Improving Ensembles and Prediction Intervals for Machine Learning on Data Streams

A thesis

submitted in fulfilment

of the requirements for the Degree

of

Doctor of Philosophy in Computer Science

at

The University of Waikato

by

Yibin Sun

THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

2025

Abstract

The rapid growth of streaming data presents significant challenges for traditional machine learning, including popular tasks like regression and classification. This thesis proposes adaptive and dynamic methods to address key issues, including concept drift, uncertainty quantification, and ensemble optimization, in evolving data streams.

The Self-Optimising K Nearest Leaves (SOKNL) regression algorithm integrates k-Nearest Neighbors (kNN) and Adaptive Random Forest Regression (ARF-Reg), dynamically optimizing neighbor selection to improve regression accuracy without relying on fixed window sizes. Extensive experimental results suggest that SOKNL outperforms the state-of-the-art streaming regression algorithms, including its origin, ARF-Reg.

For classification tasks, the Dynamic Ensemble Member Selection (DEMS) method dynamically adjusts ensemble size and selects members based on accuracy and diversity, improving predictive performance while handling concept drift. DEMS extends the idea of dynamic selection of ensemble members from SOKNL to classification tasks, with more flexible selection criteria.

The Adaptive Prediction Interval (AdaPI) framework provides robust uncertainty quantification by adaptively adjusting prediction intervals based on historical coverage, ensuring reliability in streaming regression. To evaluate prediction intervals holistically, the thesis introduces Coverage Interval Width in Non-dominated Groups (CING), a multi-objective evaluation method balancing interval width and coverage.

Aiming at analyzing the proposed methods for regression, this thesis also contributes the New Zealand Energy Pricing (NZEP) datasets, a comprehensive repository for real-time energy analytics. NZEP aims at providing a real, growing, customizable regression data source that can enrich the current regression benchmark data for stream learning, and potentially time-series.

By providing scalable, adaptive solutions for regression and classification, this research advances real-time decision-making in streaming data environments.

Acknowledgements

When I started my Ph.D., there was this tutorial aiming at acquainting us to "probably the most lonely journey in the world." Now, I am standing at the end of that journey. If it is ever possible, I would like to time-travel back to that day and tell the lady: she is wrong. It was, indeed, a long journey. Yet, this fulfilling, engaging, and full-of-accidents period has flown away when I was immersed in it.

It is hard to believe how much had happened in these four years, which was accompanied by plenty of loving and beautiful minds. They tolerated my defects, guided me to the correct way, provided me helping hands, and most importantly, cared about me, starting with my beloved supervisors.

I am profoundly grateful to my parents for their unconditional love, endless support, and unwavering faith in me. Their sacrifices and dedication have shaped who I am today and provided me with the strength to pursue my dreams. I owe this achievement to their boundless patience, guidance, and the values they instilled in me.

I also would like to express my sincerest gratitude to Prof. Bernhard Pfahringer, Prof. Albert Bifet, and Assistant Prof. Heitor Murilo Gomes, whose guidance, expertise, and encouragement have been instrumental in completing this PhD. Their support during my mental breakdown and my foolish decisions earnestly dragged me out of the whirlpool of decadence. Without them, the personal and academic growth that I have gained in this four years would only be an unattainable illusion.

My thankfulness extends to all the colleagues that have been supportive and friendly to me, including Anany Dwivedi, Guilherme Weigert Cassales, Justin Liu, Nick Lim, Syuhaida Safian, just to name a few. Our daily interactions not only joyfully but also informatively shaped up this transformative journey. To all those whose support I may have inadvertently omitted, please accept my sincere appreciation for your contributions throughout this journey.

Contents

List of Figures

List of Tables

Acronyms

AGI Artificial General Intelligence. 2

AI Artificial Intelligence. 1

AMRules Adaptive Model Rules. 6, 17, 18, 43, 46

ARF-Reg Adaptive Random Forest for Regression. viii, 6, 7, 17, 18, 32, 33, 44, 46, 47, 55, 68, 71, 120, 121

ARF Adaptive Random Forest. 23

DEMS Dynamic Ensemble Member Selection. vi, 8, 10, 94, 95, 106, 120, 122

DL Deep Learning. 1

FIMT-DD Fast Incremental Model Tree with Drift Detection. 16, 17, 32

FIRT-DD Fast Incremental Regression Tree with Drift Detection. 16, 17, 32

HT Hoeffding Tree. 22

LLM Large Language Model. 1

ML Machine Learning. 1, 2, 6, 16, 21

RAMRules Random Adaptive Model Rules. 18, 46, 47

RF Random Forest. 32

SL Stream Learning. iv, 2–6, 9, 11, 12, 16, 17, 22, 56, 81, 120

SOKNL Self-Optimising K Nearest Leaves. v, vi, viii, 7–9, 32, 33, 36–40, 42–44, 46, 47, 53, 55, 56, 68, 71, 72, 75, 94, 120, 121

SRP Streaming Random Patches. 23

`AdaPI` Adaptive Prediction Interval. v, vi, 7, 9, 56, 57, 62, 68, 76, 80, 120, 121

`EC` Evolutionary Computing. 59

`MVE` Mean and Variance Estimation. v, ix, 56, 57, 59, 62, 63, 68, 71, 74, 76, 78–80

`PI` Prediction Interval. iv, 6, 7, 11, 20, 21, 56–59, 61, 64, 78, 120

`k-means` K Means Clustering. 33

`kNN` K Nearest Neighbours. 6, 7, 16, 33–36, 38, 42, 68, 71, 72, 120

ADWIN ADaptive sliding WINdow. 15, 23

CD Concept Drift. iv, 3, 4, 6, 8, 12, 13

CE Coverage Error. 61, 62, 72, 74, 78

CING Coverage Interval width in Non-dominated Group. v, 7, 61, 62, 72, 74, 78, 120

DS Data Streams. 4

EFDT Extremely Fast Decision Tree. 22

FM Foundation Models. 2

LVB Leverage Bagging. 23

MOO Multi-Objective Optimization. 59–61, 120

NMPIW Normalized Mean Prediction Interval Width. xiii, 21, 22, 61, 62, 69, 71, 72, 74

NSGA II Non-dominated Sorting Genetic Algorithm II. v, 59–61

NZEP New Zealand Energy Pricing. vi, 8, 9, 120–122

OAUE Online Accuracy Updated Ensemble. 23

OSBoost Online Smooth Boosting. 23

SVM Support Vector Machine. 33

Chapter 1

Introduction and Background

This chapter introduces the essence of learning with evolving data streams, expresses the motivation behind our research, and poses the challenging issues in this domain. Following that, we present our contribution in tackling those challenges and provides an outline of this thesis.

1.1 Introduction

According to the published information from Statista, the volume of data/information created, captured, copied, and consumed worldwide has increased from around 2 zettabytes in 2010 to around 125 zettabytes in 2024, and will keep growing exponentially based on the forecast of the statistical model. This means that the world is producing data in not only a very high volume but also at an extremely rapid velocity.

On the other hand, the computational resources that have been poured into processing the generated or captured data have also proliferated in the last decade, as illustrated in Figure 1.1. There is absolutely no doubt that Deep Learning (DL) and Large Language Model (LLM) have revolutionized the Machine Learning (ML) and Artificial Intelligence (AI) world. Nonetheless, as plenty of researchers and practitioners have agreed ([123, 127, 54, 137, 108], and many others), the exponential increase of the computing resources can raise serious and universal concerns, such as efficiency, scalability, adaptabil-

ity, economic and environmental impacts, and so forth. Furthermore, the establishment of Foundation Models (FM) and Artificial General Intelligence (AGI) is increasing resource consumption even further [69, 82, 103].

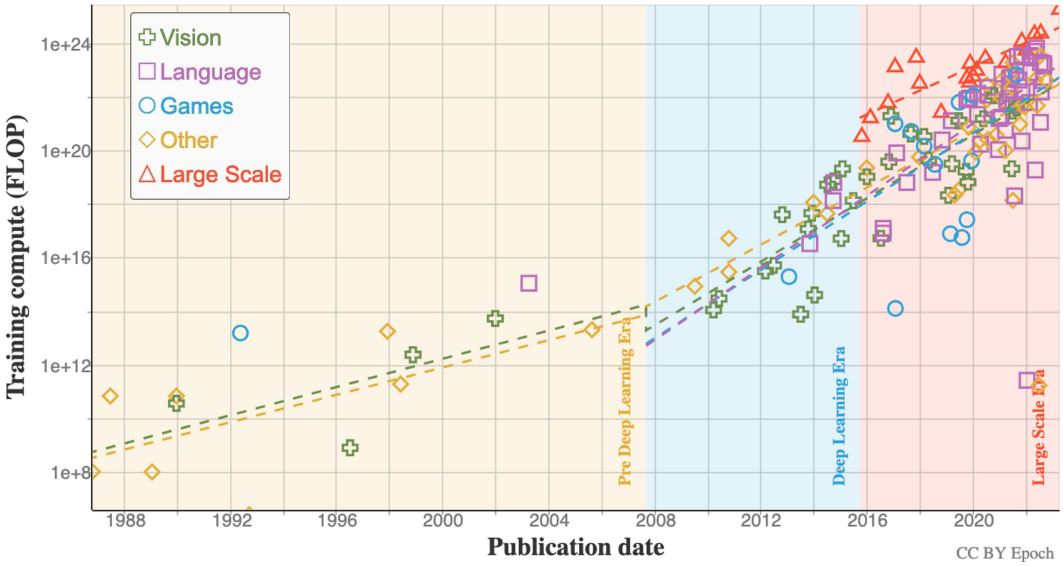


Figure 1.1: Training compute (FLOPS) of milestone ML learning systems over time. Source [109, 57]

1.1.1 Motivation

The computational resource demands and the proliferation of data, while shaping the Big Data Era, have triggered several challenges to the traditional ML techniques. Stream Learning offers promising solutions to the issues from batch learning in the following aspects (and more) [13, 52]:

- **Continuous Stream of Data.**

  Traditional batch learning assumes that data is finite and fully available at the start of the training process, making it impractical for scenarios where data is continuously generated, such as social media streams, IoT sensors, or financial transactions. SL, on the other hand, is designed to process data incrementally as it arrives, without needing to store or access the entire dataset. This capability allows stream learning to handle infinite or unbounded data sources efficiently, enabling real-time

analysis and decision-making. By continuously updating the model with incoming data, SL ensures that systems remain current and effective, even as the volume of data grows indefinitely.

- **Memory Efficiency**

  Stream Learning is efficient in both memory and computational cost, processing data incrementally – one instance or small batch at a time – and discarding it after updating the model. Unlike traditional batch learning, which stores and retrains on the entire dataset, SL minimizes memory usage and avoids the overhead of repeated retraining. This makes it ideal for large-scale, continuous data and resource-constrained environments, ensuring scalability and sustained performance even with high-velocity data streams.

- **Real-time Processing and Decision Making**

  Stream Learning enables real-time processing and decision-making by updating models incrementally as data arrives, eliminating the delays inherent in batch learning. Traditional batch methods require complete datasets and significant computational time to train models, making them unsuitable for time-sensitive applications. SL, however, processes data continuously, allowing systems to generate predictions and adapt instantly. This capability is crucial for dynamic environments like fraud detection, stock market analysis, and IoT applications, where timely insights and actions can significantly impact outcomes.

- **Concept Drift**

  Stream Learning is uniquely equipped to handle concept drift, where the underlying data distribution changes over time, rendering static models ineffective. Traditional batch learning assumes a fixed distribution, requiring frequent retraining to adapt, which is time-consuming and computationally expensive. In contrast, SL continuously updates the model as new data arrives, enabling it to adapt to shifts in patterns dynami-

cally. This makes SL ideal for environments with evolving data, such as user preferences in recommendation systems, fraud detection, or weather forecasting, ensuring consistent performance in the face of change. This phenomenon will be discussed in more detail later in Section 2.1.1.

Furthermore, we noticed that regression tasks for data streams are relatively understudied compared to its classification counterpart [21, 116, 89], which motivates us to further advance this area in this thesis. We identify two main reasons for the relative neglect of regression tasks in the stream learning scenario. First, regression tasks often involve complex numerical target distributions that are more sensitive to non-stationarity or Concept Drift compared to categorical labels in classification [122]. Second, there is a relative lack of benchmark datasets for stream regression compared to classification, limiting comparative studies and algorithm development [112].

1.1.2 Stream Learning with Evolving Concepts

Stream Learning (SL) is a specialized area of machine learning designed to process continuous, potentially infinite streams of data. SL techniques are tailored for coping with the real-time decision making and the all-time evolving data distribution issues. [39, 13, 92, 5] provide the development of the research targets over the last two decades, from basic Data Streams concepts to more advanced drift detection and adaptation techniques.

Similar to batch learning, SL can be categorized into the following sectors [13, 8]:

1. Supervised Learning: where the labels or targets for the testing instance are available after the prediction phase. Due to this characteristic, for research purposes, an evaluation process is commonly designed as: An unlabeled instance is predicted by the model at the observation moment. Then, the label will be attached to the instance immediately after the testing and the model will learn from the labeled instance. This is usually referred to as "TestThenTrain" or "Prequential" procedure [13, 47].

Figure 1.2 illustrates the process for one example in a virtual manner. This process repeats while the data stream keeps arriving.

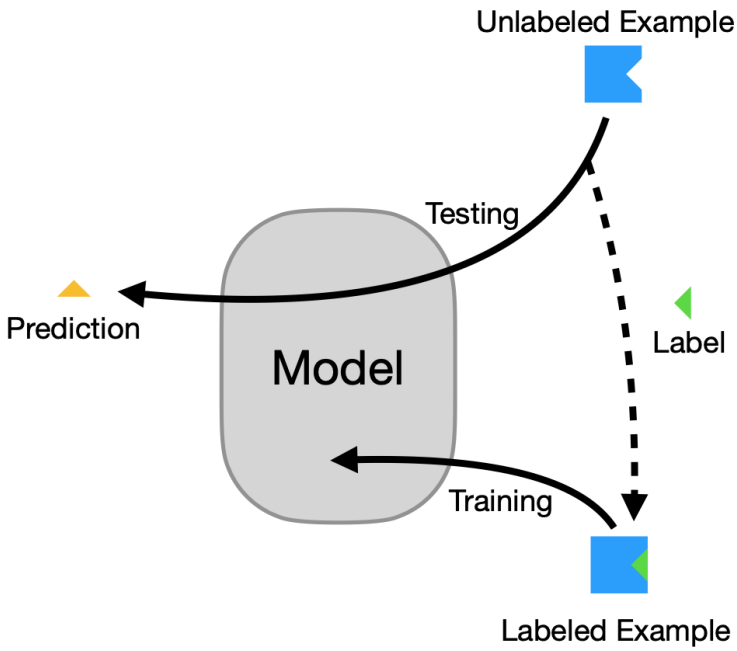


Figure 1.2: Single Step of Prequential Process on Data Streams (Simplified)

2. Unsupervised Learning: where the labels will not be revealed, including clustering [138] and anomaly detection [93][1] tasks; and

3. Semi-supervised learning: where the ground truths are not always immediately available. It also divides into the following categories [48]:

   (a) Delayed and fully labeled;

   (b) Immediate and partially labeled; and

   (c) Delayed and partially labeled.

Supervised learning is usually segmented into two parts, classification and regression. For the reasons stated in Section 1.1.1, regression remains relatively under-explored. The main goal of this thesis is to tackle the regression tasks in SL. However, we extended our attempt to classification problems, enhancing current classifiers' performances.

[1] Anomaly Detection can be supervised. However, the unsupervised area is more pervasive, thus more explored. Therefore, here we count it as a part of unsupervised learning.

1.1.3 Challenges

Stream Learning presents unique challenges due to the dynamic and evolving nature of data streams. Key issues include Concept Drift [87, 13, 22] (see in Section 2.1.1), where the statistical properties of the target variable change over time, necessitating models that adapt to new patterns without forgetting old ones prematurely. The real-time constraints of streaming data require algorithms to process data incrementally and efficiently, often within tight memory and computational limits [13, 52]. Scalability is another critical challenge, as algorithms must handle high-throughput streams and vast data volumes while maintaining accuracy and responsiveness [112, 71].

Stream regression faces challenges in overlooked aspects like evaluation metrics and Prediction Interval. Traditional metrics such as MAE or MSE fail to reflect performance under evolving data streams because they assume a static data distribution and do not account for concept drift, making them ineffective at capturing real-time model adaptability and accuracy changes over time [86]. Prediction Interval, crucial for uncertainty quantification, remain under-explored, requiring real-time updates to stay reliable amid shifting data distributions [62, 63, 119]. Addressing these gaps is essential for robust, adaptive stream regression.

1.1.4 Contributions

The contributions of this thesis can be summarized in five key points:

- This thesis presents a novel method for Stream Learning regression. Due to the scarcity of the regression algorithms for SL, the dedicated and effective regressive models in this field are from quite a while ago, such as Adaptive Model Rules (AMRules) [3, 32] in 2016 and Adaptive Random Forest for Regression (ARF-Reg) [46] in 2017. Meanwhile, the idea of K Nearest Neighbours (`kNN`) is still widely spread in the ML world, proving its effectiveness. However, in SL tasks, a `kNN` is always accompanied by a

sliding window, and the performance is largely dependent on the window size, which requires domain expertise and empirical testing to specify. Building on these ideas, we propose Self-Optimising K Nearest Leaves (SOKNL), a novel way of integrating `kNN` and ARF-Reg for better regression performance. SOKNL managed to produce improvements over ARF-Reg on almost all tasks. Furthermore, SOKNL ranked the best among 12 tested streaming regression algorithms, while remaining efficient enough.

- To improve the confidence in the predictive results of the regression models, we developed Adaptive Prediction Interval (`AdaPI`), a novel method designed to address the challenge of quantifying uncertainty in regression tasks for dynamic and evolving data streams. Traditional Prediction Interval (`PI`) methods are unsuitable for streaming contexts due to their static nature. `AdaPI` introduces a postprocessing mechanism that adaptively adjusts interval widths based on historical data to achieve a user-defined coverage level. The approach is versatile and can enhance any streaming `PI` technique. To validate `AdaPI`, I implemented an incremental variant of the Mean and Variance Estimation (MVE) method and conducted empirical evaluations on standard streaming regression tasks. Results demonstrated that `AdaPI` consistently produces compact and accurate prediction intervals with desired coverage, significantly outperforming existing methods. This work advances the field of streaming regression by providing a robust and adaptive uncertainty quantification framework.

- On top of `AdaPI`, we also proposed an evaluation framework – Coverage Interval width in Non-dominated Group (CING) – to conquer the difficulties in assessing the performance of `PI` algorithms taking into account two metrics, coverage and interval width (length). Unlike traditional optimization methods like NSGA II, CING focuses on balancing coverage and

interval width by normalizing metrics, identifying non-dominated groups, and scoring them with a weighted-sum strategy. This approach prioritizes achieving coverage close to the desired confidence level while minimizing interval width, offering a flexible and robust evaluation method for PI models in diverse scenarios.

- A notable contribution of my thesis is the introduction of a comprehensive dataset for real-time energy price analysis in New Zealand – New Zealand Energy Pricing (NZEP) – sourced from the Electricity Market Information website. This dataset addresses the scarcity of resources for streaming regression research, enabling tasks like CD detection, anomaly detection, and prediction interval generation. Through extensive experiments using advanced streaming algorithms, the dataset has demonstrated its utility in forecasting energy prices and analyzing dynamic patterns in streaming data, providing a foundation for further research in energy market analytics and data stream learning.

- We also extended the idea from SOKNL to classification tasks. However, specific adaptations were required. Based on this, we presented Dynamic Ensemble Member Selection (DEMS), a novel framework for enhancing the predictive performance of ensemble methods in data stream classification. DEMS dynamically selects a subset of classifiers for each prediction based on estimated accuracy and predictive margin, adjusting the subset size self-adaptively to optimize performance. The approach demonstrates significant improvements across diverse streaming ensemble algorithms, effectively handling CD and maintaining computational efficiency, thereby advancing the field of dynamic ensemble learning for streaming data.

1.1.5 Publications

The work listed above has been published as:

- Sun, Y., Pfahringer, B., Gomes, H. M., Bifet, A. (2022). SOKNL: A novel way of integrating K-nearest neighbours with adaptive random forest regression for data streams. Data Mining and Knowledge Discovery, 36(5), 2006-2032.

