0.4281	0.4867
	Std	0.0402	0.0274	0.0321	0.0187	0.0904	0.0472	0.1562	0.0366	0.0644	0.0326	0.0384	0.0156	0.0567	0.0858	0.0659
CPY017	100	1.0000	0.7500	**0.8571**	1.0000	0.5000	0.6667	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**
	500	1.0000	0.7500	**0.8571**	1.0000	0.5000	0.6667	1.0000	0.7500	**0.8571**	1.0000	0.3750	0.5455	0.6667	0.5000	0.5714
	1000	1.0000	0.2143	0.3529	1.0000	0.7500	**0.8571**	1.0000	0.2000	0.3333	1.0000	0.7500	**0.8571**	0.0000	0.0000	-
	5000	1.0000	0.5000	**0.6667**	1.0000	0.4286	0.6000	1.0000	0.5000	**0.6667**	1.0000	0.4286	0.6000	1.0000	0.3333	0.5000
	10,000	0.6667	0.6667	0.6667	0.6667	0.2857	0.4000	0.6667	0.6667	0.6667	1.0000	0.6000	**0.7500**	0.6667	0.6667	0.6667
	Avg	0.9333	0.5762	0.6801	0.9333	0.4929	0.6381	0.9333	0.5733	0.6762	1.0000	0.5807	**0.7219**	0.6667	0.4500	0.6488
	Std	0.1491	0.2266	0.2062	0.1491	0.1683	0.1641	0.1491	0.2323	0.2140	0.0000	0.1755	0.1443	0.4082	0.2982	0.1547
YOM009	100	0.6308	0.3178	0.4227	0.5692	0.3033	0.3957	0.6462	0.3182	**0.4264**	0.5692	0.3033	0.3957	0.5692	0.3394	0.4253
	500	0.5846	0.2734	0.3725	0.6769	0.3121	0.4272	0.6923	0.3020	0.4206	0.4769	0.4769	**0.4769**	0.4769	0.4769	**0.4769**
	1000	0.5385	0.2966	0.3825	0.6769	0.2973	0.4131	0.5538	0.3103	0.3978	0.4615	0.3947	**0.4255**	0.5846	0.3016	0.3979
	5000	0.5538	0.1818	0.2738	0.6615	0.3644	**0.4699**	0.6154	0.2581	0.3636	0.4923	0.4103	0.4476	0.5077	0.4177	0.4583
	10,000	0.4769	0.2627	0.3388	0.5692	0.2741	0.3700	0.4769	0.2605	0.3370	0.5385	0.2917	0.3784	0.4308	0.4308	**0.4308**
	Avg	0.5569	0.2665	0.3581	0.6307	0.3102	0.4152	0.5969	0.2898	0.3891	0.5077	0.3754	0.4248	0.5138	0.3933	**0.4378**
	Std	0.0570	0.0519	0.0558	0.0565	0.0334	0.0373	0.0839	0.0285	0.0382	0.0449	0.0776	0.0395	s0.0640	0.0712	0.0306

With just 500 epochs of training on CPY014 data, DRL_{Rwd} and DRL_{Valid} delivered the best F1-score of 0.8571. However, the maximum average F1-score achieved by DRL_{F1} and DRL_{Acc} was just 0.6733. When looking at the results on CPY015 data, the best models are DRL_{F1} and DRL_{Acc}. This is shown by the fact that their F1-score were the highest in many training epochs.

DRL_{Acc} was the best model for detecting anomalies in CPY016 data since it not only had the greatest F1-score in almost every training epoch but also had the highest average F1-score of 0.5714. Meanwhile, every model scored the best F1-score of 0.8571,

100 percent recall, and 0.7500 accuracy when trained with 100 epochs on CPY017, with the exception of the DRL_{F1} model, which achieved just 0.6667 F1-score. While the best models for detecting anomalies on YOM009 are DRL_{Acc} and DRL_{Valid}, which both have the same F1-score of 0.4769, the worst models are DRL while training with 5000 iterations at a 0.2728 F1-score, 0.5538 recall, and 0.1818 precision.

Figure 5.3 shows the comparison of the critical differences between the different DRL models. The number associated with each algorithm is the average rank of the DRL models on each type of dataset, and solid bars represent groups of classifiers with no significant difference. There is no statistically significant difference across the models, with DRL_{Acc} ranking first, followed by DRL_{F1}, DRL, DRL_{Rwd}, and DRL_{Valid} ranking last.

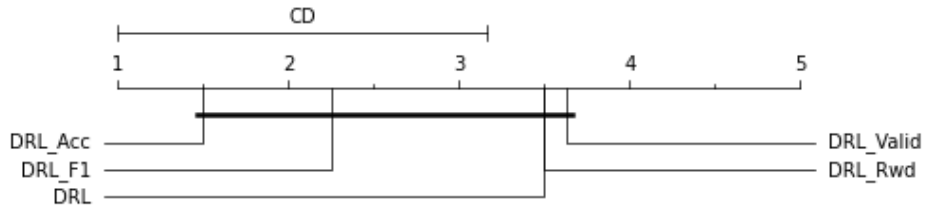
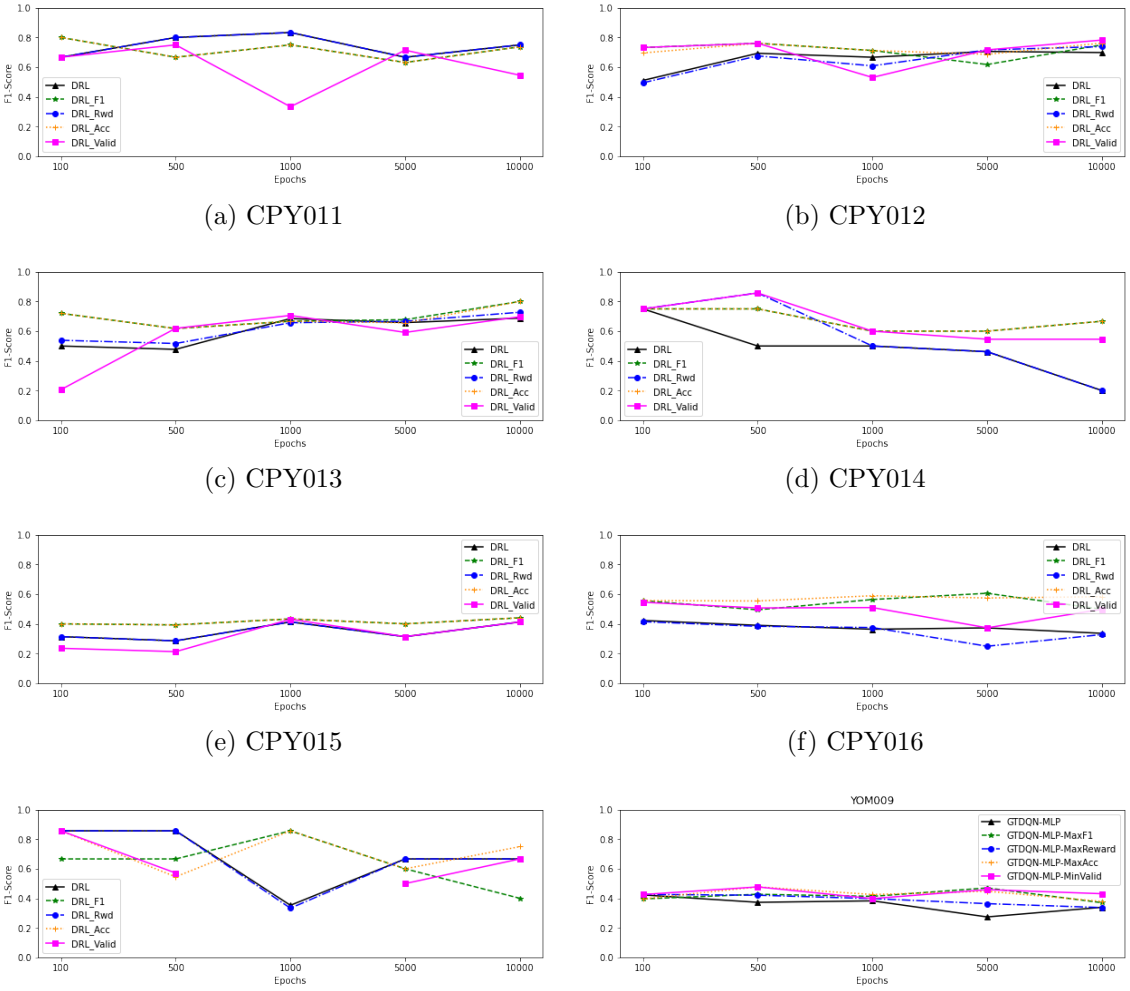


Figure 5.3: A critical difference diagram for 5 different DRL models on different datasets of telemetry water level data.

Figure 5.4 also shows a line graph of the F1-score as the number of epochs of training from each model increases. We can observe that as the number of epochs is increased, the performance of all deep reinforcement learning models using data from CPY012, CPY013, and CPY015 tends to improve. When training with CPY014 data, on the other hand, the F1-score of each model tends to stay the same or go down as the number of epochs goes up. In the case of trained models with CPY016 data, the F1-score of each model tends to stabilise and slightly decrease, with the exception of DRL_{Valid}, which tends to grow after 5000 epochs of training. When we looked at the models that were trained with the CPY017 dataset, the F1-score of DRL_{F1} went up after training with 1000 epochs and then went down. Other models, however, went up when training with more epochs, even though the performance of some models went down after 1000 epochs, while the F1-score of models that have been trained with CPY011 and YOM009 remained stable when training with more epochs.



(a) CPY011 (b) CPY012

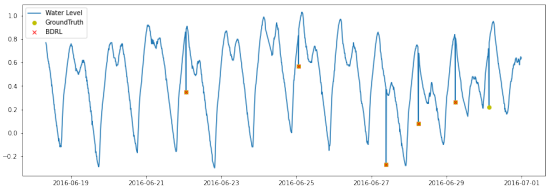
(c) CPY013 (d) CPY014

(e) CPY015 (f) CPY016

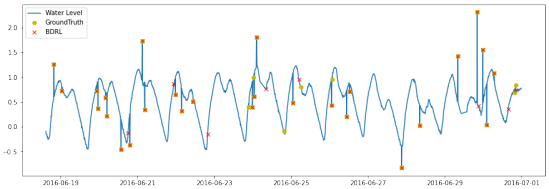
(g) CPY017 (h) YOM009

Figure 5.4: F1-score when increasing the learning epochs at each station.

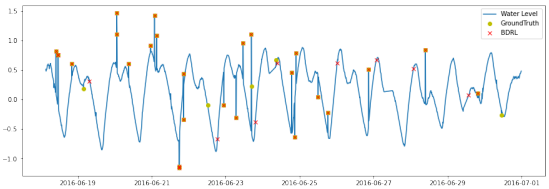
Figure 5.5 shows the findings of the best DRL model for each station. We can observe that the DRL model performs well, capturing the majority of abnormalities in testing datasets. However, it still did not work well when there were anomalies in data that changed frequently, like when there were anomalies in YOM009 data between 29 June and 1 July 2015, and in CPY015 data on 19 June 2016.



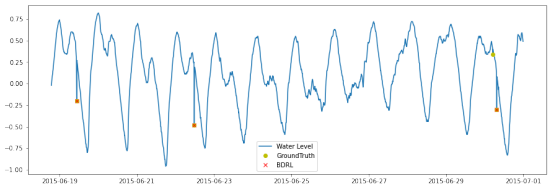
(a) CPY011



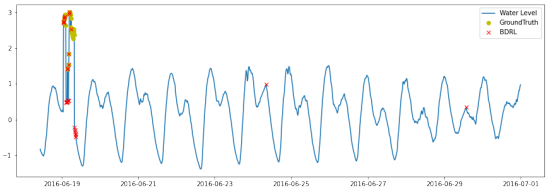
(b) CPY012



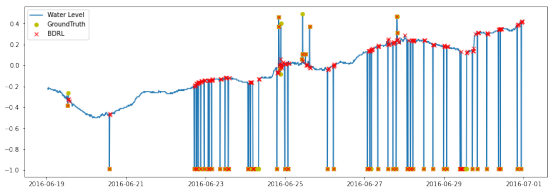
(c) CPY013



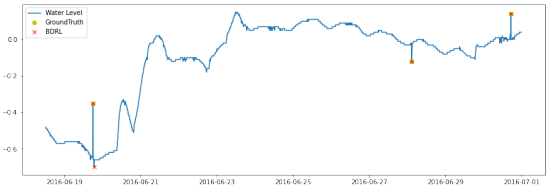
(d) CPY014



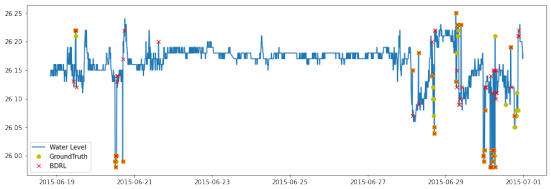
(e) CPY015



(f) CPY016



(g) CPY017



(h) YOM009

Figure 5.5: Anomaly detection from the best DRL model of each station. (**a**) CPY011 with DRL; (**b**) CPY012 with DRL_{Valid}; (**c**) CPY013 with DRL_{F1}; (**d**) CPY014 with DRL_{Rwd}; (**e**) CPY015 with DRL_{F1}; (**f**) CPY016 with DRL_{F1}; (**g**) CPY017 with DRL; (h) YOM009 with DRL_{Acc}.

5.4.2 Performance on the Same Station

We evaluated the performance of our techniques with MLP and LSTM models on eight telemetry water level datasets. The data in each station is first divided into training, validating, and testing parts in a 6:2:2 ratio. The results were averaged after being run ten times and then were compared to the averaged DRL models of each station as shown in Table 5.3.

Table 5.3: The mean F1-score and standard deviation of all DRL, MLP, and LSTM models when testing with the dataset from different stations (the best F1-score of each row is shown in bold).

Station	DRL	DRL_{F1}	DRL_{Rwd}	DRL_{Acc}	DRL_{Valid}	MLP	LSTM
CPY011	0.7433 (±0.08)	0.7170 (±0.07)	0.7433 (±0.08)	0.7170 (±0.07)	0.6020 (±0.17)	**0.8505 (±0.06)**	0.8167 (±0.04)
CPY012	0.6550 (±0.08)	0.7146 (±0.06)	0.6468 (±0.10)	0.7234 (±0.04)	0.7045 (±0.10)	**0.7822 (±0.03)**	0.7753 (±0.02)
CPY013	0.6014 (±0.10)	0.6963 (±0.07)	0.6212 (±0.09)	0.6920 (±0.07)	0.5639 (±0.21)	0.6998 (±0.03)	**0.7265 (±0.02)**
CPY014	0.4823 (±0.20)	0.6733 (±0.08)	0.5537 (±0.26)	0.6733 (±0.08)	0.6596 (±0.14)	**0.8571 (±0.00)**	**0.8571 (±0.00)**
CPY015	0.3477 (±0.06)	**0.4134 (±0.02)**	0.3477 (±0.06)	**0.4134 (±0.02)**	0.3211 (±0.10)	0.2220 (±0.10)	0.3276 (±0.09)
CPY016	0.3770 (±0.03)	0.5438 (±0.05)	0.3499 (±0.06)	0.5714 (±0.02)	0.4867 (±0.07)	0.5651 (±0.14)	**0.6252 (±0.06)**
CPY017	0.6801 (±0.21)	0.6381 (±0.16)	0.6762 (±0.21)	0.7219 (±0.14)	0.6488 (±0.15)	0.9778 (±0.07)	**0.9857 (±0.05)**
YOM009	0.3581 (±0.06)	0.4152 (±0.04)	0.3891 (±0.04)	0.4248 (±0.04)	**0.4378 (±0.03)**	0.2358 (±0.05)	0.2596 (±0.06)

It demonstrated that DRL_{F1} and DRL_{Acc} had the highest average F1-score for detecting anomalies on CPY015, with F1-score of 0.4133. MLP had the greatest average F1-score when it came to detecting anomalies on CPY011, CPY012, and CPY014 with scores of 0.8505, 0.7822, and 0.8571, respectively. On the other stations, LSTM was the top performing model. According to the CD diagram in Figure 5.6, the best LSTM model had the greatest ranking of performance, followed by DRL_{Acc} and MLP.

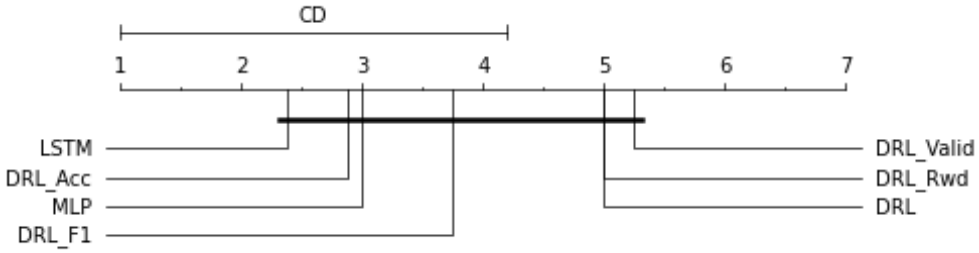


Figure 5.6: A critical difference diagram of each model.

Since RL models need time to learn until they have enough knowledge to do their task, time costing is the one important thing that we need to be interested in. We calculate the time spent by the best deep learning models (BDRL) and comparative models, as shown in Table 5.4. The MLP model requires the least training time per epoch, with an average of 0.30 seconds, followed by the LSTM model at 0.64 seconds, and the DRL model at 17.56 seconds. For MLP and LSTM training with early stopping, they needed an average of 12 and 15 training epochs, respectively, while our method requires around 4,638 epochs to get optimal results. It means that the MLP model took an average of 2.97 seconds to train, while LSTM took 9.20 seconds and DRL took an average of 78,756 seconds, which is about 22 hours.

Table 5.4: The number of training epochs and the time spent on each epoch for each model.

Station	Training Epochs			Time(s./epochs)			Total Time(s.)		
	MLP	**LSTM**	**BDRL**	**MLP**	**LSTM**	**BDRL**	**MLP**	**LSTM**	**BDRL**
CPY011	12	17	1000	0.17	0.64	18.66	2.04	10.88	18,666
CPY012	15	19	10,000	0.16	0.62	18.20	2.40	11.78	182,000
CPY013	11	17	10,000	0.18	0.64	17.88	1.98	10.88	178,000
CPY014	11	6	500	0.55	0.83	18.32	6.05	4.98	2490
CPY015	7	11	10,000	0.24	0.62	16.03	1.68	6.82	160,300
CPY016	17	21	5000	0.17	0.51	16.58	2.89	10.71	82,900
CPY017	13	17	100	0.20	0.56	16.74	2.6	9.52	1674
YOM009	6	11	500	0.69	0.73	18.82	4.14	8.03	4015
avg	12	15	4638	0.30	0.64	17.65	2.97	9.20	78,756

5.4.3 Performance on the Different Station

After generating various models on some stations' data and testing them with the same stations, we tested these models with the data collected from different stations with the intention of examining their generalisation ability. The F1-score of each model are provided in Table 5.5.

Table 5.5: The F1-score of the best DRL models when testing with the dataset from same station (show in the bracket) and different stations, while the average F1-score and standard deviations of each station were calculated without their own scores.

Tested Dataset	Trained Dataset							
	CPY011	**CPY012**	**CPY013**	**CPY014**	**CPY015**	**CPY016**	**CPY017**	**YOM009**
CPY011	(0.8333)	0.1667	0.1967	0.0255	0.4138	0.0596	0.2000	0.1556
CPY012	0.6000	(0.7826)	0.7164	0.1136	0.0533	0.5088	0.6667	0.6667
CPY013	0.6531	0.6667	(0.8000)	0.1875	0.1034	0.5421	0.5970	0.5952
CPY014	0.4000	0.8571	0.8571	(0.8571)	0.0000	0.2727	0.5871	0.6000
CPY015	0.0000	0.1538	0.1474	0.0672	(0.4407)	0.0900	0.2078	0.0839
CPY016	0.5369	0.6129	0.6550	0.1276	0.4103	(0.6058)	0.1203	0.6703
CPY017	0.5000	0.3529	0.5455	0.2308	0.0000	0.1765	(0.8571)	0.4000
YOM009	0.0000	0.3297	0.1905	0.0771	0.0000	0.4189	0.3158	(0.4769)
avg	0.3843	0.4485	0.4727	0.1185	0.1401	0.2955	0.3850	**0.4531**
std	0.2742	0.2680	0.2908	0.0713	0.1896	0.1975	0.2257	0.2457

Using DRL_{Rwd} - the best model for detecting anomalies by training with CPY011 data and then identifying anomalies from other stations, we can see that, though it works rather well, with F1-score ranging from 0.4 on CPY014 to 0.65 on CPY013 data, it is unable to detect anomalies on CPY015 and YOM009. Using the BDRL model of the CPY012 training dataset, DRL_{Valid}, although it provided good performance when identifying anomalies in the CPY013, CPY014, and CPY016 datasets with F1-score

greater than 0.61, especially CPY014 with a 0.8571 F1-score, which more than detected anomalies on its own dataset, it provided poor performance, with an F1-score lower than 0.4000, when detecting anomalies in other stations. Similar to DRL_{F1}, which was trained using CPY013 data, it not only performs well when recognising anomalies on its own dataset but also when detecting anomalies on the CPY014 dataset, with an F1-score of 0.8571. The BDRL model, DRL_{Rwd}, that was trained with CPY014 did the worst when it was used to find anomalies in other stations' data, with an F1-score of less than 0.23 for every dataset and the lowest F1-score of only 0.0255 for CPY011. Similar to the best model on CPY015 datasets, which performed poorly, with the highest F1-score on CPY011 data being 0.4138 and being unable to identify anomalies on CPY014, CPY017, and YOM009. Meanwhile, the best model for detecting anomalies on CPY016 data performed the best for detecting anomalies on CPY013 with a 0.5421 F1-score. The model that was trained on CPY017 did the best of finding anomalies in data from CPY012, CPY013, and CPY014 with an F1-score greater than 0.58. While the best model from the YOM009 training dataset achieved a low F1-score on CPY011, CPY015, and CPY017, 0.0839 is the lowest F1-score. However, when it was used to find outliers on COY012, CPY013, CPY014, and CPY016 with F1-score higher than 0.59, it did better than its own training data.

It is worth noting that models trained using CPY014 and CPY015 data perform poorly when used to identify anomalies from other stations. This may be due to the fact that the actual number of anomalies in those stations are relatively low and most of them are kind of extreme outliers, as shown in Figure 5.2, so the models were trained with only those kinds of anomalies, which may not be enough for the model to learn. In contrast to YOM009, which has a many number and types of anomalies for model to learn, as a result, it can identify abnormalities on CPY012, CPY013, CPY014, and CPY016 better than other models that were trained with another station.

Then, we tested MLP and LSTM using data from different stations to compare our method to the candidate models. Table 5.6 represents the results of the MLP models when tested with the datasets from the same and different stations. Using the CPY011 dataset, the MLP models achieved the highest F1-score of 0.5430 on CPY016, despite

Table 5.6: The F1-score of the MLP models when testing with the dataset from the same station (shown in the bracket) and different stations.

Tested Dataset	**Trained Dataset**							
	CPY011	**CPY012**	**CPY013**	**CPY014**	**CPY015**	**CPY016**	**CPY017**	**YOM009**
CPY011	(0.9231)	0.4000	0.2917	0.7778	0.0000	0.1373	0.7059	0.6087
CPY012	0.4889	(0.8387)	0.8060	0.7500	0.0541	0.6923	0.7273	0.7857
CPY013	0.4390	0.6786	(0.7500)	0.7000	0.0000	0.7123	0.6909	0.7719
CPY014	0.0000	0.8571	0.8571	(0.8571)	0.0000	0.6000	0.8571	0.8571
CPY015	0.1951	0.1509	0.2093	0.2593	(0.3529)	0.1420	0.3077	0.2414
CPY016	0.5430	0.6380	0.6587	0.6322	0.0915	(0.6629)	0.5665	0.6550
CPY017	0.5000	0.6667	0.6000	1.0000	0.0000	0.2857	(1.0000)	1.0000
YOM009	0.0000	0.2308	0.2955	0.1818	0.0000	0.4404	0.1772	(0.2857)
avg	0.3094	0.5174	0.5312	0.6144	0.0208	0.4300	0.5761	**0.7028**
std	0.2396	0.2609	0.2643	0.2929	0.0371	0.2473	0.2460	0.2408

their being unable to identify anomalies on CPY014 and YOM009. Similar to finding anomalies on CPY012, it offered good results with F1-score of more than 0.63, with the exception of CPY011, CPY015, and YOOM009, which produced F1-score of less than 0.4. The best MLP of the CPY013 training dataset provided the highest F1-score on the CPY014 dataset (0.8571 F1-score) and the lowest on CPY015 (0.2093 F1-score). Anomalies on the YOM009 dataset were the most difficult for the MLP models trained on CPY014 to detect, with an F1-score of just 0.1818. However, it performed excellent results in identifying anomalies on CPY017 with a 1.0000 F1-score. Meanwhile, the MLP model on the CPY015 dataset performed poorly when detecting abnormalities from other stations. On the other hand, the MLP models that were trained on CPY016 and CPY017 generated good results when used to identify anomalies from other stations, despite still performing poorly in some stations. In contrast, the MLP model trained on YOM009 worked well when used to detect abnormalities on other stations but performed badly when detecting anomalies on its own data. Furthermore, it performed well on CPY017 data, with a 1.000 F1-score.

In the case of the LSTM model, as depicted in Table 5.7. They performed well, with an average F1-score of more than 0.42 for each station except CPY015, which had an average F1-score of 0.1099. However, they generated poor performances in some stations, such as the LSTM of CPY016 that achieved an F1-score of only 0.1754 when used to detect anomalies on the CPY011 dataset, and it was unable to detect anomalies on CPY014, CPY017, and YOM009 datasets with the LSTM that had been trained on the CPY015 dataset. However, it provided excellent performance when

Table 5.7: The F1-score of the LSTM models when testing with the dataset from the same station (shown in the bracket) and different stations.

Tested Dataset	**Trained Dataset**							
	CPY011	**CPY012**	**CPY013**	**CPY014**	**CPY015**	**CPY016**	**CPY017**	**YOM009**
CPY011	(0.8571)	0.4828	0.2333	0.5000	0.3077	0.1754	0.7059	0.5185
CPY012	0.6400	(0.8387)	0.8308	0.7368	0.0444	0.7536	0.7500	0.7458
CPY013	0.5652	0.7333	(0.7463)	0.7213	0.1081	0.6857	0.7333	0.7188
CPY014	0.4000	0.8571	0.8571	(0.8571)	0.0000	0.8571	0.8571	0.8571
CPY015	0.3043	0.3636	0.2340	0.3939	(0.4364)	0.2045	0.3704	0.2564
CPY016	0.5679	0.6897	0.6824	0.5634	0.3089	(0.6630)	0.6296	0.6359
CPY017	0.5000	1.0000	0.5455	1.0000	0.0000	0.3333	(1.0000)	0.8571
YOM009	0.0000	0.2619	0.3146	0.2727	0.0000	0.4190	0.2857	(0.3542)
avg	0.4253	0.6269	0.5282	0.5983	0.1099	0.4898	0.6189	**0.6557**
std	0.2190	0.2679	0.2717	0.2430	0.1410	0.2747	0.2111	0.2129

detecting anomalies on CPY017 with the LSTM that has been trained on the CPY014 dataset. When the LSTM was trained on YOM009, it did well at finding anomalies from other stations, especially CPY014 and CPY017, with an F1-score of 0.8571.

Furthermore, we generated a bar chart to compare the average F1-score from each model when tested with the data collected from different stations, as shown in Figure 5.7. When evaluated with data from other stations, the models trained with CPY012 and CPY013 produced an average F1-score greater than 0.4. The models trained on CPY015 earned poor performance when used to identify anomalies from other stations, with an average F1-score lower than 0.2. DRL models that were trained with CPY015 outperform other models in detecting anomalies in data from other stations. LSTM models trained on CPY011, CPY012, CPY016, and CPY017, on the other hand, outperform other models in detecting abnormalities on other datasets. When trained with data from CPY013, CPY014, and YOM009, MLP had the best F1-score for finding outliers in other datasets.

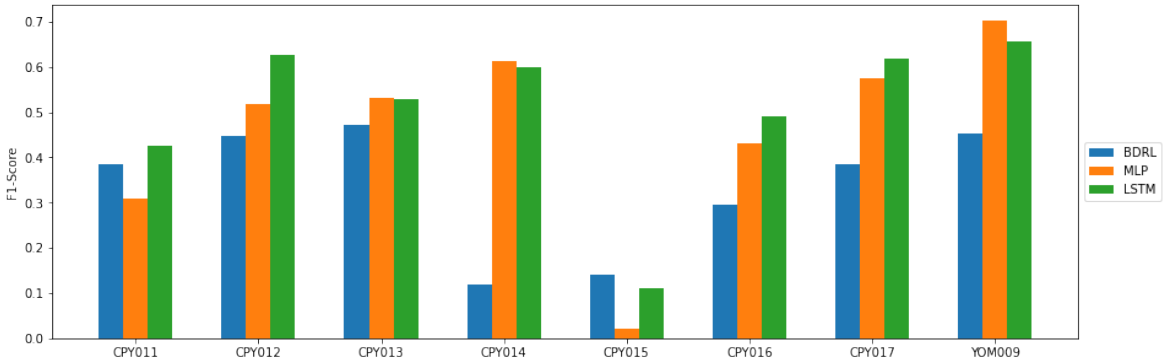


Figure 5.7: Bar charts of average F1-score of the DRL, MLP, and LSTM when tested with the data collected from different stations.

5.4.4 Ensemble Results

Since we have multiple RL models after each epoch of training, and since each model performs the best in each of the criteria, we then built an ensemble that combined the decisions of all RL models, with the aim of generating a better final decision. In model selection, we select all five models and select the three models with the highest ranking in F1-score to build our ensemble model. For decision making, we used majority voting and weighted voting strategies to make a final decision. So, we have 4 ensemble models for each epoch of training, including a majority voting ensemble model with 3 ($EDRL_3$) and 5 ($EDRL_5$) models, and a weighted ensemble model with 3 ($WEDRL_3$) and 5 ($WEDRL_5$) models.