- Sun, Y., Pfahringer, B., Murilo Gomes, H., Bifet, A. (2024, April). Adaptive Prediction Interval for Data Stream Regression. In Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 130-141). Singapore: Springer Nature Singapore.

- Sun, Y., Gomes, H.M., Pfahringer, B., Bifet, A. (2024). Real-Time Energy Pricing in New Zealand: An Evolving Stream Analysis. In: PRICAI 2024: Trends in Artificial Intelligence. Lecture Notes in Computer Science, vol 15285. Springer, Singapore.

- **(under review)** Sun, Y., Pfahringer, B., Murilo Gomes, H., Bifet, A. (2025). Adaptive Approaches towards Fully Incremental Prediction Interval for Data Stream Regression. Data Mining and Knowledge Discovery .

- **(Accepted)** Sun, Y., Pfahringer, B., Gomes, H.M., Bifet, A. (2025). Dynamic Ensemble Member Selection for Data Stream Classification. In Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25).

1.1.6 Outline

This thesis consists of seven chapters. Chapter 1 provides an overview of the basics of Stream Learning. Chapter 2 introduces preliminaries and related work. Chapter 3 details the proposed regression algorithm Self-Optimising K Nearest Leaves. Chapter 4 depicts Adaptive Prediction Interval model, followed by the introduction of the comprehensive dataset of New Zealand Energy Pricing in Chapter 5. Chapter 6 describes the details of the proposed

streaming classification framework – Dynamic Ensemble Member Selection. Finally, the thesis is concluded in Chapter 7.

Chapter 2

Preliminaries and Related Work

This chapter introduces the preparation and related work for this thesis. In particular, it provides basic concepts of SL and explains in depth the Concept Drift (CD) phenomenon, following by specific information regarding supervised learning tasks and PI. Finally, the datasets involved in this thesis are listed and described.

2.1 Stream Learning

Stream Learning is a method designed to handle data that arrives continuously over time. Unlike traditional learning approaches that work with fixed datasets, SL processes each new piece of data as it arrives, updating the model without revisiting previously seen data [13, 39]. This approach ensures that the learning process remains efficient and capable of handling data in real-time [40].

By incorporating new information on the fly, SL models can stay up-to-date with the latest trends in the data. Older data may naturally fade in influence, enabling the model to focus on more recent patterns while maintaining computational efficiency. This continuous update mechanism makes SL particularly well-suited for dynamic environments where data evolves rapidly [52, 87].

2.1.1 Concept Drift (CD)

Concept Drift (CD) is a significant and challenging issue in Stream Learning. It describes the phenomenon where the internal distributional properties of the stream change over time [13]. The changes can occur in the feature space and/or the target space [52, 131, 22]. CD is one of the essential components of Stream Learning, and as a result, a streaming technique or algorithm needs to be able to cope with CD.

2.1.1.1 Types of Drift

Concept Drift can be categorized according to different characteristics. The literature has preferred taking into account the following aspects:

- **The impact of the drift on the current decision boundary** [52]. More particularly, this is to consider if the changes in data will affect the current model's decision. Literature usually divides it into two types: Real and Virtual, where the model's performance will deteriorate for real drifts but not for virtual ones.

  Figure 2.1 is an illustration of virtual and real drift scenarios.

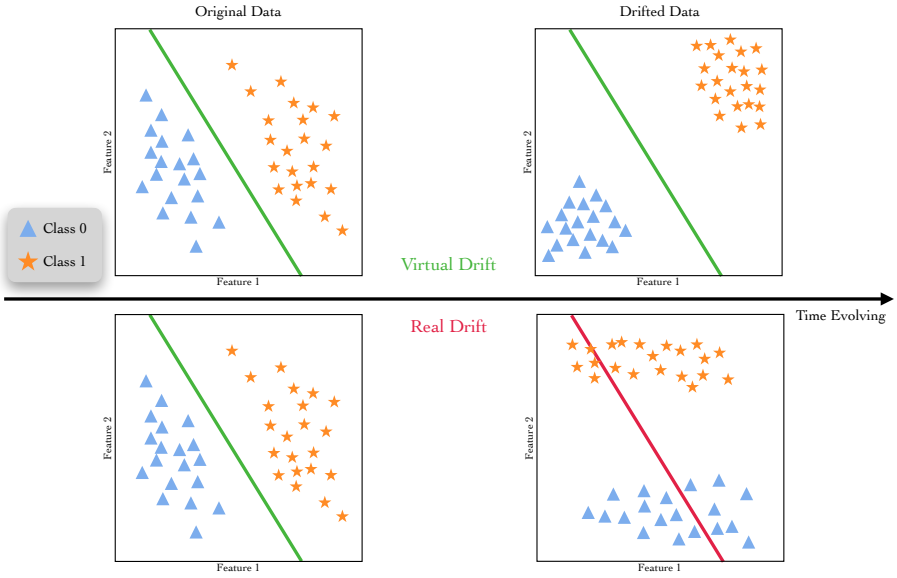


Figure 2.1: Illustration of Virtual and Real Drift.

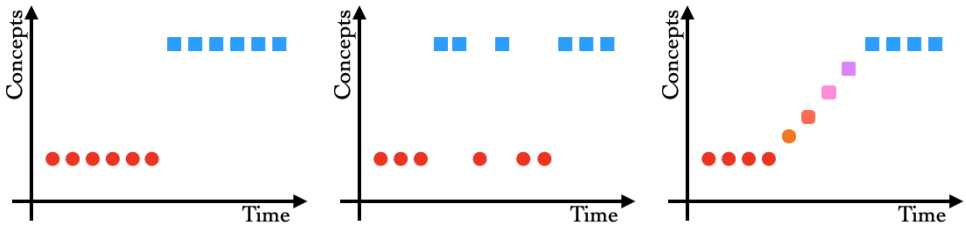
The top half describes the virtual drift, where after the drift, the decision boundary (green line) can still separate the two classes accurately. The bottom half, on the other hand, demonstrates the real drift. As observed, the separation line is no longer the optimal segmentation of the space in terms of the distribution of the data points.

- **The rate of change** [42].

  Another noticeable factor is the speed of the drift. Literature usually categorizes drifts into three different rates:

  - *Abrupt or Sudden*, where the data switch from one distribution to another immediately.
  - *Gradual*, meaning that there is a period before the drift is complete, during which data from both distributions can be observed.
  - *Incremental*, that is, the statistical properties of the data are slowly shifting, resulting in a prolonged changing period and non-detectable changes between two neighbouring instances.

  Figure 2.2 provides a conceptual overview to different changing rates of CD. In the plot, the red circled dots represent data points from one distribution (concept) while the blue squared dots come from another one.



(a) Abrupt/Sudden Drift (b) Gradual Drift (c) Incremental Drift

Figure 2.2: Illustration of Different Concept Drift Rates

- **The reach of the change** [115].

A drift can reach to different levels of the data. Specifically, a drift can have impact on (1) only the labels, (2) part of the features, (3) all the features, and (4) features and labels.

- **The recurrence of a previous concept** [115, 56].

  Recurring drifts refer to the reappearance of a particular distribution that has been encountered previously by the model in the streaming scenario. Figure 2.3 exhibits the simplified concept of a recurring drift.

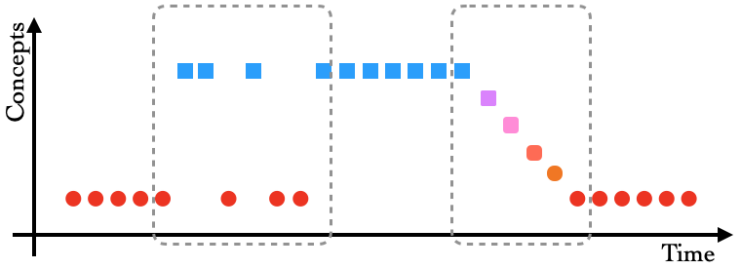


Figure 2.3: Illustration of Recurring Drift

In Figure 2.3, the data shifts from the first concept (red circles) to another one (blue squares) via a gradual drift, and thereafter switches back to the first distribution through an incremental drift. The dashed boundary boxes highlight the drifting periods, which can be substituted by any other types of drift, making recurring drifts much more complex and thus, more challenging to address.

- **The outliers or anomalous data** [13, 93][1].

  Outliers are data points that deviate significantly from the general trend or distribution from the data streams, as illustrated in Figure 2.4. They are generally rather rare and do not interfere with the overall properties of the data. Ergo, outliers and anomalies usually are not categorized as concept drifts.

[1] This is more of the outlier or anomaly detection problem, but an outlier might be mistaken as the start of a drift event.

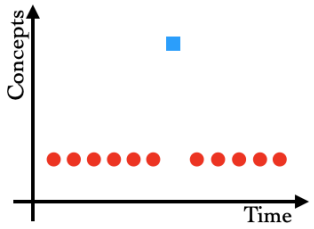


Figure 2.4: Illustration of Outliers

2.1.1.2 Drift Detection and Adaptation

Literature has proposed a substantial number of methods for detecting and adapting to drift in streaming scenarios. [92, 42, 80, 84] provide extensive overviews of drift detection and adaptation algorithms. In this thesis, only a brief introduction to a small number of drift handlers will be provided. We choose to categorize them under the taxonomy of [112, 57].

- A group of methods considers the differences in distributional properties between distinguished parts of the data streams, which is commonly achieved by windows. Frequently, we draw a null hypothesis that the data in the two windows are from the same distribution. When a selected statistical test rejects the null hypothesis, it means the detection window (holding the newer data) possesses information from a distribution different from the reference window (covering the older data). The detection windows usually are updated with the incoming data streams, while the reference window can be fixed or sliding along with the detection windows. The size of the windows can also be static or dynamic. ADaptive sliding WINdow (ADWIN) [11] and Hoeffding's Drift Detection Method (HDDM) [37] are instances of this category.

- Another set of detectors are based on sequential analysis, also known as the Sequential Probability Ratio Test (SPRT) [128]. They usually monitor the errors (or other statistics) in a sequential manner and detect drift with statistical tests. Commonly used ones in this category are the Cumulative Sum Control Chart (CUSUM) and the Page-Hinckley Test

(PHTest) [102].

- A drift can also be detected by statistical process control, where performance metrics are used as indicators for a statistical process, and then the change points are located heuristically. Drift Detection Method (DDM) [41] and Early Drift Detection Method (EDDM) [6] are, among plenty of other methods, prominent examples of this category.

2.2 Supervised Learning

Supervised learning is a highly pervasive branch of ML as well as SL.

2.2.1 Regression

Regression tasks in SL are relatively overlooked as discussed in Section 1.1.1. Some of the methods proposed over the last two decades are:

- Lazy methods have always been proven effective and elementary, and more importantly, directly applicable to regression tasks. Consequently, K Nearest Neighbours (`kNN`) [104], along with several variants ([90, 7], and more), is adapted to SL using various windowing strategies.

- Tree based algorithms are also popular approach for their efficiency, effectiveness, and interpretability. However, regression trees are less explored than classification trees.

  Hoeffding Tree Regression (HTR) [30], one of the earliest streaming regression trees, adapts the Hoeffding Tree Classification to predict numeric values.

  Fast Incremental Model Tree with Drift Detection (FIMT-DD) and Fast Incremental Regression Tree with Drift Detection (FIRT-DD) [77] are the most commonly used tree-based regressors in SL. The main difference between FIMT-DD and FIRT-DD is that FIMT-DD maintains a linear model (a Perceptron [105] more precisely) in each leaf node for

the outputs while FIRT-DD predicts with the mean values observed by each leaf. They strictly adhere to the single-pass rule of SL and apply a Page-Hinckley test at each node to detect drift. The subbranch from a certain node will be discarded and re-grown if a drift is confirmed in that node.

There are more attempts on streaming regression trees, such as Extremely Fast Decision Tree for Regression (E-Tree) [95] that guarantees a faster convergence of the tree to a batch learning tree structure.

- Rule-based algorithms are another popular approach in stream learning due to their flexibility, interpretability, and ability to handle evolving data. However, similar to regression trees, rule-based regression algorithms are relatively underexplored compared to their classification counterparts.

  Rule-based regression algorithms aim to predict numeric outputs by forming a set of human-readable rules that map specific feature conditions to target values. These rules are dynamically updated to adapt to data streams and concept drifts, ensuring efficiency and relevance over time.

  Adaptive Model Rules (AMRules) [3] represents a significant advancement in rule-based regression for streaming data. Unlike static rule models, AMRules adapts incrementally to evolving data distributions by adding or updating rules as new examples arrive. It employs adaptive drift detection mechanisms to identify changes in data patterns and replace outdated rules accordingly. Moreover, AMRules incorporates linear regression models similar to FIMT-DD to improve the accuracy of numeric predictions.

- In the literature of SL, ensembles [45] have demonstrated their strength in improving the performance upon single learners.

  Adaptive Random Forest for Regression (ARF-Reg) [46] is an adap-

tation of the well-known Adaptive Random Forest to handle numeric predictions in streaming data. It builds an ensemble of regression trees, where each tree is trained on a random subset of the features. The algorithm incorporates mechanisms to detect and adapt to concept drift using drift detectors such as the Page-Hinckley test. When drift is detected in a tree, the affected sub-tree is dropped and then rebuilt, ensuring continuous learning while maintaining model accuracy. ARF-Reg leverages bootstrap sampling to create diversity among the ensemble members, which improves robustness and reduces variance in predictions.

Random Adaptive Model Rules (RAMRules) [32] is a rule-based regression algorithm that combines the strengths of rule-based models with randomization techniques inspired by ensemble methods like the Random Forest. Unlike standard AMRules, RAMRules introduces randomness by learning rules over randomly selected feature subsets, increasing diversity in the rule set. Each rule uses adaptive linear a regression model in its consequent to predict numeric outputs, ensuring improved accuracy for complex data streams. Additionally, the algorithm includes drift detection to identify outdated rules and to replace them dynamically, maintaining adaptability in evolving environments.

With ensemble learners, there is commonly a trade-off between the efficiency and the performance [51, 44, 45], which both ARF-Reg and RAMRules aim to balance for streaming regression tasks. While ARF-Reg benefits from ensemble learning and diversity through randomization at the tree level, RAMRules achieves similar benefits at the rule level, offering a more interpretable alternative for numeric prediction in streaming environments.

2.2.1.1 Evaluation

In a stream learning scenario, regression tasks involve predicting a continuous value in a data stream, where data arrives sequentially and evolves over

time. Unlike traditional batch learning, stream learning requires models to adapt continuously while maintaining efficiency in terms of time and memory. Therefore, evaluating the performance of regression models on streaming data must be done incrementally, often using metrics that measure errors between predicted and actual values in real-time. Below, we introduce common evaluation metrics used for regression tasks in the stream learning context.

Let y_i be the actual value, $\hat{y}_i$ the predicted value, and n the total number of observations. The following metrics are commonly used:

Mean Absolute Error (MAE) The MAE calculates the average of the absolute differences between the actual and predicted values:

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

Mean Squared Error (MSE) The MSE computes the mean of the squared differences, giving more weight to larger errors:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) The RMSE is the square root of the MSE, providing an error metric in the same unit as the target variable:

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

Root Relative Squared Error (RRSE) The RRSE normalizes the RMSE by dividing it by the RMSE of a baseline model (e.g., mean predictor). This provides a relative measure of performance:

$$\text{RRSE} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where $\bar{y}$ is the mean of the actual values.

Relative Mean Absolute Percentage Error (RMAPE) The RMAPE measures the percentage error between predictions and actual values, adjusted

for the scale of the data:

$$\text{RMAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{|y_i|}$$

Coefficient of Determination (R^2) The R^2 score evaluates the proportion of variance in the target variable explained by the regression model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where $\bar{y}$ is the mean of the actual values. A higher R^2 value indicates a better fit of the model to the data, with $R^2 = 1$ representing a perfect fit.

These metrics collectively provide insights into the performance of regression models in stream learning scenarios. Each metric has its strengths, depending on whether absolute or relative errors are more critical for a given application. Regular evaluation helps to monitor a model's performance and to adapt it as the data distribution evolves over time. RMSE and R^2 are the main measurements used in this thesis.

2.2.2 Prediction Interval

A Prediction Interval (PI) is an interval within which a future observation is expected to fall with a certain probability. Unlike Confidence Interval (CI) in frequentist statistics or Credible Intervals (also CI) in Bayesian statistics, which focus on the properties of the entire population (e.g., mean, variance), PIs are concerned specifically with individual unobserved data points [20, 106]. Occasionally, PI may address multiple future observations, but here we focus on the single-observation case.

The rigorous definition of a PI is as follows [64]:

Suppose $\{W_1, W_2, \ldots, W_n\}$ is an observed sample of size n from a population with unknown properties, and $\{X_1, X_2, \ldots, X_m\}$ is a future, unobserved sample from the same population. Let $g(X_1, X_2, \ldots, X_m)$ be a statistic function (e.g., the mean), and $\mathcal{P}_l(W_1, W_2, \ldots, W_n)$ and $\mathcal{P}_u(W_1, W_2, \ldots, W_n)$ represent the lower and upper bounds of the predicted interval for $\{X_1, X_2, \ldots, X_m\}$, respectively.

1. A **two-sided** $100\gamma\%$ PI is defined as:

$$\Pr[\mathcal{P}_l(W_1, Y_2, \ldots, W_n) \leq g(X_1, X_2, \ldots, X_m) \leq \mathcal{P}_u(W_1, Y_2, \ldots, W_n)] = \gamma. \tag{2.1}$$

2. A **one-sided** $100\gamma\%$ PI (e.g., upper-bounded) is given by:

$$\Pr[-\infty \leq g(X_1, X_2, \ldots, X_m) \leq \mathcal{P}_u(W_1, W_2, \ldots, W_n)] = \gamma. \tag{2.2}$$

This general definition allows for flexibility in constructing the bounds g_L and g_U, which will be explored later in this section. Notably, when predicting a single future observation (X_1), the PI simplifies to the **single-observation case**.

PI is particularly well-suited for regression problems in ML, as they focus on predicting future individual values. This characteristic makes PI especially useful for streaming data, where only historical statistics (e.g., a few numbers) are stored in memory, and simple calculations suffice to compute the interval boundaries. This efficiency aligns perfectly with the needs of large-scale data streams.

2.2.2.1 Evaluation for Prediction Interval

Objectively measuring the quality of PI is challenging as we need to consider both the width of the interval and its coverage. To achieve a fair and comprehensive evaluation, we consider two metrics used in previous studies [60], the Coverage and Normalized Mean Prediction Interval Width (NMPIW)[2]. Coverage represents the ratio of the ground truth falling within the predicted PI, i.e.:

$$\text{Coverage} = \frac{1}{N} \sum_{i=1}^{N} I_i \tag{2.3}$$

[2]Although the term "Normalized" is used, "Relative" would be more appropriate since it is defined based on the range of the targets rather than the range of the generated intervals (see Equation 2.4). However, to maintain consistency with prior publications, we retain the term NMPIW in this work.

where $I_i = 1$ if $y \in [\mathcal{P}_l, \mathcal{P}_u]$ and $I_i = 0$ otherwise; N is the total observation number. NMPIW is expressed in Equation 2.4:

$$\text{NMPIW} = \frac{\frac{1}{N}\sum_{i=1}^{N}(\mathcal{P}_{u_i} - \mathcal{P}_{l_i})}{R} \tag{2.4}$$

where $\mathcal{P}_{u_i}$ and $\mathcal{P}_{l_i}$ denote the upper and lower bounds for the i_{th} observation, respectively, while R represents the range of observed target values. As the equation suggests, NMPIW calculates the ratio of the average prediction interval width to the range of actual values, thus measuring the PI's effectiveness. For simplicity, coverage and the confidence level will be denoted as $\mathcal{C}$ and $\mathcal{L}$ throughout this thesis.

Although wider intervals typically ensure higher coverage, they tend to be less informative. Consequently, the objective of an effective PI method is to create intervals that capture approximately the desired confidence level of targets and are as narrow as possible, as demonstrated by low NMPIW values.

2.2.3 Classification

In this section, we introduce key algorithms for classification in SL.

- Decision trees are among the most popular approaches due to their efficiency, interpretability, and ability to handle evolving data. In streaming scenarios, Hoeffding Tree (HT) [30] is a canonical algorithm. It builds trees incrementally using the Hoeffding bound to ensure that the structure converges to the batch-learning equivalent while processing the data in a single pass.