Performance on the Same Station

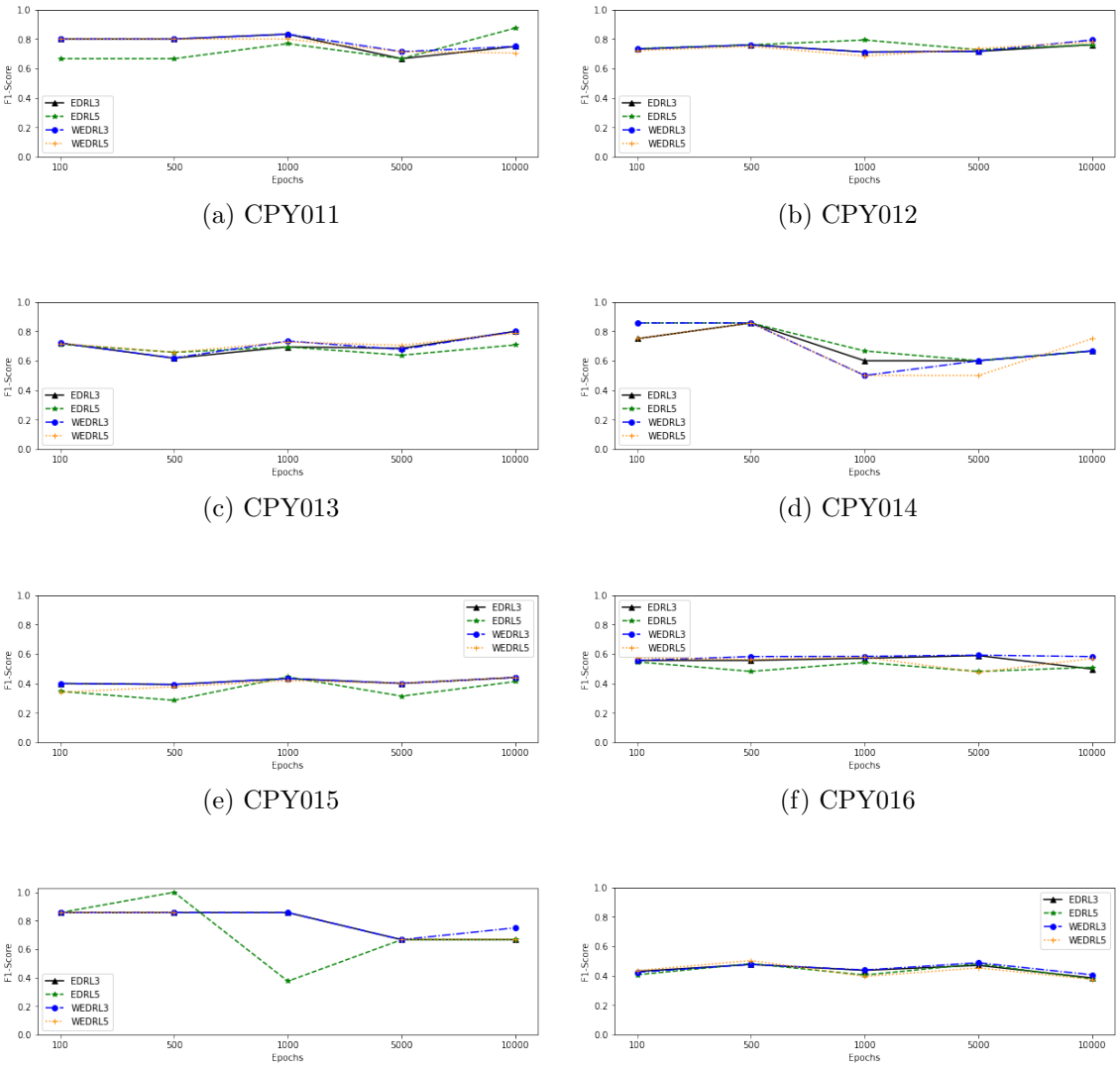
The results of our ensemble models are shown in Table 5.8 demonstrated that ensemble with majority voting and weighted voting that were generated from the top three DRL models of CPY011 provided the best with 0.8333 F1-score, while $WDRL_3$ that was generated from the DRL model after trained with 10,000 epochs is the best model to detect anomalies in CPY012 datasets with an F1-score of 0.7941. The ensemble model of CPY013 that performs the best is $EDRL_3$ and $WEDRL_3$ at 0.8000. The best ensemble model for identifying anomalies in CPY014 datasets is the ensemble model that provided the F1-score of 0.8571. With CPY015 data, the models with the highest F1-score are $EDRL_3$, $WEDRL_3$, and $WEDRL_5$. These models were built based on the individual DRL model, which was trained for 10,000 iterations. Meanwhile, $WEDRL_3$ got the highest F1-score of 0.5922 for CPY016 by combining the best three DRL models that were trained over 5000 iterations. With CPY017, $EDRL_5$ outperforms other ensemble models with a 100 percent in every metric. The ensemble results of YOM009, $WEDRL_5$, offered the highest performance with an F1-score of 0.5032 that was generated from the DRL model after 500 epochs of training.

Table 5.8: The performance of ensemble models (the best F1-score of each row is shown in bold).

Station	Epochs	$EDRL_3$			$EDRL_5$			$WEDRL_3$			$WEDRL_5$		
		Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1
CPY011	100	0.8571	0.7500	**0.8000**	0.7143	0.6250	0.6667	0.8571	0.7500	**0.8000**	0.8571	0.7500	**0.8000**
	500	0.8571	0.7500	**0.8000**	0.8571	0.6667	0.7500	0.8571	0.7500	**0.8000**	0.8571	0.6667	0.7500
	1000	0.7143	1.0000	**0.8333**	0.7143	0.8333	0.7692	0.7143	1.0000	**0.8333**	0.8571	0.7500	0.8000
	5000	0.7143	0.6250	0.6667	0.7143	0.6250	0.6667	0.7143	0.7143	**0.7143**	0.7143	0.7143	**0.7143**
	10,000	0.8571	0.6667	0.7500	1.0000	0.7778	**0.8750**	0.8571	0.6667	0.7500	0.8571	0.6000	0.7059
	Avg	0.8000	0.7583	0.7700	0.8000	0.7056	0.7455	0.8000	0.7762	**0.7795**	0.8285	0.6962	0.7540
	std	0.0782	0.1455	0.0650	0.1278	0.0949	0.0863	0.0782	0.1297	0.0471	0.0639	0.0637	0.0451
CPY012	100	0.7647	0.7027	0.7324	0.7353	0.7353	**0.7353**	0.7647	0.7027	0.7324	0.7647	0.6842	0.7222
	500	0.7941	0.7297	**0.7606**	0.7941	0.7297	**0.7606**	0.7941	0.7297	**0.7606**	0.7941	0.7105	0.7500
	1000	0.7647	0.6667	0.7123	0.7353	0.8621	**0.7937**	0.7647	0.6667	0.7123	0.7353	0.6410	0.6849
	5000	0.7059	0.7273	0.7164	0.7059	0.7500	0.7273	0.7059	0.7273	0.7164	0.7353	0.7353	**0.7353**
	10,000	0.7059	0.8276	0.7619	0.7059	0.8276	0.7619	0.7941	0.7941	**0.7941**	0.7353	0.8333	0.7812
	Avg	0.7471	0.7308	0.7367	0.7353	0.7809	**0.7558**	0.7647	0.7241	0.7432	0.7529	0.7209	0.7347
	std	0.0394	0.0598	0.0236	0.0360	0.0601	0.0261	0.0360	0.0466	0.0342	0.0263	0.0719	0.0355
CPY013	100	0.8710	0.6136	**0.7200**	0.8387	0.6190	0.7123	0.8710	0.6136	**0.7200**	0.9032	0.5957	0.7179
	500	0.8065	0.5000	0.6173	0.7742	0.5714	0.6575	0.8387	0.4906	0.6190	0.8065	0.5556	**0.6579**
	1000	0.8065	0.6098	0.6944	0.8065	0.6098	0.6944	0.9355	0.6042	**0.7342**	0.9032	0.6087	0.7273
	5000	0.8065	0.5952	0.6849	0.7097	0.5789	0.6377	0.7097	0.6471	0.6769	0.7742	0.6486	**0.7059**
	10,000	0.8387	0.7647	**0.8000**	0.7419	0.6765	0.7077	0.8387	0.7647	**0.8000**	0.8387	0.7429	0.7879
	Avg	0.8258	0.6167	0.7033	0.7742	0.6111	0.6819	0.8387	0.6240	0.7100	0.8452	0.6303	**0.7194**
	std	0.0288	0.0949	0.0660	0.0510	0.0417	0.0328	0.0822	0.0983	0.0674	0.0577	0.0712	0.0467
CPY014	100	0.7500	0.7500	0.7500	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	0.7500	0.7500
	500	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	1000	0.7500	0.5000	0.6000	0.7500	0.6000	**0.6667**	0.7500	0.3750	0.5000	0.7500	0.3750	0.5000
	5000	0.7500	0.5000	**0.6000**	0.7500	0.5000	**0.6000**	0.7500	0.5000	**0.6000**	0.7500	0.3750	0.5000
	10,000	0.7500	0.6000	0.6667	0.7500	0.6000	0.6667	0.7500	0.6000	0.6667	0.7500	0.7500	**0.7500**
	Avg	0.7500	0.6700	0.6948	0.7500	0.7400	**0.7295**	0.7500	0.6950	0.6962	0.7500	0.6500	0.6714
	std	0.0000	0.2110	0.1097	0.0000	0.2408	0.1196	0.0000	0.2896	0.1584	0.0000	0.2710	0.1625
CPY015	100	0.3235	0.5238	**0.4000**	0.2647	0.5000	0.3462	0.3235	0.5238	**0.4000**	0.2647	0.4737	0.3396
	500	0.3235	0.5000	**0.3929**	0.2059	0.4667	0.2857	0.3235	0.5000	**0.3929**	0.2941	0.5263	0.3774
	1000	0.3824	0.5000	0.4333	0.4118	0.4828	**0.4444**	0.3824	0.5000	0.4333	0.3824	0.4643	0.4194
	5000	0.3235	0.5238	**0.4000**	0.2353	0.4706	0.3137	0.3235	0.5238	**0.4000**	0.3235	0.5238	**0.4000**
	10,000	0.3824	0.5200	**0.4407**	0.3824	0.4483	0.4127	0.3824	0.5200	**0.4407**	0.3824	0.5200	**0.4407**
	Avg	0.3471	0.5135	**0.4134**	0.3000	0.4737	0.3605	0.3471	0.5135	**0.4134**	0.3294	0.5016	0.3954
	std	0.0323	0.0124	0.0219	0.0916	0.0192	0.0666	0.0323	0.0124	0.0219	0.0526	0.0300	0.0390
CPY016	100	0.5981	0.5203	0.5565	0.6075	0.4962	0.5462	0.5981	0.5203	0.5565	0.6168	0.5366	**0.5739**
	500	0.5981	0.5203	0.5565	0.6168	0.3952	0.4818	0.6075	0.5603	**0.5830**	0.5981	0.5333	0.5639
	1000	0.6168	0.5323	0.5714	0.6542	0.4636	0.5426	0.6168	0.5546	**0.5841**	0.6168	0.5455	0.5789
	5000	0.5701	0.6100	0.5894	0.5514	0.4275	0.4816	0.5701	0.6162	**0.5922**	0.5888	0.3987	0.4755
	10,000	0.6168	0.4177	0.4981	0.6262	0.4295	0.5095	0.6262	0.5447	**0.5826**	0.6449	0.5111	0.5702
	Avg	0.6000	0.5201	0.5544	0.6112	0.4424	0.5123	0.6037	0.5592	**0.5797**	0.6131	0.5050	0.5525
	std	0.0191	0.0684	0.0343	0.0377	0.0386	0.0314	0.0215	0.0353	0.0135	0.0215	0.0608	0.0434
CPY017	100	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**
	500	1.0000	0.7500	0.8571	1.0000	1.0000	**1.0000**	1.0000	0.7500	0.8571	1.0000	0.7500	0.8571
	1000	1.0000	0.7500	**0.8571**	1.0000	0.2308	0.3750	1.0000	0.7500	**0.8571**	1.0000	0.7500	**0.8571**
	5000	1.0000	0.5000	**0.6667**	1.0000	0.5000	**0.6667**	1.0000	0.5000	**0.6667**	1.0000	0.5000	**0.6667**
	10,000	0.6667	0.6667	0.6667	0.6667	0.6667	0.6667	1.0000	0.6000	**0.7500**	0.6667	0.6667	0.6667
	Avg	0.9333	0.6833	0.7809	0.9333	0.6295	0.7131	1.0000	0.6700	**0.7976**	0.9333	0.6833	0.7809
	std	0.1491	0.1087	0.1043	0.1491	0.2868	0.2354	0.0000	0.1151	0.0866	0.1491	0.1087	0.1043
YOM009	100	0.6308	0.3254	**0.4293**	0.5846	0.3115	0.4064	0.6154	0.3226	0.4233	0.6462	0.3281	0.4352
	500	0.4769	0.4769	0.4769	0.6154	0.3960	0.4819	0.4769	0.4769	0.4769	0.6000	0.4333	**0.5032**
	1000	0.5538	0.3600	0.4364	0.5538	0.3186	0.4045	0.5231	0.3778	**0.4387**	0.4923	0.3299	0.3951
	5000	0.5538	0.4091	0.4706	0.5846	0.4086	0.4810	0.6308	0.3981	**0.4881**	0.5538	0.3830	0.4528
	10,000	0.5385	0.2991	0.3846	0.4923	0.3048	0.3765	0.4154	0.3971	**0.4060**	0.4615	0.3158	0.3750
	Avg	0.5508	0.3741	0.4396	0.5661	0.3479	0.4301	0.5323	0.3945	**0.4466**	0.5508	0.3580	0.4323
	std	0.0548	0.0707	0.0371	0.0467	0.0501	0.0484	0.0914	0.0554	0.0350	0.0757	0.0494	0.0503

Figure 5.8 depicts line charts that indicate the F1-score of each ensemble model that was trained using data from each station. It was clear from the results that the ensemble models not only delivered good performances and had a tendency to either improve or keep their F1-score steady but also reduced the false alarms by increasing the precision

scores. When we compared the results of each training epoch of the individual DRL model and the ensemble model, as shown in Tables 5.2 and 5.8, we discovered that ensemble models performed better than every single DRL model in many training epochs. In particular, $EDRL_5$ on the CPY017 with 500 training epochs generated an excellent score of 1.0000 in every metrics index, resulting from a 25% increase in accuracy and a 15% increase in F1-score. Meanwhile, $EDRL_5$ on the CPY011 with 10,000 training epochs improved the performance of the best individual model with an F1-score from 0.75 to 0.8750, reached 1.00 in terms of recall, and increased precision by 20%. By combining the DRL models trained on only 500 epochs, the ensemble model on YOM009 got the highest F1-score of 0.5032.



(a) CPY011 (b) CPY012

(c) CPY013 (d) CPY014

(e) CPY015 (f) CPY016

(g) CPY017 (h) YOM009

Figure 5.8: F1-score of ensemble model when increasing the learning epochs at CPY011, CPY012, CPY013, CPY014, CPY015, CPY016, CPY017, and YOM009 (**a**-**h**).

Table 5.9: The mean F1-score and standard deviations of all of the DRL, MLP, LSTM, and ensemble of DRL-based models when testing with the dataset from different stations (the best F1-score of each station is shown in bold).

Models	CPY011	CPY012	CPY013	CPY014	CPY015	CPY016	CPY017	YOM009
DRL	0.7433 (±0.08)	0.6550 (±0.08)	0.6014 (±0.10)	0.4823 (±0.20)	0.3477 (±0.06)	0.3770 (±0.03)	0.6801 (±0.21)	0.3581 (±0.06)
DRL_{F1}	0.7170 (±0.07)	0.7146 (±0.06)	0.6963 (±0.07)	0.6733 (±0.08)	**0.4134 (±0.02)**	0.5438 (±0.05)	0.6381 (±0.16)	0.4152 (±0.04)
$DRLRwd$	0.7433 (±0.08)	0.6468 (±0.10)	0.6212 (±0.09)	0.5537 (±0.26)	0.3477 (±0.06)	0.3499 (±0.06)	0.6762 (±0.21)	0.3891 (±0.04)
DRL_{Acc}	0.7170 (±0.07)	0.7234 (±0.04)	0.6920 (±0.07)	0.6733 (±0.08)	**0.4134 (±0.02)**	0.5714 (±0.02)	0.7219 (±0.14)	0.4248 (±0.04)
DRL_{Valid}	0.6020 (±0.17)	0.7045 (±0.10)	0.5639 (±0.21)	0.6596 (±0.14)	0.3211 (±0.10)	0.4867 (±0.07)	0.6488 (±0.15)	0.4378 (±0.03)
$EDRL_3$	0.7700 (±0.06)	0.7367 (±0.02)	0.7033 (±0.07)	0.6948 (±0.11)	**0.4134 (±0.02)**	0.5544 (±0.03)	0.7809 (±0.10)	0.4396 (±0.04)
$EDRL_5$	0.7455 (±0.09)	0.7558 (±0.03)	0.6819 (±0.03)	0.7295 (±0.12)	0.3605 (±0.07)	0.5123 (±0.03)	0.7131 (±0.24)	0.4301 (±0.05)
$WEDRL_3$	0.7795 (±0.05)	0.7432 (±0.03)	0.7100 (±0.07)	0.6962 (±0.16)	**0.4134 (±0.02)**	0.5797 (±0.01)	0.7976 (±0.09)	**0.4466 (±0.03)**
$WEDRL_5$	0.7540 (±0.05)	0.7347 (±0.04)	0.7194 (±0.05)	0.6714 (±0.16)	0.3954 (±0.04)	0.5525 (±0.04)	0.7619 (±0.10)	0.4323 (±0.05)
MLP	**0.8505 (±0.06)**	**0.7822 (±0.03)**	0.6998 (±0.03)	**0.8571 (±0.00)**	0.2220 (±0.10)	0.5651 (±0.14)	0.9778 (±0.07)	0.2358 (±0.05)
LSTM	0.8167 (±0.04)	0.7753 (±0.02)	**0.7265 (±0.02)**	**0.8571 (±0.00)**	0.3276 (±0.09)	**0.6252 (±0.06)**	**0.9857 (±0.05)**	0.2596 (±0.06)

As shown in Table 5.9, we evaluated the average F1-score of each individual DRL model and ensemble of DRL models against the other neural network models. We can see that the LSTM model was the best model when detecting anomalies on CPY013, CPY014, CPY016, and CPY017, while $WEDRL_3$ provided the highest average F1-score on CPY015 and YOM009. The highest F1-score was 0.4134 on CPY015, which was provided by $DRL_{F1}, DRL_{Acc}, EDRL_3$, and $WEDRL_3$. Although MLP and LSTM beat other models in many datasets, $WEDRL_3$ has the greatest average ranking, as shown in Figure 5.9. In other words, the ensemble model not only has the potential to improve the performance of a single model, but it also has a higher reliability to deliver excellent performance than a single model.

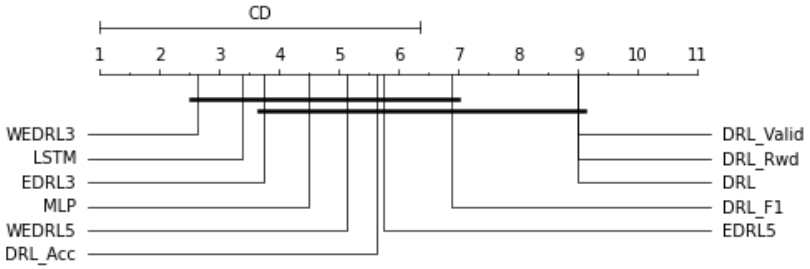


Figure 5.9: Critical difference diagram for DRL, MLP, LSTM, and ensemble of DRL-based models when testing with the dataset from different stations.

Performance on the Different Station

We then tested the generalisation ability of the best ensemble ($WEDRL_3$) with the data collected from different stations. The F1-score of each model is depicted in Table 5.10. We can observe that the ensemble model that was created from the model trained on CPY011 data performed well not only on their own dataset but also on CPY017, with an F1-score of 0.8200, similarly to $WEDRL_3$ on CPY012 and CPY013,

Table 5.10: The mean F1-score of the $WEDRL_3$ models when testing with the dataset from the same station (shown in the bracket) and different stations.

Tested Dataset	**Trained Dataset**							
	CPY011	**CPY012**	**CPY013**	**CPY014**	**CPY015**	**CPY016**	**CPY017**	**YOM009**
CPY011	(0.7795)	0.2772	0.2278	0.1718	0.4345	0.0788	0.2830	0.0900
CPY012	0.7183	(0.7432)	0.6817	0.4624	0.2182	0.5355	0.6419	0.5096
CPY013	0.6920	0.7281	(0.7101)	0.5512	0.2713	0.5625	0.5959	0.4816
CPY014	0.7276	0.8421	0.8143	(0.6962)	0.5714	0.5471	0.7948	0.3450
CPY015	0.3748	0.2683	0.1664	0.1482	(0.4134)	0.1290	0.1994	0.0739
CPY016	0.5813	0.6283	0.6450	0.3327	0.3647	(0.5797)	0.4711	0.5748
CPY017	0.8200	0.4423	0.4514	0.3803	0.3545	0.2075	(0.7976)	0.2931
YOM009	0.1084	0.3261	0.3109	0.2619	0.0568	0.4613	0.3375	(0.4466)
avg	**0.6002**	0.5320	0.5010	0.3756	0.3356	0.3877	0.5152	0.3518
std	0.2426	0.2310	0.2451	0.1887	0.1551	0.2121	0.2292	0.1889

which recognised anomalies on CPY014 better than their own dataset with F1-score of 0.8421 and 0.8143, respectively. Inversely, the ensemble model on CPY014, CPY015, and CPY016 trained datasets provided poor performance when used to detect anomalies on other stations. Even though the ensemble model trained on the CPY017 dataset got an F1-score of more than 0.5 on CPY012, CPY013, and CPY014, it did not do well on many stations, with an F1-score of less than 0.3. $WEDRL_3$ scored badly not just on their own dataset but also on others, with F1-score ranging from 0.0739 on CPY015 to 0.5748 on CPY016.

Ensemble with all Seven Models

Then, to learn more about how well the ensemble worked, we combined our developed DRL models with MLP and LSTM models. In model selection, we selected all seven models and selected the five and three models with the highest ranking in F1-score to build our ensemble model. We used the same strategy to make a final decision. So, we have 6 ensemble model for each epochs of training include majority voting ensemble model with 3 ($E3$), 5 ($E5$), and 7 ($E7$) model, and weighted ensemble model with 3 ($WE3$), 5 ($WE5$), and 7 ($WE7$) models, and the results are displayed in Table 5.11.

Table 5.11: The performance of the ensemble models built by combining DRL and candidate models (the best F1-score of each row is shown in bold).

Station	Epochs	*E3*			*E5*			*E7*			*WE3*			*WE5*			*WE7*		
		Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1	Recall	Prec	F1
CPY011	100	0.8571	1.0000	**0.9231**	0.8571	0.7500	0.8000	0.8571	0.7500	0.8000	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571
	500	0.8571	1.0000	**0.9231**	0.8571	0.7500	0.8000	0.8571	0.7500	0.8000	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571
	1000	0.8571	1.0000	**0.9231**	0.8571	1.0000	**0.9231**	0.8571	1.0000	**0.9231**	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571
	5000	0.8571	1.0000	**0.9231**	0.7143	0.7143	0.7143	0.8571	0.7500	0.8000	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571
	10,000	0.8571	1.0000	**0.9231**	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571
	avg	0.8571	1.0000	**0.9231**	0.8285	0.8143	0.8189	0.8571	0.8214	0.8360	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571	0.8571
	std	0.0000	0.0000	0.0000	0.0639	0.1168	0.0774	0.0000	0.1101	0.0546	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CPY012	100	0.7647	0.9286	**0.8387**	0.7647	0.8387	0.8000	0.7941	0.7714	0.7826	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254
	500	0.7647	0.9286	0.8387	0.7941	0.7297	0.7606	0.7941	0.9000	**0.8438**	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254
	1000	0.7647	0.9286	**0.8387**	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254
	5000	0.7647	0.9286	**0.8387**	0.7059	0.7500	0.7273	0.7353	0.8333	0.7812	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254
	10,000	0.7647	0.9286	0.8387	0.7941	0.9000	**0.8438**	0.7059	0.8276	0.7619	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7941	0.9000	**0.8438**
	avg	0.7647	0.9286	**0.8387**	0.7647	0.8230	0.7914	0.7588	0.8458	0.7990	0.7647	0.8966	0.8254	0.7647	0.8966	0.8254	0.7706	0.8973	0.8291
	std	0.0000	0.0000	0.0000	0.0360	0.0800	0.0475	0.0383	0.0537	0.0342	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0131	0.0015	0.0082
CPY013	100	0.7742	0.7273	0.7500	0.8387	0.6842	**0.7536**	0.8065	0.6757	0.7353	0.7419	0.6765	0.7077	0.7742	0.6857	0.7273	0.7742	0.6857	0.7273
	500	0.7742	0.7273	**0.7500**	0.8387	0.6341	0.7222	0.7419	0.6970	0.7188	0.7419	0.6765	0.7077	0.7419	0.6765	0.7077	0.7419	0.6765	0.7077
	1000	0.7742	0.7273	0.7500	0.8710	0.6750	**0.7606**	0.8065	0.6410	0.7143	0.7419	0.6765	0.7077	0.7742	0.6857	0.7273	0.7742	0.6857	0.7273
	5000	0.7742	0.7273	**0.7500**	0.7419	0.6765	0.7077	0.7419	0.6765	0.7077	0.7419	0.6765	0.7077	0.7419	0.6765	0.7077	0.7742	0.6857	0.7273
	10,000	0.8387	0.7647	**0.8000**	0.7742	0.7742	0.7742	0.7742	0.7742	0.7742	0.8387	0.7647	**0.8000**	0.7742	0.7273	0.7500	0.7419	0.6765	0.7077
	avg	0.7871	0.7348	**0.7600**	0.8129	0.6888	0.7437	0.7742	0.6929	0.7301	0.7613	0.6941	0.7262	0.7613	0.6903	0.7240	0.7613	0.6820	0.7195
	std	0.0288	0.0167	0.0224	0.0530	0.0516	0.0277	0.0323	0.0497	0.0267	0.0433	0.0394	0.0413	0.0177	0.0212	0.0175	0.0177	0.0050	0.0107
CPY014	100	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	500	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	1000	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	5000	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	10,000	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	0.6000	0.6667	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	avg	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	0.9200	0.8190	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**	0.7500	1.0000	**0.8571**
	std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1789	0.0851	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
CPY015	100	0.3235	0.5238	**0.4000**	0.2647	0.5000	0.3462	0.2647	0.5000	0.3462	0.3235	0.5238	**0.4000**	0.2647	0.4737	0.3396	0.2353	0.4444	0.3077
	500	0.3235	0.5000	**0.3929**	0.2353	0.5000	0.3200	0.2353	0.5000	0.3200	0.3235	0.5000	**0.3929**	0.2941	0.5263	0.3774	0.2647	0.5000	0.3462
	1000	0.3824	0.5000	0.4333	0.4118	0.4828	**0.4444**	0.2647	0.4286	0.3273	0.3824	0.5000	0.4333	0.3824	0.4643	0.4194	0.3824	0.4333	0.4062
	5000	0.3235	0.5238	**0.4000**	0.2647	0.5000	0.3462	0.2647	0.5000	0.3462	0.3235	0.5238	**0.4000**	0.2941	0.5000	0.3704	0.2941	0.5000	0.3704
	10,000	0.3824	0.5200	0.4407	0.3824	0.4483	0.4127	0.3529	0.4800	0.4068	0.3824	0.5200	0.4407	0.3824	0.5200	0.4407	0.4412	0.4839	**0.4615**
	avg	0.3471	0.5135	**0.4134**	0.3118	0.4862	0.3739	0.2765	0.4817	0.3493	0.3471	0.5135	**0.4134**	0.3235	0.4969	0.3895	0.3235	0.4723	0.3784
	std	0.0323	0.0124	0.0219	0.0795	0.0225	0.0522	0.0446	0.0309	0.0342	0.0323	0.0124	0.0219	0.0551	0.0274	0.0404	0.0858	0.0315	0.0587
CPY016	100	0.5421	0.8529	**0.6629**	0.5981	0.5470	0.5714	0.6075	0.5462	0.5752	0.5421	0.8406	0.6591	0.5421	0.8286	0.6554	0.5421	0.8286	0.6554
	500	0.5421	0.8529	**0.6629**	0.5981	0.6154	0.6066	0.5888	0.6632	0.6238	0.5421	0.8406	0.6591	0.5421	0.8406	0.6591	0.5421	0.8406	0.6591
	1000	0.5421	0.8529	**0.6629**	0.5794	0.5794	0.5794	0.6075	0.5372	0.5702	0.5421	0.8406	0.6591	0.5421	0.8286	0.6554	0.5421	0.8286	0.6554
	5000	0.5421	0.8529	0.6629	0.5607	0.7059	0.6250	0.5607	0.6122	0.5854	0.5421	0.8406	0.6591	0.5421	0.8286	0.6554	0.5607	0.8333	**0.6704**
	10,000	0.5421	0.8529	**0.6629**	0.6075	0.5752	0.5909	0.5981	0.4638	0.5224	0.5421	0.8406	0.6591	0.5421	0.8286	0.6554	0.5421	0.8529	**0.6629**
	avg	0.5421	0.8529	**0.6629**	0.5888	0.6046	0.5947	0.5925	0.5645	0.5754	0.5421	0.8406	0.6591	0.5421	0.8310	0.6561	0.5458	0.8368	0.6606
	std	0.0000	0.0000	0.0000	0.0187	0.0616	0.0215	0.0194	0.0762	0.0363	0.0000	0.0000	0.0000	0.0000	0.0054	0.0017	0.0083	0.0103	0.0063
CPY017	100	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	0.7500	0.8571	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
	500	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
	1000	1.0000	1.0000	**1.0000**	1.0000	0.7500	0.8571	1.0000	0.7500	0.8571	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
	5000	1.0000	1.0000	**1.0000**	1.0000	0.7500	0.8571	1.0000	0.7500	0.8571	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
	10,000	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	0.6667	1.0000	0.8000	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
	avg	1.0000	1.0000	**1.0000**	1.0000	0.9000	0.9428	0.9333	0.8500	0.8743	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**	1.0000	1.0000	**1.0000**
	std	0.0000	0.0000	0.0000	0.0000	0.1369	0.0783	0.1491	0.1369	0.0745	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
YOM009	100	0.6308	0.3254	0.4293	0.5846	0.3115	0.4064	0.5538	0.3303	0.4138	0.6154	0.3226	0.4233	0.6462	0.3281	**0.4352**	0.5846	0.3065	0.4021
	500	0.4769	0.4769	0.4769	0.6154	0.3960	0.4819	0.3538	0.4694	0.4035	0.4769	0.4769	0.4769	0.6000	0.4333	**0.5032**	0.6154	0.4082	0.4908
	1000	0.5538	0.3600	0.4364	0.5538	0.3186	0.4045	0.4769	0.3875	0.4276	0.5231	0.3778	**0.4387**	0.4923	0.3299	0.3951	0.5538	0.3396	0.4211
	5000	0.5538	0.4091	0.4706	0.4923	0.4384	0.4638	0.4462	0.4531	0.4496	0.6308	0.3981	**0.4881**	0.5538	0.3789	0.4500	0.5385	0.4430	0.4861
	10,000	0.5385	0.2991	0.3846	0.4923	0.3048	0.3765	0.4154	0.3293	0.3673	0.4154	0.3971	**0.4060**	0.4615	0.3158	0.3750	0.5538	0.3186	0.4045
	avg	0.5508	0.3741	0.4396	0.5477	0.3539	0.4266	0.4492	0.3939	0.4124	0.5323	0.3945	**0.4466**	0.5508	0.3572	0.4317	0.5692	0.3632	0.4409
	std	0.0548	0.0707	0.0371	0.0550	0.0599	0.0443	0.0741	0.0661	0.0305	0.0914	0.0554	0.0350	0.0757	0.0489	0.0500	0.0308	0.0595	0.0440

We can see that, on the CPY011 dataset, the ensemble of the top three models (E3) earned the greatest F1-score of 0.9231 with every epoch of training. On CPY012, the greatest F1-score of 0.8438 was obtained by E5 and WE7 with models trained with 10,000 epochs, and E7 with models trained with 500 epochs, while E3 and WE3 models trained with 10,000 epochs performed the best in identifying anomalies on the CPY013 dataset. With the CPY014 dataset, all ensemble models gave an F1-score of 0.8571, with the exception of the ensemble with majority voting of all seven models trained with 10,000 epochs, which performed badly with an F1-score of 0.6667. WE7 surpassed other ensemble models on the CPY015 and CPY016 datasets, with the greatest F1-score of 0.4615 and 0.6704, respectively. Every ensemble model on CPY017 produced outstanding results with a 1.0000 F1-score, particularly E3, WE3, WE5, and WE7,

which produced excellent results with all training epochs. The weighted ensemble with 5 models (WE5) trained with 500 epochs performed the best on the YOM009 dataset, with a 0.5032 F1-score.

Table 5.12: The mean F1-score and standard deviation of all models when testing with the dataset from different stations (the best F1-score of each station is shown in bold).

Models	CPY011	CPY012	CPY013	CPY014	CPY015	CPY016	CPY017	YOM009
DRL	0.7433 (±0.08)	0.6550 (±0.08)	0.6014 (±0.10)	0.4823 (±0.20)	0.3477 (±0.06)	0.3770 (±0.03)	0.6801 (±0.21)	0.3581 (±0.06)
DRL_{F1}	0.7170 (±0.07)	0.7146 (±0.06)	0.6963 (±0.07)	0.6733 (±0.08)	**0.4134 (±0.02)**	0.5438 (±0.05)	0.6381 (±0.16)	0.4152 (±0.04)
$DRLRwd$	0.7433 (±0.08)	0.6468 (±0.10)	0.6212 (±0.09)	0.5537 (±0.26)	0.3477 (±0.06)	0.3499 (±0.06)	0.6762 (±0.21)	0.3891 (±0.04)
DRL_{Acc}	0.7170 (±0.07)	0.7234 (±0.04)	0.6920 (±0.07)	0.6733 (±0.08)	**0.4134 (±0.02)**	0.5714 (±0.02)	0.7219 (±0.14)	0.4248 (±0.04)
DRL_{Valid}	0.6020 (±0.17)	0.7045 (±0.10)	0.5639 (±0.21)	0.6596 (±0.14)	0.3211 (±0.10)	0.4867 (±0.07)	0.6488 (±0.15)	0.4378 (±0.03)
MLP	0.8505 (±0.06)	0.7822 (±0.03)	0.6998 (±0.03)	**0.8571 (±0.00)**	0.2220 (±0.10)	0.5651 (±0.14)	0.9778 (±0.07)	0.2358 (±0.05)
LSTM	0.8167 (±0.04)	0.7753 (±0.02)	0.7265 (±0.02)	**0.8571 (±0.00)**	0.3276 (±0.09)	0.6252 (±0.06)	0.9857 (±0.05)	0.2596 (±0.06)
$EDRL_3$	0.7700 (±0.06)	0.7367 (±0.02)	0.7033 (±0.07)	0.6948 (±0.11)	**0.4134 (±0.02)**	0.5544 (±0.03)	0.7809 (±0.10)	0.4396 (±0.04)
$EDRL_5$	0.7455 (±0.09)	0.7558 (±0.03)	0.6819 (±0.03)	0.7295 (±0.12)	0.3605 (±0.07)	0.5123 (±0.03)	0.7131 (±0.24)	0.4301 (±0.05)
E3	**0.9231 (±0.00)**	**0.8387 (±0.00)**	**0.7600 (±0.02)**	**0.8571 (±0.00)**	**0.4134 (±0.02)**	**0.6629 (±0.00)**	**1.0000 (±0.00)**	**1.0000 (±0.04)**
E5	0.8189 (±0.08)	0.7914 (±0.05)	0.7437 (±0.03)	**0.8571 (±0.00)**	0.3739 (±0.05)	0.5947 (±0.02)	0.9428 (±0.08)	0.9428 (±0.04)
E7	0.8360 (±0.05)	0.7990 (±0.03)	0.7301 (±0.03)	0.8190 (±0.09)	0.3493 (±0.03)	0.5754 (±0.04)	0.8743 (±0.07)	0.8743 (±0.03)
$WEDRL_3$	0.7795 (±0.05)	0.7432 (±0.03)	0.7100 (±0.07)	0.6962 (±0.16)	**0.4134 (±0.02)**	0.5797 (±0.01)	0.7976 (±0.09)	0.4466 (±0.03)
$WEDRL_5$	0.7540 (±0.05)	0.7347 (±0.04)	0.7194 (±0.05)	0.6714 (±0.16)	0.3954 (±0.04)	0.5525 (±0.04)	0.7619 (±0.10)	0.4323 (±0.05)
WE3	0.8571 (±0.00)	0.8254 (±0.00)	0.7262 (±0.04)	**0.8571 (±0.00)**	**0.4134 (±0.02)**	0.6591 (±0.00)	**1.0000 (±0.00)**	**1.0000 (±0.03)**
WE5	0.8571 (±0.00)	0.8254 (±0.00)	0.7240 (±0.02)	**0.8571 (±0.00)**	0.3895 (±0.04)	0.6561 (±0.00)	**1.0000 (±0.00)**	**1.0000 (±0.05)**
WE7	0.8571 (±0.00)	0.8291 (±0.01)	0.7195 (±0.01)	**0.8571 (±0.00)**	0.3784 (±0.06)	0.6606 (±0.01)	**1.0000 (±0.00)**	**1.0000 (±0.04)**

As indicated in Table 5.12, we averaged the F1-score of each individual model and ensemble model to compare their performance. We can observe that E3 not only performed the best model with the greatest average F1-score on all datasets but also excellently performed with a 1.0000 F1-score on CPY017 and YOM009. Among the models tested on the CPY014 dataset, the best F1-score of 0.8571 was achieved by MLP, LSTM, E3, E5, WE3, WE5, and WE7. In contrast, on the CPY015 dataset, the model with DRL-based ($DRL_{F1}, DRl_{Acc}, EDRL_3$, and $WEDRL_3$) generated the highest F1-score of 0.4134.

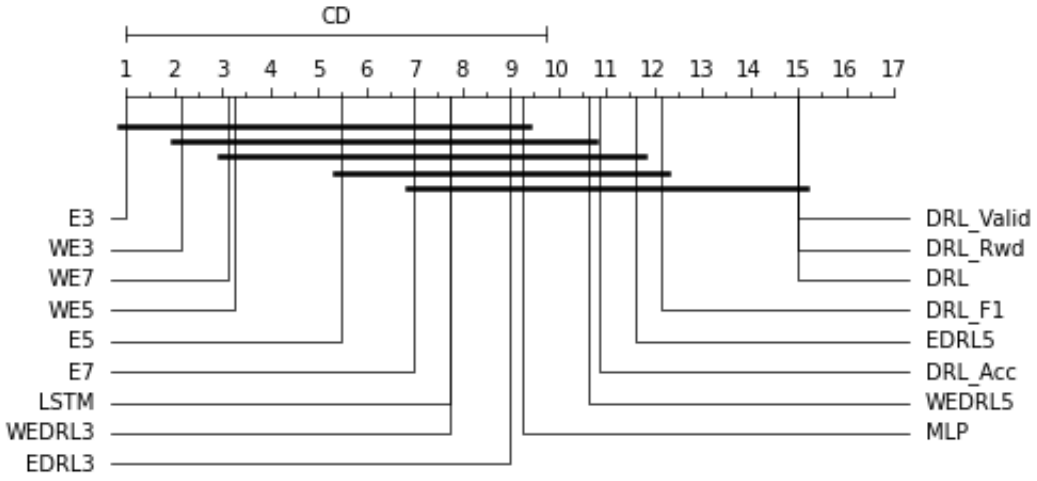


Figure 5.10: A critical difference diagram of all models when testing with the dataset from different stations.

Furthermore, as shown in Figure 5.10, the CD diagram was chosen to make a statistical comparison of our results, which revealed that E3 had the highest ranking, and the ensemble model that combined all seven individual models outperformed both the individual model and the ensemble model created using DRL models. It also demonstrated the ability of ensemble methods to improve the performance of individual DRL models because it represented a significant difference from individual models (DRL, DRL_{Rwd}, and DRL_{Valid}).

Table 5.13: The F1-score of the E3 models when testing with the dataset from the same station (shown in the bracket) and different stations.

Tested Dataset	**Trained Dataset**							
	CPY011	**CPY012**	**CPY013**	**CPY014**	**CPY015**	**CPY016**	**CPY017**	**YOM009**
CPY011	(0.9231)	0.4000	0.3222	0.7778	0.4028	0.1373	0.7059	0.6087
CPY012	0.7183	(0.8387)	0.8060	0.7500	0.1508	0.6923	0.7273	0.7857
CPY013	0.6920	0.7350	(0.7600)	0.7000	0.2117	0.7123	0.6909	0.7719
CPY014	0.6895	0.8635	0.8571	(0.8571)	0.5714	0.6000	0.8571	0.8571
CPY015	0.3444	0.2192	0.2093	0.2593	(0.4132)	0.1420	0.3077	0.2414
CPY016	0.5840	0.6398	0.6603	0.6322	0.3495	(0.6629)	0.5665	0.6580
CPY017	0.7600	0.6667	0.6000	1.0000	0.3651	0.2857	(1.0000)	1.0000
YOM009	0.1017	0.2948	0.3221	0.2908	0.0437	0.4500	0.3230	(0.4396)
avg	0.6016	0.5822	0.5671	0.6584	0.3135	0.4603	0.6473	**0.6703**
std	0.2603	0.2468	0.2496	0.2607	0.1682	0.2432	0.2412	0.2415

We then tested the generalisation ability of ensemble models with the data collected from different stations. The F1-score of each station is depicted in Table 5.13. Using ensemble E3 with CPY011 data, to identify anomalies from other stations, we can see that it works well with F1-score of more than 0.5800, but it performed poorly at detecting anomalies on CPY015 and YOM009 with F1-score of 0.3444 and 0.1017, respectively. E3 on CPY012 performed well when detecting anomalies on CPY014 with a 0.8635 F1-score. Similarly, E3 on CPY013 provided a higher F1-score on their own dataset when detecting anomalies on CPY012 and CPY014 with an F1-score of 0.8060 and 0.8571, respectively. The best ensemble on CPY014 generated excellent performance when identifying anomalies on CPY017 data. In contrast, E3 on CPY015 performed poorly on YOM009 with an F1-score of only 0.0437. While considered E3 on CPY016, although it provided good performance with an F1-score higher than 0.6 on CPY012, CPY013, and CPY014, it performed poorly on CPY011, CPY015, CPY017, and YOM009 with an F1-score lower than 0.45. E3 on CPY017 provided good results with an F1-score of more than 0.69, except on CPY015, CPY016, and YOM009 with an

F1-score lower than 0.56. Meanwhile, E3 on YOM009 generated an F1-score on its own of only 0.43, but it performed excellently when detecting anomalies on CPY017 and other datasets with an F1-score higher than 0.65, except on CPY015 with an F1-score of 0.2414.

5.5 Discussion

We can see that some of the results obtained have an F1-score of 0.5 or less. The reason for this is that the number of anomalies present in the training set is smaller compared to the number of anomalies in the validation and testing sets. For instance, in the case of CPY015, the ratio of anomalies in the training, validation, and testing sets is 0.14, 0.88, and 1.98, respectively. This means that the models might not have enough information about the anomalies to learn from the training set, which could result in lower performance on the testing set. Moreover, in the cases of CPY016 and YOM009, the data shows high fluctuations, which can make it difficult for the models to identify anomalies accurately. As a result, the performance of the models might be affected, leading to lower F1-score. However, when the number of training epochs increases, the performance of each model grows or decreases in each epoch, then drops and bounces back. This might indicate that our model is still learning or is learning too much—that is, it is difficult to decide when it is time to stop training.

Even though DRL can do better than other models, it is time-consuming—at least 50 times slower than MLP models on average—because we have to train it until it performs well enough and we cannot predict how long that will take. The size of the windows must also be taken into account. A larger window size takes more time than a smaller window size. The window size has an effect on the comparison of data in windows to identify the anomaly. Additionally, we may add additional neural networks to improve the accuracy of our technique, but training will take longer.

DRL does better than other models when it is trained on datasets with a low number of outliers. This proves the ability to detect unknown anomalies. However, its performance is insufficient, which may be due to an imbalance in our dataset. As a result, models may lack sufficient information to explore and leverage knowledge for adaptive

detection of unknown abnormalities.

Moreover, the neural structure that works well with one station may not function well with another. Hence, the problems of this topic include determining the suitable neural structure for each station. Furthermore, the primary parameter that requires further attention is the reward function, since a suitable reward will impact the model's learning process.

In the case of ensemble models, when all of the individual models in an ensemble perform similarly, majority voting is the best method for determining the final decision. However, when the accuracies of individual models are different, the weighted voting is the best way to utilise the strengths of the good models in making a decision. Furthermore, the ensemble model can also reduce the false alarm rate, as seen by an increased precision score. It should be noted that, although single models performed well on certain stations, they did poorly on others, such as the LSTM model. As a result, we cannot rely on a single model since we do not know if it is the best or not. The ensemble models, on the other hand, are more reliable, even though they may not produce the best accuracy for every station. On the whole, nevertheless, most ensembles, such as $WEDRL_3$ performed consistently very well and their accuracies are always ranked highly at every station, whilst the individual models: DRL, MLP and LSTM, are not consistent through out all the stations.

5.6 Summary

In this research, we firstly investigated how deep reinforcement learning (DRL) can be applied to detect anomalies in water level data and then devised two strategies to construct more effective and reliable ensembles. For DRL, we defined a reward function as it plays a key role in determining the success of an RL. We developed ensemble models with five deep reinforcement learning models, generated by the same DRL algorithm but with different criteria of performance measurement. We tested our ensemble approach on telemetry water level data from eight different stations. We compared our approach to two different neural network models. Moreover, we demonstrate the ability to detect unknown anomalies by using the trained model to

detect anomalies from other stations' data.

The results indicate that DRL_{Acc} models are the best individual DRL models, but they performed slightly poor than LSTM. When tested on different stations, LSTM still does better than others, but its accuracy is not satisfactory. When compared to an ensemble approach, LSTM was more accurate in some stations than other ensembles with DRL models, but less accurate in some others. On the whole, the statistical results from the CD diagram showed that our ensemble approach with only 3 members of DRL models, $WEDRL_3$, was superior. Furthermore, all ensemble models that were combined by selecting models from 5 DRL models, MLP, and LSTM outperformed both the best individual model, LSTM, and the best ensemble using DRL models, $WEDRL_3$. This is supported by the highest F1-score and rankings with the CD diagram. It is clear that ensemble methods not only increased the accuracy of a single model but also provided a higher reliability of performance.

In conclusion, DRL is applicable for detecting anomalies in telemetry water level data with added benefit of detecting unknown anomalies. Our ensemble construction methods can be used to build ensemble models from selected single DRL models in order to increase the accuracy and reliability. In general, the ensembles are consistent in producing more accurate classification, although they may not always achieve the best results. Moreover, they are superior in reducing the number of false alarms in identifying abnormalities in water level data, which is very important in real application.

However, using just water level as a single feature in our dataset may not be enough for the model to learn and correctly recognise abnormal data. Thus, feature extraction is needed to extract important information from telemetry water level data. As a result, the next step in our research will be to develop effective strategies for extracting an important feature from telemetry water level data. The work done on feature extraction will be explained in the next chapter.

Chapter 6

New Multiple Features Extraction Method for Anomaly Detection.

6.1 Introduction

Our dataset comprises only one feature, which is the water level, and this may not provide sufficient information for the model to identify anomalies in the data. Therefore, feature extraction is necessary to extract useful information from the telemetry water level data. However, existing feature extraction methods often rely on dividing the data into windows to calculate new features, and this may not always be appropriate. Selecting an unsuitable window size for feature extraction can result in irrelevant features being extracted, which can negatively affect the model's performance. Furthermore, it is critical to consider the time taken for feature extraction since detecting anomalies promptly is essential.

Moreover, most existing feature extraction methods are generic and may not be optimized to handle the unique characteristics of telemetry water level data, which is continuously streaming and dynamic. Furthermore, such methods require the entire dataset to be available before feature extraction can occur, which may not be feasible for real-time or near real-time data. Therefore, there is a need for a feature extraction method that can operate on streaming data and can handle large volumes of data.

To address these challenges, we have developed a new feature extraction approach that is specifically designed for telemetry water level data. This approach allows us to extract relevant features from the data, enabling the model to identify anomalies

accurately and quickly even in a continuously streaming or real-time environment. We have also introduced a weighted ensemble method that can enhance the model's performance.

6.2 Methods

Our study has identified that existing feature extraction methods have significant drawbacks that make them unsuitable for anomaly detection in telemetry water level data. These methods tend to be time-consuming and extract too many features. To overcome these limitations, we propose a new feature extraction approach that is specifically designed for this application. Our approach combines the saliency map (SM) method with our newly developed nearest neighbour feature extraction (NNFE) to create a new method called SM+NNFE. By using sliding windows with only two members and incorporating the SM method, we can extract essential features from telemetry water level data that are crucial for accurate anomaly detection. The SM+NNFE is a new approach, and it has been designed to enhance the accuracy of anomaly detection in telemetry water level data.

6.2.1 Saliency Map

In anomaly detection, we are looking for data point that is different from the expected value (Gupta et al., 2014). The Saliency Map (SM) method (Hou and Zhang, 2007; Cui et al., 2009; Zhang et al., 2021a), developed for discovering the salient parts of a picture in computer vision, can be adapted for this purpose. It can be achieved by transforming data from the time domain to the frequency domain and calculating the spectral residual. After that, we convert it back to a time domain to get the saliency map. The SM method has demonstrated effectiveness and success not only in detecting image saliency but also in detecting anomalies in time series data (Ren et al., 2019). Therefore, we can leverage this method for anomaly detection in telemetry water level data.

All of the steps for generating a saliency map for a given series of water level data: $x = x_1, x_2, ..., x_n$ are detailed below.

i) Convert x to frequency domain by using Fourier transform ($\mathfrak{F}$). The results comes with both of real $R(f)$ and imaginary $I(f)$ parts.

$$\begin{aligned} F(\alpha) &= \mathfrak{F}(x) \\ R &= \Re(F(\alpha)) \\ I &= \Im(F(\alpha)) \end{aligned} \tag{6.2.1}$$

ii) Calculate the amplitude $A(f)$ of the Fourier spectrum.

$$A = \sqrt{R^2 + I^2} \tag{6.2.2}$$

iii) Find the log amplitude spectrum $\mathcal{L}(f)$.

$$\mathcal{L} = log(A) \tag{6.2.3}$$

iv) Eliminate the redundant component by subtracting with the local average spectrum $\overline{\mathcal{L}(f)}$ to get the spectral residual $\mathcal{SR}(f)$.

$$\mathcal{SR} = \mathcal{L} - \overline{\mathcal{L}(f)} \tag{6.2.4}$$

v) After removing the redundant component we have to update the values of real and imaginary by

$$\begin{aligned} R &= \frac{R \times e^{\mathcal{SR}}}{A} \\ I &= \frac{I \times e^{\mathcal{SR}}}{A} \end{aligned} \tag{6.2.5}$$

vi) Finally, using the inverse Fourier Transform ($\mathfrak{F}^{-1}$), reverse the data back to the time domain to create a saliency map $S(x)$.

$$S(x) = ||\mathfrak{F}^{-1}(F(\alpha))|| \tag{6.2.6}$$

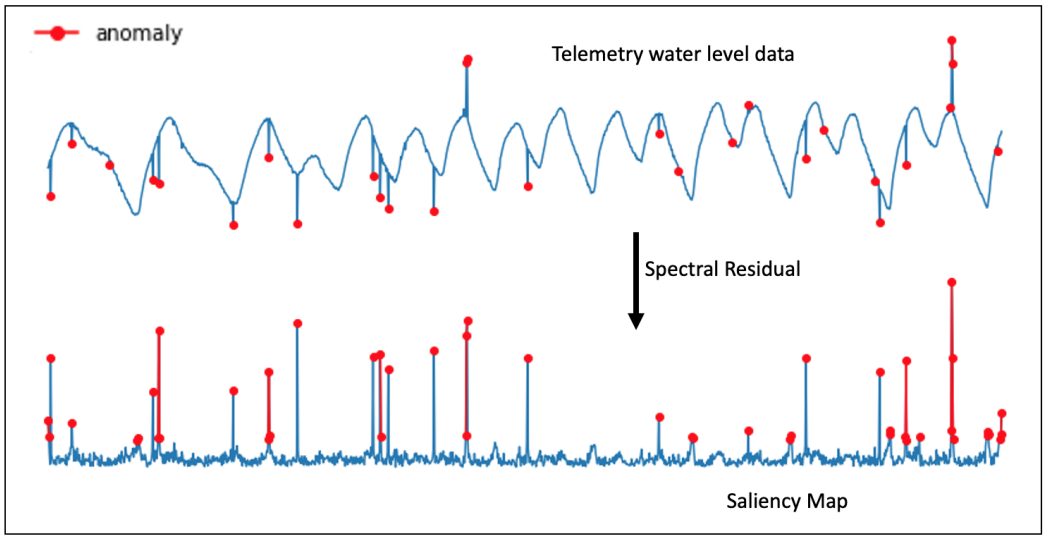


Figure 6.1: An Example of the Saliency Map

Figure 6.1 gives an example of a saliency map. This method is effective in detecting anomalies in time series data by highlighting important data points with higher values. By applying a simple threshold or clustering models, we can identify some of these highlighted values as anomalies and mark their locations. However, SM is designed to highlight only the salient parts of the data, without considering the relationships between data points. As a consequence, it has a limitation in detecting anomalies with small deviations from normal values, which are common in long periods of datasets.

In situations where the anomaly is situated at the center of the sliding window, the SM can effectively identify it because the anomaly point will be significantly different from both the preceding and succeeding values. However, in cases where the difference between the previous and most recent water level is larger than usual, such as during the rainy season or just prior to a flood, the SM values of normal data during that time may be significantly different from other times, leading to false identifications of anomalies. Therefore, relying solely on a SM to detect anomalies may not be sufficient for accurately detecting all types of anomalies in the data.

To overcome the limitations of the SM method and improve the accuracy of anomaly detection, it is necessary to combine it with other features that can capture the relationships between data points and provide a more comprehensive representation of the data.

6.2.2 Nearest Neighbour Features Extraction (NNFE)

To detect anomalies in time series data, it is common to compare a data point with its preceding value. A significant change from the previous value is more likely to indicate an abnormality. Thus, we can derive features that describe the data based on this comparison. In this study, we derived four types of features by comparing the two nearest time series values: difference, trend, angle, and percent change. The difference feature simply measures the numerical difference between the two data points. The trend feature indicates whether the data is increasing, decreasing, or remaining constant over time. The angle feature calculates the angle between the line connecting the two data points and the horizontal axis, which helps capture non-linear trends. Finally, the percent change feature measures the percentage change between the two data points relative to the overall magnitude of the data. These four types of features are calculated as follows:

i) Difference:

$$\Delta x = x_t - x_{t-1}$$

ii) Trend:

$$trend = \begin{cases} 1, & \text{if } x_{t-1} < x_t \\ 0, & \text{if } x_{t-1} = x_t \\ -1, & \text{otherwise} \end{cases}$$

iii) Angle:

$$\theta = tan^{-1}(x_t - x_{t-1})$$

iv) Percent Change:

$$\%change = 100 \times \frac{(x_t - x_{t-1})}{x_{t-1}}$$

This approach is beneficial as it has several advantages. First, it reduces the number of features extracted from the data, which helps reduce the computational cost and complexity of the anomaly detection model. Second, using data from only two time points allows for a more focused analysis of the data as it emphasises the changes between consecutive time points, which are likely to be the most informative for detecting

anomalies.

6.2.3 SM+NNFE

Although the saliency map and nearest neighbour feature extraction methods can be used for identifying anomalies in time series data, each method has its own strengths and limitations. Using only one of them may not be sufficient to accurately detect all types of anomalies. Therefore, we proposed a hybrid method called SM+NNFE that combines the features extracted from both methods. This new method uses five features in total: saliency map, difference, trend, angle, and percent change.

By combining these features, the SM+NNFE method may improve the accuracy and effectiveness of anomaly detection. For example, the SM feature is effective in detecting anomalies with significant deviations from normal values, while the NNFE can capture non-linear trends, which may be missed by other methods.

6.3 Evaluation

6.3.1 Datasets

We chose the similar dataset from 8 stations that have been used in Chapter 5 which has been summarised in Table 5.1. After doing data pre-processing, we then use our combined feature extraction method to extract five features. For each of the datasets of water level data, the data from each station is divided into three subsets: a training set 60%, a validation set 20%, and a testing set 20% using a stratified statistical sampling technique.

Water level data from telemetry stations is unlabelled in the first place. To evaluate the accuracy of our models in this study, we asked some experts at the HII to look at the data to identify and label the anomalies.

6.3.2 Evaluation Models

To evaluate the performance of our approach, we employ five popular basic machine learning methods for anomaly detection: Decision Tree (DT), k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), Gaussian Naive Bayes(GNB), and Multi-Layer Perceptron (MLP). Moreover, The selected classifiers are initially added to a pool, where they serve as candidates for the ensembles being constructed. The ensembles are constructed by choosing a specific number of classifiers from the pool, and although they are built using the same methodology, they employ different decision-making strategies (simple majority voting and weighted voting), which are outlined in Chapter 5. In order to examine whether the size of the ensemble has an impact on accuracy, we constructed our ensemble models with variable sizes of 5 and 3 models.

6.3.3 Evaluation Metrics

In order to evaluate the accuracy of classifiers for individual models and developed ensembles, in this phase, we employ measures of $precision, recall$ and $F1-score$. They are calculated with the equations, as described in Section 3.6, based on a confusion matrix (see Table 3.1).

6.3.4 Experiment Setting

The Scikit-learn Python library was used as it contains the implementation of the chosen machine learning algorithms. In our experiments, the default parameters of these learning algorithms were used, as our focus was not to optimise these algorithms.

For comparison, we also extracted features from these water level datasets using our suggested techniques, SM+NNFE, and the existing state-of-the-art time series feature extraction methods provided in the library TSFRESH (Christ et al., 2018). The SM+NNFE can extract 5 features, whereas TSFRESH can extract 787 features, which is too many. So we applied a feature selection function[1] of TSFRESH to pick only the relevant features, reducing the number of features from 787 to 137.

[1] `https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_selection.html`

We now have five sets of data: (1) WL contains only the water level data with standardisation; (2) SM contains only the saliency map feature; (3) NNFE contains 4 features that have been extracted by NNFE; (4) SM+NNFE contains the SM feature and the features extracted by NNFE, 5 features in total; (5) TSFRESH, contains 137 features. They are used separately in training the classifiers to compare which set of features are more effective in detecting anomalies. For each setting, the experiment were repeated 20 times with different random seeds in order to test the consistency of the classifiers.

It should be noted that we used the default parameters from the Scikit-Learn package without doing any hyperparameter tuning because we wanted to compare each technique under the same conditions.

6.4 Results

The average testing results of the 20 experiments from traditional models and ensemble models for each experimental setting are reported as follows.

Results of individual models

Five different models were trained on a specific set of data, and their performance was assessed using a different datasets. The findings of this study are presented in Table 6.1, which displays the average results of 20 experiments conducted. The outcomes show that without any feature extraction, the average of F1-score, precision, and recall of water level data are 0.3199, 0.6122, and 0.3215, respectively. While assessing individual models, the decision tree (DT) model showed the highest F1-score of 0.3765, whereas the Gaussian Naive Bayes (GNB) model displayed the lowest score of 0.2354. All models showed low recall, indicating that they could only detect a few anomalies in each dataset. In other words, training a model on only water level data may not provide enough information to identify anomalies effectively.

Utilising feature extraction with SM on water level data resulted in an average increase of 25% in the F1-score of all models. The most substantial improvement was observed in the MLP model, which showed a 30% improvement, while the DT model had the smallest improvement of 16%. In addition, all models displayed an increase in recall,

indicating that their ability to detect anomalies was enhanced with greater precision compared to using only water level data.

The use of NNFE as a feature extraction method resulted in an increase of almost 23% in the F1-score of all models compared to models with no feature extraction. The k-NN model demonstrated the most significant improvement, rising approximately 33%. Moreover, not only did the average F1-score improve, but the standard deviation also decreased, suggesting a higher degree of consistency and reduced variability compared to using solely water level data as an input for each anomaly detection model.

Table 6.1: Average performance scores of five machine learning models, trained with three different sets of features: Saliency Map (SM), our new feature set, SM+NNFE, and the baseline feature set, TSFRESH.

Methods	Avg. extracting time(sec.)	Model	F1-Score	Precision	Recall
WL	-	DT	0.3765	0.6346	0.2921
		k-NN	0.3477	0.6084	0.2720
		SVM	0.3257	0.6049	0.2478
		GNB	0.2354	0.6082	0.1688
		MLP	0.3142	0.6049	0.2376
		Avg	0.3199	0.6122	0.3215
		Std	0.2985	0.4411	0.3149
SM	0.014	DT	0.5383	0.7035	0.4514
		k-NN	0.5943	0.7692	0.5148
		SVM	0.6069	0.7998	0.5232
		GNB	0.5412	0.4748	0.7344
		MLP	0.5603	0.7773	0.4709
		Avg	0.5683	0.7049	0.5389
		Std	0.2244	0.2341	0.2656
NNFE	0.043	DT	0.6968	0.7765	0.6498
		k-NN	0.6738	0.8026	0.5888
		SVM	0.5116	0.8134	0.3984
		GNB	0.4184	0.3900	0.5593
		MLP	0.6182	0.8162	0.5203
		Avg	0.5838	0.7197	0.5433
		Std	0.2145	0.2540	0.2195
SM+NNFE	0.072	DT	0.7374	0.7795	0.7028
		k-NN	0.7268	0.8700	0.6317
		SVM	0.6515	0.9190	0.5365
		GNB	0.5377	0.4593	0.7305
		MLP	0.7147	0.8618	0.6272
		Avg	0.6736	0.7779	0.6658
		Std	0.1641	0.2067	0.1903
TSFRESH	245	DT	0.7425	0.7699	0.7234
		k-NN	0.7260	0.8633	0.6422
		SVM	0.6010	0.9781	0.4654
		GNB	0.3099	0.2052	0.8656
		MLP	0.7865	0.8719	0.7274
		Avg	0.6332	0.7377	0.6848
		Std	0.2378	0.3090	0.2121

Combining SM with NNFE resulted in a substantial increase in the average score for all metrics, with the most significant improvement observed in recall and the F1-score, which doubled compared to no feature extraction. Additionally, this approach yielded the lowest average standard deviation among all strategies.

The application of TSFRESH as a feature extraction technique resulted in a noteworthy increase in the average F1-score of all models by 0.31, with the MLP model showing the most substantial improvement of 0.47 when compared to no feature extraction. Moreover, the average F1-score and recall significantly increased from 0.3199 to 0.6332 and 0.3215 to 0.6848, respectively.

When comparing different feature extraction techniques, models utilising NNFE outperformed models utilising SM by approximately 0.2 in terms of the average F1-score. Additionally, NNFE provided a lower standard deviation, indicating higher consistency and reduced variability. When SM and NNFE were combined, the average F1-score increased by approximately 0.10 when compared to NNFE and SM used individually.

Although TSFRESH has a higher average F1-score than that of WL, SM, and NNFE, its average F1-score is lower when compared with SM+NNFE. Moreover, TSFRESH was too time-consuming to extract features and took about 4 minutes, whereas SM, NNFE, and SM+NNFE only took less than one second. So, TSFRESH is not only less accurate but also much less efficient, about several hundred times slower than the other two methods.

The critical difference diagram in Figure 6.2 summarises the final results of SM+NNFE against the 4 feature extraction techniques. When we consider only the group of our proposed feature extraction techniques, we can observe that WL has the lowest performance, and there is no difference between SM and NNFE, but when they are combined, they are significantly higher ranked than one separate methodology. When compared to the existing feature extraction algorithm, TSFRESH outperforms WL and SM, but it still has a lower ranking than SM+NNFE.

Table 6.2: Average performance score with ensemble models.

Methods	Score	5 Models		Top 3 Models	
		Majority	Weighted	Majority	Weighted
WL	F1	0.3142	0.3587	0.3218	0.3678
	Precision	0.6049	0.6785	0.6049	0.6785
	Recall	0.2376	0.2696	0.2442	0.2765
SM	F1	0.5609	0.6124	0.5757	0.6291
	Precision	0.8820	0.7045	0.8355	0.6660
	Recall	0.4618	0.5850	0.4796	0.6789
NNFE	F1	0.5925	0.7080	0.5920	0.7192
	Precision	0.9176	0.8675	0.9102	0.8608
	Recall	0.4622	0.6067	0.4684	0.6327
SM+NNFE	F1	0.7135	0.7825	0.6852	0.7701
	Precision	0.9341	0.8888	0.9581	0.8499
	Recall	0.5932	0.7061	0.5608	0.7102
TSFRESH	F1	0.7239	0.8018	0.6919	0.8124
	Precision	0.9640	0.9040	0.9870	0.9234
	Recall	0.5961	0.7291	0.5580	0.7389

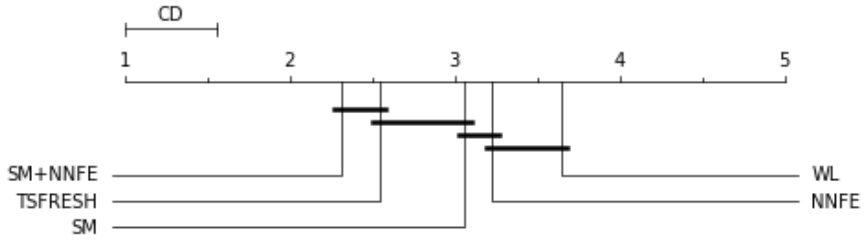


Figure 6.2: The critical difference diagram for 3 different feature extraction techniques against the existing feature extraction technique (TSFRESH) and the telemetry water level data before applying any feature extraction technique (WL) to the telemetry water level.