  The Extremely Fast Decision Tree (EFDT) [95] improves upon HT by enabling faster updates and more efficient splitting decisions, which accelerate convergence to an accurate tree structure.

  More recently, the PLASTIC algorithm [68] was proposed as an improvement to EFDT by leveraging decision tree plasticity to restructure subtrees instead of pruning them during split revision.

- Ensemble methods have demonstrated superior performance in classification tasks for streaming data [45]. They combine multiple classifiers to enhance robustness and accuracy, particularly in non-stationary environments.

  Adaptive Random Forest (ARF) [44] extends the Random Forest framework to data streams. It uses an ensemble of Hoeffding Trees, where each tree only considers a random subset of features when splitting and adapts to concept drifts through drift detection mechanisms such as ADWIN.

  Leverage Bagging (LVB) [12] builds upon the standard bagging technique and incorporates ensemble diversity using randomization and under/over-sampling of incoming data streams. It efficiently adapts to evolving data distributions and concept drifts.

  Online Accuracy Updated Ensemble (OAUE) [16] focuses on updating ensemble weights dynamically based on individual classifiers' accuracy on recent data, ensuring adaptability to streaming data.

  Online Smooth Boosting (OSBoost) [19] applies boosting techniques to streaming data. It smoothens ensemble updates to reduce overfitting while maintaining adaptability in evolving data streams.

  Streaming Random Patches (SRP) [51] is an ensemble method designed for data streams, combining the strengths of bagging and random subspace techniques. Each base learner is trained on a randomly selected subset of features and examples, ensuring diversity and reducing overfitting. SRP efficiently adapts to evolving data streams by dynamically handling concept drifts and maintaining lightweight models, making it suitable for high-dimensional and large-scale streaming regression tasks.

2.2.3.1 Evaluation

In a stream learning scenario, classification tasks involve assigning class labels to instances in the data stream. Given the sequential nature of streaming

data, evaluation metrics must adapt to real-time, incremental learning scenarios, which is usually achieved by updating statistics instead of recalculating them from scratch. Below are the common evaluation metrics for streaming classification models.

Accuracy Accuracy measures the proportion of correctly classified instances:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Kappa Statistic (κ) The Kappa statistic evaluates the classifier's accuracy while accounting for the possibility of random chance:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

where P_o is the observed accuracy and P_e is the expected accuracy by random chance.

There are several variants of the Kappa Statistic, such as Kappa Temporal (κ_T) and Kappa Majority κ_M [10]. The key difference is the estimation of P_e. κ_T considers a no-change classifier for P_e, and κ_M utilizes a majority class classifier as the base-line.

Recall (True Positive Rate) Recall measures the ability of the classifier to correctly identify positive instances:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Precision Precision quantifies the proportion of predicted positive instances that are correctly classified:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

F1-Score The F1-Score is the harmonic mean of precision and recall, balancing their trade-off:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics provide a comprehensive evaluation of classification models in streaming scenarios. Accuracy and κ are widely used for general performance, while κ_T and κ_M are preferred for temporal and multi-class evaluations. Precision, recall, and F1-score are particularly relevant for imbalanced data streams, ensuring the model's reliability across all class distributions. In this thesis, Accuracy and κ are the primary metrics used for evaluation.

2.3 Datasets

The datasets used in this thesis will be listed and elaborated in this section.

2.3.1 Regression Datasets

Twenty datasets with real-valued targets are handpicked to serve the main focus of this research thesis – Regression. Table 2.1 exhibits an overview of these regression datasets.

Details of the datasets are listed as follows:

- Abalone [99]: aims at predicting the age of abalones based on their physical measurements;
- CPU Activity*[3]: predicts portion of time that cpus run in user mode;
- Naval Propulsion Plant [23]: predicts the GT Turbine decay state coefficient of naval propulsion plants based on other features;
- Ailerons [124]: artificial data addressing a control problem for flying a F16 aircraft;
- Miami Housing*: predicts sale price of single family houses in Miami;
- Elevators [124]: also predicts a control problem for flying a F16 aircraft;
- Bike [36]: predicts the usage of sharing bikes, both casual and registered;

[3]Datasets with * can be found at OpenML.

Table 2.1: Regression Datasets Overview

Drift Types – A: Abrupt; G: Gradual; N: No Drift; ?: Unknown

Dataset	Abbr.	#Feature	#Instance	Source	Drift
Abalone	ABL	8	4,177	Real	?
CPU Activity	CPU	22	8,192	Real	?
Naval Propulsion Plant	NPP	15	11,934	Real	?
Ailerons	AIL	40	13,750	Synth.	N
Miami Housing	MIA	16	13,932	Real	?
Elevators	ELV	18	16,599	Synth.	N
Bike	–	12	17,379	Real	?
California Housing	CAH	10	20,640	Real	?
Superconductivity	SCD	82	21,263	Real	?
Health Insurance	HI	12	22,272	Real	?
House8L	HOU	8	22,784	Real	?
NZ Energy Price	NZEP	11	26,121	Real	?
MetroTraffic	MT	7	48,204	Real	?
Diamonds	DIA	10	53,940	Real	?
Video Transcoding	VT	18	68,784	Real	?
Wave Energy	WE	49	71,993	Real	?
FriedmanGra	FR1	10	100,000	Synth.	A
FriedmanGsg	FR2	10	100,000	Synth.	G
FriedmanLea	FR3	10	100,000	Synth.	A
Hyper_a	HPR	10	500,000	Synth.	A

- California Housing [101]: predicts the median house price in California, US, in 1990s;
- Superconductivity [66]: predicts the critical temperature of superconductors based on other features;

- Health Insurance*: predicts the working hours of house wives based on their insurance information;
- House8L*: predicts the house price in 1990s, US;
- NZ Energy Price[116]: provided by Electricity Market Information website (EMI) based on difference locations (Point of Connections), aims at predicting the electricity price 24 hours in advance at real time. The Hamilton region is presented in this work, available at a Github repository. Details can be found in Chatper 5;
- MetroTraffic*: predicting the impacts of weather information on traffic volume;
- Diamonds [130]: predicts the price of diamonds based on the physical attributes;
- Video Transcoding[4] [28]: predicts the video transcoding time according to input information and output format;
- Wave Energy*: consists of positions and absorbed power outputs of wave energy converters, used to predict the total energy converted;
- FriedmanGra/Gsg/Lea: Friedman synthetic datasets family proposed in [77], publicly accessible generator at river machine learning platform. *Gra* – Global Recurring Abrupt drift; *Gsg* – Global and Slow Gradual drift; and *Lea* – Local Expanding Abrupt drift;
- Hyper_a [75]: Simulated datasets with 3 abrupt drifts, predicting the distance from a random point to a hyperplane in the space; the generator is available at CapyMOA [49] Online ML Repository;

[4]The feature "ID" was in String format, therefore removed from our experiments.

2.3.2 Classification Datasets

Classification tasks are also an important part of this thesis, and the datasets that are utilized are also carefully selected. Table 2.2 gives a general view of their respective properties, followed by thorough introduction.

Table 2.2: Classification Datasets Overview

Drift Types – A: Abrupt; G: Gradual; I_m: Moderate Incremental and; I_f: Fast Incremental; ?: Unknown

Datasets	#Instances	#Features	Types	Drifts	#Labels
AGR_a	1,000,000	9	Synth.	A	2
AGR_g	1,000,000	9	Synth.	G	2
LED_a	1,000,000	24	Synth.	A	10
LED_g	1,000,000	24	Synth.	G	10
RBF_f	1,000,000	10	Synth.	I_f	5
RBF_m	1,000,000	10	Synth.	I_m	5
RTG_a	100,000	10	Synth.	A	2
SEA_a	100,000	3	Synth.	A	2
SEA_g	100,000	3	Synth.	G	2
Airlines	539,383	7	Real	?	2
$CovT_S$	581,012	54	Real	-	7
CovT	581,012	54	Real	?	7
ElecX	45,312	8	Real	?	2
Elec	45,312	6	Real	?	2
KDD	4,898,431	41	Real	?	23
Nomao	34,465	118	Real	?	2
Sensor	2,219,803	5	Real	?	58
WISDM	5,418	45	Real	?	6

- AGRAWAL

  The AGRAWAL generator, introduced in [2], employs ten functions to

partition random data points, considering six categorical and three continuous attributes, into two classes. A 10% perturbation ratio introduces noise from a uniform random distribution to the features. The dataset includes two variants: AGR_a and AGR_g, featuring three abrupt and gradual drifts, respectively.

- LED

  In [15], Breiman et al. proposed the LED synthetic dataset generation, which creates datasets with 7 relevant and 17 irrelevant boolean features. The goal is to predict the number on an digital LED display based on the seven relevant features, 10% of which are inverted as noise. LED_a and LED_g simulate three abrupt and gradual concept drifts, respectively.

- RBF

  Radial Basis Function (RBF) generator creates moving (for simulating incremental drifts) centres with assigned weights and a random standard deviation. New instances are created based on the centre and the SD. We generate data with 50 centres, which move with a speed of 0.0001 for RBF_m, and 0.001 for RBF_f.

- RTG

  Random Tree Generator (RTG) [31] creates decision trees where the split attributes and the class values are randomly selected. New instances are also randomly generated from a uniform distribution and are assigned a class value by the corresponding leaf. The RTG_a dataset used in this work includes two abrupt drifts.

- SEA

  The SEA generator [114] yields three features, the last one of which is irrelevant to the label. The instances are generated in this way: the two dimensional space is separated into four blocks. Each block is associated with a different parameter θ. Random instance are generated and then assigned with a binary class label according to the blocks and the

associated θ. SEA_a and SEA_g simulate three abrupt and gradual drifts, respectively.

- Airlines

  The Airlines dataset was originally a regression dataset to predict the time delay of flights [78] and then transferred to a classification version that predicts whether the flights will be delayed or not. Four nominal and three numeric attributes are included in the Airlines dataset.

- Covertype

  Seven types of 30 × 30 meters of forest coverage, which are imbalanced, are included in the Covertype (CovT) dataset. Two variants of CovT datasets are included in this research. CovT is the normalised version of the original covertype dataset. CovT_S is a randomly shuffled version of CovT to eliminate any possible chronological relationships among the instances.

- Electricity

  Electricity (Elec) and extended Electricity (ElecX) were first proposed by Harries in [67]. They aim at predicting the electricity price tendency (UP or DOWN). The data was collected in a state in Australia in over 2 years, where the electricity prices depend on supply and demand in that area. In comparison to the Elec dataset, ElecX includes two more attributes: date and day of the week.

- KDD

  KDD'99 dataset was first used in the third Knowledge Discovery and Data Mining Tool Competition held in 1999 [113]. This dataset aims at predicting 22 types of cyber attacks along with a normal status. This dataset can be treated as a data stream since the instances are presented sequentially [1].

- Nomao

Nomao is a real-world application: a location search engine. [17] launched a machine learning challenge and provided the associated dataset to the UCI machine learning dataset repository [4]. The Nomao dataset contains 34,465 data points. The goal is to predict either one of the two places every data point belongs to, based on information like names, phones, and so on.

- Sensor

  Sensor dataset contains sensor data and the recording time during an MIT experiment.

Chapter 3

Self-Optimising K Nearest Leaves (SOKNL)

Regression for data stream learning is relatively overlooked compared to its classification counterpart. This chapter introduces the proposed streaming ensemble-based regression algorithm – Self-Optimising K Nearest Leaves (SOKNL).

3.1 Introduction

Adaptive Random Forest for Regression (ARF-Reg), as one of the most used ensemble regressors for data stream mining, adapts the traditional Random Forest (RF) to evolving data streams technique and yields accurate continuous predictions. Using Fast Incremental Regression Tree with Drift Detection (FIRT-DD)[1] as the base learner, ARF-Reg induces diversity to the ensemble members with bootstrapping and random feature space techniques.

Distance information usually plays an important role in the machine learning research, due to its intuitiveness and simplicity. As a measurement for quantifying the similarity (or dissimilarity) between two data points, distance

[1] Although the Fast Incremental Model Tree with Drift Detection is more accurate in most cases, ARF-Reg uses the a modified version – FIRT-DD – as the base learner, which utilizes the average value of the observed targets as the prediction, since the linear model in the leaves causes overflow problem due to its SGD-based nature.

metrics include Manhattan Distance, Cosine Similarity, and the most commonly used member – Euclidean Distance. Many algorithms rely on a distance metric for their learning and prediction procedure, such as K Means Clustering (`k-means`), Support Vector Machine (SVM), and K Nearest Neighbours (`kNN`), which is highly relevant for the content of this chapter. In the streaming scenario, `kNN` is usually implemented with a sliding window.

The predictive performance of `kNN` and ARF-Reg is remarkable (as shown by previous studies [46, 50]), but they still have some shortcomings. `kNN` requires many distance calculations during prediction, which can be slow and prohibitive. On the other hand, due to the random nature of ARF-Reg, not every tree in it may produce accurate predictions, which is adding unnecessary noise to the final prediction when all those individual trees are aggregated. This motivates our work to integrate both techniques into a single algorithm for improved accuracy.

The following contributions are outlined:

- a novel approach to regression analysis for data streams by combining ARF-Reg and KNN into the Self-Optimising K Nearest Leaves (SOKNL) algorithm;
- through the combination of both algorithms, we improve the predictive performance of ARF-Reg without adding too much extra pressure on computational resources.
- a dynamic parameter-choosing methodology enabling the algorithm to self-adapt the value of k;
- an extensive empirical evaluation and a statistical test show how the new method compares with other previous state-of-the-art online regression algorithms.

3.2 Self-Optimising K Nearest Leaves

As mentioned in Chapter 2, both kNN and ARF-Reg have their own specific inherent shortcomings.

Streaming versions of KNN rely only on a sliding window where older instances are forgotten to limit memory usage. This strategy can be beneficial in situations where older instances no longer represent the current concept (i.e. a drift has happened); however, the older instances are relevant to building a more robust model in many situations. Under the circumstances where the concept of the data streams has not drifted, which means all observed data points are valuable for establishing learning model, we may want to somehow store those information from older instances. One of the core questions for streaming kNN is how to retain some of that older but still relevant information.

ARF-Reg randomly trains and grows ensemble trees on different subspaces of the datasets and features. Thus this approach relinquishes at random some bits of informative data for each ensemble member to introduce diversity into the ensemble. For instance, the artificial dataset "fried", which is used in our experiments, contains ten features, only five out of which are related to the target value. If some trees in ARF-Reg are trained mainly on the irrelevant features, they can potentially yield inaccurate single predictions, negatively impacting the aggregated ensemble prediction.

Consequently, the idea of k Nearest Leaves (KNL) – which is the integration of kNN and ARF-Reg (see in Section 3.2.1) – emerged. The intuition is to overcome the transient behaviour of kNN by using the trees in the ARF-Reg ensemble to keep information for much longer, by maintaining a centroid for each leaf in each tree; and to improve the ensemble prediction aggregation procedure using kNN over the centroids selected by each tree at prediction time. Using centroids to summarize information is common, one of the most famous applications is the Clustering Feature Tree (CF-Tree) from BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [135]. In our approach, each leaf is mapped to a "data point" for later use of the kNN procedure.

In general, there are two perspectives to comprehend the KNL algorithm. From the standpoint of kNN, ARF-Reg is providing the leaves – which can also be regarded as micro-clusters, condensed into one compact and robust representation: a centroid. From the ARF-Reg perspective, the prediction is more robust by excluding leaves that may be too dissimilar to the current prediction instance.

Figure 3.1 is a diagram visualising the idea of KNL.

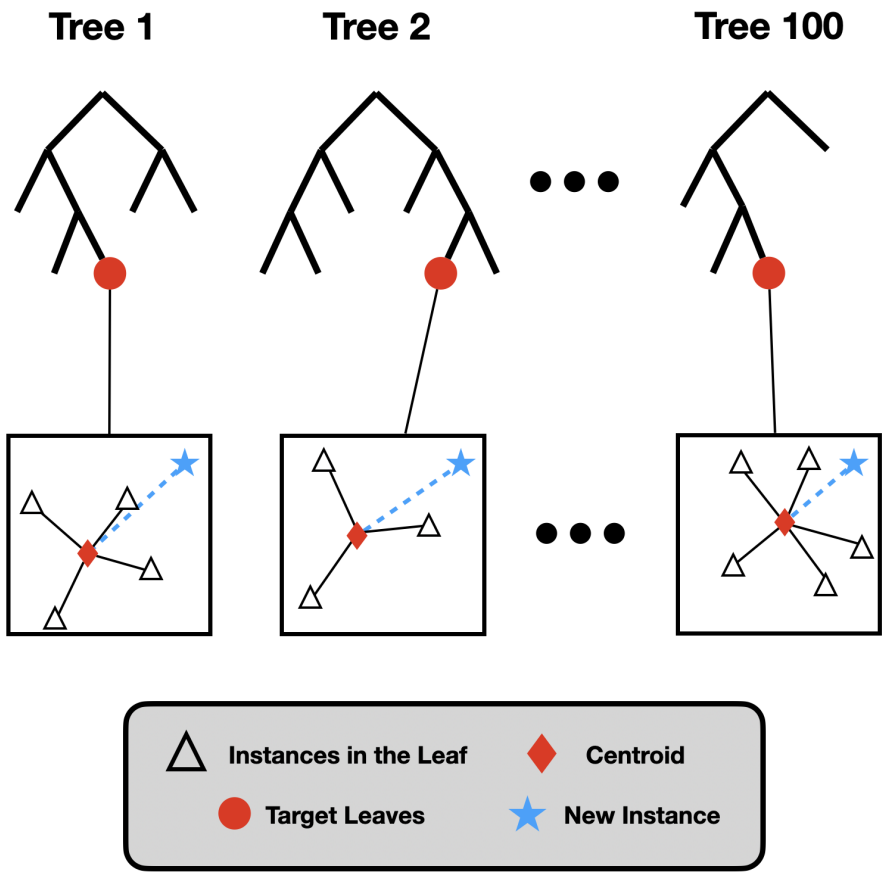


Figure 3.1: Diagram of K Nearest Leaves

Our approach has a virtue as Fittingly for a streaming algorithm, all the calculations, learning, updates, and comparisons can be accomplished by only using the statistics store in the system. Therefore, instances are not stored in the trees or the forests. For instances, the centroid in the leaf could be calculated by a counter and an array of the sums of values of individual features of all instances that have been seen at a leaf. Memory constraint is complied

with via thus approach.

There is a potential downside to this combination, as KNL now adds one more hyperparameter to the set of hyperparameters of ARF-Reg: the k value needed for selecting the closest leaves. To simplify the application of KNL, we also introduce a technique for automatically and dynamically selecting a good value for this "k" hyperparameter. More details are given below in Section 3.2.2. This modified version of KNL is called Self-Optimising k Nearest Leaves, abbreviated as SOKNL.

In the rest of this section, SOKNL will be explained more specifically. In general, our contributions can be separated into two parts: a) Integrating `kNN` with ARF-Reg, and b) the Self-Optimising Strategy. See the pseudo-code of SOKNL in Algorithm 1 for an intuitive understanding.

3.2.1 Integrating K-Nearest Procedure with ARF-Reg

As aforementioned in Chapter 2, ARF-Reg is an ensemble of multiple tree learners. Every tree will sort an incoming instance into exactly one of its respective leaves. Thus an ensemble of n trees will return n leaves for prediction.

3.2.1.1 Distance Measurements

Our approach requires a technique to measure the distance from an instance to a leaf (a cluster of instances). In principle, there are at least two options available:

- Calculate all the distances from the incoming instance to the instances in the leaf, and use the average as the measurement of the distance between an instance and a leaf.
- Maintain a centroid in each leaf, which is an average of all features of all the leaf's instances. Thereupon, the distance from an instance to a leaf can be defined as the distance to the leaf's centroid.

Algorithm 1: **Self-Optimising K Nearest Leaves**

$\mathcal{E}$: Ensemble; $\mathcal{N}_{\mathcal{E}}$: Ensemble size; $\mathcal{E}_i$: i_{th} ensemble member; E: Evaluators; E_i: i_{th} evaluator; $\mathcal{C}_i$: i_{th} centroid; $\mathcal{K}_{best}$: the best $\mathcal{K}$ observed; $\mathbb{R}$: list of the recorded performance for each $\mathcal{K}_i$ value.