Results of ensembles

As mentioned earlier, our ensemble methods combine the outputs of member models with different weights, which are computed by our function defined in equation 3.5.1. Basically, it uses the validation data set to evaluate the models, then uses their F1-score and ranking positions to determine the weight score for each model. The experimental results of the ensembles are presented in Table 6.2.

Without applying any feature extraction approach (WL), or applying SM or NNFE as feature extraction, the average performance of majority voting ensemble models is not different from the average performance before combining models. While it has improved by around 5% by weighted voting ensemble models. In the case of SM+NNFE, their performance has improved by around 10% with weighted ensemble techniques. When

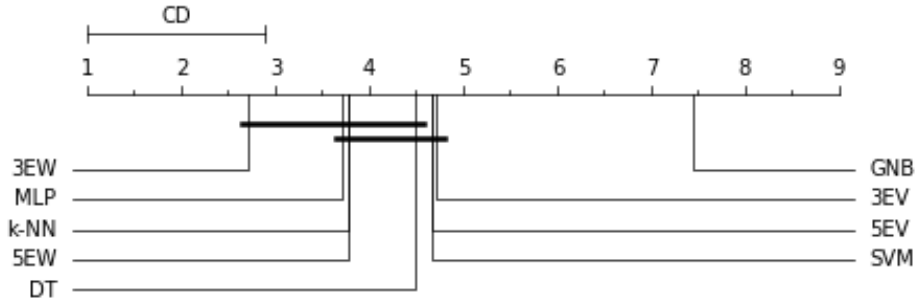


Figure 6.3: The critical difference diagram for four different ensemble models versus five individual models on telemetry water level. Ensemble generated the highest ranked.

compared to TSFRESH, it has considerably improved with both ensemble methodologies and has the greatest F1-score from a weighted voting ensemble among the top 3 models at 81.24%.

Figure 6.3 depicts a comparative analysis of the critical difference between distinct individual models and ensembles. The results indicate that the weighted ensemble of the best three models (3EW) generally exhibits the highest performance, although its advantage is not always statistically significant when compared to a few other models. On the other hand, the individual model GNB shows the lowest accuracy amongst all models. When examining the ensembles exclusively, the weighted ensemble models perform better than the majority voting ensemble models. It is important to note that the simple ensembles (3EV and 5EV) rank lower than all individual models except GNB. This suggests that a mere combination of models without proper consideration may not enhance accuracy. Conversely, ensembles built with appropriate models and their outputs combined with a suitable weighted function can significantly improve accuracy and consistency over the individual models.

Discussion

Only telemetry water level data provides insufficient for machine learning models to accurately detect anomalies, as the results from many anomaly detection model provided low accuracy. While SM that convert the data to frequency domain to represent the salient part which could be the anomaly in our datasets. However, in some event the difference between the previous and most recent reported water levels is greater than usual, such as rainy season or before a flood, which difficult for models to classify.

NNFE generates new features by computing the difference between neighbouring data points to highlight the significant dissimilarities. However, it still has the same issue with SM approach.

TSFRESH extracts features by extracting features across multiple domains (such as time domain and frequency domain) to obtain a wide range of features suitable for diverse tasks. Although extracting the feature can give new insights into the time series and their dynamics which models may achieved the better results, adding redundant variables reduces the model's generalisation capability and may also reduce the overall accuracy of a classifier. Moreover, This extensive feature extraction process can be time-consuming , as seen on the time for extracting features with TSFRESH are significantly higher than other candidate techniques. Additionally, the process for selecting the optimal features for a particular application also needs more time.

With the combination of SM and NNFE can address those issue and improved the performance of each model. Moreover, our suggested approach generates new features by computing the difference between neighbouring data points to highlight the significant dissimilarities, combined with the saliency map, improve the model accuracy to detect anomalies. While simultaneously reducing time costs, as it only employs two members and performs basic computations.

Furthermore, the accuracy of anomaly detection can be improved by utilizing ensemble techniques. However, the decision-making method employed is a crucial factor that significantly impacts the performance of the ensemble model. As we can see, there is no variation in ensemble performance with varying numbers of model members. However, when our weighted voting strategy is used in decision-making, the accuracy has been noticeably improved when water levels changed rapidly from some events.

6.5 Summary

In this experiment, we introduced a new feature extraction approach that combined the saliency map and the nearest neighbour feature extraction to extract significant features for the model to learn. As a result, the models are more diverse from each other and

thus reduce their chance of making the same errors, which leads to an improvement in their performance. In addition, we proposed a weighted ensemble approach to combine the models selected for building an ensemble with different weights so that the good models contribute more than the poor models in determining the final output of the ensemble.

We used an existing library, TSFRESH, for extracting features from time series data as a baseline for comparison. Five types of models have been trained with different feature sets: SM, NNFE, WL, SM+NNFE, and TSFRESH. Then some of these trained models were selected based on their accuracy on the validation data to build an ensemble. We devised a weighting function to determine the weight for each model in an ensemble. We ran several series of experiments using data from eight water level data sets. Each experiment was repeated 20 times with different partitions of the data to investigate the consistency of the models. The accuracies were measured by *recall, precision* and *F1-score*. The results were evaluated with a statistical significance test.

The results show that the new feature extraction methods, SM+NNFE, do produce more informative and diverse features and represent more underlying information about the water level time series data than the existing feature extraction library, TSFRESH. The ensemble methods are more accurate and reliable compared with the individual models. The weighted ensemble trained with SM+NNFE features outperformed the others in terms of accuracy and consistency. Additionally, our proposed feature extraction methods were significantly more efficient, taking approximately 0.072 seconds, thousands of times faster than TSFRESH (245 seconds), indicating that our methods are suitable for real-time anomaly detection, a critical requirement for an early warning system.

Anomalies are often deleted or excluded from a series of data, which is problematic since many modelling processes need whole data sets to avoid biases or incorrect research conclusions. As a result, the next step in our research will be to develop effective strategies for addressing data mistakes. The work done on anomaly correction will be explained in the next chapter.

Chapter 7

Subsequence Matching Approaches for Data Correction and Imputation

Contributing Publications

- Khampuengson, T. and Wang, W. (2023). Novel methods for imputing missing values in water level monitoring data. *Water Resources Management*, pp.1-28.

7.1 Introduction

Using telemetry stations is a cost-effective approach to automatically collect the hydrological data for monitoring water levels in real time. However, there are several factors that might disrupt the operation of stations such as environmental, technological issues and human activities, and consequently result in anomalous or missing data in the collected water level data. Although, there are many methods, as discussed by Blázquez-García et al. (2020) and Yang et al. (2017), for discovering anomalies and missing data, they often remove anomalies from a series of data, or replace them with some constants, which are problematic because the missing data or inaccurate data can lead to erroneous analysis results. Thus, effective approaches for predicting missing values from accessible data are needed.

The goal of this research is to improve the accuracy and efficiency of correcting missing or anomalous values in time series data, specifically in telemetry surface water level data. We have developed a new technique that utilises the patterns of water levels that tend to repeat themselves over a year. By identifying the most similar subsequence in

the historical data, we can reproduce the pattern and effectively correct the missing data. Our approach involves simple operations for searching and matching patterns, making it an efficient and effective solution for data correction.

7.2 Methods

We recognise that removing anomalies from time series data can create gaps that need to be treated as missing values, and we include them in our imputation approach. This research aims to develop an efficient and effective framework for imputing missing values in water level data, and we propose a novel approach called full subsequence matching (FSM). To evaluate the effectiveness of our approach, we compare it to the traditional approach of partial subsequence matching (PSM), which involves splitting the subsequences before and after the missing gaps. By comparing the two methods, we can demonstrate the advantages of our FSM approach in terms of accuracy and efficiency. Each method is described in detail below:

7.2.1 Full Subsequence Matching (FSM)

In a time series, the data surrounding a missing gap can contain valuable information related to the gap and hence should be used for imputation of the missing points. A key question is how these pieces of useful information can be extracted and utilised in an efficient and effective manner. In this research, rather than separating the subsequence into two parts, before and after the missing gap, we replaced the missing gap with some temporary constant values to construct a dummy full sequence. Then, for parity in searching with historical data, we set the data in each sliding window at the same position as the same replaced constant values in the dummy full sequence. So, we can search for similar subsequences with the subsequences before and after a missing gap at the same time.

Our proposed FSM method consists of four main steps, which are explained as follows:

For a given time series $X = \{x_1, ..., x_N\}$, where N is the length, i.e. the number of data point in a time series.

Step one - Identifying a missing gap: Firstly, we identify the first missing point at time x_t and the last missing point x_{t+T} of a missing gap with T number of consecutive points, $[x_t, ..., x_{t+T}]$.

Step two - Extracting an extended subsequence: We then extract a subsequence I that contains the identified missing gap sandwiched with two subsequences of m and n consecutive data points at the left side and the right side of the gap respectively. This extended subsequence can be represent as:

$$I = \{x_{t-m}, ...x_{t-1}, [x_t, ..., x_{t+T}], x_{t+T+1}, ..., x_{t+T+n}\}$$

We then assign constant values c for every value of missing values in I as follows:

$$I = \{x_{t-m}, ..., x_{t-1}, [c, ..., c], x_{t+T+1}, ..., x_{t+T+n}\}.$$

Step three - Matching: This step searches and matches I with other subsequences in X. It is done by a sliding window technique. We set $W = \{w_1, w_2, ..., w_i\}$ to denote the subsequence in a sliding window where w_i is the set of consecutive values of X at position i with length z. We then compute the Euclidean distance of I with each subsequence in W. However, because the missing values in I have been replaced with constants, before computing the distance, we must replace all values in each subsequence of the sliding windows at the same position as the missing values in I with the same constant c. The most similar subsequence, denoted by S, is the one with the shortest distance, as shown in equation 7.2.1.

$$S = min\{d(I, W)\} \tag{7.2.1}$$

Step four - Imputation : We developed two different techniques to impute missing values: difference imputation and scaling imputation, as shown in Algorithm 2. They are explained further below.

1. *Difference Imputation (FSM_D):* If we know the difference between every two

consecutive values in the any sequence, we can recreate the original series even if some values are missing. We calculate the difference between each pair of consecutive values in S, starting with the first pair of values at the same position of missing data in I. Then addition those value with the first values before the missing gap in I to calculate the first missing values. The difference between the following pairwise values in S is computed and added to the latest imputed values in I. We do so until all missing values have been imputed.

2. *Scaling Imputation (FSM_S):* The scale of the query subsequence and the scale of the most similar subsequence should be the same or almost the same. Hence, we can adjust the scale of the most similar subsequence to the scale of query subsequence to regenerate the values in missing gaps.

Algorithm 2 FSM Imputation

Input: S - the most similar subsequence.
I - query subsequence.
m - the number of points of the right subsequence.
T - the length of missing gap.
procedure FSMD(S, I, m, T)
  $D = \{\}$
  for i=0 to $T-1$ **do**
    $D \leftarrow S[m+1+i] - S[m+i]$ ▷ difference between two consecutive values
  $imp = \{I[m]\}$
  for index i in D **do**
    $val = imp[-1] + D[i]$
    $imp \leftarrow val$
  return imp
procedure FSMS(S, I, m, T)
  $diff = S[m] - I[m]$ ▷ difference between two subsequences
  $max = Max(S[m : m+T+1]) - diff$
  $min = Min(S[m : m+T+1]) - diff$
  $val = I[m : m+T+1]$
  $imp = MinMaxScaler(val, feature\_range = (min, max))$
  return imp

7.2.2 Partial Subsequence Matching (PSM)

The basic idea behind the partial subsequence matching method is that instead of using full subsequences for search and matching, only partial subsequences are used, which could speed up the process. The PSM is explained in detail as follows:

Step one - Identifying a missing gap: This step is the same as that of the FSM, i.e. finding the start and end indices of the missing gap in X.

Step two - Dividing: We then extract subsequence with m points from left (L) and n points from right (R) side of the missing gap in X. That is , we have that

$$L = \{x_{t-m}, ..., x_{t-1}\}$$

and

$$R = \{x_{t+T+1}, ..., x_{t+T+n}\}$$

Step three - Matching: We then search the most similar subsequences to L and R, denoted by S_L and S_R, respectively. It is done by computing the Euclidean distance of L and R with each subsequence in sliding windows W. The most similar subsequences can be represent by

$$S_L = min\{d(L, W)\}$$

and

$$S_R = min\{d(R, W)\}$$

Step four - imputation: Four different techniques have been developed to impute the missing values as represented in Algorithm 3. We use S_L to generate the forward subsequence, and S_R to generate the backward subsequence, then combine those generated subsequences to impute the missing values. The missing values have been imputed by 4 different methods, as follows:

1. *Average Imputation (PSM_A):* We extracted the consecutive subsequence on the right side of S_L, and on the left side of S_R that was the same length as the missing gap. Then, to calculate the difference between each pair of consecutive values, we combined them with the average method before using them to impute missing values.

2. *Forward Imputation (PSM_F):* Instead of using an average difference from both side of the most similar subsequence, we then use only the calculated difference from subsequence on the right side of S_L to impute the missing values.

3. *Backward Imputation (PSM_B):* We used only the calculated difference from the

subsequence on the left side S_R to impute the missing values.

4. *Weighted Imputation (PSM_W):* The basic idea is that the values closest to the missing gap have more effect than the values farthest away. We will assign higher weights to closer points and decrease the weight as the time interval grows. The missing values are then imputed by multiplying the difference between each pair of consecutive subsequences by their weighted score.

Algorithm 3 PSM Imputation

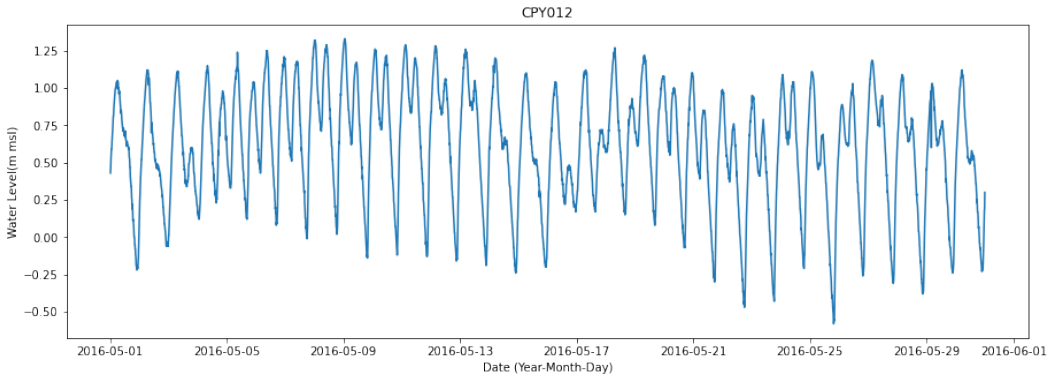
```
        p - last index of S_L
        k - first index of S_R
procedure PSMA(S_L, S_R, m, T, X)
    D_L = {}
    D_R = {}
    for i=0 to T − 1 do
        D_L ← X[p + i] − X[p − 1 + i]
        D_R ← X[k − i] − X[k − 1 − i]
    D_avg = avg(D_L, D_R)
    imp = {X[p − 1]}
    for index i in D_avg do
        val = imp[−1] + D_avg[i]
        imp ← val
    return imp
procedure PSMF(X, p, T)
    D_L = {}
    for i=0 to T − 1 do
        D_L ← X[p + i] − X[p − 1 + i]                ▷ difference between two neighbours
    imp = {X[p − 1]}
    for index i in D_L do
        val = imp[−1] + D_L[i]
        imp ← val
    return imp
procedure PSMB(X, k, T)
    D_R = {}
    for i=1 to T do
        D_R ← X[k − 1] − X[k − 1 − i]                ▷ difference between two neighbours
    imp = {X[k]}
    for index i in D_R do
        val = imp[−1] + D_R[i]
        imp ← val
    return inverse(imp)
procedure PSMW(X, p, k, T)
    weightNumber = {1, 2, 3, ..., T}
    Weight_fwd = MinMaxScaler(weightNumber, feature_range = (0, 1))
    Weight_bwd = Inverse(Weight_fwd)
    imp = {L[m]}
    for i=1 to T do
        val = (D_L[i]*Weight_fwd[i] + D_R[T−i]*Weight_bwd[i]) / (Weight_fwd[i] + Weight_bwd[i])
        imp ← val
    return imp
```

7.3 Evaluation

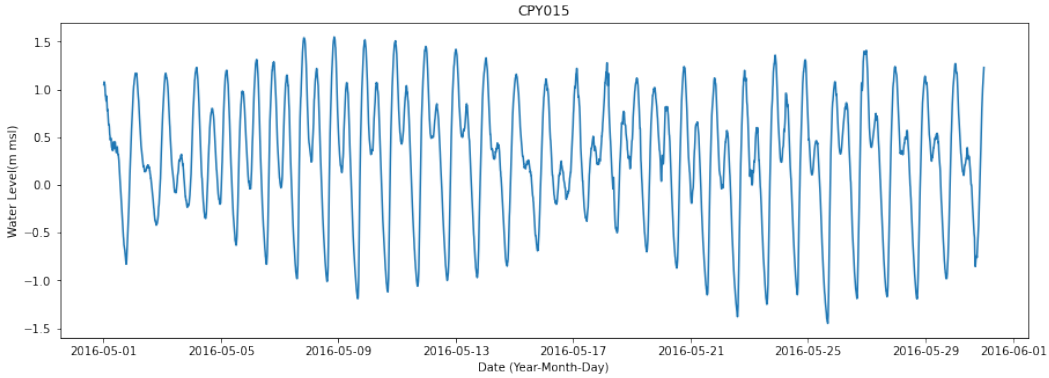
7.3.1 Dataset

Indeed, we were unable to examine the capacity of imputation algorithms on genuine missing data because the true values were not available. As a result, we must generate

simulated missing values on complete data in order to evaluate the performance of imputation approaches. So we considered datasets that were either complete or had the fewest number of missing or anomalous data points. After data preprocessing, we chose two years of data (2015 and 2016) from six representative stations (CPY012, CPY015, CPY016, CPY017, CHM003, and CHR004) to test the accuracy and generalisation of our proposed methods when dealing with different data behaviours. CPY012 and CPY015 stations represent the data with tidal effects that have strong periodic patterns, as depicted in Figure 7.1. While the data from CPY016 and CPY017 that have fluctuation characteristics with few upward and downward as a result of irrigation operating in the canal that has telemetry stations installed, as shown in Figure 7.2. Additionally, the data from CHM003 and CHR004 show fluctuations and many upward and downward patterns as a result of the rain effect, as shown in Figure 7.3.

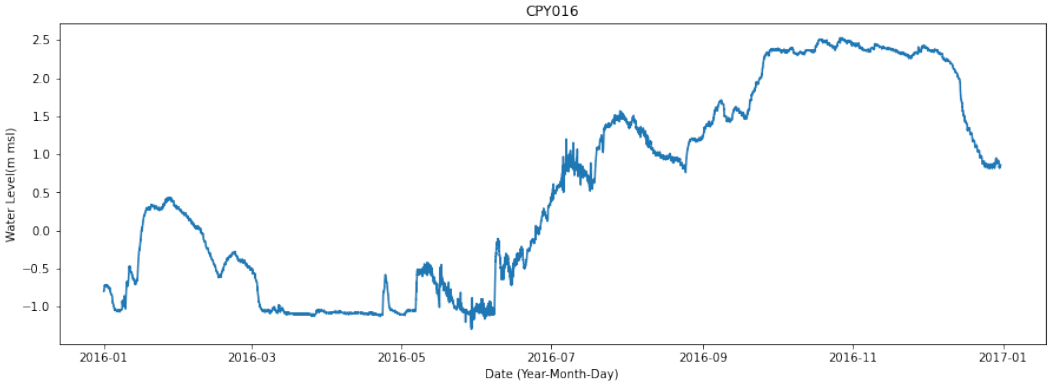


(a) CPY012

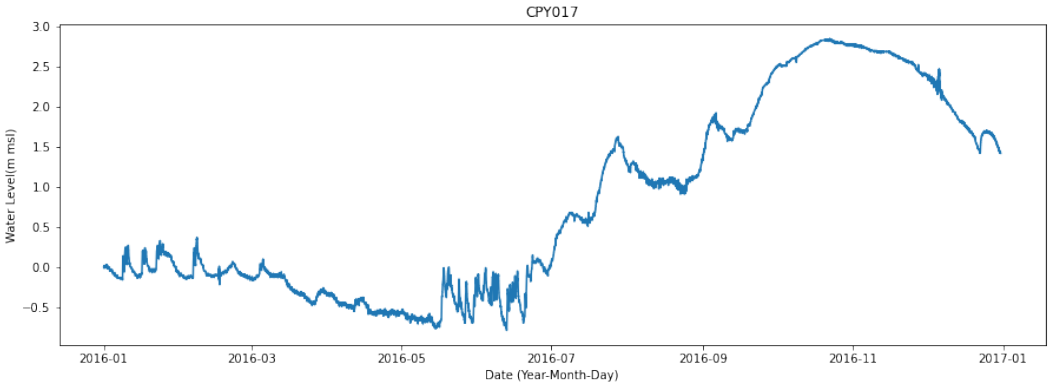


(b) CPY015

Figure 7.1: Water level data with tidal influences.

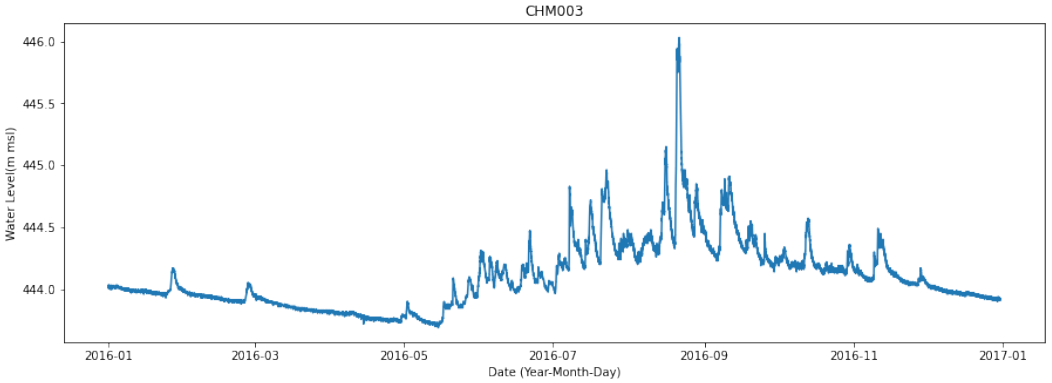


(a) CPY016

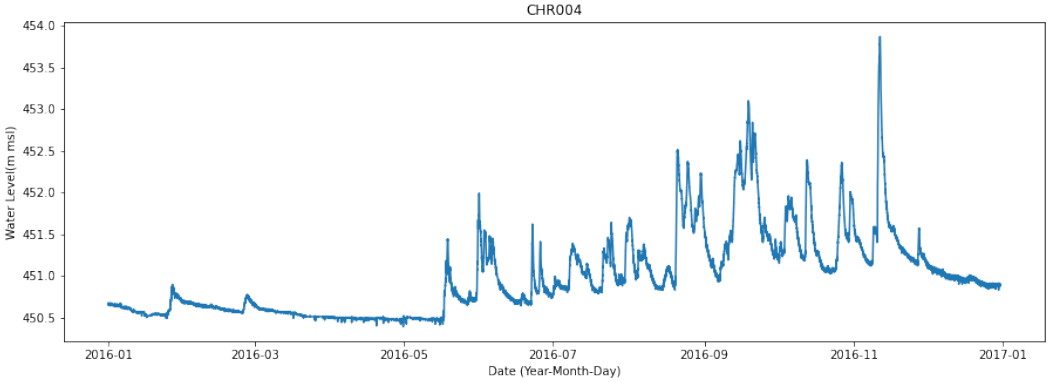


(b) CPY017

Figure 7.2: Water level data with irrigation influences.



(a) CHM003



(b) CHR004

Figure 7.3: Water level data with rain influences.

7.3.2 Missing data generation

We were unable to examine the accuracy of imputation algorithms on genuine missing data because the true values were not available. But we can simulate some missing data with some methods on complete data in order to evaluate the performance of imputation approaches. To produce datasets with missing data, we delete consecutive values from the dataset under the assumption that it happens at random.

To simulate various missing data situations, we generated missing gaps of sizes 6, 12, 18, 36, 72, and 144 (1 hour, 2 hours, 3 hours, 6 hours, 12 hours, and 1 day), and the length of a consecutive subsequence before and after the missing gap equal to the size of the missing gap. For instance, if the missing gap is six in length, the lengths of the subsequences before and after the missing gap are also six.

7.3.3 Comparision imputation methods

We chose some well-known representative imputation methods for comparing our methods. These are interpolation of linear and polynomial, k-Nearest Neighbours (k-NN), MissForest (MF), and a deep learning method - long short term memory (LSTM). In case of LSTM, we utilised two LSTM models ($LSTM_F$ and $LSTM_B$) with the same architecture for predicting and backcasting the missing gaps. In the event of backcasting, we invert the position of the subsequence after the missing data and feed it into the LSTM_B model. Furthermore, we created two new imputations based on the results of both LSTM_F and LSTM_B. The first is $LSTM_A$, which takes the average of the outputs of both models and uses it as the final output. The second is $LSTM_W$, which we weight the values of output from LSTM_F and LSTM_B using the same notion of weighting imputation as in FSM_W by assigning the greatest weighting score to the data that is closest to the current values.

7.3.4 Experimental Setting

The datasets are divided depending on each method, and we run each method 500 times and average the results. The dataset is divided into training/searching and testing/removing. The training data is used to fit neural network model and search

the most similar subsequence, while the testing data is used to generate the missing subsequences and assess model performance. For training purposes, we used data from 2015, whereas for testing purposes, we used data from 2016.

For interpolation technique we used interpolate class in panda.DataFrame[1] python library, which is a method for filling missing value using an interpolation method. We specified a linear approach for linear interpolation and a polynomial method with an order of 2 for polynomial interpolation.

We used the grid search technique to find the best k number of nearest neighbours, ranging from 2 to the number of members in the query subsequence for k-NN models. Since MF models require multivariate data, we chose the simplest way to convert our data from univariate to multivariate by dividing the query sequence into three subsequences to use as input for MF models.

For partial subsequence matching, we used the matrix profile python library called STUMPY (Law, 2019) which is a powerful and scalable library for searching the most similar subsequence. In order to properly train LSTM models, we used one LSTM layer and one hidden layer that uses only dense layer, 30 training epochs, prevent time-consuming and over-fitting with early stopping with the patience values of 5 and mini-batch size of 128. In practical, we tried with different number of neurons per layer (64, 128, and 256) and found that 128 neuron per layer give the best result. The input of LSTM is the subsequence of before and after missing values for prediction the missing values.

All the experiments were coded with Python Programming Language (V3.6) and TensorFlow 2.8, and run on a personal computer with an Intel Core i5-7500 CPU @ 3.4 GHz, 32 GB RAM, 64-Bit Operating System.

7.3.5 Evaluating imputation methods

We assess the performance of our proposed approaches to current imputation methods using three separate measures: RMSE, MAE, and Sim, as described in Section 3.6.3.

[1] `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html`

7.4 Results

Incomplete subsequence matching methods as described in Section 7.2 guarantee that our suggested model will be capable of producing imputation results with varying lengths. Aside from that, the number of available data points around the missing values can also be adjusted. As a result, the FMS is built to deal with random size of data gaps in time series. We random remove the subsequence 500 times from telemetry water level data and the results described as follow.

Table 7.1: The average imputation performance indexes of 14 methods on telemetry water level data with tidal influence.

Method	Gap	RMSE	MAE	Sim	Gap	RMSE	MAE	Sim	Gap	RMSE	MAE	Sim
Li-Inter	6	**0.0180**	**0.0158**	**0.8525**	12	0.0419	0.0376	0.8625	18	0.0726	0.0652	0.8485
Poly-Deg2		0.0202	0.0177	0.8305		0.0343	0.0297	0.8728		0.0507	0.0433	0.8742
k-NN		0.0818	0.0768	0.7251		0.1136	0.1030	0.7697		0.1763	0.1593	0.7538
MF		0.0835	0.0790	0.6295		0.1976	0.1862	0.6074		0.3253	0.3011	0.5919
PSM_F		0.0647	0.0560	0.6915		0.1231	0.1040	0.7340		0.1776	0.1487	0.7311
PSM_B		0.0688	0.0618	0.6696		0.1315	0.1174	0.7149		0.2044	0.1809	0.7031
PSM_A		0.0522	0.0460	0.7167		0.0944	0.0822	0.7645		0.1417	0.1223	0.7600
PSM_W		0.0509	0.0448	0.7187		0.0912	0.0792	0.7674		0.1339	0.1149	0.7643
FSM_D		0.0231	0.0198	0.8223		0.0349	0.0300	0.8704		0.0460	0.0395	0.8788
FSM_S		0.0208	0.0178	0.8356		**0.0290**	**0.0247**	**0.8872**		**0.0383**	**0.0327**	**0.8984**
$LSTM_F$		0.0634	0.0551	0.6888		0.1062	0.0901	0.7551		0.1513	0.1257	0.7610
$LSTM_B$		0.0540	0.0470	0.7203		0.1096	0.0944	0.7395		0.1251	0.1060	0.7790
$LSTM_A$		0.0376	0.0340	0.7537		0.0797	0.0729	0.7613		0.1064	0.0950	0.7883
$LSTM_W$		0.0398	0.0345	0.7767		0.0679	0.0605	0.8050		0.0836	0.0730	0.8384
Li-Inter	36	0.2034	0.1792	0.8136	72	0.4464	0.3729	0.7820	144	0.6121	0.5056	0.7760
Poly-Deg2		0.1125	0.0939	0.8736		0.3777	0.3107	0.8033		0.7508	0.5970	0.7453
k-NN		0.2685	0.2375	0.7732		0.3229	0.2937	0.8085		0.2695	0.2451	0.8633
MF		0.6674	0.5987	0.5881		0.6402	0.5792	0.6781		**0.1253**	**0.1014**	**0.9331**
PSM_F		0.2754	0.2273	0.7736		0.2531	0.2142	0.8340		0.2180	0.1786	0.8785
PSM_B		0.3202	0.2776	0.7435		0.2666	0.2303	0.8254		0.2503	0.2137	0.8646
PSM_A		0.2220	0.1885	0.7940		0.2128	0.1822	0.8506		0.1923	0.1616	0.8873
PSM_W		0.2121	0.1790	0.8024		0.2338	0.2001	0.8410		0.1954	0.1639	0.8847
FSM_D		0.1029	0.0887	0.8707		0.2329	0.1982	0.8392		0.1904	0.1593	0.8877
FSM_S		**0.0921**	**0.0790**	**0.8794**		0.2066	0.1737	0.8542		0.1692	0.1391	0.8977
$LSTM_F$		0.2205	0.1869	0.7933		0.2405	0.2073	0.8372		0.2724	0.2375	0.8622
$LSTM_B$		0.2406	0.2092	0.7810		0.2612	0.2237	0.8291		0.2574	0.2227	0.8719
$LSTM_A$		0.1895	0.1729	0.7948		0.2208	0.1962	0.8408		0.2334	0.2094	0.8758
$LSTM_W$		0.1394	0.1250	0.8391		**0.1949**	**0.1679**	**0.8564**		0.2172	0.1889	0.8807

We first consider the tidal-influenced data whose recurrent upward and downward trends are noticeably and frequent changing with a similar magnitude. The average imputation performance of each methods are depicted in Table 7.1. As expected, when the size of the missing gaps is small, e.g., 6, linear interpolation techniques (Li-Inter) achieve the best performance for RMSE, MAE, and Sim with 0.0180, 0.0158, and 0.8525, respectively. However, their performance degrades steadily when dealing with gaps bigger than 6. Similar to polynomial interpolation (Poly-Deg2), which performed

well when imputed missing data with a small gap but poorly when the gap increased. MF technique performed the poorest results with gap size lower than 144, particularly on gap size 36, with 0.6674 (RMSE), 0.5987 (MAE), and 0.5991 (Sim), but it performed best when it filled in the missing gaps with a size of 144, with 0.1253 (RMSE), 0.1014 (MAE), and 0.9331 (Sim). Our proposed solution, FSM_S, outperforms all others when imputed missing gaps of sizes 12, 18, and 36. $LSTM_W$ performed best on gap size 72, with RMSE, MAE, and Sim of 0.1949, 0.1678, and 0.8564, respectively. When the missing gaps were imputed at size 144, MF beat other models with the lowest RMSE of 0.0383, the lowest MAE of 0.0322, and the highest Sim of 0.9331.