Data: Data Streams : $\mathcal{D}$

Result: Prediction : $\mathcal{P}$

`/* Initialization */`

$\mathcal{K}_{best} \leftarrow N_{\mathcal{E}}$;

for $i < N_{\mathcal{E}}$ **do**

  build $\mathcal{E}_i$ and E_i;

`/* Execution */`

while $\mathcal{D}$ *has more examples* **do**

  `/* test example` **e** `*/`

  Initialize $\mathbb{D}$; `/* list of distances with all 0s */`

  for $i < N_{\mathcal{E}}$ **do**

    $\mathcal{C}_i \leftarrow \mathcal{E}_i$.getCentroid(**e**);

    $\mathbb{D}_i \leftarrow$calculateDistance($\mathcal{C}_i$, **e**);

  sort $\mathcal{E}$ by $\mathbb{D}$;

  `/* predict by the closest` $\mathcal{K}_{best}$ `leaves */`

  for $i < \mathcal{K}_{best}$ **do**

    $\mathcal{P}_i \leftarrow \mathcal{E}_i$.predict(**e**);

  return $\mathcal{P} \leftarrow \frac{\sum_{i=0}^{\mathcal{K}_{best}} \mathcal{P}_i}{\mathcal{K}_{best}}$;

  `/* train with example` **e** `*/`

  for $i < N_{\mathcal{E}}$ **do**

    evaluate $\mathcal{E}_i$; `/* update` E_i `&` $\mathcal{A}_i$ `*/`

    evaluate $\mathcal{K} = i$; `/* update` $\mathbb{R}$ `*/`

    $\mathcal{L}_i$.train(**e**); `/* do regular train */`

  $\mathcal{K}_{best} \leftarrow \underset{index}{argmin}\mathbb{R} + 1$; `/* update` $\mathcal{K}_{best}$ `*/`

Both options are feasible and have their own pros and cons. The former one keeps more information which gives it more flexibility. It also needs much more memory and runtime. Due to the data stream algorithms' requirements for timeliness and memory space utilisation, the latter option is more promising, and consequently suitable for SOKNL.

3.2.1.2 Selection of K Nearest Leaves

Instead of aggregating the leaf predictions of all trees in the ensemble, SOKNL only averages the target values of the k nearest leaves. Since ARF-Reg trees are growing semi-randomly, some leaves will be more informative than others, for a given instance. SOKNL is able to select the most relevant leaves, thanks to the centroids stored within each leaf.

The construction of centroids in our method is simple. For all the instances in a leaf, we take the mean values of each feature. For example, we average the values of Feature 1 of all the instances in the leaf and put it at the Feature 1 position of the centroid. With this method, all the information of the instances in a certain leaf is compressed into a robust, space- and memory-efficient, and incrementally calculable representative instance – the centroid.

Another notable point is, although SOKNL also requires distance calculation, it only performs one distance calculation per tree or 100 if it is for an ensemble of 100 trees (it is already a quite large choice for ensemble number). In the Sliding Window `kNN` case, the number of distance calculations equals to the window length, which is typically several thousands or even more. Hence, in terms of time for calculating distances, SOKNL manages to surpass the `kNN` algorithms easily.

3.2.2 Self-Optimising Strategy

Choosing good k values for `kNN` requires specialised knowledge and sometimes "luck". Even for experts, there is no convenient way for finding the best k. Moreover, there may even not be a "global best" k due to concept drifts or

other causes. Therefore, a self-optimising regime is introduced into SOKNL to automatically determine the currently best-performing value for k.

Self-optimising, or self-tuning, is a popular way to reduce the labour of hyperparameter tuning [73] [126] [94]. Here, the self-optimising mechanism measures performance using the Sum Squared Error (SSE). For every possible value of k, for k in $1..k_{max}$, an evaluator keeps track of the SSE for this k value. Updating all k evaluators can be done very efficiently: first, all leaves are sorted by distance, and then the predictions for larger and larger values of k are incrementally computed in one linear sweep over the sorted leaves, resulting in $O(k_{max} * log(k_{max})$ runtime. Even though the possibility of two k values maintaining exact same SSE is extremely small, there is no guarantee for that not to happen, especially at the beginning of the learning progress. We simply choose the smaller k value in such cases.

3.2.3 Change Adaptation

How our approach adapts to drift is straightforward. It relies on the built-in adaption methodology in ARF-Reg as well as implicit adaptation of the centroids.

In ARF-Reg, the concept drift detection and adaptation exist in both trees and ensembles. In every node of the tree learners, there is a Page-Hinckley (PH) test [97], which is an extension of the CUSUM test [102]. PH maintains two variables in the system, a cumulative value m_t and the minimum of m_t until the current moment M_t. m_t is defined as the sum of the difference between the current target value and the average value ($\bar{x}$) of the whole time and an indicative parameter α, which is responsible for controlling how much of the change ought to be identified as a drift. Equations 3.1 and 3.2 denote m_t and M_t respectively.

$$m_t = \sum_{t=1}^{N}(x_t - \bar{x} - \alpha) \tag{3.1}$$

$$M_t = min(m_t, t = 1, 2, ..., N) \tag{3.2}$$

Consequently, the PH test at the moment of t is defined by Equation 3.3:

$$\mathrm{PH}_t = m_t - M_t \tag{3.3}$$

If the PH_t is larger than a threshold parameter λ that is specified by the user to command the sensitivity of the test, a drift is confirmed. Then the associated node will be removed and a new subtree will be built.

ARF-Reg has an ADaptive WINdow (ADWIN) [11] algorithm as an external change detector at the ensemble level. The operation of ADWIN requires storing two windows that have window lengths of $\mathcal{W}_{old}$ and $\mathcal{W}_{new}$. The mean values of each window, μ_{old} and μ_{new} , are used to compare to the corresponding Hoeffding Bound ϵ by inequality 3.4.

$$\mid \mu_{old} - \mu_{new} \mid \geq \epsilon = \sqrt{\frac{1}{2m} \cdot \ln \frac{4 \mid \mathcal{W} \mid}{\delta}} \tag{3.4}$$

where δ denotes a use-defined confidence level in the range of $[0, 1]$, and m is the harmonic mean of the sub-windows' length as in Equation 3.5.

$$m = \frac{2}{\frac{1}{|\mathcal{W}_{old}|} + \frac{1}{|\mathcal{W}_{new}|}} \tag{3.5}$$

If inequality 3.4 holds, a drift is detected and the old window is dropped.

In addition, SOKNL has another way to adapt to drift. The centroids in SOKNL are moving according to incoming instances. If instances on average shift, the centroids will shift as well, thus implicitly providing a certain capability for drift adaptation, on top of the explicit change detectors. Furthermore, when a leaf is split into two new leaves, the old centroid is deleted, and two new centroids will be computed from the newly arriving instances, yet again supporting adaptation to potential drifts implicitly.

3.3 Experimental Settings

This section introduces the experimental settings, including the data preprocessing, the involved algorithms, and the evaluation manners.

3.3.1 Data Pre-processing

Data pre-processing techniques are commonly used in the regression tasks. In this work, we exploited two simple yet powerful ones – Min-Max Normalization and Z-score Standardization.

- **Min-Max Normalization**

  scales the values of a dataset to fit within a specific range, typically between 0 and 1. This method is particularly useful when the features of a dataset have different units or ranges, as it brings all features to a common scale without distorting the relationships between them.

  The Min-Max normalization formula is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.6}$$

  Where:

  - x is the original data point,
  - $\min(x)$ and $\max(x)$ are the minimum and maximum values of the feature, respectively,
  - x' is the normalized value.

  By applying Min-Max normalization, the minimum value of the feature becomes 0, and the maximum value becomes 1, while other values are linearly transformed to fit within this range. This technique is particularly suitable for algorithms that rely on the magnitude of values, such as gradient-based optimization methods (e.g., perceptron), where normalized data helps improve the convergence rate and performance.

- **Z-score Standardization**

  also known as standardization or zero-mean normalization, is a technique used to transform data so that it has a mean of 0 and a standard deviation of 1. This process is particularly useful in machine learning and statistical

analysis when different features in the data have varying units or scales, allowing them to be compared on the same scale.

The Z-score standardization formula is:

$$z = \frac{x - \mu}{\sigma}$$

Where:

- x is the original data point,
- μ is the mean of the data,
- σ is the standard deviation of the data,
- z is the standardized value (also called the Z-score).

After applying Z-score standardization, the transformed data will have a mean of 0 and a standard deviation of 1. This technique is particularly helpful for algorithms that rely on the assumption that the data is normally distributed.

It is worth mentioning that these two techniques are modified for the streaming scenario. This is achieved by calculating the necessary values using only the updated statistics up to the current moment.

3.3.2 Algorithms

For comparison, experiments on the above datasets with some other algorithms are conducted, including Traditional `kNN`; Self-Optimising `kNN`[2]; FIMT-DD; ORTO; AMRules; ARF-Reg; and k Nearest Leaves(KNL)[3].

Most of these algorithms have been introduced in Section 2, yet there are several things to be specified here. In order to make our experiments reproducible, the hyperparameters for the algorithms are included here. Moreover,

[2]A variant of `kNN` with the proposed self-optimising method.

[3]The intermediate version of SOKNL without self-optimising.

the hyperparameters for ARF-Reg are introduced particularly since our approach is based on the ARF-Reg and the basic hyperparameters are the same. Notably, the ensemble numbers of ARF-Reg and SOKNL in our experiments are arbitrarily fixed to 100. That is because the number could be too small, otherwise, the centroid selecting procedure would be too limited to be effective. Also, large numbers could cause a dramatic increase in the computational resource requirements. We made some attempts with some "appropriate" ensemble numbers and chose 100 as it is the best in terms of balancing effectiveness and efficiency.

- **Traditional KNN:** The classic KNN with fixed $k \in \{1, 5, and 10\}$ and a window length of 1000.
- **Self-Optimising k Nearest Neighbours(SOKNN):** A variant of KNN with the same auto-selecting k value method as used in SOKNL.
- **FIMT-DD:** The hyper-parameter values are taken from [77].
- **ORTO:** Option trees using the default parameter setting from [79].
- **AMRules:** Adaptive Model Rules (AMRules) is the first rule algorithm in regression for data streams [3]. We use the same algorithm setup as in that paper, except for the split confidence parameter δ, where $1E-7$ is used instead of 0.01 to ensure a faster split.
- **ARF-Reg:** See details in Chapter 2 and the algorithm hyper-parameters are set as follows: *a*) **Ensemble learner:** FIMT-DD; *b*) **Ensemble Size:** 100; *c*) **Feature Subspace Size:** 60%; *d*) λ **for Poisson Distribution:** 6 (for simulating the bootstrapping procedure) .
- **K Nearest Leaves:** For comparison to SOKNL, a plain KNL with a fixed k value is also in included. Values used for a k are 1, 5, and 10. We choose these k values as they illustrate the improvements by increasing the number of neighbours. The differences are more significant when k values are small. After 10, the improvements start getting inconspicuous.

3.4 Experimental Results

In this section, we illustrate the experimental results and relevant discussion. Tables demonstrated in this section adhere to the following rules:

- cell values are in the format of "$\text{Mean}_{\text{stdev}}$".
- small standard deviations (< 0.001) are omitted.
- **bold** font highlights the best results in the same row (dataset).
- underlines indicate the cases where SOKNL outperformed ARF-Reg.
- " – "s fill the blanks where the values were "nan"s or overflows (see 3.1).
- the last row shows the average ranking of each algorithm[4].

3.4.1 Regular Results

Tables 3.1 and 3.2 present the Coefficient of Determination (R^2) and Root Mean Squared Error (RMSE), respectively. Similar conclusions can be drawn from these two tables, therefore, our analysis in this section focuses on Table 3.1.

Overall, SOKNL performed the best on 9 out of 20 datasets. Its average ranking of 2.7 among 12 algorithms further affirms its outstanding performance.

The second key takeaway is that SOKNL improves upon ARF-Reg in 19 datasets. The only exception is the NZEP dataset, where the difference is less than 1%. This aligns with the purpose of the SOKNL algorithm, which is to enhance the performance of ARF-Reg.

Notably, there are several cases where results show overflow. Most of these occur with tree- and rule-based learners, due to the "Perceptron" model used by these learners. This SGD-based linear model is commonly applied for stream regression tasks. If not properly controlled, some predicted values

[4] overflowed ones are considered lowest ranking

Table 3.1: Full Coefficient of Determination Results on Regular Datasets

Datasets	AMRules	FIMT-DD	ORTO	KNN-1	KNN-5	KNN-10	RAMRules	ARF-Reg	KNL-1	KNL-5	KNL-10	SOKNL
ABL	$0.544_{0.022}$	0.368	-5.26	0.329	0.492	0.5	$0.575_{0.001}$	$0.485_{0.002}$	$0.025_{0.021}$	$0.381_{0.009}$	$0.455_{0.007}$	**$\underline{0.58}_{0.003}$**
AIL	–	–	–	0.597	0.726	**0.73**	–	$0.55_{0.002}$	$0.155_{0.009}$	$0.29_{0.009}$	$0.33_{0.009}$	$\underline{0.56}_{0.004}$
Bike	$0.42_{0.09}$	–	–	0.548	0.482	0.473	$0.603_{0.005}$	$0.744_{0.004}$	$0.613_{0.008}$	$0.735_{0.004}$	$0.764_{0.004}$	**$\underline{0.796}_{0.003}$**
CAH	$0.714_{0.005}$	0.567	-0.633	0.66	0.739	0.727	$0.745_{0.002}$	0.722	$0.02_{0.012}$	$0.476_{0.005}$	$0.611_{0.004}$	**$\underline{0.775}$**
CPU	$0.54_{0.034}$	0.343	-18.152	0.907	0.914	0.896	$0.72_{0.009}$	$0.768_{0.003}$	$0.793_{0.009}$	$0.856_{0.008}$	$0.88_{0.007}$	**$\underline{0.93}_{0.002}$**
DIA	$0.931_{0.001}$	**0.975**	0.264	0.925	0.916	0.911	$0.948_{0.002}$	0.914	$0.55_{0.008}$	$0.82_{0.004}$	$0.85_{0.003}$	$\underline{0.928}$
ELV	$0.443_{0.065}$	–	–	0.486	**0.617**	0.604	$0.555_{0.019}$	$0.387_{0.005}$	$0.094_{0.006}$	$0.206_{0.007}$	$0.246_{0.007}$	$\underline{0.465}_{0.007}$
FR1	0.745	0.738	-1.341	0.442	0.696	0.707	0.768	$0.653_{0.002}$	$0.642_{0.002}$	$0.767_{0.001}$	$0.785_{0.001}$	**$\underline{0.81}_{0.001}$**
FR2	0.747	0.74	-2.192	0.432	0.69	0.701	$0.778_{0.001}$	$0.644_{0.002}$	$0.628_{0.002}$	$0.756_{0.002}$	$0.774_{0.002}$	**$\underline{0.801}_{0.002}$**
FR3	0.738	0.766	-1.112	0.386	0.649	0.659	$0.776_{0.001}$	$0.669_{0.002}$	$0.651_{0.003}$	$0.765_{0.002}$	$0.782_{0.002}$	**$\underline{0.809}_{0.002}$**
NZEP	$-0.017_{0.218}$	-1.524	0.219	0.393	0.463	0.469	$0.461_{0.009}$	**$0.585_{0.003}$**	$0.187_{0.009}$	$0.41_{0.005}$	$0.455_{0.002}$	$0.584_{0.006}$
HI	0.321	0.038	-0.928	-0.248	0.23	0.284	**$0.347_{0.001}$**	$0.105_{0.001}$	$-0.071_{0.02}$	$0.067_{0.004}$	$0.099_{0.003}$	$\underline{0.136}$
HOU	$0.419_{0.001}$	0.4	-1.528	0.23	0.428	0.429	$0.482_{0.003}$	$0.54_{0.002}$	$0.401_{0.007}$	$0.514_{0.005}$	$0.533_{0.004}$	**$\underline{0.578}_{0.003}$**
HPR	0.927	**0.937**	-3.498	0.658	0.825	0.835	0.933	0.606	$0.728_{0.001}$	0.834	0.848	$\underline{0.859}$
MT	$-2.962_{1.079}$	–	–	-0.454	-0.017	0.036	$-0.303_{0.096}$	0.264	$-0.104_{0.007}$	$0.132_{0.001}$	0.162	**$\underline{0.308}$**
MIA	–	0.524	-0.705	0.778	**0.829**	0.81	–	$0.733_{0.002}$	$0.246_{0.015}$	$0.499_{0.004}$	$0.589_{0.005}$	$\underline{0.766}_{0.002}$
NPP	0.998	–	–	0.737	0.727	0.728	**1.0**	$-1.612_{0.058}$	$0.706_{0.082}$	$0.736_{0.003}$	0.737	$\underline{0.739}$
SCD	-0.274	–	–	**0.853**	0.826	0.798	$0.459_{0.017}$	$0.691_{0.001}$	$0.391_{0.008}$	$0.609_{0.004}$	$0.651_{0.003}$	$\underline{0.697}_{0.001}$
VT	–	–	–	**0.952**	0.905	0.839	–	$0.546_{0.003}$	$0.4_{0.014}$	$0.748_{0.004}$	$0.715_{0.004}$	$\underline{0.751}_{0.004}$
WE	$0.99_{0.012}$	0.876	–	0.499	0.673	0.679	**0.998**	$-4.053_{0.027}$	$-0.765_{0.168}$	$0.427_{0.008}$	$0.508_{0.003}$	$\underline{0.626}$
Avg. Rank	6.4	8.6	11.75	6.9	5	4.8	4.95	6.3	9.1	6.6	5.3	**2.7**

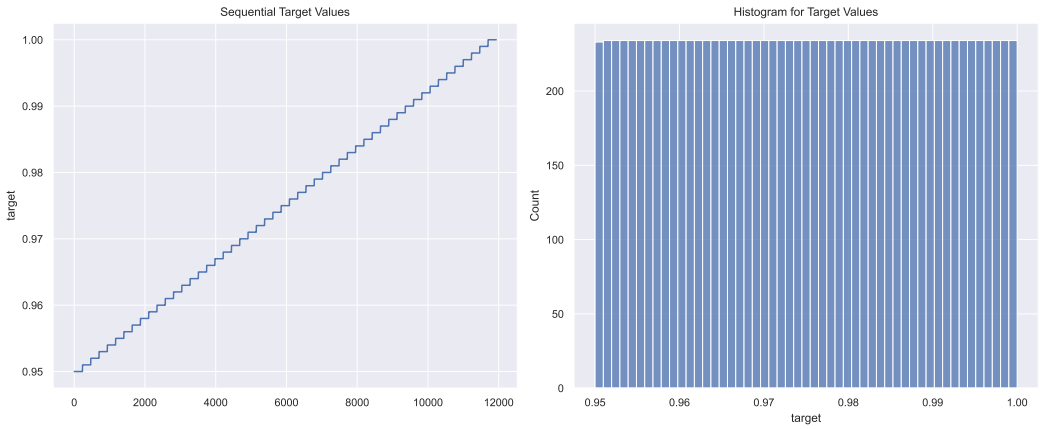


Figure 3.2: Visualization of Target Values from NPP Dataset

can deviate significantly from the targets, distorting error-based measurements such as R^2 and RMSE. ARF-Reg and SOKNL avoid this by using the average value in the leaf node as the output, instead of relying on the prediction from a linear model.

However, this approach has its limitations. While it helps prevent overflows, the performance of the average model is generally inferior to that of a linear model. In certain datasets, such as DIA and WE, perceptron-based algorithms deliver much better results. For instance, RAMRules achieves an R^2 score of 0.998 on the WE dataset, indicating an almost perfect match between the predicted values and the ground truths.

Moreover, for more challenging datasets like Health Insurance (HI), AMRules and RAMRules hold significant advantages over other algorithms.

The most extreme case is seen with RAMRules on the NPP dataset, where it produces an R^2 score of 1.0, which is almost unheard of in regression research. This result is likely more attributable to the nature of the NPP dataset rather than the perceptron model itself.

Figure 3.2 includes both a sequential plot and a histogram of the NPP target values, clearly showing a distinct pattern. The values gradually increase from 0.95 to 1, with each value persisting for a similar duration. This makes it easier for rule-based learners to capture, which explains the near-perfect performance of AMRules and RAMRules. However, it is unusual for a

dataset to exhibit such an obvious pattern. Further discussion on this issue will be considered in future work.