We also plotted the average imputation performance for 14 methods using telemetry water level data with tidal features, as illustrated in Figure 7.5. As we can see, the interpolation method, Li-Inter and Poly-Deg2, appeared to decrease in performance as the number of missing gaps rose. When filling in data gaps with sizes of 12, 18, and 36, our suggested method, FSM_S, is clearly better than all others. After increasing the gap size to 144, performance of k-NN often improves. While performance of MF was the poorest while trying to impute missing data with a gap size of 72 or less, it improved to the best when the gap size was increased to 144. Although the set of PSM techniques performs poorly when the missing gap size is small, their performance improves when the missing gap size is equal to or higher than 72. It is interesting to note that as the amount of input data goes up, the performance of LSTM approaches gets better as the number of missing gaps goes up.

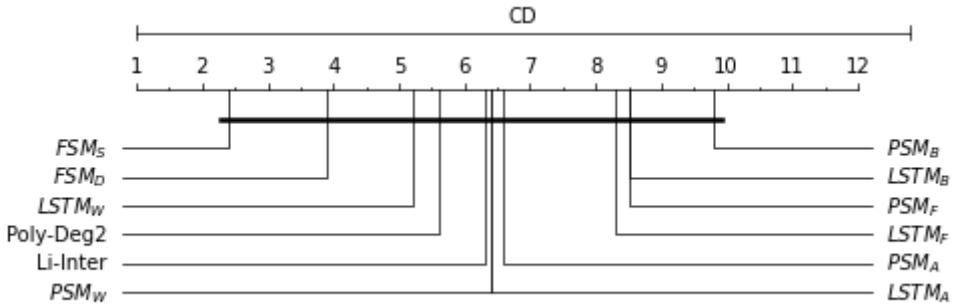
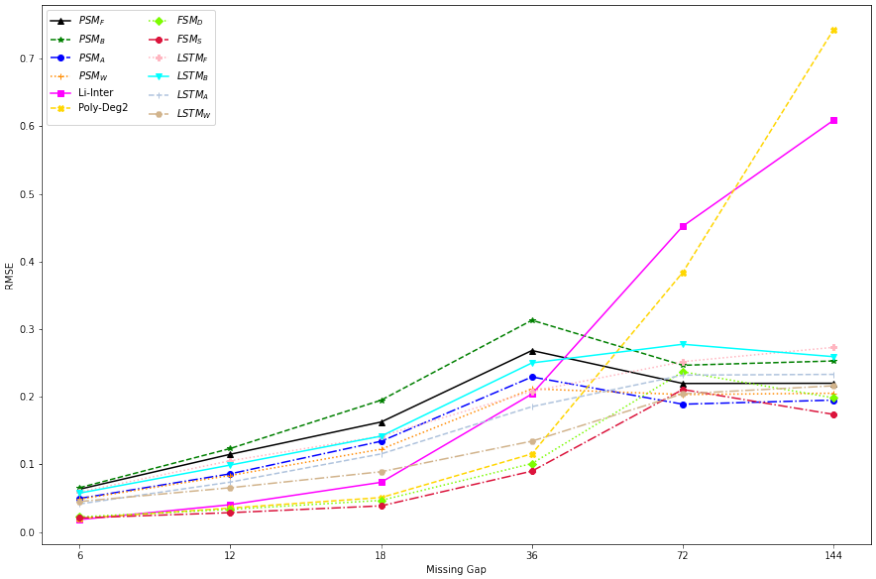
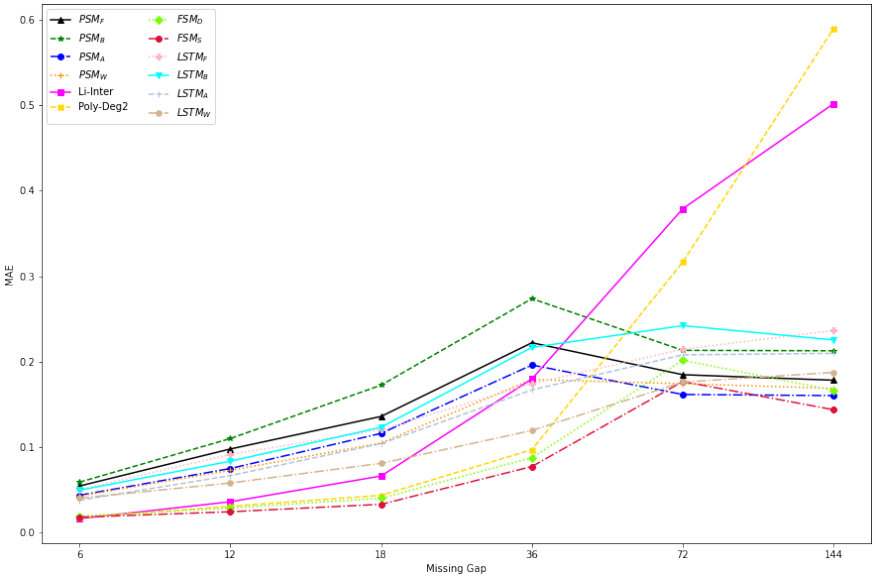


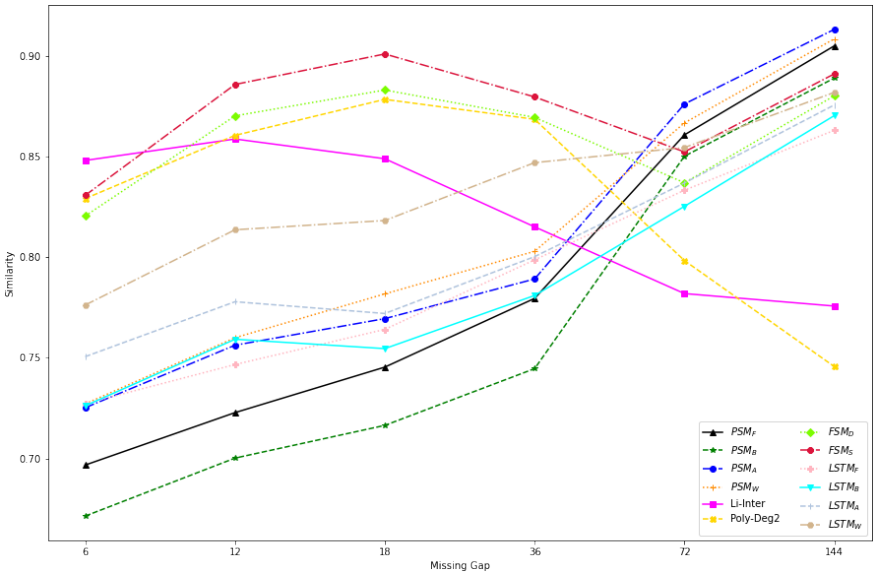
Figure 7.4: A critical difference diagram for 14 different imputation techniques on tidal influence datasets of telemetry water level data.



(a) RMSE



(b) MAE



(c) Similarity

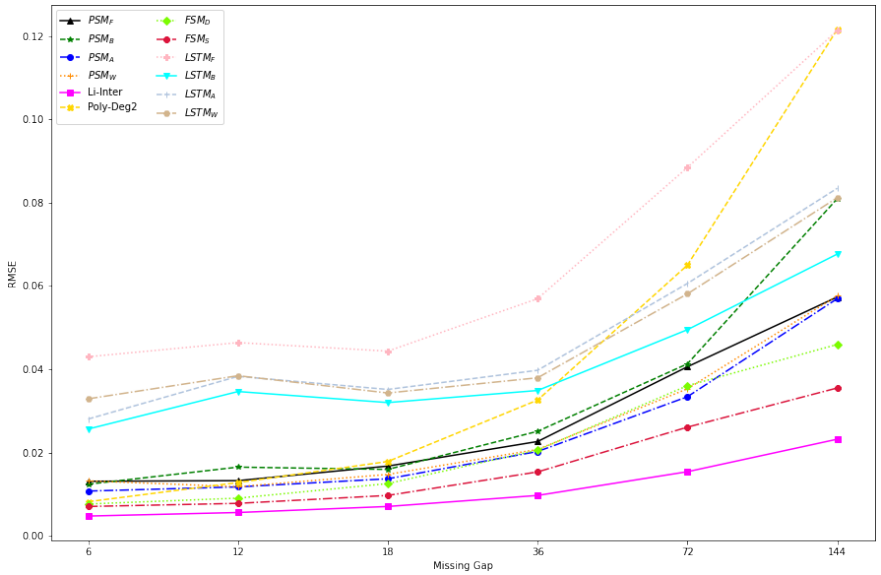
Figure 7.5: The performance for imputing telemetry water level data with tidal influence for various missing gap size.

Figure 7.4 shows the comparison of the critical difference between the different imputation models. The number associated with each algorithm is the average rank of the imputation models on each type of datasets and solid bar group classifiers with no significant difference. For the data type with tidal effect, FSM_S achieved the top rank follow with FSM_D, $LSTM_W$, and Poly-Deg2, respectively. MF not only provided the lowest ranking but also significant difference from FSM-based technique.

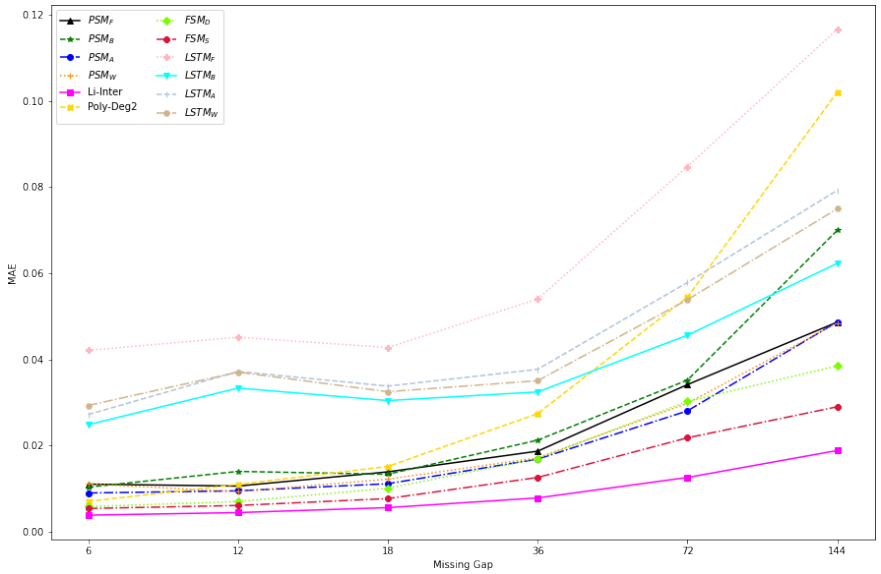
Table 7.2 shows the imputing findings for the irrigation-affected data. Li-Inter is not only the best imputation model for missing gaps of size 6, but also performs well for larger missing gaps. k-NN, on the other hand, performed the poorest with every missing gap size and has a score difference from Li-Inter of roughly 0.08 in every performance metric.

Table 7.2: The average imputation performance indexes of 14 methods on telemetry water level data with irrigation influence.

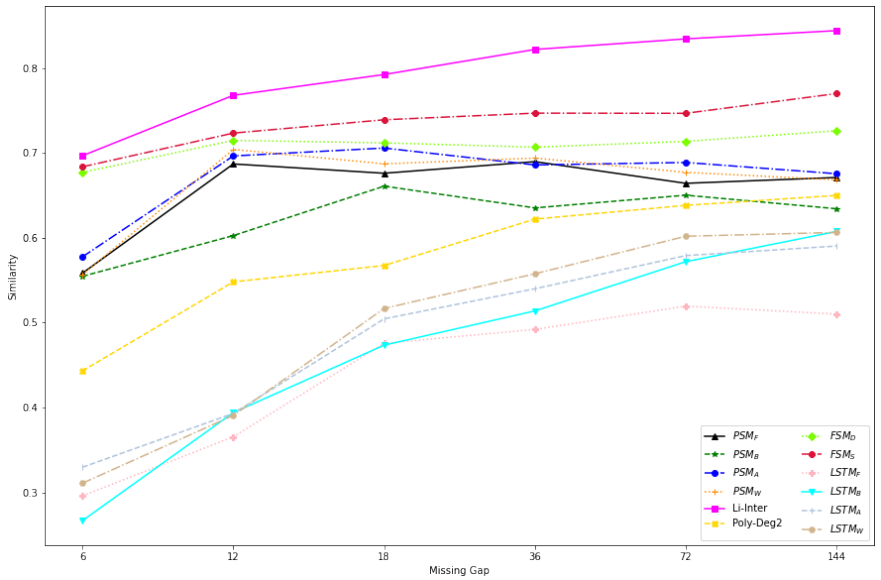
Method	Gap	RMSE	MAE	Sim	Gap	RMSE	MAE	Sim	Gap	RMSE	MAE	Sim
Li-Inter	6	**0.0050**	**0.0042**	**0.7004**	12	**0.0066**	**0.0054**	**0.7641**	18	**0.0069**	**0.0055**	**0.7992**
Poly-Deg2		0.0083	0.0071	0.4525		0.0138	0.0117	0.5570		0.0177	0.0150	0.5596
k-NN		0.0832	0.0822	0.3505		0.0933	0.0918	0.4006		0.1061	0.1042	0.4173
MF		0.0075	0.0058	0.7271		0.0113	0.0089	0.7454		0.0124	0.0100	0.7452
PSM_F		0.0140	0.0120	0.5239		0.0151	0.0123	0.6745		0.0183	0.0151	0.6610
PSM_B		0.0133	0.0112	0.5251		0.0178	0.0151	0.6179		0.0168	0.0140	0.6631
PSM_A		0.0117	0.0099	0.5542		0.0140	0.0115	0.6651		0.0147	0.0119	0.6962
PSM_W		0.0138	0.0117	0.5389		0.0136	0.0109	0.6916		0.0158	0.0130	0.6827
FSM_D		0.0082	0.0063	0.6689		0.0103	0.0081	0.7214		0.0119	0.0094	0.7165
FSM_S		0.0075	0.0058	0.6831		0.0090	0.0071	0.7342		0.0097	0.0077	0.7418
$LSTM_F$		0.0394	0.0386	0.2696		0.0333	0.0319	0.4421		0.0512	0.0498	0.3514
$LSTM_B$		0.0233	0.0225	0.3171		0.0287	0.0273	0.4167		0.0324	0.0309	0.4655
$LSTM_A$		0.0279	0.0271	0.3283		0.0284	0.0272	0.4529		0.0360	0.0346	0.4579
$LSTM_W$		0.0302	0.0281	0.3116		0.0284	0.0268	0.4591		0.0381	0.0356	0.4424
Li-Inter	36	**0.0092**	**0.0074**	**0.8261**	72	**0.0155**	**0.0126**	**0.8367**	144	**0.0240**	**0.0193**	**0.8424**
Poly-Deg2		0.0320	0.0269	0.6305		0.0609	0.0510	0.6556		0.1271	0.1070	0.6252
k-NN		0.1146	0.1118	0.4511		0.1235	0.1194	0.4759		0.1600	0.1544	0.4988
MF		0.0207	0.0176	0.7131		0.0335	0.0289	0.7069		0.0445	0.0386	0.7529
PSM_F		0.0265	0.0222	0.6639		0.0482	0.0405	0.6407		0.0683	0.0580	0.6479
PSM_B		0.0266	0.0227	0.6404		0.0506	0.0429	0.6225		0.0675	0.0571	0.6500
PSM_A		0.0225	0.0188	0.6837		0.0416	0.0351	0.6589		0.0560	0.0474	0.6786
PSM_W		0.0233	0.0195	0.6811		0.0443	0.0374	0.6465		0.0625	0.0531	0.6579
FSM_D		0.0200	0.0166	0.7001		0.0371	0.0312	0.7050		0.0458	0.0388	0.7222
FSM_S		0.0153	0.0127	0.7416		0.0274	0.0229	0.7407		0.0344	0.0283	0.7729
$LSTM_F$		0.0588	0.0564	0.4748		0.0784	0.0745	0.5279		0.1222	0.1170	0.5406
$LSTM_B$		0.0488	0.0467	0.4684		0.0627	0.0590	0.5661		0.0737	0.0680	0.5986
$LSTM_A$		0.0463	0.0444	0.5264		0.0631	0.0602	0.5735		0.0890	0.0844	0.5880
$LSTM_W$		0.0457	0.0434	0.5241		0.0600	0.0567	0.5963		0.0860	0.0798	0.6040



(a) RMSE



(b) MAE



(c) Similarity

Figure 7.6: The performance for imputing telemetry water level data with irrigation influence for various missing gap size.

As seen in Figure 7.6, the performance of all approaches fell progressively as the size of the missing gap rose, with the exception of the similarity score of LSTM models, which tended to improve performance as the gap size increases.

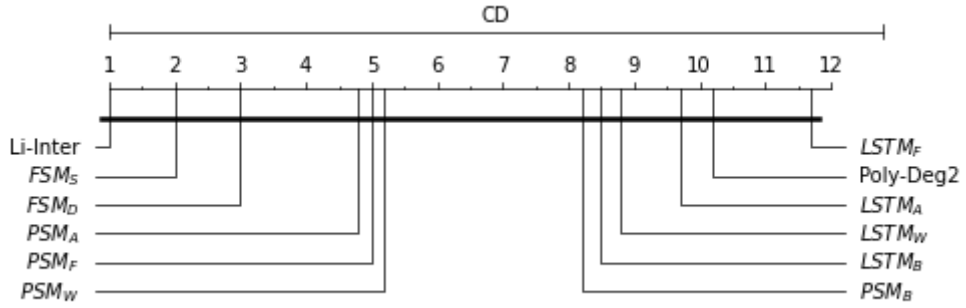


Figure 7.7: A critical difference diagram for 14 different imputation techniques on irrigation influence datasets of telemetry water level data.

The CD diagram in Figure 7.7 revealed that Li-Inter took first place, followed by our suggested technique (FSM_S), and MF, respectively, while the group of LSTM models performed the worst. A collection of PSM and FSM models works well and offers a considerable improvement over LSTM-based imputation approaches.

Regarding the impacts of rain, Li-Inter outperformed across all gap sizes and evaluation metrics, with the exception of gap 6, where performance was slightly lower than MF, around 0.0259, for Sim score. Moreover, the line charts in Figure 7.9 present the performance for imputing telemetry water level data with rain influence for various missing gap sizes. As we can see, the set of LSTM models scores the lowest on all evaluation metrics. Li-Inter and Poly-Deg2 have a tendency to perform worse as the number of missing values grows, while the set of PSM and FSM strategies maintain a consistent level of performance.

When used to impute the missing data on the water level with rain-effected, Li-Inter, FSM_S, and MF maintained their top rankings. However, k-NN dropped to the bottom of the list. LSTM-based techniques still provided the low ranking and significant difference from Li-Inter and FSM_S, as shown in Figure 7.8.

Table 7.3: The average imputation performance indexes of 14 methods on telemetry water level data with rain influence.

Method	Gap	RMSE	MAE	Sim	Gap	RMSE	MAE	Sim	Gap	RMSE	MAE	Sim
Li-Inter	6	**0.0066**	**0.0053**	0.7200	12	**0.0079**	**0.0062**	**0.7949**	18	**0.0080**	**0.0063**	**0.8148**
Poly-Deg2		0.0126	0.0110	0.5195		0.0211	0.0184	0.5577		0.0273	0.0236	0.5621
k-NN		0.0274	0.0263	0.5021		0.0358	0.0338	0.5668		0.0384	0.0364	0.5620
MF		0.0089	0.0069	**0.7459**		0.0128	0.0102	0.7701		0.0145	0.0117	0.7685
PSM_F		0.0131	0.0106	0.6658		0.0163	0.0129	0.7248		0.0188	0.0151	0.7210
PSM_B		0.0147	0.0124	0.5719		0.0196	0.0164	0.6628		0.0223	0.0186	0.6912
PSM_A		0.0120	0.0097	0.6395		0.0151	0.0122	0.7218		0.0173	0.0140	0.7305
PSM_W		0.0122	0.0098	0.6694		0.0153	0.0121	0.7330		0.0173	0.0139	0.7291
FSM_D		0.0110	0.0086	0.6773		0.0137	0.0109	0.7114		0.0165	0.0137	0.6853
FSM_S		0.0100	0.0078	0.6804		0.0115	0.0092	0.6959		0.0124	0.0099	0.7346
$LSTM_F$		0.3161	0.3159	0.0952		0.3216	0.3211	0.1272		0.3219	0.3215	0.1364
$LSTM_B$		0.3204	0.3202	0.0912		0.3268	0.3263	0.1204		0.3252	0.3248	0.1354
$LSTM_A$		0.3182	0.3180	0.0912		0.3241	0.3236	0.1239		0.3235	0.3230	0.1357
$LSTM_W$		0.3184	0.3180	0.0922		0.3243	0.3236	0.1238		0.3237	0.3231	0.1361
Li-Inter	36	**0.0105**	**0.0084**	**0.8453**	72	**0.0173**	**0.0143**	**0.8500**	144	**0.0277**	**0.0229**	**0.8561**
Poly-Deg2		0.0506	0.0436	0.5753		0.0944	0.0804	0.5647		0.1895	0.1618	0.5348
k-NN		0.0545	0.0512	0.5877		0.0850	0.0790	0.5786		0.1158	0.1080	0.5818
MF		0.0256	0.0218	0.7656		0.0438	0.0380	0.7548		0.0609	0.0527	0.7813
PSM_F		0.0360	0.0299	0.7001		0.0691	0.0578	0.6509		0.0945	0.0798	0.6358
PSM_B		0.0391	0.0331	0.6928		0.0830	0.0693	0.6315		0.1394	0.1169	0.6097
PSM_A		0.0300	0.0249	0.7234		0.0620	0.0517	0.6591		0.0955	0.0803	0.6342
PSM_W		0.0313	0.0261	0.7148		0.0649	0.0537	0.6549		0.0964	0.0795	0.6304
FSM_D		0.0221	0.0182	0.7452		0.0352	0.0295	0.7518		0.0625	0.0528	0.7296
FSM_S		0.0164	0.0134	0.7607		0.0249	0.0208	0.7711		0.0437	0.0367	0.7750
$LSTM_F$		0.3190	0.3181	0.1874		0.3328	0.3307	0.2219		0.3219	0.3182	0.2862
$LSTM_B$		0.3237	0.3228	0.1827		0.3371	0.3351	0.2185		0.3275	0.3237	0.2824
$LSTM_A$		0.3212	0.3203	0.1844		0.3348	0.3328	0.2204		0.3246	0.3208	0.2842
$LSTM_W$		0.3214	0.3203	0.1849		0.3351	0.3329	0.2202		0.3249	0.3210	0.2840

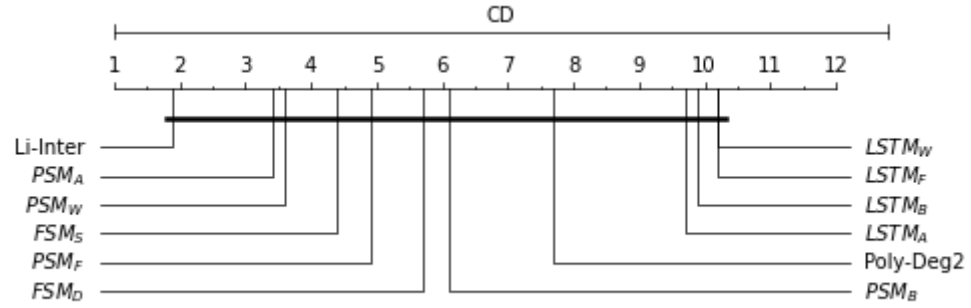
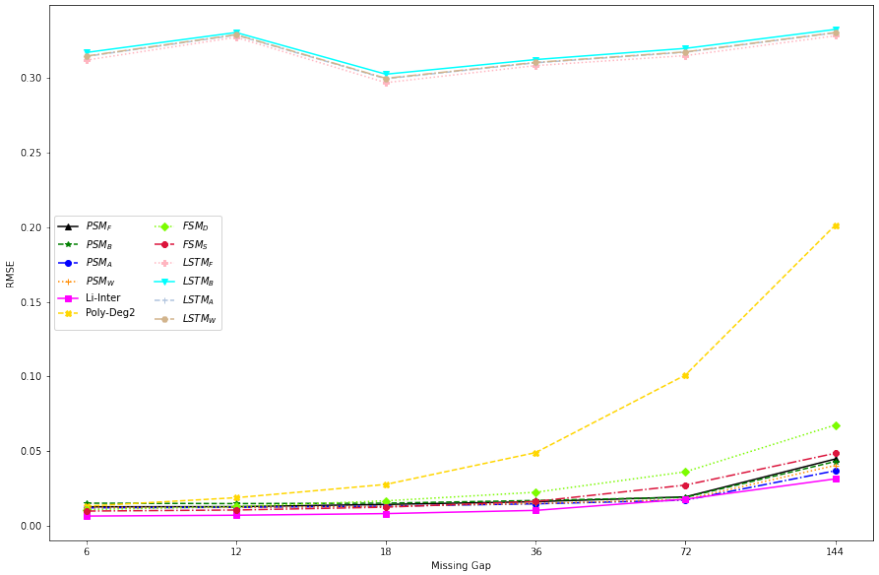
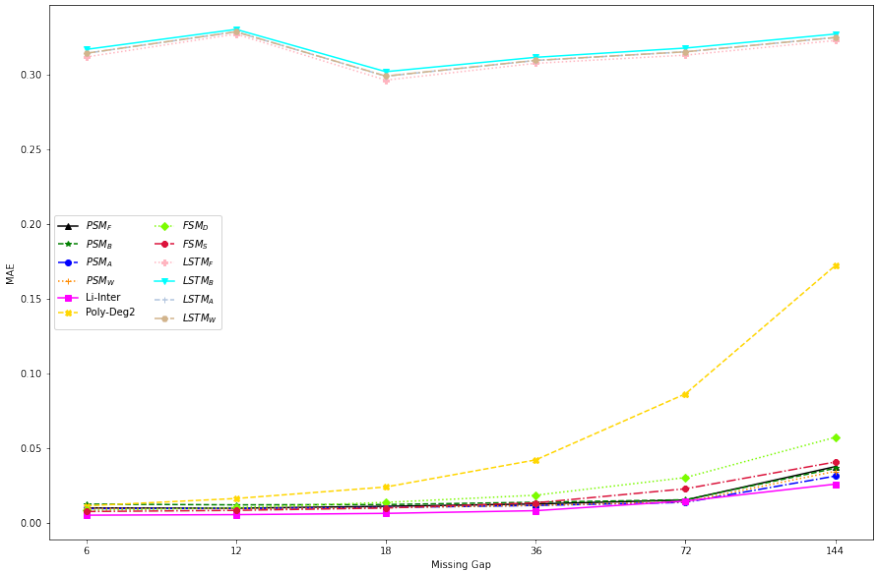


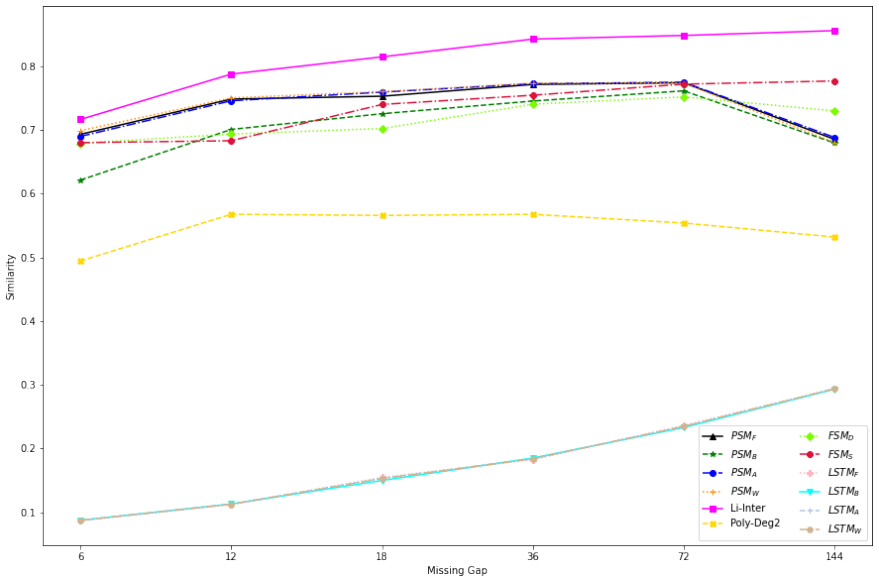
Figure 7.8: A critical difference diagram for 14 different imputation techniques on rain influence datasets of telemetry water level data.



(a) RMSE



(b) MAE



(c) Similarity

Figure 7.9: The performance for imputing telemetry water level data with rain influence for various missing gap size.

7.4.1 Discussion

Li-inter and Poly-Deg2 produced the best performance on data with non-cyclical and periodic patterns, like data with rain and irrigation effects. Moreover, with the small missing gap size Li-inter outperformed the others with all data behaviours, which is expected.

When imputing the missing values on water level data with tidal influence, our approaches FSM_S outperformed the others. It is mostly due to the fact that some cyclical patterns of tidal influence are repeated in water level data over time, and our approaches are capable of finding and matching the most similar pattern in the past to impute the missing data more accurately. However, when we took a close look into the very short missing gaps, which still seem more like linear, then it is not surprising to see that Li-inter performed better.

The utilized LSTM model was trained on the input subsequence and its reversed copy that preserved both past and future information of a specific time frame. With this advantage, LSTM models are able to understand the context better and thus in principle they should be suitable for imputing the missing data in telemetry water level. However, they did not produce the best performance. According to our experiments on the data with different behaviours, LSTM models incorrectly estimated strongly fluctuated data, for example the data with some raining effects. On the other hand, they are capable of imputing the missing values on the data with tidal and irrigation effects with periodic and without frequent trends.

Since MF does not use data for training, it performed poorly when dealing with short subsequences but produced excellent outcomes for missing data with large gap sizes. In other words, MF performed better when appropriate data is available. However, since MF models need multivariate data, which is not always the case in this application and transformation from univariate data to multivariate data can introduce noise or misrepresentations. Splitting the sequence into many subsequences is the simplest technique to generate multivariate data. This raises the challenge of determining the optimal number of subsequence splits.

7.5 Summary

This experiment introduced two subsequence matching methods: full and partial methods, for searching the most similar subsequence and filling in the missing values in telemetry water level data. They were tested with real-world water level data collected from 6 water level monitoring stations, and their results were compared with a range of other existing methods, including some commonly used methods and the latest state-of-the-art deep learning methods - LSTM. The results showed that our new methods, particularly full subsequence matching with scaling imputation technique (FSM_S), were better than all of them.

The FSM approach uses the Euclidean distance technique to search for the most similar subsequence of the query subsequences, then calculate missing data values based on the pattern of those subsequence. However, rather than dividing it into two subsequences we replaced missing data with constant values and searched as a single subsequence. The proposed methods were evaluated using missing data simulated on six time series water level data with three distinct data behaviours. The results indicate that FSM with scaling imputation, FSM_S, outperforms other imputation methods when dealing with large missing gap sizes.