Table 3.2 shows the RMSE results from the same set of experiments in Table 3.1. Because of rounding up, more ties can be found in RMSE results, especially when the error values are considerably small. As a consequence, the outcomes are slightly different from Table 3.1. In Table 3.2, SOKNL also performed best on 9 datasets, but got an average ranking at 2.85. SOKNL outperformed ARF-Reg in 18 cases and tied once.

3.4.2 Normalized Results

This section exhibits the results on normalized datasets. However, most of the tendencies are similar to regular datasets. Therefore, we only highlight important points here.

With the datasets normalized, SOKNL managed to outperform ARF-Reg on all datasets. It also ranked 2.8 among 12 algorithms.

It is worth mentioning that, normalization sometime is considered as an approach to avoid overflow. This can be observed in Table 3.3 and 3.4. Table 3.2 reduced the number of overflowed cases from 21 in Table 3.1 to 19. There is no overflows in RAMRules after normalization. In terms of RMSE, the overflowed cases decreased from 12 (Table 3.2) to 6 (Table 3.4).

3.4.3 Standardized Results

The observation in standardized datasets are quite similar to normalized ones. Therefore, we will not elaborate further here.

3.4.4 CD Diagram of RMSE with Normalized Data

In this section, we present a Critical Difference Diagram (CDD) in generated with the RMSE results from the normalized data (Section 3.4.2). Note that the missing values ("–"s in the table) are replaced with +inf to ensure they

Table 3.2: Full RMSE Results on Regular Datasets

Datasets	AMRules	FIMT-DD	ORTO	KNN-1	KNN-5	KNN-10	RAMRules	ARF-Reg	KNL-1	KNL-5	KNL-10	SOKNL
ALB	$2.248_{0.053}$	2.652	8.345	2.731	2.378	2.358	$2.169_{0.003}$	$2.393_{0.006}$	$3.293_{0.036}$	$2.624_{0.019}$	$2.462_{0.015}$	$\underline{\mathbf{2.162}}_{0.007}$
AIL	0.568	1.036	1.944	0.072	**0.059**	**0.059**	$0.625_{0.016}$	0.076	0.104	0.095	0.093	$\underline{0.075}$
Bike	138_{10}	572	2883	122	131	131	114_{1}	92_{1}	113_{1}	93_{1}	88_{1}	$\underline{\mathbf{82}}_{1}$
CAH	61697_{573}	75972	147522	67306	59010	60277	58310_{190}	60824_{80}	114319_{725}	83560_{393}	72007_{360}	$\underline{\mathbf{54732}}_{90}$
CPU	$12.494_{0.449}$	14.948	80.679	5.637	5.414	5.945	$9.754_{0.152}$	$8.875_{0.049}$	$8.39_{0.191}$	$6.986_{0.183}$	$6.372_{0.199}$	$\underline{\mathbf{4.887}}_{0.067}$
DIA	1049_{11}	**633**	3423	1089	1156	1187	907_{16}	1171_{5}	2675_{23}	1694_{16}	1545_{14}	$\underline{1068}_{7}$
ELV	0.005	0.338	0.071	0.005	**0.004**	**0.004**	**0.004**	0.005	0.006	0.006	0.006	0.005
FR1	2.517	2.548	7.621	3.722	2.749	2.696	$2.399_{0.004}$	$2.933_{0.007}$	$2.981_{0.008}$	$2.404_{0.007}$	$2.312_{0.007}$	$\underline{\mathbf{2.174}}_{0.007}$
FR2	2.503	2.538	8.896	3.751	2.773	2.721	$2.347_{0.006}$	$2.97_{0.01}$	$3.038_{0.01}$	$2.458_{0.012}$	$2.365_{0.012}$	$\underline{\mathbf{2.224}}_{0.013}$
FR3	2.768	2.613	7.855	4.236	3.203	3.155	$2.558_{0.006}$	$3.109_{0.01}$	$3.193_{0.015}$	$2.619_{0.011}$	$2.526_{0.01}$	$\underline{\mathbf{2.362}}_{0.012}$
NZEP	108.987_{11}	172.54	96.0	84.631	79.59	79.172	$79.771_{0.684}$	**69.983**	97.936	83.414	80.215	70.09
HI	$15.425_{0.001}$	18.36	25.987	20.907	16.421	15.838	$\mathbf{15.12}_{0.014}$	$17.706_{0.014}$	$19.366_{0.18}$	$18.08_{0.034}$	$17.768_{0.027}$	$\underline{17.396}_{0.008}$
HOU	40296_{42}	40945	84042	46387	39983	39936	38026_{104}	35863_{87}	40927_{226}	36842_{184}	36135_{164}	$\underline{\mathbf{34344}}_{117}$
HPR	$2.018_{0.003}$	**1.88**	15.869	4.374	3.134	3.041	$1.934_{0.003}$	$4.697_{0.004}$	$3.902_{0.008}$	$3.047_{0.005}$	$2.921_{0.005}$	$\underline{2.815}_{0.006}$
MT	3925_{510}	–	–	2396	2004	1950	2267_{82}	1705_{1}	2087_{6}	1851_{1}	1818_{1}	$\underline{\mathbf{1652}}_{1}$
MIA	–	218931	414405	149613	**131434**	138294	–	164074_{476}	275671_{2767}	224622_{882}	203484_{1151}	$\underline{153552}_{799}$
NPP	0.001	–	–	0.009	0.009	0.009	**0.0**	0.028	$0.009_{0.001}$	0.009	0.009	$\underline{0.009}$
SCD	38.668	–	–	**13.119**	14.279	15.416	$25.202_{0.406}$	$19.038_{0.041}$	$26.741_{0.179}$	$21.43_{0.11}$	$20.225_{0.074}$	$\underline{18.843}_{0.033}$
VT	–	–	–	**3.517**	4.971	6.471	–	$10.849_{0.034}$	$12.473_{0.147}$	$8.08_{0.066}$	$8.601_{0.06}$	$\underline{8.036}_{0.067}$
WE	10041_{5784}	39849	506150	79952	64562	64054	$\mathbf{5047}_{43}$	253959_{682}	149942_{7177}	85491_{600}	79211_{236}	$\underline{69072}_{63}$
Avg. Rank	6.5	8.5	11.55	7.05	4.8	4.7	4.6	6.3	9.1	6.6	5.4	**2.85**

Table 3.3: Full Coefficient of Determination Results on Normalized Datasets

Datasets	AMRules	FIMT-DD	ORTO	KNN-1	KNN-5	KNN-10	RAMRules	ARF-Reg	KNL-1	KNL-5	KNL-10	SOKNL
ABL	$0.573_{0.006}$	0.374	–	0.309	0.505	0.513	**0.587**	$0.477_{0.002}$	$-0.001_{0.013}$	$0.344_{0.007}$	$0.424_{0.005}$	$\underline{0.568}_{0.003}$
AIL	$-1.394_{0.106}$	–	–	0.605	**0.721**	0.713	$-1.21_{0.045}$	$0.551_{0.004}$	$0.253_{0.012}$	$0.347_{0.009}$	$0.381_{0.006}$	$\underline{0.569}_{0.001}$
Bike	$0.062_{0.266}$	–	–	0.551	0.509	0.498	$0.614_{0.004}$	$0.745_{0.004}$	$0.536_{0.015}$	$0.692_{0.009}$	$0.729_{0.006}$	$\underline{\mathbf{0.794}}_{0.003}$
CAH	$0.705_{0.003}$	0.644	-0.524	0.655	0.736	0.723	$0.747_{0.001}$	0.722	$0.007_{0.022}$	$0.488_{0.004}$	$0.624_{0.003}$	$\underline{\mathbf{0.777}}$
CPU	$0.82_{0.012}$	0.441	–	0.897	0.907	0.886	$0.878_{0.005}$	$0.79_{0.002}$	$0.657_{0.036}$	$0.805_{0.014}$	$0.859_{0.011}$	$\underline{\mathbf{0.934}}_{0.003}$
DIA	$0.921_{0.004}$	0.966	0.53	0.923	0.916	0.912	$\mathbf{0.975}_{0.002}$	$0.913_{0.001}$	$0.522_{0.009}$	$0.797_{0.004}$	$0.832_{0.003}$	$\underline{0.927}_{0.001}$
ELV	–	–	–	0.475	0.6	0.583	$\mathbf{0.645}_{0.133}$	$0.394_{0.003}$	$0.208_{0.016}$	$0.232_{0.011}$	$0.25_{0.009}$	$\underline{0.454}_{0.006}$
FR1	0.734	0.735	-1.206	0.442	0.695	0.707	$0.767_{0.001}$	$0.655_{0.001}$	$0.639_{0.003}$	$0.765_{0.001}$	$0.783_{0.001}$	$\underline{\mathbf{0.809}}_{0.002}$
FR2	0.727	0.735	-1.23	0.432	0.69	0.701	0.777	$0.646_{0.002}$	$0.625_{0.002}$	$0.755_{0.001}$	$0.774_{0.001}$	$\underline{\mathbf{0.8}}_{0.001}$
FR3	0.743	0.752	-1.03	0.386	0.649	0.659	$0.776_{0.001}$	$0.672_{0.002}$	$0.647_{0.002}$	$0.763_{0.002}$	$0.78_{0.002}$	$\underline{\mathbf{0.808}}_{0.002}$
NZEP	$-0.165_{0.24}$	–	–	0.399	0.484	0.463	$0.416_{0.014}$	$0.584_{0.003}$	$0.161_{0.01}$	$0.392_{0.009}$	$0.436_{0.006}$	$\underline{\mathbf{0.593}}_{0.005}$
HI	0.323	0.001	-0.788	-0.247	0.231	0.284	$\mathbf{0.347}_{0.001}$	$0.106_{0.002}$	$0.079_{0.033}$	$0.154_{0.006}$	$0.167_{0.002}$	$\underline{0.173}_{0.001}$
HOU	0.448	0.451	-1.076	0.22	0.419	0.422	$0.496_{0.003}$	$0.518_{0.003}$	$0.414_{0.01}$	$0.506_{0.005}$	$0.525_{0.006}$	$\underline{\mathbf{0.56}}_{0.002}$
HPR	0.928	**0.937**	–	0.658	0.825	0.835	0.933	0.606	$0.729_{0.001}$	0.835	0.848	$\underline{0.859}$
MT	$-0.208_{0.13}$	–	–	-0.435	-0.012	0.037	$0.058_{0.007}$	0.263	$-0.107_{0.005}$	$0.138_{0.001}$	$0.172_{0.001}$	$\underline{\mathbf{0.307}}$
MIA	$0.695_{0.335}$	0.541	-0.796	0.785	0.825	0.805	$\mathbf{0.867}_{0.005}$	$0.731_{0.001}$	$0.181_{0.018}$	$0.44_{0.009}$	$0.539_{0.007}$	$\underline{0.761}_{0.002}$
NPP	0.997	–	–	0.737	0.727	0.728	**0.999**	$-1.548_{0.062}$	$0.695_{0.082}$	$0.73_{0.003}$	0.734	$\underline{0.74}$
SCD	$0.547_{0.004}$	0.362	-1.161	**0.847**	0.819	0.789	$0.581_{0.004}$	$0.688_{0.002}$	$0.347_{0.008}$	$0.552_{0.003}$	$0.6_{0.001}$	$\underline{0.695}_{0.001}$
VT	$0.566_{0.009}$	–	–	**0.953**	0.904	0.839	$0.753_{0.034}$	$0.547_{0.003}$	$0.081_{0.006}$	$0.306_{0.004}$	$0.342_{0.004}$	$\underline{0.649}_{0.003}$
WE	0.996	0.88	–	0.499	0.673	0.678	**0.997**	$-4.052_{0.02}$	$-0.663_{0.134}$	$0.43_{0.011}$	$0.507_{0.004}$	$\underline{0.627}$
Avg. Rank	6.25	8.25	11.9	6.85	5.1	5.2	3.25	6.5	9.7	6.95	5.5	**2.8**

Table 3.4: Full RMSE Results on Normalized Datasets

Datasets	AMRules	FIMT-DD	ORTO	KNN-1	KNN-5	KNN-10	RAMRules	ARF-Reg	KNL-1	KNL-5	KNL-10	SOKNL
ABL	$2.176_{0.015}$	2.638	9.368	2.773	2.346	2.329	$\mathbf{2.14}_{0.003}$	$2.412_{0.005}$	$3.338_{0.022}$	$2.702_{0.014}$	$2.532_{0.012}$	$\underline{2.192}_{0.007}$
AIL	$0.175_{0.004}$	5.884	8.859	0.071	**0.06**	0.061	$0.168_{0.002}$	0.076	0.098	0.092	0.089	$\underline{0.074}$
Bike	173_{27}	569	443	122	127	128	113_{1}	92_{1}	124_{2}	101_{1}	94_{1}	$\underline{\mathbf{82}}_{1}$
CAH	62717_{332}	68845	142544	67817	59371	60738	58038_{138}	60888_{65}	115018_{1284}	82644_{311}	70841_{267}	$\underline{\mathbf{54556}}_{53}$
CPU	$7.828_{0.259}$	13.789	78.334	5.919	5.62	6.218	$6.441_{0.123}$	$8.447_{0.032}$	$10.776_{0.558}$	$8.13_{0.293}$	$6.925_{0.27}$	$\underline{\mathbf{4.741}}_{0.124}$
DIA	1118_{31}	737	2736	1109	1158	1183	$\mathbf{634}_{21}$	1176_{7}	2757_{27}	1795_{18}	1636_{15}	$\underline{1078}_{9}$
ELV	$0.014_{0.011}$	0.168	0.061	0.005	**0.004**	**0.004**	**0.004**	0.005	0.006	0.006	0.006	0.005
FR1	2.57	2.562	7.398	3.722	2.749	2.696	$2.402_{0.005}$	$2.927_{0.005}$	$2.991_{0.012}$	$2.414_{0.006}$	$2.32_{0.007}$	$\underline{\mathbf{2.179}}_{0.009}$
FR2	2.6	2.562	7.436	3.751	2.773	2.721	$2.352_{0.005}$	$2.964_{0.007}$	$3.05_{0.01}$	$2.464_{0.005}$	$2.369_{0.006}$	$\underline{\mathbf{2.226}}_{0.006}$
FR3	2.742	2.692	7.702	4.237	3.203	3.155	$2.558_{0.008}$	$3.097_{0.007}$	$3.211_{0.01}$	$2.631_{0.01}$	$2.536_{0.011}$	$\underline{\mathbf{2.368}}_{0.011}$
NZEP	117_{13}	240	243	84	78	80	83_{1}	70_{0}	99_{1}	85_{1}	82	$\underline{\mathbf{69}}$
HI	$15.398_{0.004}$	18.704	25.026	20.903	16.409	15.841	$\mathbf{15.119}_{0.017}$	$17.693_{0.015}$	$17.955_{0.322}$	$17.212_{0.066}$	$17.082_{0.023}$	$\underline{17.024}_{0.012}$
HOU	39272_{25}	39184	76158	46686	40301	40204	37533_{97}	36680_{102}	40452_{336}	37140_{205}	36443_{221}	$\mathbf{35078}_{98}$
HPR	2.005	**1.882**	15.752	4.374	3.134	3.041	$1.933_{0.002}$	$4.697_{0.004}$	$3.898_{0.008}$	$3.039_{0.006}$	$2.915_{0.006}$	$\underline{2.807}_{0.006}$
MT	2181_{115}	–	–	2380	1999	1950	1929_{7}	1706_{1}	2091_{4}	1845_{1}	1809_{1}	$\underline{\mathbf{1654}}_{1}$
MIA	160709_{73565}	214937	425392	147118	132954	140175	$\mathbf{115545}_{2256}$	164753_{320}	287298_{3117}	237433_{1823}	215519_{1710}	$\underline{155094}_{800}$
NPP	0.001	–	–	0.009	0.009	0.009	**0.0**	0.027	$0.009_{0.001}$	0.009	0.009	$\underline{0.009}$
SCD	$23.049_{0.104}$	27.371	50.367	**13.421**	14.573	15.749	$22.184_{0.098}$	$19.125_{0.049}$	$27.676_{0.167}$	$22.937_{0.064}$	$21.669_{0.038}$	$\underline{18.909}_{0.032}$
VT	$10.606_{0.108}$	–	–	**3.494**	4.98	6.472	$7.987_{0.521}$	$10.847_{0.036}$	$15.44_{0.049}$	$13.416_{0.043}$	$13.066_{0.039}$	$\underline{9.541}_{0.037}$
WE	7071_{200}	39100	496897	79979	64592	64060	$\mathbf{5940}_{35}$	253919_{508}	145589_{5796}	85289_{831}	79303_{305}	$\underline{69039}_{70}$
Avg. Rank	5.9	7.55	11.1	6.7	4.7	4.9	3.25	6.05	9.1	6.55	5.2	**2.65**

Table 3.5: Full Coefficient of Determination Results on Standardized Datasets

Datasets	AMRules	FIMT-DD	ORTO	KNN-1	KNN-5	KNN-10	RAMRules	ARF-Reg	KNL-1	KNL-5	KNL-10	SOKNL
ABL	0.062	0.2	-5.423	0.321	0.501	0.507	0.07	$0.462_{0.003}$	$-0.029_{0.038}$	$0.343_{0.013}$	$0.426_{0.01}$	$\underline{\mathbf{0.555}}_{0.003}$
AIL	0.202	–	-2.979	0.605	**0.721**	0.713	$0.206_{0.008}$	$0.566_{0.003}$	$0.243_{0.013}$	$0.329_{0.01}$	$0.361_{0.008}$	$\underline{0.571}_{0.002}$
Bike	0.314	–	–	0.535	0.436	0.431	$0.321_{0.005}$	$0.665_{0.007}$	$0.405_{0.011}$	$0.559_{0.008}$	$0.59_{0.006}$	$\underline{\mathbf{0.687}}_{0.004}$
CAH	0.344	0.56	0.106	0.641	0.721	0.71	$0.432_{0.008}$	0.722	$0.011_{0.017}$	$0.473_{0.003}$	$0.612_{0.002}$	$\underline{\mathbf{0.776}}$
CPU	0.375	0.745	–	0.908	0.911	0.893	$0.442_{0.013}$	$0.794_{0.002}$	$0.703_{0.032}$	$0.845_{0.011}$	$0.877_{0.005}$	$\underline{\mathbf{0.922}}_{0.003}$
DIA	0.215	**0.976**	0.637	0.924	0.914	0.909	$0.623_{0.019}$	0.903	$0.582_{0.006}$	$0.839_{0.001}$	$0.862_{0.001}$	$\underline{0.919}_{0.001}$
ELV	0.084	–	-1.729	0.559	**0.612**	0.581	$0.115_{0.002}$	$0.413_{0.005}$	$0.259_{0.013}$	$0.274_{0.012}$	$0.276_{0.009}$	$\underline{0.476}_{0.003}$
FR1	0.412	0.649	-2.167	0.441	0.695	0.707	$0.524_{0.003}$	$0.654_{0.001}$	$0.637_{0.003}$	$0.764_{0.002}$	$0.782_{0.002}$	$\underline{\mathbf{0.808}}_{0.002}$
FR2	0.452	0.623	-2.266	0.432	0.689	0.701	$0.53_{0.002}$	$0.644_{0.001}$	$0.621_{0.004}$	$0.753_{0.002}$	$0.772_{0.002}$	$\underline{\mathbf{0.799}}_{0.002}$
FR3	0.475	0.692	-2.216	0.385	0.648	0.659	$0.573_{0.003}$	$0.671_{0.002}$	$0.645_{0.004}$	$0.761_{0.003}$	$0.778_{0.003}$	$\underline{\mathbf{0.807}}_{0.003}$
NZEP	0.113	0.388	0.191	0.395	0.483	0.473	$0.232_{0.01}$	$\mathbf{0.569}_{0.003}$	$0.088_{0.01}$	$0.357_{0.005}$	$0.411_{0.004}$	$0.56_{0.004}$
HI	0.283	0.124	-1.85	-0.246	0.23	0.284	$\mathbf{0.293}_{0.002}$	0.11	$0.063_{0.029}$	$0.153_{0.003}$	$0.161_{0.001}$	$\underline{0.164}_{0.001}$
HOU	0.141	0.462	-0.126	0.229	0.429	0.431	$0.212_{0.005}$	$0.533_{0.003}$	$0.464_{0.006}$	$0.552_{0.003}$	$0.564_{0.002}$	$\underline{\mathbf{0.574}}_{0.002}$
HPR	0.368	0.681	-1.477	0.658	0.824	0.835	$0.546_{0.001}$	0.604	$0.726_{0.001}$	0.833	0.847	$\underline{\mathbf{0.858}}$
MT	0.03	–	–	-0.09	0.037	0.042	0.037	0.275	$-0.286_{0.007}$	$0.117_{0.002}$	$0.152_{0.001}$	$\underline{\mathbf{0.307}}$
MIA	0.178	0.566	0.199	0.777	**0.822**	0.805	$0.223_{0.005}$	$0.735_{0.001}$	$0.193_{0.016}$	$0.453_{0.006}$	$0.553_{0.004}$	$\underline{0.761}_{0.001}$
NPP	0.433	–	–	**0.737**	0.723	0.726	$0.331_{0.005}$	$-1.751_{0.065}$	$0.383_{0.125}$	$0.434_{0.082}$	$0.508_{0.047}$	$\underline{0.678}_{0.011}$
SCD	0.359	0.608	0.365	**0.847**	0.825	0.798	$0.472_{0.004}$	0.683	$0.33_{0.011}$	$0.546_{0.003}$	$0.592_{0.002}$	$\underline{0.686}_{0.001}$
VT	0.244	–	–	**0.937**	0.892	0.826	$0.311_{0.003}$	–	–	–	–	–
WE	0.236	0.507	–	0.5	0.676	**0.681**	$0.307_{0.003}$	$-3.951_{0.02}$	$-0.764_{0.107}$	$0.411_{0.008}$	$0.491_{0.003}$	$\underline{0.616}_{0.001}$
Avg. Rank	9.6	7.5	11.1	5.7	3.65	3.6	8.4	5.5	9.05	5.8	4.65	**2.45**