The FSM_S performs well on data that has strong periodic and cyclical pattern such as data of water level with tidal effects. While the linear interpolation approach works well with data that fluctuates and has a number of up and down trends such as water level data with rain and irrigation effects. LSTM and MF models show increased performance when the missing gap size is increased. However, for large datasets, LSTM is computationally costly and time-consuming. Although we may train the model using long periods of historical data with high performance computing (HPC) to shorten training time, we have no way of knowing when we need to retrain the model, which is a significant downside of this approach. While MF needs to transform univariate data to multivariate data which difficult to find the appropriate techniques of transformation.

Chapter 8

Evaluation

8.1 Introduction

In Chapters 4 - 7, we presented our work on anomaly detection and data imputation. In this chapter, we overview all the methodologies used in our study, then evaluate and discuss them and the results. Moreover, we have also performed a fully independent validation of our work by evaluating our techniques on external methodologies and datasets.

8.2 Overview of the Research

This study examined the issue with telemetry water level data from two perspectives. The first component was to detect anomalous data. The second part was data imputation.

For anomaly detection, we first considered the statistical models to determine how well those methods were when dealing with telemetry water level data. Although an individual statistical model can be used to identify anomalies, it still produces a large number of false alarms. DRL is an algorithm that has obtained outstanding performance during the last decade. Therefore, we created DRL models for detecting anomalies in telemetry water level data. Although DRL is applicable for identifying abnormalities, it takes a long time because we have to train it until it works well enough and there are a lot of hyperparameters to tune. We then developed a feature extraction algorithm called SM+NNFE that combined the saliency map and nearest

neighbour extracted features to improve the performance of models. Furthermore, we combined the individual models with ensemble methods, which provided more accurate and consistent classification. Although they may not always produce the best results, they outperformed in terms of minimising the frequency of false alarms in detecting anomalies, which is critical in real-world applications.

In the case of data imputation, the water level is seasonal, which means it follows the same pattern throughout the year. For these characteristics, we may replicate the most similar pattern from the prior data to impute the missing data. The results of experiments suggest that our method is better than others when dealing with data that has a strong periodic pattern.

8.3 Evaluation

In each experiment, we picked data from a distinct time period and station. Because each model has its limitations, such as statistical models do not need data for training, we have tested using data that has a sufficient consecutive sequence with a low proportion of missing data. While machine learning models need data for training, our problem is imbalanced datasets. Therefore, we picked data from certain stations with a low proportion of missing data and an appropriate number of anomalies to conduct the experiment. Also, because DRL takes a lot of time, we only used data from short time periods to evaluate the models. As a consequence, the evaluation of the accuracy of our models is needed. In this section, we explain how we evaluated our research methods and compared the results of our approaches.

8.3.1 Evaluation of the Research Methods

While assessing a research methodology, there are three primary aspects that should be addressed. These are accuracy, reliability, and time-consuming. The accuracy was measured using the F1-score for anomaly detection and RMSE for data imputation, in addition to the critical difference diagram for statistical comparison. The reliability was determined by doing each experiment five times with variations, and then the mean and standard deviation of the results were reported. The time-consuming was

measured by the time spent on training each model.

8.3.2 Comparison between our anomaly detection methods

We analyse data gathered over a variety of periods and from different stations to assess the accuracy, consistency, and reliability of each model. We picked data from 33 sites that covered all different water level behaviours. The summary of the data is shown in Table 8.1.

Table 8.1: Summary of data from 33 telemetry stations.

Data Behavior	Station	Date Interval	No. of Records	No. of Anomaly	% of Anomaly Data
Irrigation	DIV001	1 Jan 2015 - 31 Dec 2015	51798	11	0.02
	PIN002	1 Jun 2016 - 1 Oct 2017	68760	2	0.00
	PIN003	1 Apr 2015 - 1 Oct 2016	77858	696	0.89
	VLGE13	1 Jan 2017 - 31 Dec 2017	52290	4	0.01
	YOM012	9 Apr 2014 - 31 Dec 2016	140489	6	0.00
Rain	CHM001	1 Mar 2014 - 31 Dec 2014	39317	2	0.01
	CHM002	1 May 2014 - 31 Dec 2017	181878	2	3.72
	CHM003	1 Apr 2014 - 31 Dec 2016	136462	8	0.01
	CHM005	1 Jan 2015 - 31 Dec 2016	99447	2	0.00
	CHM006	1 Apr 2014 - 31 Dec 2017	187008	1909	1.02
	CHR004	1 Apr 2014 - 30 Sep 2017	173610	57	0.03
	CHR005	1 May 2014 - 28 Feb 2015	40623	4	0.01
	CPY001	1 Jan 2013 - 31 Dec 2014	105120	41	0.04
	CPY002	1 Nov 2013 - 31 Dec 2014	61344	3	0.00
	CPY004	1 Jan 2013 - 31 Jul 2014	83088	158	0.19
	CPY005	1 Jan 2013 - 31 Oct 2014	96336	2	0.00
	CPY006	1 Dec 2014 - 31 Oct 2017	153504	12	0.01
	DIV004	16 May 2014 - 30 Jul 2016	114370	46	0.04
	PIN005	1 May 2014 - 30 Nov 2015	77391	6449	8.33
	PIN006	16 Aug 2015 - 1 May 2016	37165	2943	7.92
	WAN001	1 Jan 2015 - 31 Jul 2017	125840	2207	1.75
	WAN002	1 Mar 2014 - 30 Sep 2016	122946	5	0.00
	WAN003	1 May 2016 - 31 Dec 2017	86483	3	0.00
	WAN005	1 Mar 2014 - 31 Dec 2015	92652	60	0.06
	YOM009	15 Dec 2014 - 15 Mar 2016	65153	1773	2.72
	YOM010	1 Jan 2014 - 30 Nov 2015	90645	3372	3.72
Tidal	CPY011	1 Apr 2016 - 31 Mar 2017	52560	109	0.21
	CPY012	1 Jan 2015 - 31 Jul 2016	83232	1292	1.55
	CPY013	1 May 2016 - 31 Dec 2016	35280	578	1.64
	CPY014	1 Jan 2014 - 30 Aug 2015	87408	371	0.42
	CPY015	1 Jan 2014 - 31 Dec 2016	157824	94	0.06
	CPY016	1 Apr 2014 - 31 Dec 2016	144864	2265	1.56
	CPY017	1 Jan 2014 - 31 Dec 2016	157824	1028	0.65
		Avg	99411.18	773.15	1.11

For comparison, we selected seven statistical models in Chapter 4, MLP and LSTM in Chapter 5, and five machine learning models in Chapter 6. We experimented with univariate and multivariate data that had been extracted by SM+NNFE as described in Chapter 6. We used F1-score to evaluate the results. Additionally, the statistical comparison with the critical difference diagram is also represented.

With univariate data.

We start with comparing the accuracy of each model by using only telemetry water level data (univariate). The F1-score of each model, as depicted in Table 8.2. With statistical models, IQR had the greatest overall average F1-score of 0.2425, followed by

MAS and Ksigma, which had 0.1781 and 0.1441, respectively. KNN had the highest overall average F1-score of 0.1970 for ML models, while GNB had the lowest overall average F1-score of 0.0190. Both LSTM and MLP models produced an overall average F1-score of roughly 0.2, which was slightly lower than the IQR model. Overall, IQR provided the best performance, followed by LSTM, MLP, and KNN. GNB, on the other hand, had the lowest overall average F1-score of 0.0190. When we analysed the accuracy of each model based on data behaviour, we discovered that ML and NN models provided low accuracy when dealing with data with irrigation behaviour; almost all of them, with the exception of KNN on the PIN003 dataset, were unable to identify any abnormality. KNN had the greatest average F1-score of 0.2386 for data with rain behaviour, followed by DT with a 0.1624 F1-score and LSTM with a 0.1255 F1-score, respectively. On tidal behaviour data, NN models achieved the highest accuracy, with an average F1-score of 0.5953 by LSTM and 0.5856 by MLP.

Table 8.2: The F1-score of each model on univariate telemetry water level data with different characteristics.

Data Behaviour	Station	Statistical Models							ML Models					NN Models	
		AR	DB	IQR	KSigma	MAS	SA	ZScore	DT	GNB	IsoForest	KNN	SVC	LSTM	MLP
Irrigation	DIV001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PIN002	0.0001	0.1176	1.0000	0.0104	0.0084	0.6667	0.0012	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PIN003	0.0009	0.1087	0.0571	0.0665	0.0140	0.0000	0.0156	0.0000	0.0000	0.0007	0.0474	0.0000	0.0000	0.0000
	VLGE13	0.0000	0.1429	0.0037	0.0056	0.0014	0.0000	0.0008	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	YOM012	0.0000	0.0909	0.0025	0.0034	0.0142	0.0000	0.0006	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
	Avg	0.0002	0.0920	0.2127	0.0172	0.0076	0.1333	0.0036	0.0000	0.0000	0.0001	0.0095	0.0000	0.0000	0.0000
	Std	0.0004	0.0548	0.4408	0.0278	0.0067	0.2982	0.0067	0.0000	0.0000	0.0003	0.0212	0.0000	0.0000	0.0000
Rain	CHM001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM003	0.0000	0.1212	0.0020	0.0061	0.0005	0.0000	0.0007	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM006	0.0004	0.2462	0.0195	0.0136	0.0034	0.0000	0.0023	1.0000	0.0000	0.0004	0.0000	0.0000	0.0000	0.0000
	CHR004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHR005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CPY001	0.0008	0.0212	0.0017	0.0040	0.0027	0.0000	0.0017	0.0000	0.0000	0.0004	0.0000	0.0000	0.6356	0.7846
	CPY002	0.0001	0.0870	0.1600	0.0066	0.0833	0.0000	0.0008	0.0000	0.1733	0.0010	0.0000	0.0000	0.0000	0.0000
	CPY004	0.0061	0.2814	0.4436	0.3174	0.1363	0.0000	0.0295	0.0405	0.0000	0.0020	0.0253	0.0000	0.8880	0.8329
	CPY005	0.0007	0.2927	0.0040	0.0194	0.0037	0.0000	0.0021	0.0000	0.0000	0.0005	0.0000	0.0000	0.4240	0.4746
	CPY006	0.0004	0.3556	0.0015	0.0129	0.0022	0.0000	0.0010	0.0000	0.0000	0.0015	0.0000	0.0000	0.0800	0.1133
	DIV004	0.0001	0.1379	0.0089	0.0053	0.0007	0.0000	0.0006	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PIN005	0.2789	0.1012	0.3828	0.0615	0.9992	0.0398	0.3312	0.0002	0.0000	0.1154	0.9321	0.0231	0.0783	0.0923
	PIN006	0.3201	0.1321	0.9484	0.1799	1.0000	0.2044	0.9652	0.8974	0.0468	0.1379	0.9394	0.0044	0.4503	0.1385
	WAN001	0.0257	0.0443	0.2038	0.1018	0.1495	0.0000	0.0440	0.0000	0.0000	0.0726	0.9505	0.0000	0.0000	0.0000
	WAN002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	WAN003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	WAN005	0.0003	0.0656	0.0505	0.0310	0.0115	0.0000	0.0080	0.0000	0.0000	0.0040	0.2725	0.0000	0.0000	0.0000
	YOM009	0.0173	0.1421	0.5118	0.1918	0.3488	0.0000	0.2905	0.2734	0.0000	0.0187	0.9472	0.0000	0.0800	0.1003
	YOM010	0.1206	0.0947	0.2195	0.1716	0.3461	0.0000	0.1510	0.9491	0.4064	0.2860	0.9446	0.6642	0.0000	0.0000
	Avg	0.0367	0.1011	0.1409	0.0535	0.1470	0.0116	0.0871	0.1624	0.0298	0.0305	0.2386	0.0329	0.1255	0.1208
	Std	0.0915	0.1100	0.2456	0.0881	0.3028	0.0450	0.2227	0.3385	0.0945	0.0708	0.4076	0.1447	0.2513	0.2529
Tidal	CPY011	0.0003	0.4643	0.4235	0.6778	0.0274	0.0000	0.1787	0.0917	0.0000	0.0030	0.0083	0.0000	0.8729	0.8090
	CPY012	0.0158	0.4273	0.7475	0.6836	0.3131	0.0628	0.6128	0.4386	0.0000	0.0198	0.3849	0.0000	0.8131	0.8120
	CPY013	0.0465	0.4495	0.7611	0.7164	0.4987	0.1558	0.5625	0.1982	0.0000	0.0310	0.0821	0.0000	0.8185	0.8083
	CPY014	0.0067	0.1842	0.5631	0.6975	0.6424	0.1253	0.3902	0.0845	0.0000	0.0019	0.0126	0.0000	0.5506	0.6627
	CPY015	0.0021	0.1429	0.5564	0.1295	0.0031	0.1386	0.0669	0.0000	0.0000	0.0012	0.0000	0.0000	0.0000	0.0000
	CPY016	0.0626	0.1314	0.4373	0.2248	0.6225	0.0052	0.0736	0.0895	0.0000	0.0203	0.5911	0.0000	0.4855	0.3968
	CPY017	0.0214	0.2432	0.4910	0.4182	0.6430	0.0136	0.0668	0.2894	0.0000	0.0143	0.3619	0.0000	0.6267	0.6103
	Avg	0.0222	0.2918	0.5686	0.5068	0.3929	0.0716	0.2788	0.1703	0.0000	0.0131	0.2058	0.0000	0.5953	0.5856
	Std	0.0238	0.1499	0.1376	0.2485	0.2830	0.0676	0.2403	0.1506	0.0000	0.0114	0.2377	0.0000	0.3010	0.2989
	Overall Avg	0.0281	0.1402	0.2425	0.1441	0.1781	0.0428	0.1151	0.1395	0.0190	0.0222	0.1970	0.0210	0.2062	0.2011
	Overall Std	0.0743	0.1362	0.3094	0.2307	0.2960	0.1238	0.2240	0.2818	0.0761	0.0574	0.3481	0.1155	0.3170	0.3156

With multivariate data.

The created feature extraction technique, SM+NNFE, is then used to extract more features, as explained in Chapter 6. Because statistical models are univariate, the only

results from ML and NN models are unambiguous. Table 8.3 displays the findings. DT not only had the highest overall average f1-score of 0.3102, but it also performed the best on all data behaviour, particularly tidal behaviour, with a 0.7089 f1-score. Data with irrigation influenced is the most difficult for models to identify anomalies in, as seen by the average f1-score of every model being less than 0.0600.

Table 8.3: F1-score of each machine learning models with multivariate telemetry water level datasets.

Data Behaviour	Station	ML Models					NN Models	
		DT	GNB	IsoForest	KNN	SVC	LSTM	MLP
Irrigation	DIV001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PIN002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PIN003	0.2718	0.1298	0.0143	0.0000	0.0333	0.0000	0.0000
	VLGE13	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	YOM012	0.0000	0.0041	0.0007	0.0000	0.0000	0.0000	0.0000
	Avg	0.0544	0.0268	0.0030	0.0000	0.0067	0.0000	0.0000
	Std	0.1216	0.0576	0.0063	0.0000	0.0149	0.0000	0.0000
Rain	CHM001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM006	0.1249	0.0132	0.0020	0.0000	0.1019	0.0000	0.0000
	CHR004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHR005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CPY001	0.0000	0.0000	0.0022	0.0000	0.0000	0.0000	0.0000
	CPY002	0.3000	0.0675	0.0010	0.0000	0.0000	0.0000	0.0000
	CPY004	0.8699	0.5557	0.0439	0.8893	0.8913	0.7130	0.1101
	CPY005	0.3090	0.0208	0.0028	0.2000	0.2267	0.0000	0.0000
	CPY006	0.6267	0.0292	0.0024	0.4991	0.4667	0.0000	0.0000
	DIV004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	PIN005	0.5768	0.1367	0.4590	0.5707	0.0757	0.0000	0.0000
	PIN006	0.8653	0.3426	0.4572	0.8325	0.1992	0.9799	0.9740
	WAN001	0.1936	0.2352	0.1530	0.2920	0.1430	0.0000	0.0000
	WAN002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	WAN003	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	WAN005	0.0733	0.0452	0.0047	0.0000	0.0000	0.0000	0.0000
	YOM009	0.5493	0.4309	0.3028	0.5227	0.2899	0.4991	0.4122
	YOM010	0.5137	0.4919	0.3551	0.5403	0.2643	0.3730	0.4181
	Avg	0.2382	0.1128	0.0851	0.2070	0.1266	0.1221	0.0912
	Std	0.3040	0.1829	0.1598	0.3041	0.2188	0.2778	0.2378
Tidal	CPY011	0.8528	0.4675	0.0419	0.7759	0.7646	0.8687	0.2424
	CPY012	0.7920	0.5756	0.2504	0.7238	0.6977	0.8470	0.8369
	CPY013	0.8190	0.6032	0.2827	0.6282	0.5789	0.8743	0.8463
	CPY014	0.7256	0.3876	0.0788	0.6280	0.6403	0.8707	0.7145
	CPY015	0.3986	0.1978	0.0110	0.2892	0.2002	0.0000	0.0000
	CPY016	0.6728	0.5719	0.2467	0.6690	0.5486	0.4329	0.1963
	CPY017	0.7012	0.5775	0.1216	0.6705	0.6260	0.6790	0.5038
	Avg	0.7089	0.4830	0.1476	0.6264	0.5795	0.6532	0.4772
	Std	0.1515	0.1475	0.1110	0.1578	0.1821	0.3307	0.3377
	Overall Avg	0.3102	0.1783	0.0859	0.2646	0.2045	0.2163	0.1592
	Overall Std	0.3351	0.2283	0.1421	0.3228	0.2775	0.3516	0.2930

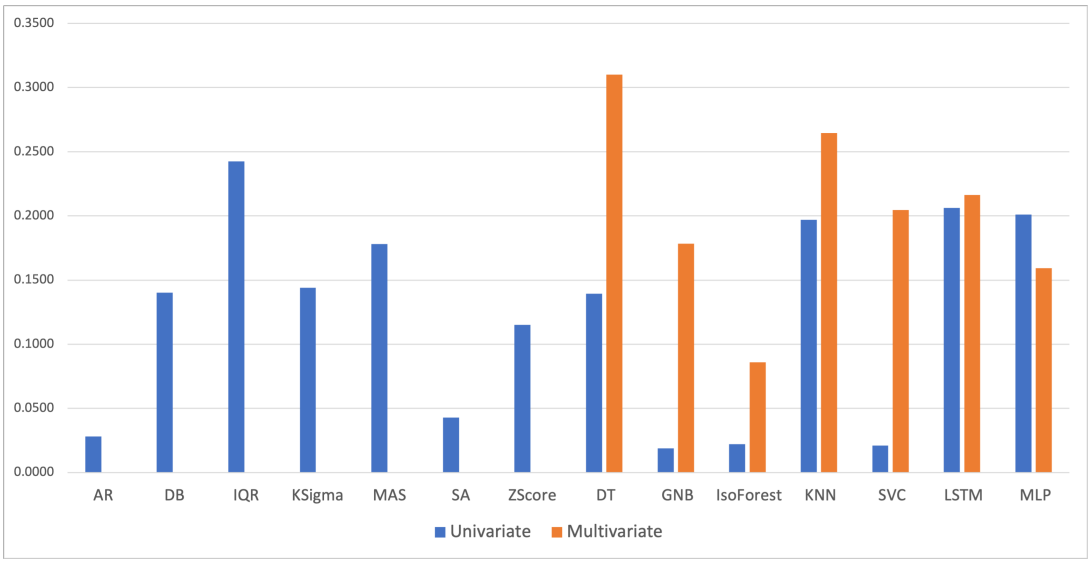


Figure 8.1: Bar charts of average F1-score different models with univariate and multivariate telemetry water level data.

In addition, as shown in Figure 8.1, we created a bar chart to compare the average F1-score from each model when evaluated with data from diverse data characteristics. DT models utilising multivariate data generated the highest average f1-score, which was about twice as high as univariate. Multivariate models outperformed univariate models, except for multivariate MLP which performed worse than univariate MLP. Multivariate LSTM slightly outperforms univariate LSTM. When compared to univariate SVC, multivariate SVC significantly increased the f1-score by about ten times.

With sliding windows technique.

Since our data sets are imbalanced time-series data. As a result, the normal train-test split or time series train-test split cannot divide data into train and test datasets with the same normal/anomaly ratio. As a consequence, we use the sliding windows approach described in Chapter 4 to construct the sliding windows. Furthermore, we only use this strategy with ML models since statistics and NN models already use it. Table 8.4 displays the effects of training using this strategy.

Table 8.4: The f1-score of machine learning model with univariate data when used and not used sliding windows techniques.

Data Behaviour	Station	DT		GNB		IsoForest		KNN		SVC	
		Uni	Uni-SW	Uni	Uni-SW	Uni	Uni-SW	Uni	Uni-SW	Uni	Uni-SW
Irrigation	DIV001	0.0000	0.3543	0.0000	0.0010	0.0000	0.0000	0.0000	0.0500	0.0000	0.0000
	PIN002	0.0000	1.0000	0.0000	0.0000	0.0000	0.0009	0.0000	0.0000	0.0000	0.0000
	PIN003	0.0000	0.9028	0.0000	0.0507	0.0007	0.0111	0.0474	0.9285	0.0000	0.6294
	VLGE13	0.0000	0.3000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000
	YOM012	0.0000	0.2405	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000
	Avg	0.0000	0.5595	0.0000	0.0103	0.0001	0.0024	0.0095	0.1957	0.0000	0.1259
	Std	0.0000	0.3616	0.0000	0.0226	0.0003	0.0049	0.0212	0.4102	0.0000	0.2815
Rain	CHM001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000
	CHM003	0.0000	0.3919	0.0000	0.0000	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000
	CHM005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM006	1.0000	0.8961	0.0000	0.0822	0.0004	0.0374	0.0000	0.9208	0.0000	0.8462
	CHR004	0.2500	0.2551	0.0000	0.0000	0.0000	0.0002	0.0000	0.0111	0.0000	0.0000
	CHR005	0.0000	0.2000	0.0000	0.0000	0.0000	0.0003	0.0000	0.0000	0.0000	0.0000
	CPY001	0.0000	0.0000	0.0000	0.0000	0.0004	0.0004	0.0000	0.0000	0.0000	0.0000
	CPY002	0.0000	0.9000	0.1733	0.2000	0.0010	0.0007	0.0000	0.0000	0.0000	0.0000
	CPY004	0.0405	0.7659	0.0000	0.0000	0.0020	0.0025	0.0253	0.8289	0.0000	0.7924
	CPY005	0.0000	0.6667	0.0000	0.0000	0.0005	0.0000	0.0000	0.0000	0.0000	0.0000
	CPY006	0.0000	0.1175	0.0000	0.0000	0.0015	0.0002	0.0000	0.0000	0.0000	0.0000
	DIV004	0.0000	0.7129	0.0000	0.0098	0.0000	0.0004	0.0000	0.8223	0.0000	0.5960
	PIN005	0.0002	0.9996	0.0000	0.9994	0.1154	0.3082	0.9321	0.9995	0.0231	0.9995
	PIN006	0.8974	1.0000	0.0468	1.0000	0.1379	0.3071	0.9394	1.0000	0.0044	1.0000
	WAN001	0.0000	0.9259	0.0000	0.2859	0.0726	0.0880	0.9505	0.9271	0.0000	0.4009
	WAN002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000
	WAN003	0.0000	0.0000	0.0000	0.0002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	WAN005	0.0000	0.7867	0.0000	0.0030	0.0040	0.0042	0.2725	0.6777	0.0000	0.0000
	YOM009	0.2734	0.8866	0.0000	0.0726	0.0187	0.0057	0.9472	0.8909	0.0000	0.6883
	YOM010	0.9491	0.9200	0.4064	0.5918	0.2860	0.2742	0.9446	0.9473	0.6642	0.8484
	Avg	0.1624	0.4964	0.0298	0.1545	0.0305	0.0491	0.2386	0.3822	0.0329	0.2939
	Std	0.3385	0.4097	0.0945	0.3148	0.0708	0.1056	0.4076	0.4557	0.1447	0.4024
Tidal	CPY011	0.0917	0.7720	0.0000	0.0000	0.0030	0.0027	0.0083	0.7874	0.0000	0.7566
	CPY012	0.4386	0.7595	0.0000	0.0840	0.0198	0.0415	0.3849	0.8046	0.0000	0.7612
	CPY013	0.1982	0.7801	0.0000	0.0262	0.0310	0.0646	0.0821	0.8172	0.0000	0.6437
	CPY014	0.0845	0.8174	0.0000	0.1754	0.0019	0.0089	0.0126	0.8368	0.0000	0.6017
	CPY015	0.0000	0.7025	0.0000	0.0650	0.0012	0.0052	0.0000	0.7723	0.0000	0.7545
	CPY016	0.0895	0.9290	0.0000	0.0000	0.0203	0.0481	0.5911	0.9531	0.0000	0.8821
	CPY017	0.2894	0.8189	0.0000	0.0000	0.0143	0.0236	0.3619	0.8265	0.0000	0.6769
	Avg	0.1703	0.7971	0.0000	0.0501	0.0131	0.0278	0.2058	0.8283	0.0000	0.7253
	Std	0.1506	0.0702	0.0000	0.0648	0.0114	0.0240	0.2377	0.0594	0.0000	0.0931
	Overall Avg	0.1395	0.5698	0.0190	0.1105	0.0222	0.0375	0.1970	0.4485	0.0210	0.3599
	Overall Std	0.2818	0.3702	0.0761	0.2577	0.0574	0.0859	0.3481	0.4426	0.1155	0.3916

The f1-score of every model has been improved using the sliding window approach. DT's f1-score on PIN002 got the perfect score of 1.0000. Although most classifier models used to identify anomalies in PIN006 data had an f1-score of 1.000, IsoForest had an f1-score of just 0.3071. DT has the highest overall average f1-score, followed by KNN and KSigma.

The performance of multivariate machine learning using sliding windows approaches was then compared, as shown in table 8.5. We can observe that when we use the sliding windows approach, the average f1-score of DT increases from 0.3102 to 0.4495, KNN increases from 0.2646 to 0.3074, and SVC increases from 0.2045 to 0.2466. On the other hand, GNB and IsoForest have seen their average f1-score slightly drop from 0.1783 to 0.1743, and 0.0859 to 0.0810, respectively. The DT with the sliding windows approach not only delivered the greatest overall f1-score but also performed the best when used to identify abnormalities in datasets that have been impacted by irrigation and rain.

Table 8.5: The f1-score of machine learning model with multivariate data when used and not used sliding windows techniques.

Data Behaviour	Station	DT		GNB		IsoForest		KNN		SVC	
		Multi	Multi-SW	Multi	Multi-SW	Multi	Multi-SW	Multi	Multi-SW	Multi	Multi-SW
Irrigation	DIV001	0.0000	0.2821	0.0000	0.0253	0.0000	0.0038	0.0000	0.1400	0.0000	0.0000
	PIN002	0.0000	0.3000	0.0000	0.0000	0.0000	0.0006	0.0000	0.0000	0.0000	0.0000
	PIN003	0.2718	0.4096	0.1298	0.3611	0.0143	0.1173	0.0000	0.2976	0.0333	0.1661
	VLGE13	0.0000	0.1000	0.0000	0.0066	0.0000	0.0013	0.0000	0.0000	0.0000	0.0000
	YOM012	0.0000	0.0750	0.0041	0.0012	0.0007	0.0009	0.0000	0.0000	0.0000	0.0000
	Avg	0.0544	0.2334	0.0268	0.0788	0.0030	0.0248	0.0000	0.0875	0.0067	0.0332
	Std	0.1216	0.1421	0.0576	0.1581	0.0063	0.0517	0.0000	0.1322	0.0149	0.0743
Rain	CHM001	0.0000	0.3667	0.0000	0.0000	0.0000	0.0017	0.0000	0.0000	0.0000	0.0000
	CHM002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0004	0.0000	0.0000	0.0000	0.0000
	CHM003	0.0000	0.1749	0.0000	0.0217	0.0000	0.0008	0.0000	0.0000	0.0000	0.0000
	CHM005	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	CHM006	0.1249	0.7152	0.0132	0.5248	0.0020	0.1827	0.0000	0.6170	0.1019	0.6849
	CHR004	0.0000	0.2186	0.0000	0.0228	0.0000	0.0044	0.0000	0.0732	0.0000	0.0000
	CHR005	0.0000	0.0667	0.0000	0.0249	0.0000	0.0016	0.0000	0.0000	0.0000	0.0000
	CPY001	0.0000	0.0548	0.0000	0.0016	0.0022	0.0004	0.0000	0.0000	0.0000	0.0000
	CPY002	0.3000	0.0667	0.0675	0.0000	0.0010	0.0011	0.0000	0.0000	0.0000	0.0000
	CPY004	0.8699	0.8938	0.5557	0.3623	0.0439	0.0434	0.8893	0.8711	0.8913	0.7161
	CPY005	0.3090	0.7000	0.0208	0.0000	0.0028	0.0006	0.2000	0.0000	0.2267	0.0000
	CPY006	0.6267	0.2413	0.0292	0.0119	0.0024	0.0013	0.4991	0.0000	0.4667	0.0000
	DIV004	0.0000	0.6385	0.0000	0.1079	0.0000	0.0075	0.0000	0.3698	0.0000	0.0000
	PIN005	0.5768	0.8179	0.1367	0.1888	0.4590	0.2538	0.5707	0.6675	0.0757	0.2446
	PIN006	0.8653	0.9112	0.3426	0.5382	0.4572	0.3810	0.8325	0.8393	0.1992	0.6983
	WAN001	0.1936	0.4468	0.2352	0.3469	0.1530	0.1388	0.2920	0.2609	0.1430	0.1743
	WAN002	0.0000	0.5067	0.0000	0.0009	0.0000	0.0004	0.0000	0.0000	0.0000	0.0000
	WAN003	0.0000	0.2567	0.0000	0.0230	0.0000	0.0007	0.0000	0.0000	0.0000	0.0000
	WAN005	0.0733	0.2925	0.0452	0.0454	0.0047	0.0079	0.0000	0.2604	0.0000	0.0211
	YOM009	0.5493	0.6148	0.4309	0.4084	0.3028	0.3008	0.5227	0.4630	0.2899	0.4428
	YOM010	0.5137	0.5811	0.4919	0.5246	0.3551	0.3461	0.5403	0.5047	0.2643	0.5372
	Avg	0.2382	0.4078	0.1128	0.1502	0.0851	0.0798	0.2070	0.2346	0.1266	0.1676
	Std	0.3040	0.3017	0.1829	0.2045	0.1598	0.1305	0.3041	0.3066	0.2188	0.2697
Tidal	CPY011	0.8528	0.8168	0.4675	0.3025	0.0419	0.0342	0.7759	0.8165	0.7646	0.8408
	CPY012	0.7920	0.8011	0.5756	0.4240	0.2504	0.1902	0.7238	0.8342	0.6977	0.8239
	CPY013	0.8190	0.8649	0.6032	0.4037	0.2827	0.2175	0.6282	0.8243	0.5789	0.7868
	CPY014	0.7256	0.8171	0.3876	0.2118	0.0788	0.0678	0.6280	0.8274	0.6403	0.7763
	CPY015	0.3986	0.3830	0.1978	0.0481	0.0110	0.0097	0.2892	0.2660	0.2002	0.1151
	CPY016	0.6728	0.6694	0.5719	0.4616	0.2467	0.2365	0.6690	0.5949	0.5486	0.5679
	CPY017	0.7012	0.7515	0.5775	0.3510	0.1216	0.1172	0.6705	0.6170	0.6260	0.5420
	Avg	0.7089	0.7291	0.4830	0.3147	0.1476	0.1247	0.6264	0.6829	0.5795	0.6361
	Std	0.1515	0.1648	0.1475	0.1441	0.1110	0.0914	0.1578	0.2111	0.1821	0.2596
	Overall Avg	0.3102	0.4495	0.1783	0.1743	0.0859	0.0810	0.2646	0.3074	0.2045	0.2466
	Overall Std	0.3351	0.3001	0.2283	0.1982	0.1421	0.1160	0.3228	0.3334	0.2775	0.3212

Figure 8.2 shows bar charts of average F1-score for comparing the performance of each machine learning on univariate and multivariate data with and without the sliding windows technique. DT with univariate sliding windows outperforms candidate models. Furthermore, sliding window approaches have the potential to increase the performance of all models, except GNB and IsoForest. DT, KNN, and SVC with univariate sliding windows outperformed univariate, multivariate, and multivariate with sliding windows.