Table 3.6: Full RMSE Results on Standardized Datasets

Datasets	AMRules	FIMT-DD	ORTO	KNN-1	KNN-5	KNN-10	RAMRules	ARF-Reg	KNL-1	KNL-5	KNL-10	SOKNL
ABL	3.225	2.983	8.453	2.748	2.356	2.341	$3.21_{0.001}$	$2.447_{0.006}$	$3.383_{0.063}$	$2.703_{0.027}$	$2.527_{0.021}$	$\underline{\mathbf{2.225}}_{0.007}$
AIL	0.101	0.823	0.226	0.071	**0.06**	0.061	0.101	0.075	0.099	0.093	0.091	$\underline{0.074}$
Bike	150	1382	1580	124	136	137	149	105_{1}	140_{1}	121_{1}	116_{1}	$\underline{\mathbf{101}}_{1}$
CAH	93510	76552	109169	69221	60956	62175	86993_{580}	60848_{39}	114842_{972}	83851_{275}	71924_{228}	$\underline{\mathbf{54629}}_{97}$
CPU	14.572	9.307	72.644	5.603	5.487	6.036	$13.77_{0.167}$	$8.359_{0.039}$	$10.027_{0.536}$	$7.253_{0.262}$	$6.477_{0.143}$	$\underline{\mathbf{5.155}}_{0.085}$
DIA	3535	**615**	2402	1102	1170	1200	2448_{63}	1244_{3}	2578_{19}	1599_{6}	1481_{8}	$\underline{1134}_{8}$
ELV	0.006	0.132	0.011	**0.004**	**0.004**	**0.004**	0.006	0.005	0.006	0.006	0.006	0.005
FR1	3.818	2.949	8.864	3.724	2.751	2.698	$3.437_{0.009}$	$2.929_{0.006}$	$3.003_{0.012}$	$2.42_{0.01}$	$2.323_{0.009}$	$\underline{\mathbf{2.181}}_{0.01}$
FR2	3.684	3.059	8.998	3.754	2.775	2.723	$3.414_{0.008}$	$2.972_{0.005}$	$3.065_{0.016}$	$2.475_{0.011}$	$2.38_{0.011}$	$\underline{\mathbf{2.235}}_{0.01}$
FR3	3.916	2.999	9.694	4.237	3.205	3.157	$3.533_{0.011}$	$3.101_{0.009}$	$3.22_{0.019}$	$2.641_{0.018}$	$2.545_{0.018}$	$\underline{\mathbf{2.375}}_{0.018}$
NZEP	102	85	98	85	78	79	95_{1}	**71**	104_{1}	87	83	72
HI	15.851	17.521	31.598	20.893	16.419	15.84	$\mathbf{15.732}_{0.025}$	$17.652_{0.009}$	$18.113_{0.276}$	$17.227_{0.035}$	$17.141_{0.015}$	$\underline{17.109}_{0.011}$
HOU	48975	38781	56095	46418	39928	39887	46903_{147}	36113_{127}	38684_{207}	35392_{115}	34916_{84}	$\underline{\mathbf{34496}}_{80}$
HPR	5.95	4.227	11.777	4.374	3.135	3.041	$5.045_{0.006}$	$4.707_{0.006}$	$3.914_{0.01}$	$3.055_{0.008}$	$2.93_{0.008}$	$\underline{\mathbf{2.824}}_{0.008}$
MT	1958	–	–	2075	1950	1944	1950_{1}	1692_{1}	2253_{6}	1868_{2}	1830_{1}	$\underline{\mathbf{1654}}_{1}$
MIA	287749	209145	283992	149782	**133806**	140069	279740_{874}	163258_{337}	285149_{2784}	234710_{1294}	212222_{935}	$\underline{155269}_{400}$
NPP	0.013	–	–	**0.009**	**0.009**	**0.009**	0.014	0.028	$0.013_{0.001}$	0.013	0.012	$\underline{0.01}$
SCD	27.429	21.463	27.294	**13.395**	14.344	15.404	$24.896_{0.098}$	$19.297_{0.029}$	$28.047_{0.235}$	$23.094_{0.084}$	$21.876_{0.063}$	$\underline{19.207}_{0.042}$
VT	14.009	–	–	**4.043**	5.289	6.715	$13.366_{0.034}$	–	–	–	–	–
WE	98802	79304	–	79893	64346	**63826**	94130_{211}	251362_{519}	149972_{4516}	86691_{555}	80613_{275}	$\underline{69978}_{119}$
Avg. Rank	9.45	7.25	10.85	5.6	3.65	3.6	8.1	5.3	8.9	5.7	4.55	**2.35**

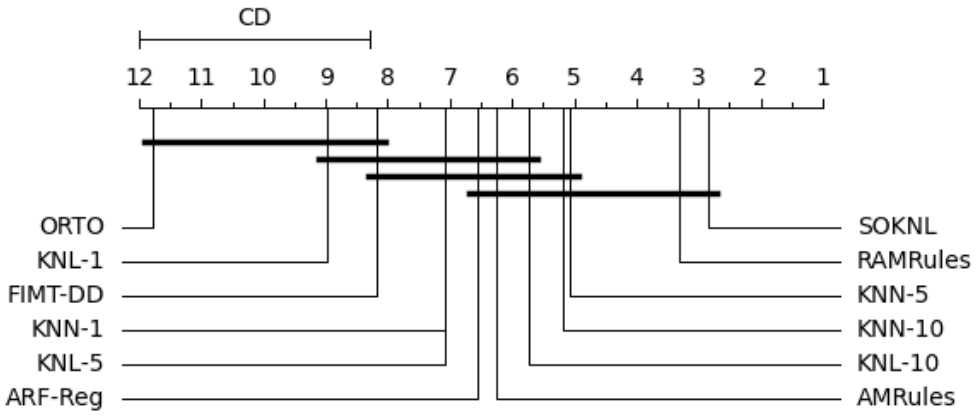


Figure 3.3: The Critical Difference Diagram across Twelve Methods with RMSE Results on Normalized Data

are ranked the lowest in the comparison.

Figure 6.1 compares the average ranks of the twelve evaluated algorithms, with lower ranks indicating better performance. SOKNL achieves the best overall performance, ranking first and showing no statistically significant difference from RAMRules, which is the second-best. These two algorithms form the top-performing group. KNN-5, KNN-10, KNL-10, AMRules, and ARF-Reg occupy the intermediate range, with overlapping significance bars indicating no statistically significant differences within this group. ORTO, KNL-1, and FIMT-DD obtain the highest (worst) ranks and are significantly outperformed by the top group. Overall, the diagram highlights SOKNL as the leading algorithm, with a clear separation between the best- and worst-performing methods.

3.4.5 $\mathcal{K}$ Value Analysis for SOKNL

In this section, we provide illustrations and a discussion of the $\mathcal{K}$ values over time. Figure 3.4 presents an overview of the selected $\mathcal{K}$ values over time for all datasets. Note that the $\mathcal{K}$ values shown are averaged across ten runs on normalized datasets, each using different random seeds.

The $\mathcal{K}$ values for different datasets fluctuate within varying ranges. Most of

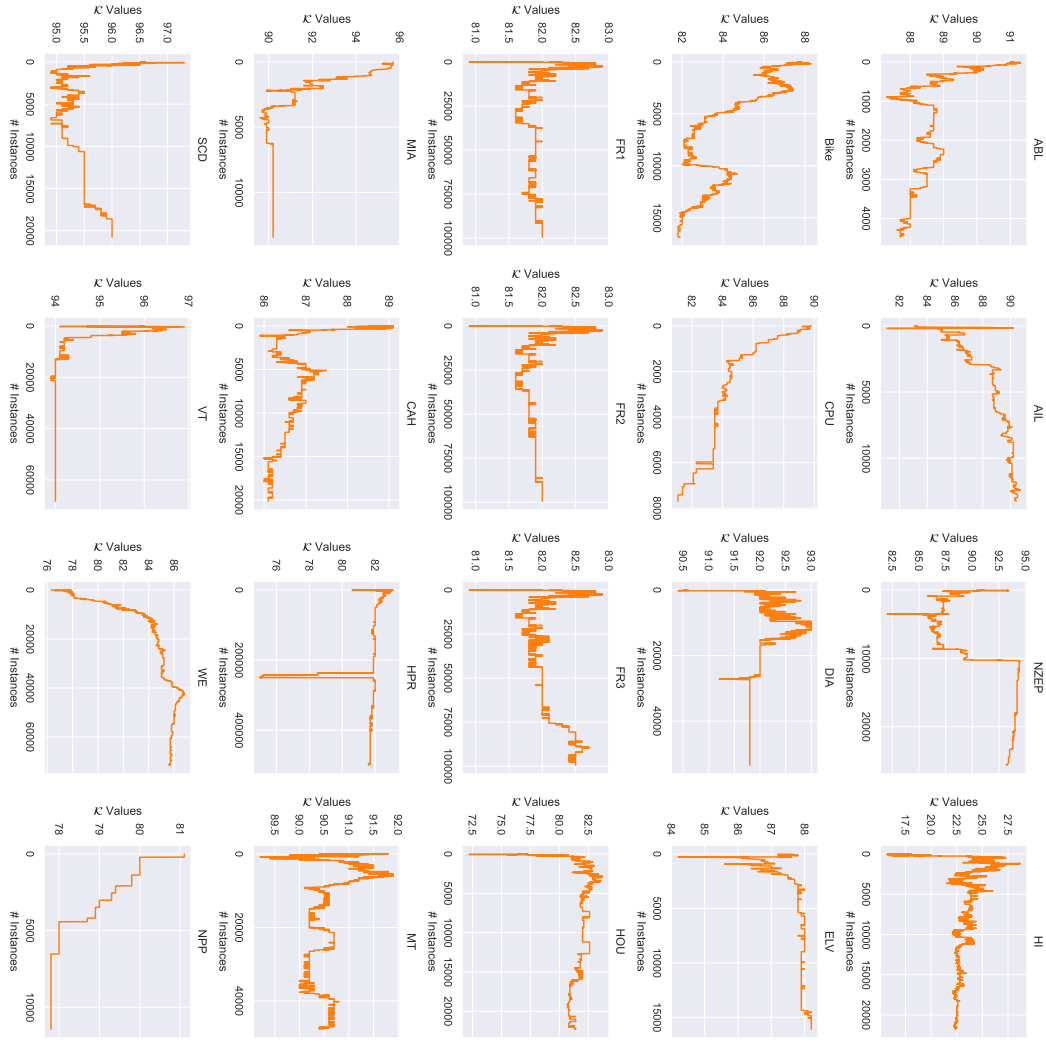


Figure 3.4: Selected $\mathcal{K}$ Values for 20 Datasets Over Time

the $\mathcal{K}$ values converge to high values, typically between 70 and 90. However, smaller converged values, such as 22 for the Health Insurance (HI) dataset, are also observed. The variation in $\mathcal{K}$ values indicates that the $\mathcal{K}$ selection mechanism adapts to different concepts based on specific contexts.

3.4.6 Runtime Results

This section provides an analysis of the runtime overhead introduced by SOKNL compared to ARF-Reg. Although a variety of algorithms were included in our experiments, comparing the runtime of single learners with ensemble learners is both unnecessary and uninformative. Therefore, we focus solely on the comparison between ARF-Reg and SOKNL. Additionally, different preprocessing

techniques introduce only minor variations in runtime, which do not significantly impact the overall results. For clarity, we present the runtime results on the regular datasets in Figure 3.5:

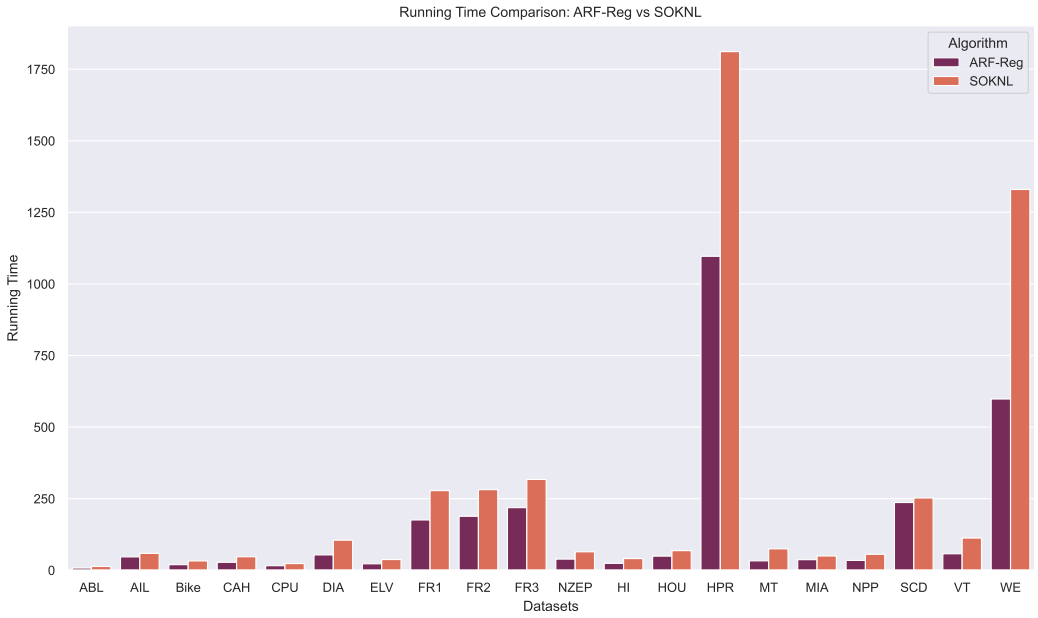


Figure 3.5: Runtime Comparison between ARF-Reg and SOKNL

The added overhead varies between 6.8% (for the Superconductivity dataset) and 128% (for the MetroTraffic dataset), with an average of 67%. This increase in computational resources is considered acceptable, given the significant performance improvements that SOKNL offers over ARF-Reg.

Chapter 4

Adaptive Prediction Interval (AdaPI)

In the previous chapter, we address the regression SL tasks by proposing SOKNL. Nonetheless, there is a pain-point left. Unlike the classification predictions that can be marked as correct or incorrect, regressive single-valued predictions are almost impossible to match the ground truths. Consequently, the analysis on regression tasks is typically based on "error based" metrics, which compromises the capability of the regressive models for uncertainty measurement[1]. Prediction Interval (PI) is a powerful technique for quantifying the uncertainty of regression tasks. However, research on PI for data streams has received limited attention. Traditional PI-generating approaches are not directly applicable due to the dynamic and evolving nature of data streams. This chapter presents Adaptive Prediction Interval (AdaPI), a novel framework that can automatically adjust the interval width by an appropriate amount according to historical information to achieve a user-defined coverage percentage. AdaPI can be applied as a postprocessing step to any streaming PI technique. This chapter proposes an incremental variant of the pervasive Mean and Variance Estimation (MVE) method for use with AdaPI.

[1] For instance, in classification the accuracy measurements can be an indicator for uncertainty.

4.1 Introduction

The machine learning literature has thoroughly investigated learning algorithms for regression tasks [50]. However, a single-valued prediction is insufficient for many applications [65]. A PI [62] provides more informativeness and uncertainty measurements [110] to a regression model since it generates intervals that encompass the expected range of true values with a desired confidence level.

Several techniques for establishing prediction intervals were introduced based on different mathematical theories since the seminal technique in [62]. Zhao et al. summarized the most commonly used ones, such as bootstrapping techniques and the delta method [136].

There are few PI methods available for data streams (or time series), which often rely on windowed versions of existing techniques [61]. Such approaches do not agree with current state of the art methods for data stream regression [117], which are fully incremental (as discussed in Section 2). The lack of a fully incremental PI method motivated this work.

The first stage of this work implemented Mean and Variance Estimation (MVE) prediction interval method in a streaming fashion and proposed the initial version of Adaptive Prediction Interval (AdaPI) methodology [119]. AdaPI is meant to be a post-calibration tool that can be applied to any streaming PI method. AdaPI incrementally adjusts the generated interval widths according to the current coverage, ensuring that the coverage converges towards the desired confidence level. AdaPI also adapts automatically to concept drifts.

The second stage of this research contributes to this area in two ways:

- extending the AdaPI algorithm into a family that can be customized by demands; and
- presenting a novel evaluation method for prediction interval to handle the dual-criteria evaluation problem (see in Section 4.3.2.2).

4.2 Problem Setting

Consider a linear regression problem:

$$y = \sum_{k=1}^{K} \beta_k x_k + \epsilon \tag{4.1}$$

where K is the number of independent variables, x_k the input independent variables, y is the predicted dependent variable, β_k the coefficients to be determined, ϵ the noise term. If we collect N samples, the problem can be stated in a compact matrix form such as:

$$\mathcal{Y} = \mathcal{X}\beta + \mathcal{E} \tag{4.2}$$

with $\mathcal{Y}$, $\mathcal{E}$, $\beta \in \mathbb{R}^N$, and $\mathcal{X} \in \mathbb{R}^{N \times K}$.

This research considers the PI problem as an extension of the regression problem. Specifically, instead of using $\mathcal{Y}$ as the prediction, an interval $[\mathcal{P}_l, \mathcal{P}_u]$ is required to expectingly contain the true value with an α probability (confidence level, denoted as $\mathcal{L}$ for the rest of this article), where $\mathcal{P}_l$ and $\mathcal{P}_u$ are the lower and upper prediction interval bounds, i.e.: $Pr(\mathcal{P}_l \leq \mathcal{Y} \leq \mathcal{P}_u) = \alpha$.

Note that there is no guarantee for coverage to achieve the expected confidence level α due to the strong statistical assumption underlying most PI methods, which are often violated in practice [120]. This issue is more challenging in streaming scenario due to the evolving nature and the concept drift phenomena [13].

Our goal is to propose a prediction interval method that is capable of providing as narrow as possible intervals while the coverage stays at the desired confidence level.

4.3 Background

In this section, we introduce relevant information to this research.

4.3.1 Mean and Variance Estimation (MVE)

Mean and Variance Estimation (MVE) [100] is one of the most straightforward and widely applied approaches for PI. MVE assumes the predictive errors follow a Gaussian distribution (Normal Assumption) which results in a generalized linear model, and therefore leads to a prediction interval illustrated as in Equation 4.3:

$$\text{PI} \in \left(\mathcal{Y} - G^{-1}(0,\gamma) \times \sigma_\epsilon, \mathcal{Y} + G^{-1}(0,\gamma) \times \sigma_\epsilon\right) \tag{4.3}$$

where $G^{-1}(0,\gamma)$ is the inverse Gaussian distribution function with 0 mean and γ probability, and $\mathcal{Y}$ is the prediction output. According to the normal assumption, $\mathcal{E} \sim \mathcal{N}(0,\sigma_\epsilon)$. Hence, when $\gamma = 95\%$, $G^{-1}(0,0.95) \approx 1.96$.

4.3.2 Multi-objective Evaluation

As detailed in 2.2.2.1, two metrics are involved in evaluating PI tasks. Thus, a multi-target evaluation technique is required. The literature has proposed several optimization and evaluation methodologies for multi-objective (see in [55]), especially in Evolutionary Computing (EC) [33].

This section introduces a popular algorithm in Multi-Objective Optimization (MOO) – Non-dominated Sorting Genetic Algorithm II (NSGA II) – and proposes a modified version of NSGA II as an evaluation technique for prediction interval tasks.

4.3.2.1 Non-dominated Sorting Genetic Algorithm II

First introduced by Deb et al. in [26], NSGA II is widely used in MOO tasks to obtain Pareto-optimal solutions [35].

NSGA II consists of two parts – 1. non-dominated sorting; and 2. crowding distance. For the non-dominated sorting part, we first define the Pareto Dominance. In the objective space, where f denotes the objective dimensions, k is the number of the dimensions, and $\mathcal{O}$ denotes the observation. Then, we

say an observation $\mathcal{O}^d$ dominates another observation $\mathcal{O}^t$, when:

$$\forall i \in \{1, ..., k\} : \mathcal{O}^d_{f_i} \leq \mathcal{O}^t_{f_i} \tag{4.4}$$

and

$$\exists j \in \{1, ..., k\} : \mathcal{O}^d_{f_j} < \mathcal{O}^t_{f_j} \tag{4.5}$$

Note that this is in the case of minimization where smaller value are preferable. For maximization, the criteria are reversed.

All observations are separated into two groups[2] – dominated and non-dominated. By definition, the observations within the dominated group are "clearly" inferior to those in the non-dominated group. If there is more than one member in the non-dominated group, a crowding distance ranking is executed in NSGA II.

The definition of crowding distance is as follow: an observation $\mathcal{O}$'s contribution c with respect to the i_{th} objective, i.e. c_{f_i}, can be expressed as in Equation 4.6:

$$c(\mathcal{O}_{f_i}) = \mathcal{O}^R_{f_i} - \mathcal{O}^L_{f_i} \tag{4.6}$$

where $\mathcal{O}^R$ and $\mathcal{O}^L$ are the observations right or left next to the observation $\mathcal{O}$. The crowding distance[3] for an observation $\mathcal{O}$ is explained by Equation 4.7:

$$C(\mathcal{O}) = \sum_{i=1}^{k} c(\mathcal{O}_{f_i}) \tag{4.7}$$

To express it in words, the crowding distance of a observation is the sum of contributions of this observation in regards to all objectives. In a two dimensional scenario, it also corresponds to the Manhattan distance between the two neighbours of the observation.

[2] In MOO problems, there can be multiple layers of observations, that is, the dominated group can also be divided into different levels of dominance. However, for our double-objective evaluation task, we consider only the simple scenario as described in this context.

[3] The term "Distance" intuitively involves two objects. However, in the original work [26], the "crowding distance" was defined as a volume around the target observation. We keep the original term in this work.

4.3.2.2 Coverage Interval width in Non-dominated Group Evaluation

NSGA II provides remarkable insights on multi-objective evaluation. Nonetheless, the second stage of NSGA II algorithm identifies sparser regions in the objective space as "optimal" to ensure a diverse and well-distributed set of Pareto-optimal solutions – an intrinsic characteristic of MOO problems. However, the diversity maintenance is not essential for evaluating PI tasks.

Therefore, in this section, we propose an evaluation approach for prediction interval problem – Coverage Interval width in Non-dominated Group (CING). CING works as follows:

1. Apply a linear Max-Min normalization (Equation 3.6) step to the Coverage Error and NMPIW across all observations;
2. Determine the non-dominated group with the same technique as in NSGA II; and
3. Calculate scores using a weighted-sum strategy within the non-dominated group.

The Coverage Error (CE) is the absolute value of the difference from the coverage and the confidence level, as expressed in Equation 4.8:

$$CE = Abs(\mathcal{C} - \mathcal{L}) \tag{4.8}$$

where $\mathcal{C}$ denotes the coverage, and $\mathcal{L}$ represents the confidence level.

A weighted-sum strategy is employed here because of its flexibility and universality. Euclidean distance from the observations and the optimal point[4] has been considered. However, using classic Euclidean distance implies a equal significance of both metrics (coverage and interval width). As explained in the Section 2.2.2.1, our goal is to make the coverage converge to the confidence

[4]In our minimization case, the optimal point is (0,0), reaching which means that the coverage has converged to the desired confidence level, and the prediction interval is narrowed down to a point prediction.

level while narrowing the interval width down as possible. This implicitly indicates that the coverage related metric (CE, in our case) should possess a higher importance. A weighted-sum strategy allows us to manipulate the importance of different metrics. Moreover, utilizing a equal weight (0.5-0.5) is essentially identical to using the Euclidean distance.

As the coverage is supposed to converge to the confidence level, one should minimise the CE when designing a prediction interval model. Intuitively, the NMPIW also needs to be shrunk. As a consequence, in our evaluation procedure, the left bottom corner is the optimal point. After the third step of CING, each PI approach obtains a score, then the top one (in an ascending order) is the best approach under the current weights. Different weights can be assigned in different scenario, based on the purposes and emphases of the use cases.

4.4 Adaptive Prediction Interval (AdaPI)

MVE assumes that errors are normally distributed (Normal Assumption), which is frequently violated in the real applications. Intuitively, adjusting the interval width when the normal assumption is compromised over time for streaming data is a practical and efficient method. A coefficient applied to the generated intervals, which is tightly linked to the current coverage, could be advantageous.

As mentioned in Section 4.3.1, in practical scenarios, it is rarely accurate to assume that predictive errors conform to a Gaussian distribution. Consequently, prediction intervals relying on statistical Gaussian measures might fail to adequately encompass actual observations. This mismatch leads to the coverage deviating from the confidence level established in the MVE approach. When regression models demonstrate exceptional (or poor) accuracy in forecasting specific data streams, MVE tends to produce overly narrow (or wide) intervals, jeopardizing the informativeness and the uncertainty measurement

of the PI methods. Thus, we introduce a factor $\mathcal{F}$[5] to the MVE approach, as formulated in Equation 4.9:

$$\text{PI} = \left(\mathcal{Y} - \mathcal{F} \times G^{-1}(0, \gamma) \times \sigma_\epsilon, \mathcal{Y} + \mathcal{F} \times G^{-1}(0, \gamma) \times \sigma_\epsilon\right) \tag{4.9}$$

where the $\mathcal{F}$ is determined according to the real-time coverage of the model.

For the adaptive approaches to work properly, the factor $\mathcal{F}$ must exhibits several key characteristics:

1. To ensure that the adaptive mechanism does not disrupt the interval generation when the model performs as desired, $\mathcal{F}$ should equal one when the coverage equals to the confidence level, i.e. $\mathcal{F} = 1$ if $\mathcal{C} = \mathcal{L}$;

2. To facilitate the coverage converging to the confidence level, the interval should be expanded when coverage is insufficient, thus allowing the modified interval to include more ground truth values. Conversely, when the coverage is excessively high, the interval should be contracted to maintain an acceptable uncertainty level. Ergo, $\mathcal{F} > 1$ if $\mathcal{C} < \mathcal{L}$, and $\mathcal{F} < 1$ if $\mathcal{L} < \mathcal{C} < 100\%$;

3. A too narrow prediction interval is also not informative as it might frequently miss the ground truth. Therefore, a reasonably conservative lower limit for $\mathcal{F}$ can be beneficial, i.e., $\mathcal{F} = \text{limit}$ if $\mathcal{C} = 100\%$. The limit will be denoted as $\mathcal{M}$ in the rest of t his work.

Based on these characteristics, we outline the following generating curves for the adaptive approaches. Note that coverage inherently ranges of [0, 100] as it is a percentage.

4.4.1 Logarithm Approach

The first approach defines $\mathcal{F}_{Log}$ as in Equation 4.10:

[5] In our published work [119], it was referred to as a factors $\mathcal{F}$, but is now termed "factor" for broader applicability.

$$\mathcal{F}_{Log} = \begin{cases} 100 - \mathcal{C} & \text{if } \mathcal{C} < 2\mathcal{L} - 100 \\ \log_{\mathcal{L}} \frac{100-\mathcal{C}}{100-\mathcal{L}} + 1 & \text{if } \mathcal{C} \geq \mathcal{L} \\ (\mathcal{L} - 100) \log_{\mathcal{L}} \frac{100+\mathcal{C}-2\mathcal{L}}{100-\mathcal{L}} + 1 & \text{otherwise} \end{cases} \tag{4.10}$$

where $\mathcal{C}$ is the current coverage, and $\mathcal{L}$ represents the confidence level. This curve varies with different confidence levels $\mathcal{L}$. Figure 4.1 illustrates the curve graph of the factor with a fixed confidence level at 95%, i.e. $\mathcal{L} = 95$.

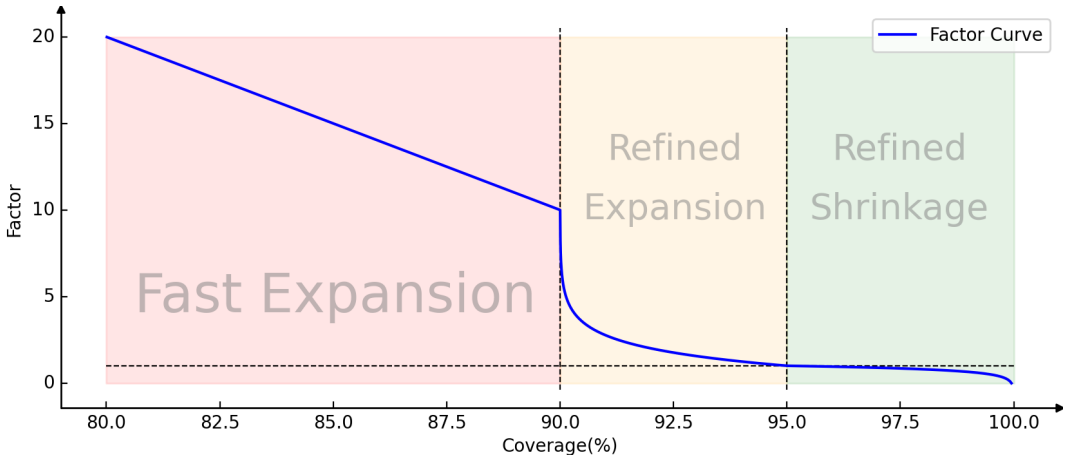


Figure 4.1: Logarithm Factor with a 95% Confidence Level ($\mathcal{C} \in [80, 100]$) and a 0.01 lower limit. Dotted lines denote significant values in the figure. The horizontal line is 1.0, the right vertical line is the desired confidence level (95%). The refined expansion and the refined shrinkage areas have the same width.

Three segments are illustrated in Figure 4.1. The red segment, termed "Fast Expansion," represents a phase where factors are determined linearly based on the current coverage. This phase aims to rapidly enhance coverage towards the confidence level. The orange segment, referred to as "Refined Expansion," also expands the PI by generating factors greater than one. However, these factors are produced more subtly via a logarithmic function. As the coverage approaches the confidence level, the rate of change in the factors diminishes. Conversely, the green segment, labeled "Refined Shrinkage," seeks to reduce the PI when the coverage surpasses the confidence level. Similar to the orange segment, the factors in this phase are updated moderately as the

coverage nears the confidence level, preventing excessive adjustments.

The logarithmic approach offers several advantages: 1. Radical adjustments occur for substantial coverage-confidence disparities; 2. Modifications are moderate when coverage approaches the confidence level, preventing overreactions; and 3. Auto cessation at the desired confidence level is achieved without adding extra hyperparameters.

4.4.2 Huber Loss Inspired (HLI) Approach

As shown in Figure 4.1, there is an inflection point in the logarithmic approach. To mitigate it, we leverage the Huber Loss [74] function for factor generation. Originally, the Huber Loss was employed as a loss function that is more robust to outliers than a regular quadratic loss function, and it is defined as in Equation 4.11:

$$L_\delta = \begin{cases} \frac{1}{2}e^2 & \text{if } |e| < \delta \\ \delta\left(e - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases} \tag{4.11}$$

where e is the predictive error of the model, and the δ is a pre-defined parameter that determines the width of the quadratic portion. The Huber Loss facilitates a smooth transition between the linear and quadratic sections.

In our adaptation, the curve needs to fulfill the characteristics mentioned previously in Section 4.3. Therefore, we modify the Huber Loss function to satisfy our requirements. Specifically, the symmetric line of Huber Loss function is aligned with the confidence level. The right linear section is discarded. The δ value is set such that the right end of the quadratic portion corresponds to 100% coverage. Then, the right half of the quadratic portion is inverted and scaled to fit the lower limit. After this transformation, our Huber Loss Inspired curve is defined as in Equation 4.12. From this point onwards, this

adaptation will be referred to as the Huber Loss Inspired (HLI) approach:

$$\mathcal{F}_{HLI} = \begin{cases} \frac{(\mathcal{M}-1)(\mathcal{C}-\mathcal{L})^2}{(100-\mathcal{L})^2} + 1 & \text{if } \mathcal{C} \geq \mathcal{L} \\ 2(\frac{\mathcal{C}-2\mathcal{L}+100}{\mathcal{L}-100} + 1) & \text{if } \mathcal{C} < 2\mathcal{L} - 100 \\ (\frac{\mathcal{C}-\mathcal{L}}{100-\mathcal{L}})^2 + 1 & \text{otherwise} \end{cases} \tag{4.12}$$

where $\mathcal{F}_H$ is factor from HLI, $\mathcal{M}$ is the lower limit, and the other notations are the same as in Equation 4.10.

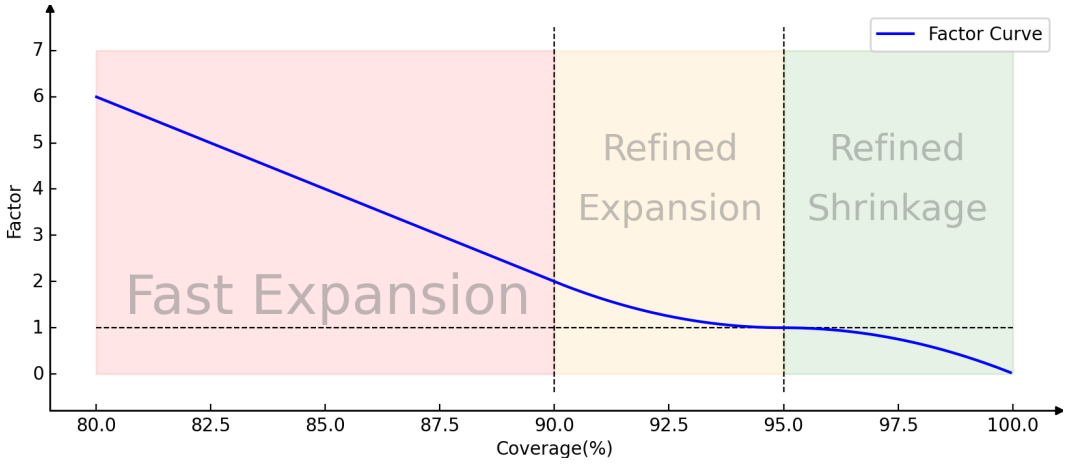


Figure 4.2: HLI Factor with a 95% Confidence Level ($\mathcal{C} \in [80, 100]$) and a 0.01 lower limit. Dotted lines denote significant values in the figure. The horizontal line is 1.0, the right vertical line is the desired confidence level (95%). The refined expansion and the refined shrinkage areas have the same width.

Similar to Figure 4.1, Figure 4.2 is also divided into three sections: fast expansion, refined expansion, and refined shrinkage. Unlike the logarithmic function, the HLI factor provides a smoother transition among the different sections.

4.4.3 Linear Lever

Lastly, we present the simplest yet powerful method for generating the factor – a linear model. However, to maintain an adaptive approach to the changing properties of the PI tasks, the linear models must still adhere to the requirements outlined in 4.4. Two specific points can be positioned under these

constraints: the factor is one when coverage converges to confidence level, $f(\mathcal{L}) = 1$, and factor equals to the lower limit when coverage reaches 100%, i.e. $f(100) = limit$. Thus, the linear model can be generally represented as shown in Equation 4.13:

$$\mathcal{F}_L = \frac{1 - \mathcal{M}}{\mathcal{L} - 100}\,\mathcal{C} + \frac{\mathcal{M}\mathcal{L} - 100}{\mathcal{L} - 100} \tag{4.13}$$

where $\mathcal{F}_L$ is the Linear Lever Factor, and the other notations are the same with Equation 4.10 and 4.12.

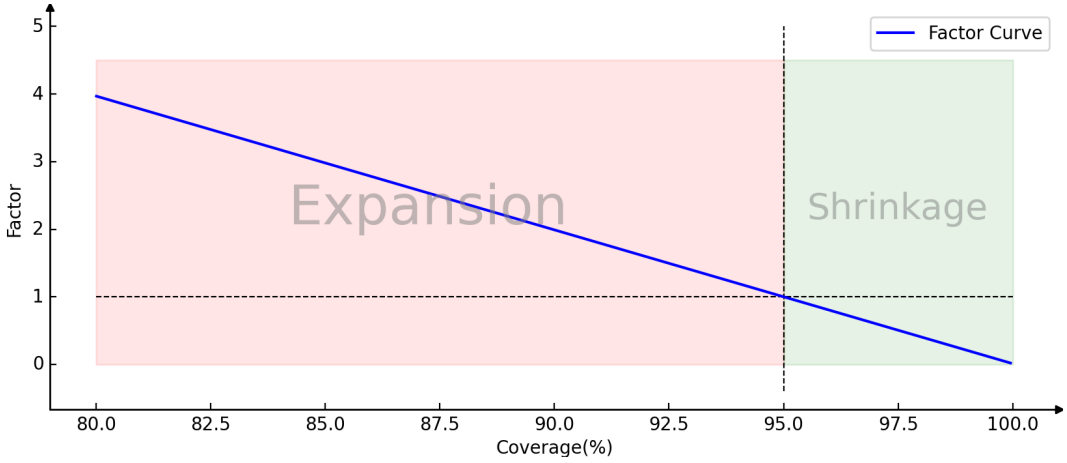


Figure 4.3: Linear Lever Factor with a 95% Confidence Level ($\mathcal{C} \in [80, 100]$) and a 0.01 lower limit. Dotted line denotes significant values in the figure. The horizontal line is 1.0, the vertical line is the desired confidence level (95%).

As shown in Figure 4.3, for the Linear approach, there are only two sections. Using the confidence level as the dividing line, the values generated on the left will be used to expand the prediction interval, while the values on the right will be associated with shrinkage.

4.5 Experiments

We provide the information concerning our experiments in this section.

4.5.1 Regression Models

The MVE and the adaptive approaches proposed in this work require regression models as the base learner to generate single-value predictions and the residuals for calculating the intervals. We include three regression algorithms in the experiments: K Nearest Neighbours (kNN), Adaptive Random Forest for Regression (ARF-Reg) [46], and Self-Optimising K Nearest Leaves (SOKNL) [117]. For a more detailed description of these regression models, please see Chapter 2.

4.5.2 Experimental Setting

Hyperparameter settings are usually significant for experiments. In this section, we introduce the detailed settings used in this work. At the base learner level, the parameters are set as defaults in MOA.

Furthermore, at the prediction interval level, we execute several sets of experiments with the following parametrisation:

- $\mathcal{C} \in \{80\%, 90\%, 95\%, 99\%\}$; and
- $\mathcal{M} \in \{0.05, 0.1, 0.5\}$.[6]

All of our experiments include a warm-up phase of 100 examples to allow for a reasonably robust MVE estimation before applying any adaptation to the interval width by AdaPI.

4.6 Results and Discussion

Experimental results and relevant discussion are presented in this section.

4.6.1 Raw Results

Table 4.1 and 4.2 present parts of our experimental results. To avoid excessively lengthy outcome displays and to mitigate redundancy in space utiliza-

[6] Limit $\mathcal{M}$ (see Section 4.4) is not applicable to MVE.