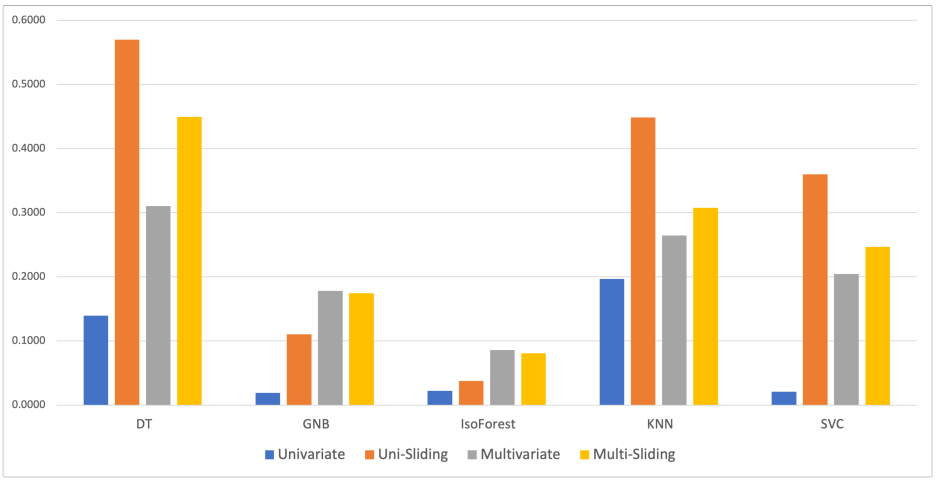


Figure 8.2: Bar charts of average F1-score of different machine learning models with and without the sliding windows techniques.

As illustrated in Figure 8.3, we next do a statistical comparison using a critical difference diagram to compare the accuracy of each approach. The highest ranking went to DT with univariate sliding windows (DT-UW), followed by DT with multivariate sliding windows (DT-MW) and KNN with univariate sliding windows (KNN-UW). Meanwhile, SVC-U was ranked last, followed by GNB-U and SA-U, respectively. In other words, the machine learning models with the sliding windows technique achieved the best. Only IQR is an exception among the statistical models, which reach high accuracy.

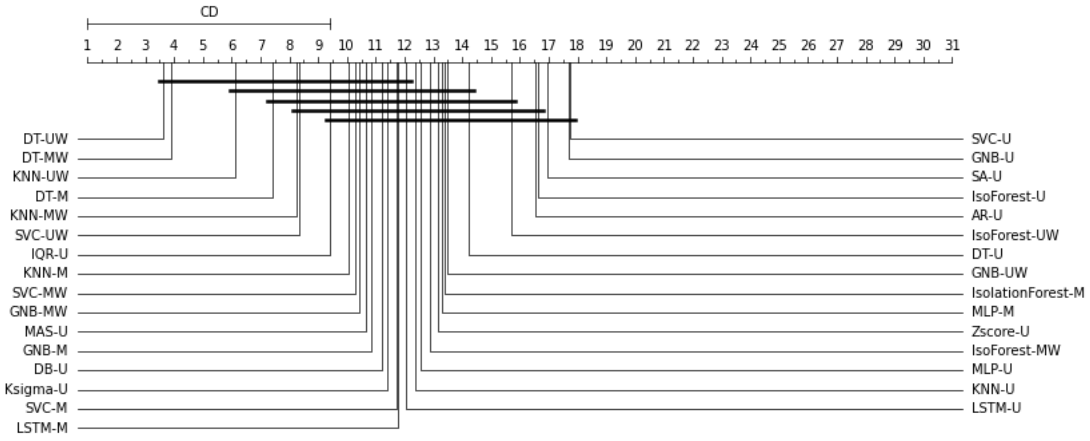


Figure 8.3: A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW).

However, when we consider the data characteristics, as depicted in Figure 8.4 - 8.6. We found that, although DT-UW had the highest ranking on data with irrigation-influenced, it had no significant difference with other models. Meanwhile, data with rain-influenced, DT-UW generated the highest ranking, followed by DT-MW, with a significant difference from SA-U, SVC-U, and GNB-U. In the case of data with tidal-influenced, KNN-UW provided the best, with the highest ranking, followed by DT-MW and DT-M.

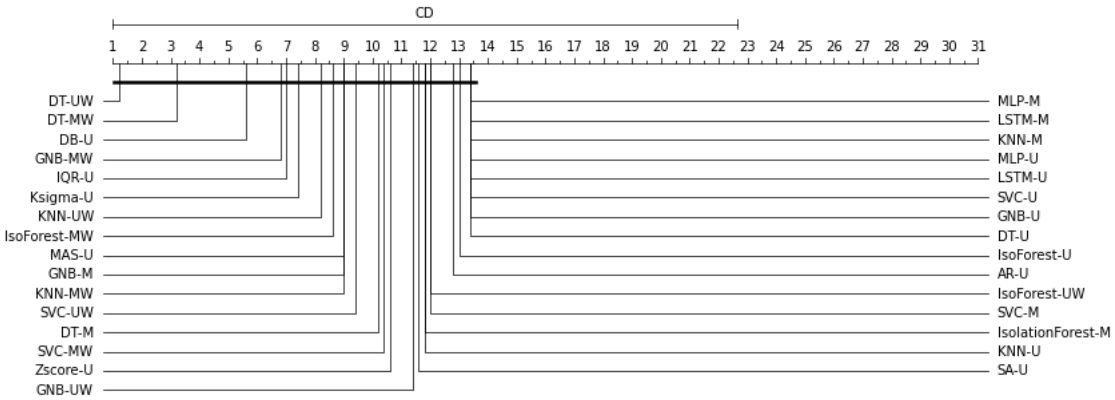


Figure 8.4: A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW) on irrigation data behaviours.

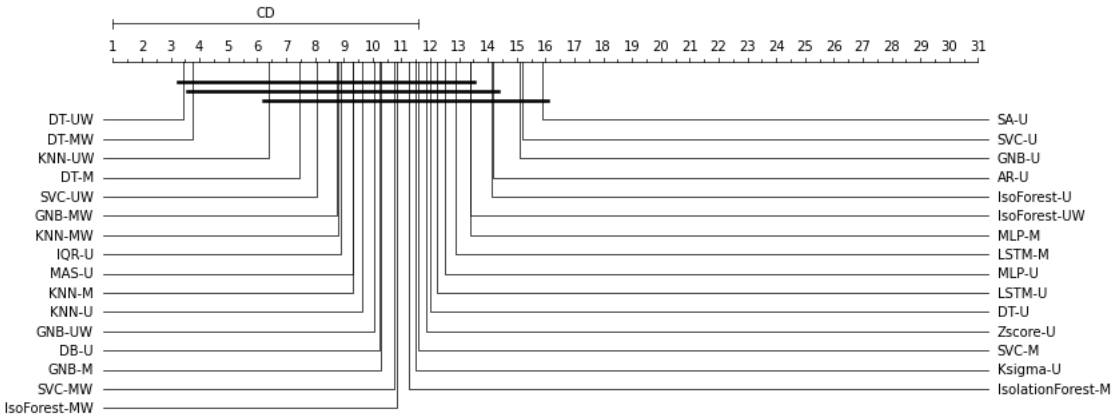


Figure 8.5: A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW) on rain data behaviours.

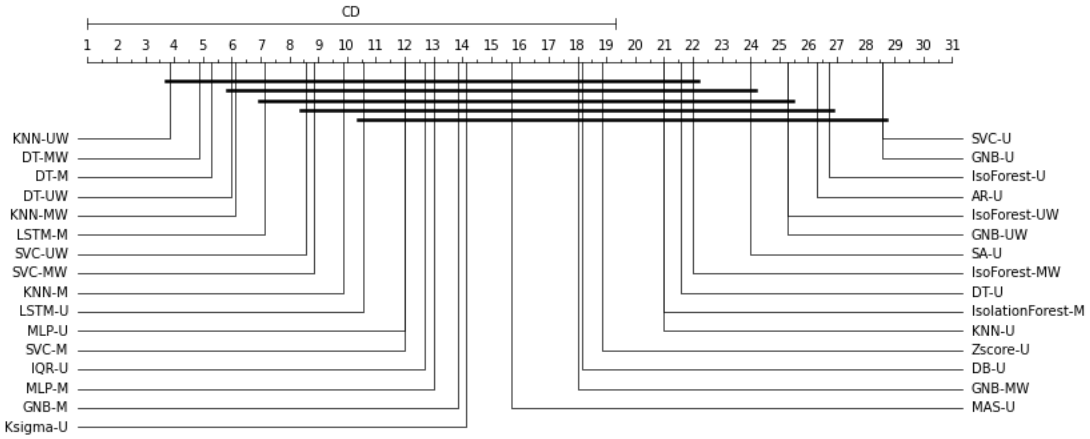


Figure 8.6: A critical difference diagram of all models with univariate (U), multivariate (M), univariate sliding windows (UW), and multivariate sliding windows (MW) on tidal data behaviours.

Comparison with External Datasets

To evaluate our anomaly detection methods, we selected Yahoo published time-series datasets (Yahoo! Webscope, 2010) which consist of real and synthesis data, and have characteristics like our telemetry water level datasets for comparison. The data can be separated into four main datasets, which are detailed as follows:

- **A1**: This is a univariate time-series dataset that contains traffic to Yahoo services. The abnormalities are identified by humans. This data set has 67 different time-series, and each one has about 1400 timestamps. 1.9% of the timestamps are anomalies.

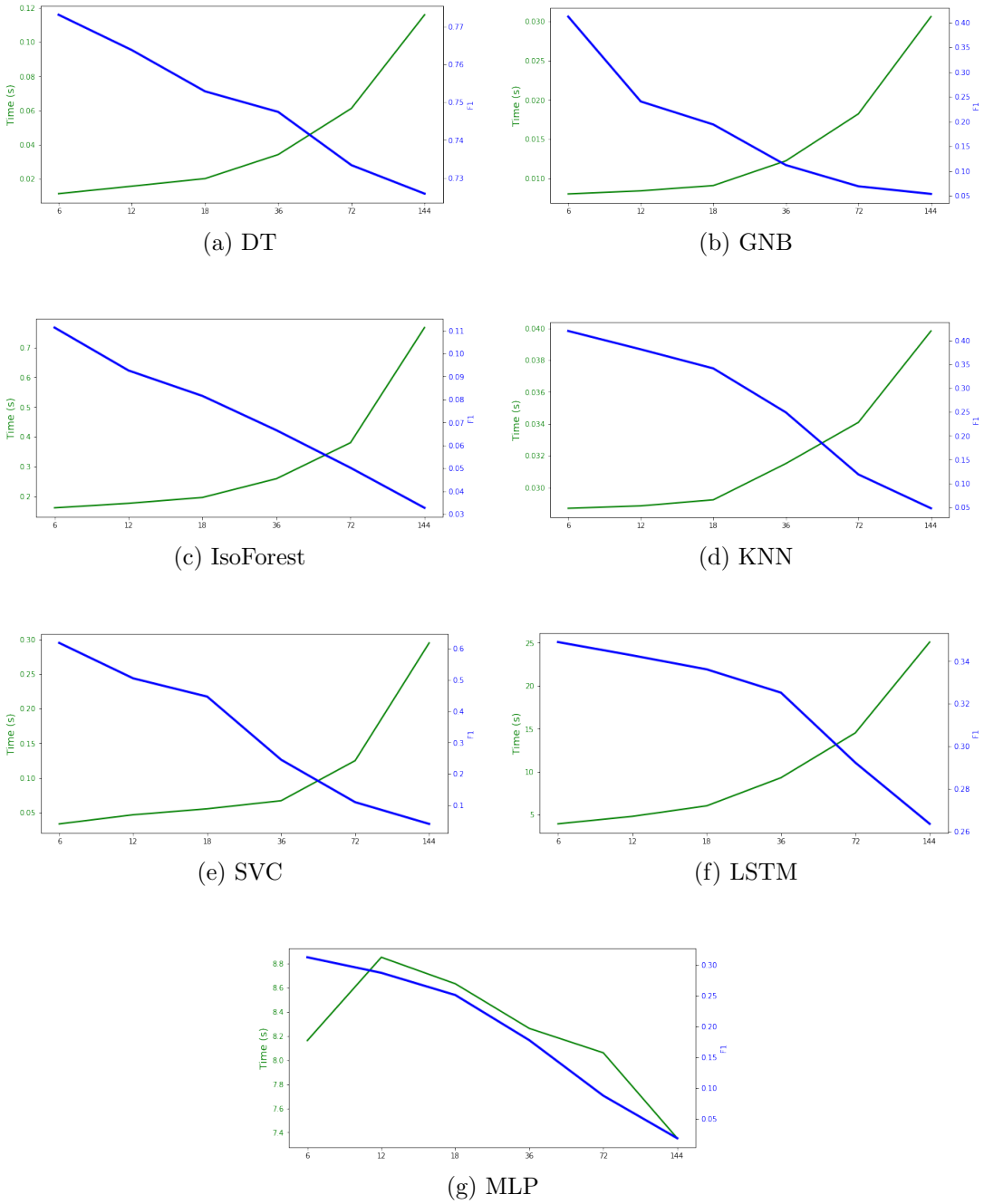
- **A2**: This dataset contains 100 synthetic univariate time-series data with anomalies. Each time series has around 1421 timestamps. The anomalies were inserted at random, therefore representing point anomalies. Each time series has 0.3% anomalies on average.

- **A3**: This dataset also includes 100 synthetic univariate time series with around 1680 timestamps per series. The anomalies are placed at random locations to indicate the change-points, and the rate of anomaly in each time series is around 0.3%.

- **A4**: The dataset also includes one hundred synthetic univariate time series. Each time series consists of around 1680 timestamps. An average of 0.5% of the dataset

is change point anomalies.

For our evaluation, we focus on the main anomalous points and ignore distinguishing between anomalous types. In addition, since we use the sliding window approach, each dataset is rather tiny. Then, we test the influence of window size on each model's accuracy by using several window sizes. We were tested with window sizes of 6, 12, 18, 36, 72, and 144 and found that window size 6 provided the best, as shown in Figure 8.7.



(a) DT

(b) GNB

(c) IsoForest

(d) KNN

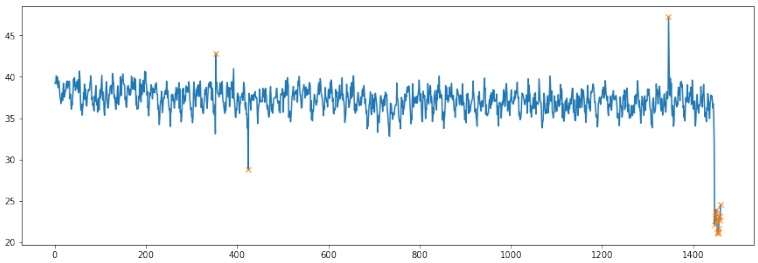
(e) SVC

(f) LSTM

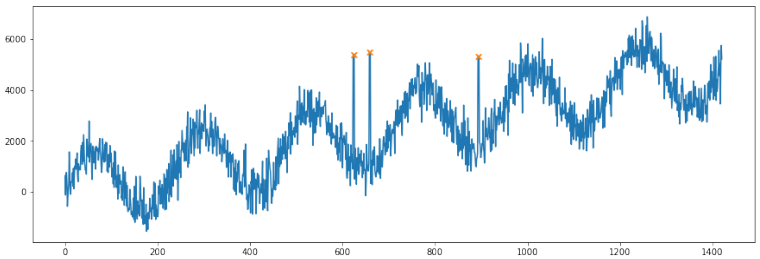
(g) MLP

Figure 8.7: The accuracy of each model on different windows size.

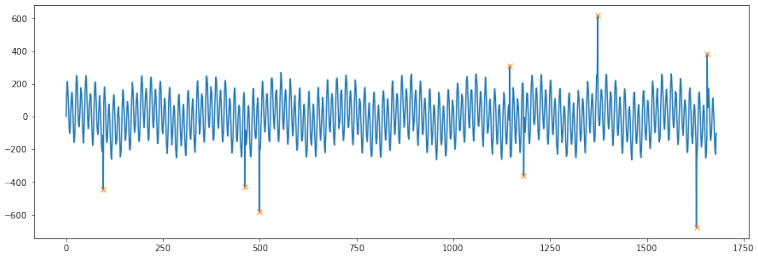
Figure 8.8 shows an example of data from the Yahoo dataset. We conducted experiments with our statistical models, ML models, and NN models using univariate, multivariate, and sliding window techniques.



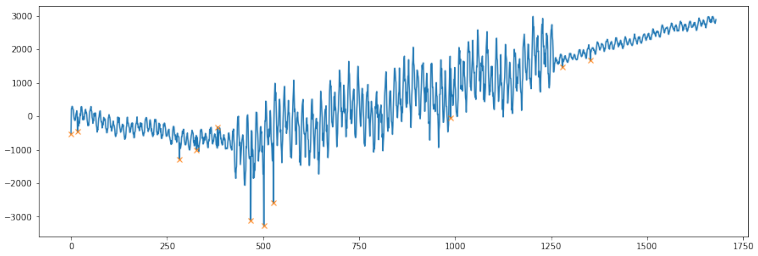
(a) A1



(b) A2



(c) A3



(d) A4

Figure 8.8: Example data from Yahoo datasets with an anomaly on the orange cross, x (**a-d**).

The average F1-score of each model with the best window size is shown in Table 8.6. With an average F1-score of 0.4915, IQR produced the highest results among statistical models, followed by Zscore with an average F1-score of 0.3073, while AR did badly with

Table 8.6: The F1-score of different models on the Yahoo dataset.

Dataset	Statistical Models						
	AR	DB	IQR	KSigma	MAS	SA	ZScore
A1	0.0046	0.0409	0.3493	0.0197	0.2292	0.0752	0.3342
A2	0.0000	0.0028	0.5818	0.0000	0.3227	0.0069	0.2601
A3	0.0000	0.0051	0.5534	0.0000	0.0352	0.0113	0.3441
A4	0.0000	0.0033	0.4815	0.0000	0.0301	0.0095	0.2909
Avg	0.0011	0.0130	0.4915	0.0049	0.1543	0.0257	0.3073
Std	0.0023	0.0186	0.1038	0.0099	0.1456	0.0330	0.0390

	ML Models					NN Models	
	DT-U	GNB-U	IsoForest-U	KNN-U	SVC-U	LSTM-U	MLP-U
A1	0.1200	0.1466	0.2149	0.0243	0.5105	0.0261	0.1071
A2	0.0000	0.0000	0.0000	0.0000	0.6234	0.0000	0.0000
A3	0.1071	0.0869	0.0342	0.0107	0.0615	0.0000	0.1254
A4	0.0339	0.0601	0.0343	0.0021	0.0274	0.0000	0.0778
Avg	0.0653	0.0734	0.0709	0.0093	0.3057	0.0065	0.0776
Std	0.0577	0.0608	0.0974	0.0111	0.3055	0.0131	0.0553

	ML Models				
	DT-UW	GNB-UW	IsoForest-UW	KNN-UW	SVC-UW
A1	0.1269	0.0125	0.0821	0.1004	0.5315
A2	0.0000	0.0000	0.0159	0.0000	0.9012
A3	0.1370	0.0564	0.0172	0.0140	0.1075
A4	0.0553	0.0298	0.0149	0.0113	0.0406
Avg	0.0798	0.0247	0.0325	0.0314	0.3952
Std	0.0644	0.0244	0.0331	0.0464	0.4013

	ML Models					NN Models	
	DT-M	GNB-M	IsoForest-M	KNN-M	SVC-M	LSTM-M	MLP-M
A1	0.1462	0.0378	0.1820	0.0220	0.1796	0.1725	0.1225
A2	0.0000	0.0000	0.0000	0.0000	0.5661	0.0000	0.0000
A3	0.7144	0.4644	0.1146	0.6290	0.7096	0.7406	0.6860
A4	0.4518	0.2890	0.0842	0.3175	0.4125	0.4984	0.4422
Avg	0.3281	0.1978	0.0952	0.2421	0.4669	0.3529	0.3127
Std	0.3190	0.2192	0.0755	0.2957	0.2267	0.3309	0.3110

	ML Models				
	DT-MW	GNB-MW	IsoForest-MW	KNN-MW	SVC-MW
A1	0.517	0.2948	0.1784	0.2486	0.2235
A2	0.9186	0.4854	0.0897	0.0000	0.9355
A3	0.8929	0.5837	0.1022	0.8516	0.8046
A4	0.7641	0.2842	0.0791	0.5808	0.5105
Avg	0.7732	0.4120	0.1124	0.4203	0.6185
Std	0.1837	0.1471	0.0450	0.3732	0.3177

an average F1-score of just 0.0011. AR, DB, KSigma, and SA models performed poorly when identifying anomalies in datasets A2, A3, and A4 with an average maximum F1-score of 0.0113 and a minimum of 0.0000. Despite having the highest average F1-score among machine learning methods for univariate data, SVC performed worse than IQR. Similar to the poor performance of NN models on univariate data, with the maximum average F1-score from MLP being 0.0776. With the exception of GNB and IsoForest, the accuracy of univariate ML models has been enhanced by the use of sliding window approaches. With multivariate ML models, the accuracy of each model has grown dramatically, particularly KNN, which boosted the average F1-score by 26 times, from 0.0093 to 0.2421. Our work on feature extraction not only improved the ability for ML models to identify anomalies, but it also made NN models more accurate by improving the performance of LSTM and MLP by 54 and 4 times, respectively. However, data containing fluctuations, such as the A2 dataset, makes it difficult to identify anomalies, as shown by the fact that the majority of models had a low F1-score and some models had a F1-score of 0.0000. When sliding windows are used in multivariate ML models, accuracy has gone up a lot, especially for DT-MW, which has seen its F1-score double.

Figure 8.9 depicted a bar chart comparing the total average F1-score of each approach in desending order. In other words, better models are on the left, while worse models are on the right. The highest overall F1-score was obtained by DT-MW, followed by SVC-MW and IQR, in that order. Most of the multivariate models are on the left, whereas most of the univariate models are on the right.

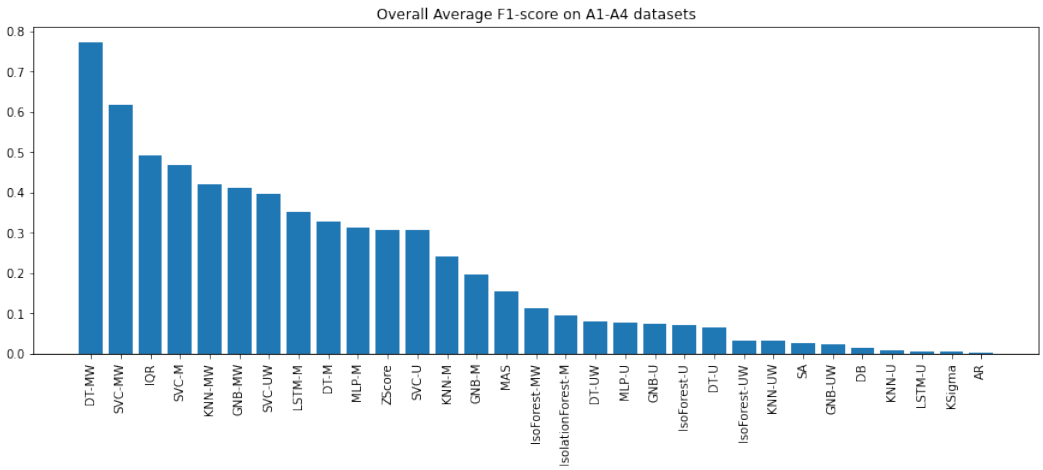


Figure 8.9: Overall average F1-Score from different anomaly detection techniques on A1-A4 datasets.

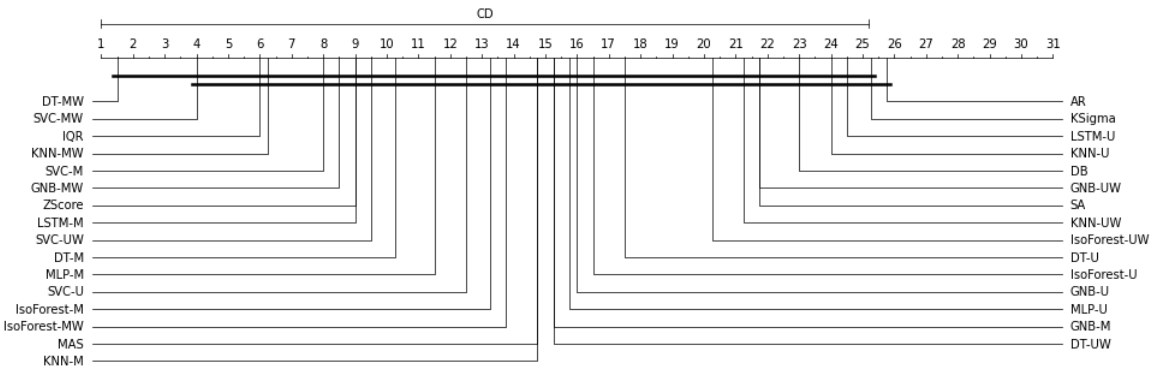


Figure 8.10: A critical difference diagram of all models with Yahoo time-series dataset.

However, the overall average F1-score may not represent the real potential of each model. The CD diagram is then plotted to establish statistical comparisons, as shown in Figure 8.10. We discovered that DT-MW had the highest ranking, followed by SVC-MW and IQR, respectively. Three of the top four are multivariate models with sliding windows. AR not only got the lowest score, but it was also significantly different from the model that got the best score, which was DT-MW.

8.3.3 Comparison of our data imputation methods with external methods.

We compared our methods with the work presented by Kulanuwat et al. (2021). They applied 3 methods for imputing missing data, including linear interpolation, spline interpolation, and bidirectional LSTM for data imputation on HII telemetry stations. Their findings showed that spline interpolation worked better on non-cyclical data, but bidirectional long short-term memory (BiLSTM) outperformed other interpolation techniques on a specific tidal data pattern. Our results of data imputation in Chapter 7 are similar to the results of them in the case of non-cyclical data (rain and irrigation behaviours). Moreover, from the literature review, KNN is one of the methods that is widely used for univariate data imputation. So we then compare our approach with BiLSTM and KNN on data with tidal behaviour from 4 stations (CPY011, GLF002, THA009, and TNG002) on data from 2020 to 2021. The data and the hyperparameter settings of BiLSTM are followed by the settings from their research. The results are shown in Table 8.7.

Table 8.7: The average RMSE of eight methods on telemetry water level data with tidal behaviour on different missing gap size.

Gap	PSM_F	PSM_B	PSM_A	PSM_W	KNN	FSM_D	FSM_S	$BiLSTM$
6	0.0707 (±0.03)	0.0786 (±0.05)	0.0572 (±0.03)	0.0593 (±0.04)	0.6990 (±1.19)	0.0277 (±0.01)	0.0240 (±0.01)	0.6655 (±1.15)
12	0.1305 (±0.07)	0.1221 (±0.05)	0.0949 (±0.05)	0.0987 (±0.06)	0.8857 (±1.52)	0.0674 (±0.06)	0.0790 (±0.11)	0.6246 (±0.79)
18	0.1667 (±0.07)	0.1851 (±0.09)	0.1312 (±0.06)	0.1288 (±0.06)	0.8556 (±1.40)	0.0735 (±0.04)	0.0562 (±0.05)	0.8290 (±1.13)
36	0.2697 (±0.09)	0.3371 (±0.14)	0.2375 (±0.09)	0.2137 (±0.07)	0.9902 (±1.42)	0.1460 (±0.05)	0.1165 (±0.07)	0.8532 (±1.35)
72	0.2724 (±0.14)	0.3018 (±0.18)	0.2451 (±0.15)	0.2488 (±0.13)	1.0258 (±1.39)	0.3008 (±0.12)	0.2227 (±0.10)	0.8731 (±1.51)
144	0.2811 (±0.20)	0.3005 (±0.19)	0.2388 (±0.18)	0.2528 (±0.19)	1.1103 (±1.68)	0.2848 (±0.19)	0.2105 (±0.15)	0.7144 (±1.20)
Avg	0.1985 (±0.13)	0.2209 (±0.15)	0.1674 (±0.12)	0.1670 (±0.12)	0.9278 (±1.28)	0.1500 (±0.14)	0.1181 (±0.11)	0.7600 (±1.07)

We can see that our technique FSM_S clearly beats candidate models with the lowest overall average RMSE of 0.1181. Furthermore, FSM_S performed best when imputed missing data with all gap sizes except gap size 12, where FSM_D performed better. KNN performed poorly, achieving the highest RMSE across all gap sizes. Similar to BiLSTM, which produced poor results with an overall average RMSE of 0.76.

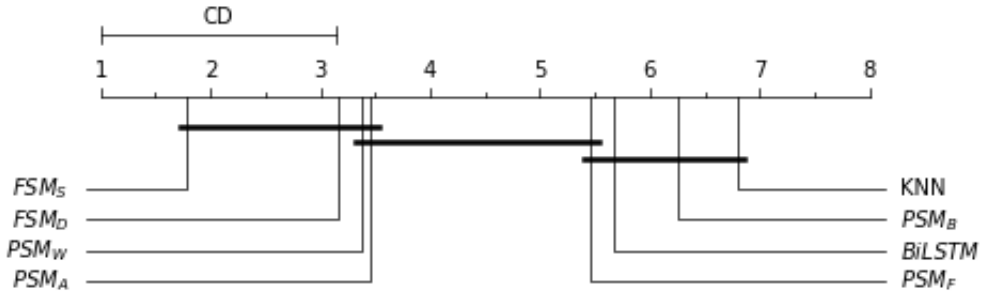


Figure 8.11: A critical difference diagram of our approach and candidate imputation models on tidal influence dataset of telemetry water level data.

The comparison of the critical difference between the various imputation models, as represented in Figure 8.11, validated the superior performance of our method in comparison to candidate models. Not only did FSM_S and FSM_D have the highest ranking, but they also had a statistically significant difference compared to KNN and BiLSTM.

We also plotted the RMSE results for all models, as demonstrated in Figure 8.12. As we can see, the performance of KNN decreased as the number of missing gaps increased. While the performance of BiLSTM models tends to decrease as the number of missing data points increases, their performance improves when the number of missing data points reaches 144. When the gap sizes are increased, our approach still retains a steady level with a slight drop in performance.

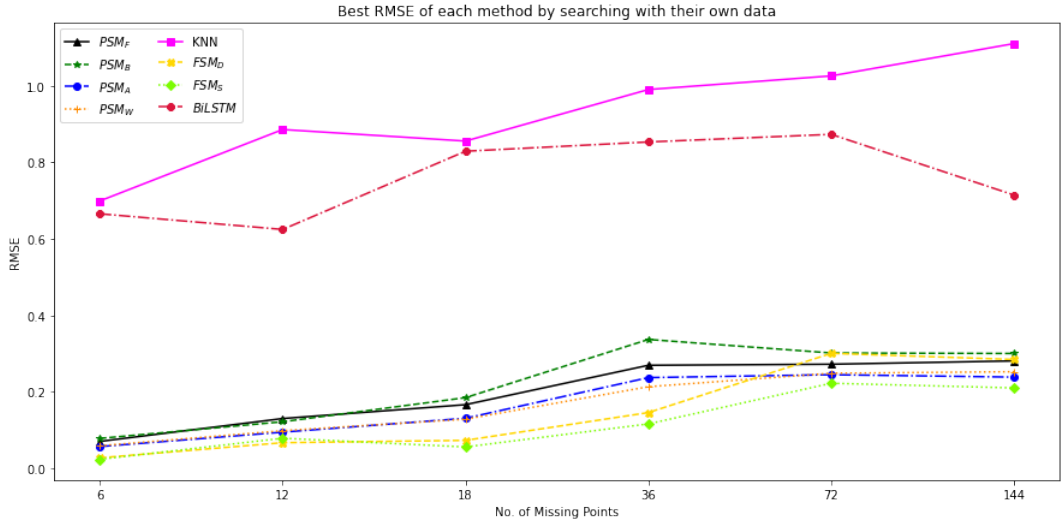


Figure 8.12: The RMSE for imputing telemetry water level data with tidal influence for various missing gap sizes.

8.4 Summary

In this chapter, we have presented a comprehensive overview of our methodology and the results of our four main experiments. We begin with the use of statistical models to detect anomalies, how applicable RL is for detecting anomalies, how to improve model accuracy with our feature extraction techniques, and the most similar pattern searching for data imputation. Finally, we compared all our experiments on anomaly detection approaches on various datasets while comparing our data imputation approaches with external methods.

In the case of anomaly detection, we observed that machine learning algorithms using the sliding window methodology outperform all others. Statistical techniques beat neural networks while performing worse than machine learning models. Despite the fact that neural network approaches have received a lot of attention from the artificial intelligence community, our findings show that they are not typically able to achieve higher accuracy than the best statistical model. In the case of data imputation, we compared our methods to those of other researchers by using their datasets as well as our own. We discovered that our methods produced better results.

The conclusions of this research are presented in the following chapter, along with suggestions for further work.

Chapter 9

Conclusion and Further Work

9.1 Summary

This study aimed to experimentally investigate the application of detecting and correcting errors in telemetry water level data. As we mentioned in Chapter 1, to ensure the validity, reliability, and effectiveness of early warning systems, the ability to promptly identify and correct data errors in telemetry water levels is critical. We investigated various existing anomaly detection and correction methods and also developed new methods to address those issues. Moreover, we applied ensemble techniques to improve the performance of individual models. We begin with an overall view of the thesis and then look at more specific conclusions. Our entire research is summarised next.

First, in Chapter 1, we identified some research question namely:

- How efficient are the statistical models in detecting anomalies in telemetry water level data?
- Is DRL applicable and effective for identifying abnormalities in water level data?
- Can multi features data improve the performance of anomaly detection models?
- How efficient is pattern-matching-based data imputation for imputing data errors in telemetry water level data?
- Can ensemble methods improve the performance of single models?

We also proposed specific objectives for carrying out our experimentation in order to

answer the predefined question and achieve our goals.

We next described details of telemetry systems and the description of datasets used in this research in Chapter 2. The issues of anomaly identification and correction in time series data were discovered by the current literature and are explored in Chapter 3. According to our review of the literature, machine learning is frequently employed in anomaly detection and correction in time-series data. However, most of them employ regression and prediction approaches to detect abnormalities, which requires time and effort to set the optimum threshold, which may not be adequate with the amount and frequency of data from telemetry water level stations. Furthermore, most studies used multivariate data, which is in contrast to our univariate data. We also noticed that minimal efforts had been made to identify and repair problems in real-time water level data. This prompted us to recognise that there was a need for further research in this area. Thus, there is a need to develop machine learning to detect and correct errors in telemetry water level data.

After that, in Chapter 4, we investigated the standard statistical models for detecting anomalies in water level data. We tested our initial research question by looking at how efficient the statistical models were in detecting anomalies in telemetry water level data. The experimental findings demonstrated that deployed statistical models performed poorly when detecting anomalies in flood events, such as a rapid increase in water level before a flood. Although the ensemble of statistical models may improve performance, they do not perform adequately. This supported the need for further work on other machine learning algorithms.

We next carried out an extended experiment to investigate the efficiency of deep reinforcement learning models and deep learning for detecting anomalies in Chapter 5. Here we answer the second research question regarding the ability of the proposed methods. We answer this by implementing reinforcement learning models and the state-of-the-art deep learning models (MLP and LSTM) and comparing their performance. Our finding showed that in general, deep learning performs better than reinforcement learning, not only in terms of performance but also time spent. Although the accuracy of basic deep learning models is not sufficient in data from certain stations, it may be improved with

parameter adjustment and the addition of hidden layers. As a result, deep learning appears to be the preferred option for dealing with anomaly detection. Also, generated ensemble models can improve the detection of anomalies in water level data, making them more accurate and reliable, including reducing the number of false alarms.

We also tested here our third question related to the effect of the multivariate variable on the algorithm performance. We then developed the new techniques to extract more features from water level data to improve the performance of anomaly detection models. We developed a nearest neighbour algorithmic method, in conjunction with a saliency map, so called SM+NNFE, to extract more features for improving the performance of anomaly detection models in Chapter 6. We compared the performances by training the models with only water level data and the extracted features from SM+NNFE. In comparison with one of the existing state-of-the-art time series feature extraction library. Furthermore, ensemble models were also used, using weighted ensemble voting. The results proved that our approach not only increased the performance of learning algorithms but also outperformed the candidate feature extraction library in terms of accuracy and time efficiency. Also, the idea of ensemble techniques shows that they can not only improve the performance of a single model but also improve the performance of a group of models.

Finally, in Chapter 7, we proposed a technique for imputing telemetry water level data with subsequence matching to test the fourth research question. Instead of splitting the subsequence in two, the missing data are temporarily replaced with some constant values to produce a dummy full subsequence, and then a sliding window is used to search for the most similar historical data subsequence. The identified subsequence will be adapted to fit the missing part based on their similarity. We found out that our developed methods outperforms other imputation methods when dealing with large missing gap sizes and has strong periodic pattern such as data of water level with tidal-influenced.

To find the answer to our final research question on ensemble techniques' abilities. For each experiment, ensemble models were developed, and they all showed that the idea of ensemble approaches can improve not only the performance of a single model but

also the performance of a group of models.

9.2 Conclusion

Upon completing this research on two main issues: anomaly detection and anomaly correction, the following conclusion can be drawn.

9.2.1 Anomaly Detection

The first focus of this thesis is to detect anomalies in telemetry water level data. We first considered the statistical models to determine how well those methods worked when dealing with real-world datasets. These results showed that they are unsuitable for this purpose due to a high frequency of false alarms because the water level will dramatically rise before the flood, which differs considerably from the other data points. As a consequence, the majority of statistical models will identify such points as anomalous, but in fact they are normal points, which provides very useful information in flood management.

So, we then developed reinforcement learning based models to solve such limitations and compared them with the deep learning models. These findings reveal that the reinforcement learning models not only provided the best performance but also had the ability to detect unseen anomalies. However, the rewards function and determining the right training duration are the most challenging and lead to time-consuming problems. Although deep learning models perform well, the neural structure that works well with one station may not function well with another. So, one of the problems with this topic is figuring out which neural structure is best for each station.

However, with the time-consuming of RL that is not appropriate for use with real-time data, we have developed a feature extraction algorithm called SM+NNFE that combines the saliency map and nearest neighbour extracted feature to improve the performance of ML models. The results demonstrated that when models are given more information to learn, their performance improves.

Individual models built by certain learning algorithms may learn separate sections of

the issue and make different decisions. Therefore, it is best to combine different models in order to work together. This method inspired us to develop an ensemble model in order to improve the performance of our model. The results show that ensemble methods can increase the accuracy of anomaly detection in water level data by selecting suitable models and final decision-making.

In conclusion, machine learning with sliding windows technique is the first option to use for detecting anomalies in telemetry water level data. Moreover, we can apply a reinforcement learning model to detect anomalies in critical stations that need more attention. Although deep learning and reinforcement learning models need time to train and fine-tune parameters, they can detect anomalies in real-time. As a result, we can schedule batch processing training and tuning at regular intervals, such as weekly, monthly, and so on. Statistical models, on the other hand, work well with slightly changed data, making them ideal for detecting abnormalities during the dry season. We also developed the firmware of the telemetry station to be able to detect anomaly values by itself with the statistical model. But depending on just one or a few models may be too risky since their performance may diminish in the future without awareness. Thus, built models using ensemble techniques are effective solutions for reducing errors from such issues.

9.2.2 Anomaly Correction

In the case of data imputation, the water level is seasonal, which means it has a similar pattern in each season. With those characteristics, we can impute the missing data by imitating the most similar pattern from the previous data. We created a technique called full subsequence matching (FSM) for locating a subsequence that contains missing data without breaking it into two parts. Experimental results show that our approach outperforms other methods when dealing with data that has a strong periodic pattern (tidal effect). However, accuracy is determined by the amount and quality of past data. But for data that has few changes or patterns, like data from stations with rain or irrigation effects, linear or polynomial interpolation are the appropriate methods. It not only provided good results, but also less computed cost

and less time-consuming.

9.3 Suggestions for Further Work

This research has highlighted a number of areas that could be explored further in the future; these are:

1. We aim to develop this system in a real-world setting in order to test the capability of our approach. The overview system is depicted in Figure 9.1. Telemetry water level data was sent from the telemetry station to the data centre via the MQTT protocol. The raw data was saved in the raw database, and the data that was found to be an anomaly by the anomaly detection service, which finds anomalies in real time, was stored in the cleaned database and marked as an anomaly. The data imputation service then imputes the anomalous and missing data from the cleaned database at regular intervals (depending on user settings, such as daily, weekly, or monthly) and stores it in the completed database. Users can access data through an online service, which can be utilised for a number of objectives. For example, a researcher may need raw data or data that has been cleaned for further analysis, while other users may use completed data to make a report. However, it is difficult to assess how effectively our model works in the real world since it requires a human to compare the outcomes of what the model accomplishes to what occurs in the real world. So, feedback from online users and labelled data from a database like this are the best ways to measure how well our processes are working. This is especially helpful for figuring out when our system needs to be retrained.

2. Nowadays, HII established the "National Hydroinformatics Data Center" (NHC)[1] to serve as a focal point for the integration of national water resources information in order to maximise benefits in managing water resources, analysing data, forecasting, and assisting decision-making in normal and crisis situations. When data is pooled from several sources, it might be inconsistent since they use various types of sensors for different purposes, making identifying abnormalities more

[1] `https://www.thaiwater.net/`

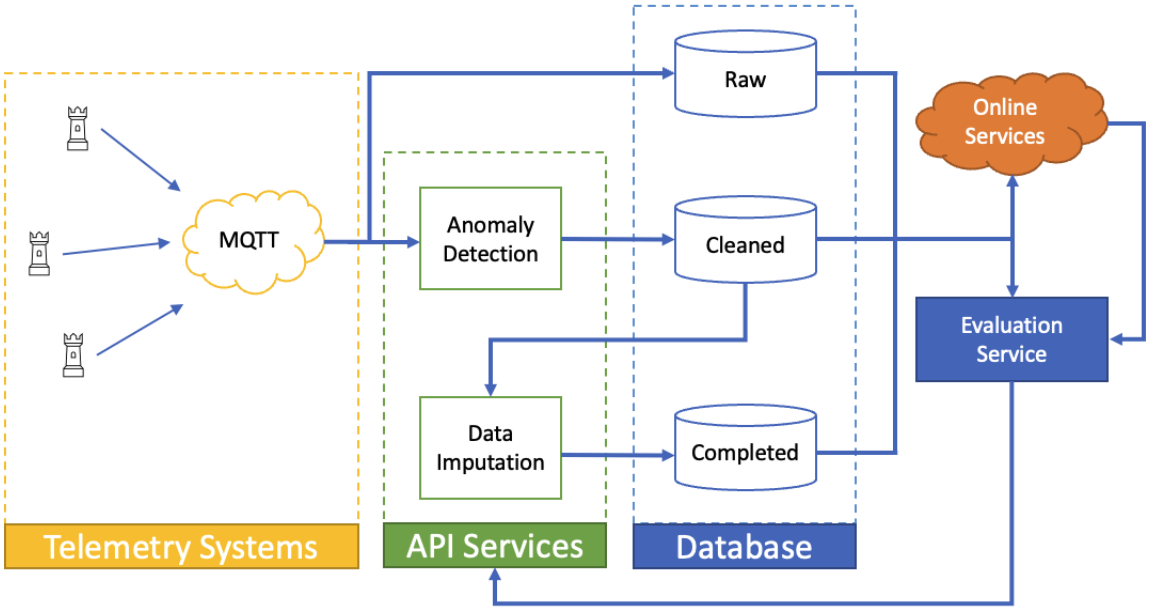


Figure 9.1: System Overview

challenging. This will be the challenge of our future study to identify anomalies from different sources.

3. HII has developed survey technologies such as the seafloor echoboat unmanned surface vessel (USV), the mobile mapping system (MMS), and the points cloud map. The data has been collected, which is the kinds of time series, with multiple kinds of scanners (e.g., laser and radar) that may have anomalies. So, we might adapt our study to detect and correct any anomalous data in the survey method.

Bibliography

Aggarwal, C. C. (2017). An introduction to outlier analysis. In *Outlier analysis*, pages 1–34. Springer.

Akouemo, H. N. and Povinelli, R. J. (2014). Time series outlier detection and imputation. In *2014 IEEE PES General Meeting— Conference & Exposition*, pages 1–5. IEEE.

Alasadi, S. A. and Bhaya, W. S. (2017). Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107.

Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367.

Apostol, E.-S., Truică, C.-O., Pop, F., and Esposito, C. (2021). Change point enhanced anomaly detection for iot time series data. *Water*, 13(12):1633.

Asesh, A. (2022). Normalization and bias in time series data. In *Digital Interaction and Machine Intelligence: Proceedings of MIDI'2021–9th Machine Intelligence and Digital Interaction Conference, December 9-10, 2021, Warsaw, Poland*, pages 88–97. Springer.

Atienza, R. (2018). *Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more*. Packt Publishing Ltd.

Bank, W. (2012). *Thai Flood 2011: Rapid assessment for resilient recovery and reconstruction planning*. World Bank.

Basu, S. and Meckesheimer, M. (2007). Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154.

Berkhahn, S., Fuchs, L., and Neuweiler, I. (2019). An ensemble neural network model for real-time prediction of urban floods. *Journal of hydrology*, 575:743–754.

Bernacki, J. and Kołaczek, G. (2015). Anomaly detection in network traffic using selected methods of time series analysis. *IJ Computer Network and Information Security*, 9:10–18.

Blázquez-García, A., Conde, A., Mori, U., and Lozano, J. A. (2020). A review on outlier/anomaly detection in time series data. *arXiv preprint arXiv:2002.04236*.

Bokde, N., Beck, M. W., Álvarez, F. M., and Kulat, K. (2018). A novel imputation methodology for time series based on pattern sequence forecasting. *Pattern recognition letters*, 116:88–96.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Braei, M. and Wagner, S. (2020). Anomaly detection in univariate time-series: A survey on the state-of-the-art. *arXiv preprint arXiv:2004.00433*.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104.

Caillault, E. P., Bigand, A., et al. (2016). Comparative study on supervised learning methods for identifying phytoplankton species. In *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, pages 283–288. IEEE.

Caillault, É. P., Lefebvre, A., Bigand, A., et al. (2020). Dynamic time warping-based imputation for univariate time series data. *Pattern Recognition Letters*, 139:139–147.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. (2018). Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31.

Casciaro, G., Ferrari, F., and Mazzino, A. (2021). Comparing novel strategies of ensemble model output statistics (emos) for calibrating wind speed/power forecasts. *arXiv preprint arXiv:2108.12174*.

Chang, L.-C., Chang, F.-J., Yang, S.-N., Kao, I.-F., Ku, Y.-Y., Kuo, C.-L., and Amin, I. M. Z. b. M. (2018). Building an intelligent hydroinformatics integration platform for regional flood inundation warning systems.

Chang, L.-C., Chang, F.-J., Yang, S.-N., Tsai, F.-H., Chang, T.-H., and Herricks, E. E. (2020). Self-organizing maps of typhoon tracks allow for flood forecasts up to two days in advance. *Nature communications*, 11(1):1–13.

Chang, L.-C., Liou, J.-Y., and Chang, F.-J. (2022). Spatial-temporal flood inundation nowcasts by fusing machine learning methods and principal component analysis. *Journal of Hydrology*, 612:128086.

Chen, J., Sathe, S., Aggarwal, C., and Turaga, D. (2017). Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 90–98. SIAM.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Chiang, H.-T., Hsieh, Y.-Y., Fu, S.-W., Hung, K.-H., Tsao, Y., and Chien, S.-Y. (2019). Noise reduction in ecg signals using fully convolutional denoising autoencoders. *IEEE Access*, 7:60806–60813.

Christ, M., Braun, N., Neuffer, J., and Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77.

Ciocarlie, G. F., Lindqvist, U., Nováczki, S., and Sanneck, H. (2013). Detecting anomalies in cellular networks using an ensemble method. In *Proceedings of the 9th international conference on network and service management (CNSM 2013)*, pages 171–174. IEEE.

Cui, X., Liu, Q., and Metaxas, D. (2009). Temporal spectral residual: fast motion saliency detection. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 617–620.

Curiac, D.-I. and Volosencu, C. (2012). Ensemble based sensing anomaly detection in wireless sensor networks. *Expert Systems with Applications*, 39(10):9087–9096.

Dao, C., Liu, X., Sim, A., Tull, C., and Wu, K. (2018). Modeling data transfers: change point and anomaly detection. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1589–1594. IEEE.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30.

Ding, Z. and Fei, M. (2013). An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20):12–17.

Dwivedi, D., Mital, U., Faybishenko, B., Dafflon, B., Varadharajan, C., Agarwal, D., Williams, K. H., Steefel, C. I., and Hubbard, S. S. (2022). Imputation of contiguous gaps and extremes of subhourly groundwater time series using random forests. *Journal of Machine Learning for Modeling and Computing*, 3(2).

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fang, C. and Wang, C. (2020). Time series data imputation: A survey on deep learning approaches. *arXiv preprint arXiv:2011.11347*.

Gao, S., Zhang, Y., Jia, K., Lu, J., and Zhang, Y. (2015). Single sample face recognition via learning deep supervised autoencoders. *IEEE transactions on information forensics and security*, 10(10):2108–2118.

Gao, Y., Merz, C., Lischeid, G., and Schneider, M. (2018). A review on missing hydrological data processing. *Environmental earth sciences*, 77(2):1–12.

Gneiting, T. and Raftery, A. E. (2005). Weather forecasting with ensemble methods. *Science*, 310(5746):248–249.

Golab, L. (2004). Querying sliding windows over online data streams. In *International Conference on Extending Database Technology*, pages 1–11. Springer.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M. R., Venkatesh, S., and Hengel, A. v. d. (2019). Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1705–1714.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Günnemann, N., Günnemann, S., and Faloutsos, C. (2014). Robust multivariate autoregression for anomaly detection in dynamic product ratings. In *Proceedings of the 23rd international conference on World wide web*, pages 361–372.

Gupta, M., Gao, J., Aggarwal, C., and Han, J. (2014). Outlier detection for temporal data. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 5(1):1–129.

Hill, D. J., Minsker, B. S., and Amir, E. (2007). Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the Congress-International Association for Hydraulic Research*, volume 32, page 503. Citeseer.

Hou, X. and Zhang, L. (2007). Saliency detection: A spectral residual approach. In *2007 IEEE Conference on computer vision and pattern recognition*, pages 1–8. Ieee.

Hsu, Y.-F. and Matsuoka, M. (2020). A deep reinforcement learning approach for anomaly network intrusion detection system. In *2020 IEEE 9th International Conference on Cloud Networking (CloudNet)*, pages 1–6. IEEE.

Huang, C., Wu, Y., Zuo, Y., Pei, K., and Min, G. (2018). Towards experienced anomaly detector through reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Iftikhar, N., Baattrup-Andersen, T., Nordbjerg, F. E., and Jeppesen, K. (2020). Outlier detection in sensor data using ensemble learning. *Procedia Computer Science*, 176:1160–1169.

Jiang, D., Liu, J., Xu, Z., and Qin, W. (2011). Network traffic anomaly detection based on sliding window. In *2011 International Conference on Electrical and Control Engineering*, pages 4830–4833. IEEE.

Jiang, G., Xie, P., He, H., and Yan, J. (2017). Wind turbine fault detection using a denoising autoencoder with temporal information. *IEEE/Asme transactions on mechatronics*, 23(1):89–100.

Kao, I.-F., Liou, J.-Y., Lee, M.-H., and Chang, F.-J. (2021). Fusing stacked autoencoder and long short-term memory for regional multistep-ahead flood inundation forecasts. *Journal of Hydrology*, 598:126371.

Keogh, E. J. and Pazzani, M. J. (2001). Derivative dynamic time warping. In *Proceedings of the 2001 SIAM international conference on data mining*, pages 1–11. SIAM.

Khampuengson, T., Bagnall, A., and Wang, W. (2020). Developing ensemble methods for detecting anomalies in water level data. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 145–151. Springer.

Khampuengson, T. and Wang, W. (2022). Deep reinforcement learning ensemble for detecting anomaly in telemetry water level data. *Water*, 14(16):2492.

Khampuengson, T. and Wang, W. (2023). Novel methods for imputing missing values in water level monitoring data. *Water Resources Management*, pages 1–28.

Kieu, T., Yang, B., Guo, C., and Jensen, C. S. (2019). Outlier detection for time series with recurrent autoencoder ensembles. In *IJCAI*, pages 2725–2732.

Kim, M., Baek, S., Ligaray, M., Pyo, J., Park, M., and Cho, K. H. (2015). Comparative studies of different imputation methods for recovering streamflow observation. *Water*, 7(12):6847–6860.

Kormushev, P., Calinon, S., and Caldwell, D. G. (2013). Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122–148.

Kulanuwat, L., Chantrapornchai, C., Maleewong, M., Wongchaisuwat, P., Wimala, S., Sarinnapakorn, K., and Boonya-aroonnet, S. (2021). Anomaly detection using a sliding window technique and data imputation with machine learning for hydrological time series. *Water*, 13(13):1862.

Kumar, A., Srivastava, A., Bansal, N., and Goel, A. (2012). Real time data anomaly detection in operating engines by statistical smoothing technique. In *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*, pages 1–5. IEEE.

Lai, W. Y. and Kuok, K. (2019). A study on bayesian principal component analysis for addressing missing rainfall data. *Water Resources Management*, 33(8):2615–2628.

Law, S. M. (2019). STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39):1504.

Le, N., Rathour, V. S., Yamazaki, K., Luu, K., and Savvides, M. (2021). Deep reinforcement learning in computer vision: a comprehensive survey. *Artificial Intelligence Review*, pages 1–87.

Li, L., Wang, H., Wang, Y., Chen, M., and Wei, T. (2022). Improving iot data availability via feedback-and voting-based anomaly imputation. *Future Generation Computer Systems*, 135:194–204.

Lin, E., Chen, Q., and Qi, X. (2020). Deep reinforcement learning for imbalanced classification. *Applied Intelligence*, pages 1–15.

Lin, J., Sheng, G., Yan, Y., Zhang, Q., and Jiang, X. (2018). Online monitoring data cleaning of transformer considering time series correlation. In *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*, pages 1–9. IEEE.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE.

Liu, W., Jiang, H., Che, D., Chen, L., and Jiang, Q. (2020). A real-time temperature anomaly detection method for iot data. In *IoTBDS*, pages 112–118.

Liu, Y., Hou, G., Huang, F., Qin, H., Wang, B., and Yi, L. (2022). Directed graph deep neural network for multi-step daily streamflow forecasting. *Journal of Hydrology*, 607:127515.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. (2019). A survey of reinforcement learning informed by natural language. *arXiv preprint arXiv:1906.03926*.

Ma, J., Cheng, J. C., Jiang, F., Chen, W., Wang, M., and Zhai, C. (2020). A bi-directional missing data imputation scheme based on lstm and transfer learning for building energy data. *Energy and Buildings*, 216:109941.

Ma, L., Gu, X., and Wang, B. (2017). Correction of outliers in temperature time series based on sliding window prediction in meteorological sensor network. *Information*, 8(2):60.

Maas, A., Le, Q. V., O'neil, T. M., Vinyals, O., Nguyen, P., and Ng, A. Y. (2012). Recurrent neural networks for noise reduction in robust asr.

Marathe, A., Walambe, R., and Kotecha, K. (2021). Evaluating the performance of ensemble methods and voting strategies for dense 2d pedestrian detection in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3575–3584.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Moahmed, T. A., El Gayar, N., and Atiya, A. F. (2014). Forward and backward forecasting ensembles for the estimation of time series missing data. In *IAPR workshop on artificial neural networks in pattern recognition*, pages 93–104. Springer.

Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.

Ostroumov, I. and Kuzmenko, N. (2021). Outliers detection in unmanned aerial system data. In *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*, pages 591–594. IEEE.

Ouyang, Z., Sun, X., and Yue, D. (2017). Hierarchical time series feature extraction for power consumption anomaly detection. In *Advanced Computational Methods in Energy, Power, Electric Vehicles, and Their Integration*, pages 267–275. Springer.

Pan, Q., Hu, W., and Zhu, J. (2020). Series saliency: Temporal interpretation for multivariate time series forecasting. *arXiv preprint arXiv:2012.09324*.

Pang, G., van den Hengel, A., Shen, C., and Cao, L. (2020). Deep reinforcement learning for unknown anomaly detection. *arXiv preprint arXiv:2009.06847*.

Park, K., Jung, Y., Seong, Y., and Lee, S. (2022). Development of deep learning models to improve the accuracy of water levels time series prediction through multivariate hydrological data. *Water*, 14(3):469.

Parker, D., Tapsell, S., and McCarthy, S. (2007). Enhancing the human benefits of flood warnings. *Natural Hazards*, 43(3):397–414.

Pereira, J. and Silveira, M. (2018). Unsupervised anomaly detection in energy time series data using variational recurrent autoencoders with attention. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1275–1282. IEEE.

Peterson, L. E. (2009). K-nearest neighbor. *Scholarpedia*, 4(2):1883.

Phan, T.-T.-H. (2020). Machine learning for univariate time series imputation. In *2020 International Conference on Multimedia Analysis and Pattern Recognition (MAPR)*, pages 1–6.

Polydoros, A. S. and Nalpantidis, L. (2017). Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173.

Pratama, I., Permanasari, A. E., Ardiyanto, I., and Indrayani, R. (2016). A review of missing values handling methods on time-series data. In *2016 International Conference on Information Technology Systems and Innovation (ICITSI)*, pages 1–6. IEEE.

Raja, P. and Thangavel, K. (2020). Missing value imputation using unsupervised machine learning techniques. *Soft Computing*, 24(6):4361–4392.

Ramrez-Gallego, S., Krawczyk, B., Garca, S., Woniak, M., and Herrera, F. (2017). A survey on data preprocessing for data stream mining. *Neurocomputing*, 239(C):39–57.

Rashid, W. and Gupta, M. K. (2021). A perspective of missing value imputation approaches. In *Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2019*, pages 307–315. Springer.

Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., and Zhang, Q. (2019). Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3009–3017.

Rousseeuw, P. J. and Hubert, M. (2011). Robust statistics for outlier detection. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 1(1):73–79.

Saad, M., Nassar, L., Karray, F., and Gaudet, V. (2020). Tackling imputation across time series models using deep learning and ensemble learning. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3084–3090. IEEE.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2017). Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.

Senay, G., Velpuri, N. M., Bohms, S., Budde, M., Young, C., Rowland, J., and Verdin, J. (2015). Drought monitoring and assessment: remote sensing and modeling approaches for the famine early warning systems network. In *Hydro-meteorological hazards, risks and disasters*, pages 233–262. Elsevier.

Sharma, A. R. and Kaushik, P. (2017). Literature survey of statistical, deep and reinforcement learning in natural language processing. In *2017 International conference on computing, communication and automation (ICCCA)*, pages 350–354. IEEE.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Siris, V. A. and Papagalou, F. (2004). Application of anomaly detection algorithms for detecting syn flooding attacks. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, volume 4, pages 2050–2054. IEEE.

Stekhoven, D. J. and Bühlmann, P. (2012). Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118.

Tan, A. C. and Gilbert, D. (2003). Ensemble machine learning on gene expression data for cancer classification.

Tormene, P., Giorgino, T., Quaglini, S., and Stefanelli, M. (2009). Matching incomplete time series with dynamic time warping: an algorithm and an application to post-stroke rehabilitation. *Artificial intelligence in medicine*, 45(1):11–34.

Truong, C., Oudre, L., and Vayatis, N. (2020). Selective review of offline change point detection methods. *Signal Processing*, 167:107299.

Twala, B. (2009). An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5):373–405.

van de Wiel, L., van Es, D. M., and Feelders, A. J. (2020). Real-time outlier detection in time series data of water sensors. In Lemaire, V., Malinowski, S., Bagnall, A., Guyet, T., Tavenard, R., and Ifrim, G., editors, *Advanced Analytics and Learning on Temporal Data*, pages 155–170, Cham. Springer International Publishing.

Vinutha, H., Poornima, B., and Sagar, B. (2018). Detection of outliers using interquartile range technique from intrusion dataset. In *Information and Decision Sciences: Proceedings of the 6th International Conference on FICTA*, pages 511–518. Springer.

Vu, M., Jardani, A., Massei, N., and Fournier, M. (2021). Reconstruction of missing groundwater level data by using long short-term memory (lstm) deep neural network. *Journal of Hydrology*, 597:125776.

Wan, Y., Chen, J., Xu, C.-Y., Xie, P., Qi, W., Li, D., and Zhang, S. (2021). Performance dependence of multi-model combination methods on hydrological model calibration strategy and ensemble size. *Journal of Hydrology*, page 127065.

Wang, W. (2008). Some fundamental issues in ensemble methods. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 2243–2250. IEEE.

Xu, C., Liu, Q., and Ye, M. (2017). Age invariant face recognition and retrieval by coupled auto-encoder networks. *Neurocomputing*, 222:62–71.

Yahoo! Webscope (2010). Yahoo! webscope dataset ydata-labeled-time-series-anomalies-v1_0. `http://labs.yahoo.com/Academic_Relations`.

Yang, J.-H., Cheng, C.-H., and Chan, C.-P. (2017). A time-series water level forecasting model based on imputation and variable selection method. *Computational intelligence and neuroscience*, 2017.

Yang, Z., Abbasi, I. A., Mustafa, E. E., Ali, S., and Zhang, M. (2021). An anomaly detection algorithm selection service for iot stream data based on tsfresh tool and genetic algorithm. *Security and Communication Networks*, 2021.

Yeh, C.-C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Silva, D. F., Mueen, A., and Keogh, E. (2016). Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1317–1322. Ieee.

Yu, L., Tan, S. K., and Chua, L. H. (2017). Online ensemble modeling for real time water level forecasts. *Water resources management*, 31(4):1105–1119.

Yu, Y., Zhu, Y., Li, S., and Wan, D. (2014). Time series outlier detection based on sliding window prediction. *Mathematical problems in Engineering*, 2014.

Yuan, M. (2017). Getting to know mqtt. *IBM Developer*, 7.

Zelaya, C. V. G. (2019). Towards explaining the effects of data preprocessing on machine learning. In *2019 IEEE 35th international conference on data engineering (ICDE)*, pages 2086–2090. IEEE.

Zhang, J., Aviles-Rivero, A. I., Heydecker, D., Zhuang, X., Chan, R., and Schönlieb, C.-B. (2021a). Dynamic spectral residual superpixels. *Pattern Recognition*, 112:107705.

Zhang, Y. and Thorburn, P. J. (2021). A dual-head attention model for time series data imputation. *Computers and Electronics in Agriculture*, 189:106377.

Zhang, Y., Zhou, B., Cai, X., Guo, W., Ding, X., and Yuan, X. (2021b). Missing value imputation in multivariate time series with end-to-end generative adversarial networks. *Information Sciences*, 551:67–82.

Zhou, C. and Paffenroth, R. C. (2017). Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674.

Zhou, Y., Guo, S., and Chang, F.-J. (2019). Explore an evolutionary recurrent anfis for modelling multi-step-ahead flood forecasts. *Journal of hydrology*, 570:343–355.

Zimek, A. and Filzmoser, P. (2018). There and back again: Outlier detection between statistical reasoning and data mining algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(6):e1280.

Zimek, A. and Schubert, E. (2017). Outlier detection. *Encyclopedia of Database Systems*, pages 1–5.

Zong, B., Song, Q., Min, M. R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International Conference on Learning Representations*.