Table 4.1: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 95\%$ and Limit $\mathcal{M} = 0.1$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	94.88	35.12	95.3	30.92	95.26	34.36
HuberLoss	94.72	36.11	95.36	32.34	**94.98**	35.21
Linear	94.82	35.17	95.1	**30.69**	95.2	34.3
MVE	94.68	34.98	95.62	31.9	94.96	33.81
Abalone						
Logarithm	97.81	33.85	97.6	23.75	98.0	24.49
HuberLoss	97.25	33.15	97.23	23.17	97.51	22.78
Linear	**96.14**	31.17	96.36	20.48	96.67	**19.19**
MVE	99.71	40.54	98.44	27.0	98.68	30.34
CPU Activity						
Logarithm	96.53	11.42	99.99	1.2	99.92	1.31
HuberLoss	96.37	10.91	99.99	**0.68**	99.85	0.75
Linear	**95.44**	10.46	99.99	**0.68**	99.72	0.74
MVE	97.1	13.24	99.99	6.78	99.99	6.82
Naval Propulsion Plant						
Logarithm	95.2	30.34	95.4	28.97	94.92	23.4
HuberLoss	95.05	30.43	95.37	29.18	94.71	23.3
Linear	**95.03**	30.0	95.11	28.31	94.88	23.34
MVE	94.8	29.62	95.16	28.71	94.41	**22.8**
Ailerons						
Logarithm	97.28	29.58	97.37	27.42	97.39	24.52
HuberLoss	96.95	28.51	97.02	26.83	97.06	24.06
Linear	97.08	26.96	97.19	24.93	97.1	**22.53**
MVE	97.03	29.52	**96.91**	27.59	97.17	24.99
Miami Housing						
Logarithm	96.86	4.54	97.27	3.88	96.8	4.01
HuberLoss	96.89	4.53	97.22	3.83	96.85	4.03
Linear	96.57	4.19	96.72	**3.46**	**96.49**	3.73
MVE	96.95	4.72	97.45	4.2	96.99	4.2
NZ Energy Price						
Logarithm	97.7	5.64	97.85	5.61	97.4	6.57
HuberLoss	97.42	5.42	97.51	5.37	97.14	6.38
Linear	96.62	4.83	96.64	**4.76**	**96.31**	5.79
MVE	98.88	8.3	98.17	6.34	97.66	7.17
Elevators						
Logarithm	93.85	39.56	**93.97**	33.39	93.76	53.74
HuberLoss	92.58	35.61	92.69	30.48	92.32	49.37
Linear	93.3	37.46	93.47	32.01	93.05	51.45
MVE	89.53	28.88	89.56	**25.86**	88.16	39.1
Bike						
Logarithm	94.55	51.59	**94.62**	44.56	94.53	49.55
HuberLoss	93.71	49.29	93.89	42.42	93.65	47.28
Linear	94.33	51.38	94.48	44.15	94.34	49.34
MVE	92.65	45.52	93.2	**39.92**	92.75	44.23
California Housing						
Logarithm	96.31	46.29	96.34	44.5	96.55	41.09
HuberLoss	96.34	45.37	96.35	44.5	96.32	39.84
Linear	95.91	42.83	**95.9**	41.85	96.16	**37.82**
MVE	96.55	46.72	96.54	45.82	96.12	39.69
Superconductivity						

Table 4.2: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 95\%$ and Limit $\mathcal{M} = 0.1$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	97.15	68.97	96.5	70.56	95.43	67.76
HLI	96.86	68.38	96.39	70.57	95.54	68.6
Linear	95.86	**65.89**	95.53	68.87	**95.16**	67.33
MVE	98.85	77.9	98.63	76.5	95.75	69.03
Health Insurance						
Logarithm	95.95	28.56	96.19	27.13	95.37	31.46
HLI	95.98	28.87	96.17	27.27	95.43	31.94
Linear	95.45	26.71	95.56	**25.08**	**95.15**	30.91
MVE	96.32	29.99	96.61	28.83	95.46	32.01
House8L						
Logarithm	95.65	90.11	95.71	86.26	95.79	102.34
HLI	95.77	90.79	95.83	86.99	95.91	102.82
Linear	**95.24**	88.54	95.26	**84.71**	**95.24**	100.55
MVE	96.22	92.77	96.32	88.96	96.88	106.66
MetroTraffic						
Logarithm	96.23	29.23	96.08	27.48	96.12	26.9
HLI	95.35	27.21	95.16	25.11	95.55	23.57
Linear	95.66	28.21	95.39	26.22	95.78	24.93
MVE	93.06	22.49	93.43	**19.75**	**94.89**	20.92
Diamonds						
Logarithm	95.15	19.3	95.33	11.54	95.17	11.59
HLI	95.1	19.18	95.37	11.6	95.09	11.42
Linear	95.08	19.18	95.13	11.3	95.12	11.54
MVE	94.96	19.04	95.36	11.62	**94.98**	**11.26**
Video Transcoding						
Logarithm	96.38	20.66	**94.74**	6.17	94.6	5.93
HLI	96.34	20.58	94.28	5.96	93.94	**5.68**
Linear	95.54	19.96	94.63	5.98	94.44	5.75
MVE	98.28	22.84	94.75	6.47	94.27	6.14
Wave Energy						
Logarithm	95.26	36.76	95.32	26.99	95.01	33.38
HLI	95.38	36.99	95.4	27.13	95.01	33.41
Linear	95.08	36.38	95.11	**26.7**	**95.0**	33.36
MVE	95.73	37.64	95.92	27.96	95.09	33.5
FriedmanGra						
Logarithm	95.17	36.78	95.15	27.12	95.02	33.48
HLI	95.27	36.98	95.17	27.17	94.93	33.37
Linear	95.05	36.48	95.05	**26.93**	**95.0**	33.46
MVE	95.59	37.57	95.62	27.92	**95.0**	33.44
FriedmanGsg						
Logarithm	94.86	30.0	94.78	22.0	94.2	29.72
HLI	94.6	29.76	94.56	**21.79**	93.35	28.45
Linear	94.77	29.85	94.63	21.81	93.99	29.38
MVE	94.86	30.17	**95.04**	22.48	92.93	27.86
FriedmanLea						
Logarithm	94.9	28.19	94.68	17.11	94.64	18.27
HLI	94.86	28.14	94.18	**16.63**	94.04	17.7
Linear	94.84	28.09	94.62	17.02	94.57	18.19
MVE	**95.03**	28.41	94.17	16.68	93.7	17.39
Hyper_a						

tion, we provide only the results for a lower limit of 0.1 with a confidence level of 95%. We choose 0.1 out of 3 choices in our setting for the lower limit because it is the median of the 3 values, preventing it to be too moderate or too radical. A 95% of confidence level is selected due to its universality in research and real-world problems. Results for other parameter settings can be found in the Appendix A.1.

Each patch in the tables represents a full set of results for a single dataset, with the dataset's name indicated below the patch. Bold text highlights the best values in the associated patches. If the bold is in the "coverage" columns, it indicates that the value is the closest to the confidence level. If the bold text is in the "NMPIW" columns, it signifies the smallest value in terms of interval width.

The tables indicate that the proposed adaptive approaches generally provide better performance compared to the baseline `MVE` method. Among the adaptive methods, the linear approach often yields the best balance between coverage and NMPIW. The performance varies depending on the dataset and the regression model, but overall, the adaptive methods demonstrate their capability to improve prediction interval estimates effectively.

More specifically, it is readily apparent that the performance of all the prediction interval methods in our experiments varies based on the base learners. For instance, the NMPIW results from the linear method in the "MetroTraffic" (Table 4.2) patch vary significantly between `kNN` and ARF-Reg (88.54% and 100.55%, respectively), despite achieving same coverage (95.24%).

SOKNL tends to correlate with narrower interval width, primarily due to its remarkable regressive performance (lower RMSE). According to Equation 4.9, interval width by MVE-based methods is strongly dependent on the RMSE or the regression model.

An obvious outlier are the results for the Naval Propulsion Plant dataset in Table 4.1. It exhibits the case where the predictive residuals do not follow the assumed distribution. The approaches that use SOKNL and `kNN` as the

base learners achieve very high coverage, with some also presenting very short NMPIW. This seems to be the ultimate goal of the prediction interval task. However, in our experiments, this is more likely caused by the dataset itself. The target values in the dataset have a very narrow range ([0.95, 1]). Furthermore, the ground truths are sorted in ascending order. Thus, regressive models that utilize feature space information (`kNN` and SOKNL) perform extremely well, leading to "perfect" prediction interval results. Nonetheless, this success cannot be replicated with other data streams. Please find more details concerning this dataset on OpenML. More investigation is required regarding this issue.

Undoubtedly, the performance of each method can vary depending on the dataset and the specific regression algorithm used. Unfortunately, these tabular results are too excessive to analyse and extract meaningful information from. The CING evaluation is intended to provide a clearer picture.

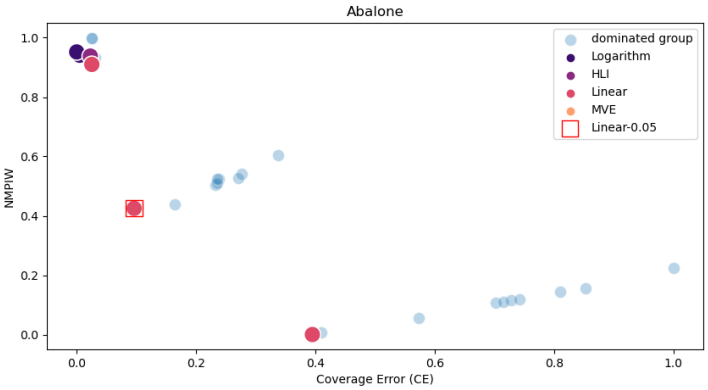
4.6.2 Results from CING evaluation

As introduced in 4.3.2.2, a score can be assigned to a single execution of a PI task, based on the CE and the NMPIW.

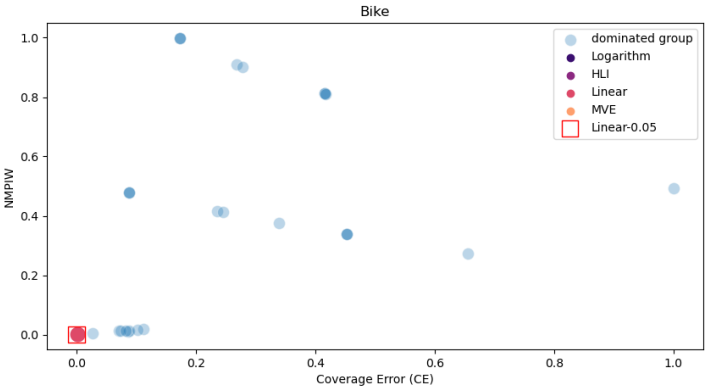
One of the advantages of CE is that it can mitigate the difficulty in comparing coverage results for diverse confidence levels. Due to this characteristic, we are able to summarize all the experiments with different hyperparameter settings for a particular dataset into a single figure.

Figure 4.4 illustrate parts of the results from our experiments using the CING evaluation approach. Each plot represents for specific dataset, and each dot within the plots marks an execution. Note that only six representative datasets are included in Figure 4.4; results for other datasets can be found in Figure A.1 and Figure A.2, located in Appendix A.2.

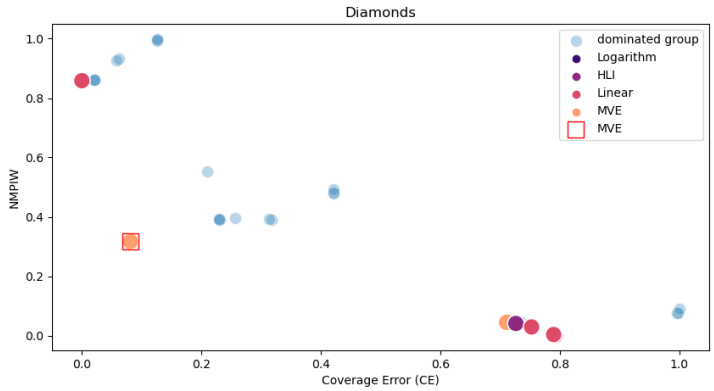
In the CING evaluation, the outcomes are first separated into dominated and non-dominated groups. We denote the dominated group with translucent blue dots, as they are significantly worse than the members in the non-



(a) Abalone



(b) Bike



(c) Diamonds

Figure 4.4: CING Evaluation Results for Different Datasets (Part I)

dominated group. The colourful dots, on the other hand, represent the non-dominated group. Furthermore, we framed the "best" approach in red, where the best score is calculated as the weighted sum of the normalized CE and NMPIW. In this work, given our higher emphasis on the coverage metric, we assign CE a weight of 0.8 and the NMPIW receives a weight of 0.2. Nevertheless, the weighting coefficients should be adjusted based on experimental requirements and real-world conditions as necessary. The "best" instance will also switch with different weight settings. In Figure 4.4, we have blurred out the parameter-related information to enable a focus on comparing and analyzing different approaches.

Figure 4.4a demonstrates the CING result on Abalone dataset. It represents for a typical situation where multiple adaptive approaches are categorized in the non-dominated group. This indicates that each adaptive approach has certain advantages for this dataset. The framed instance is from the linear approach with 0.05 limit setting. Although the red instance to the lower right has a better NMPIW, it is worse in terms of coverage. Due to our weighting strategy, the framed one achieves a better score.

Figure 4.4b represents a relatively rare case where there is only one instance in the non-dominated group. This means all the other approaches are worse than or at most equal to the framed instance regarding coverage and NMPIW. Therefore, the weighted-sum strategy does not impact the outcomes. Undoubtedly, this approach is the best for the Bike dataset.

In Figure 4.4c, unusually, an `MVE` solution is identified as the best one. This can occur when the adaptive approaches significantly modify the NMPIW without greatly affecting the coverage. Hence, when a 0.8-0.2 weight is applied to the results, `MVE` can take advantage and outperform other instances.

4.6.3 Dealing with Concept Drifts

In this section, we discuss how our adaptive prediction interval approaches perform under drifting conditions.

Here, we present the experimental results on the HyperA datasets, as it is frequently cited in the literature (e.g. [46, 117, 75]) and contains three abrupt simulated drifts.

Figure 4.5 illustrates the variations over time in the windowed coverage and the corresponding adjustment factors generated by the proposed adaptive prediction interval methods.

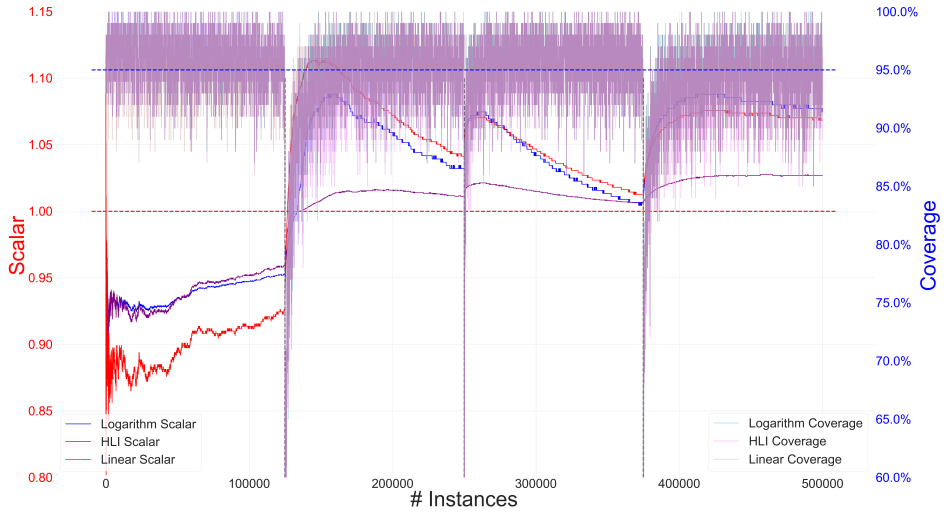


Figure 4.5: Factors from different adaptive approaches and the correlated coverage for SOKNL on HyperA dataset over time. The black vertical lines highlight positions of concept drifts; the red horizontal line marks the value 1.0; and the blue horizontal line emphasizes 95% confidence level.

Due to the nature of our adaptive approaches, the generation of the factors is determined according to the current coverage, allowing them to passively react to concept drifts. Particularly, when a concept drift occurs, the regressive model (e.g. SOKNL) will have a reaction period, during which the performance of the model (e.g. RMSE) will diminish. This means that the predictions produced by the model may deviate significantly from the truth values, resulting in generated intervals that do not cover enough target values, hence, causing the coverage to drop. Since the factor is generated by a function related to the current coverage, a decrease in coverage will lead to larger

factors, thereby widening the prediction interval and ensuring that more true values are covered.

This trend can be clearly observed in Figure 4.5. The HyperA dataset contains three abrupt drifts, locating at instances #125000, #250000, and #375000. The decreases in coverage coincide with the positions of the drifts. The produced factors also react accordingly to the coverage decreases. The enlarged factors ensure a rapid recovery from the drift in terms of coverage, which can be confirmed by the coverage returning to around 95% shortly after drifts. Therefore, the effectiveness of our adaptive approaches in adapting to drifts is evident.

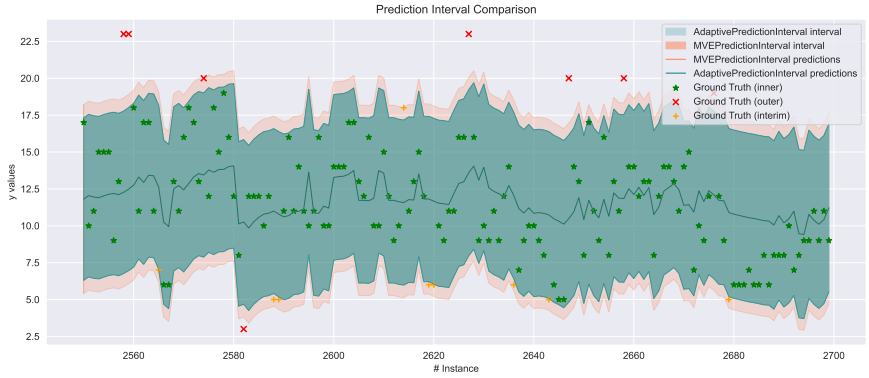
HLI always produces more "gentle" factors than the other two approaches, while the linear approach seems more radical. This suggests a potential assumption about this family of online prediction interval functions: more radical factor could be more beneficial to the final results.

4.6.4 Behavioural Visualisation

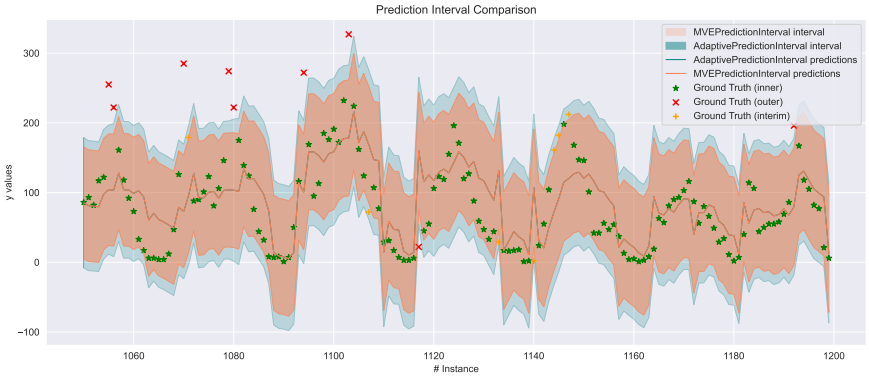
In this section, we present three images depicting the effectiveness of `AdaPI` for online prediction interval tasks. Although the three curves we use for the adaptive methods are different, their behaviours resemble each other on a broader perspective; that is, shrinking the `MVE` when the coverage is higher than desired and expanding the `MVE` otherwise. Therefore, we use only the logarithmic approach as an example here. The confidence level $\mathcal{L}$ and the limit $\mathcal{M}$ are set to 95% and 0.1, respectively.

In Figure 4.6 we present three images for different situations. Figure 4.6a shows results from logarithmic approach and `MVE` on part of the Abalone dataset. In this case, the coverage is slightly higher than the confidence level, Therefore, the adaptive approach generates smaller than 1 factor, resulting in constant shrinkage.

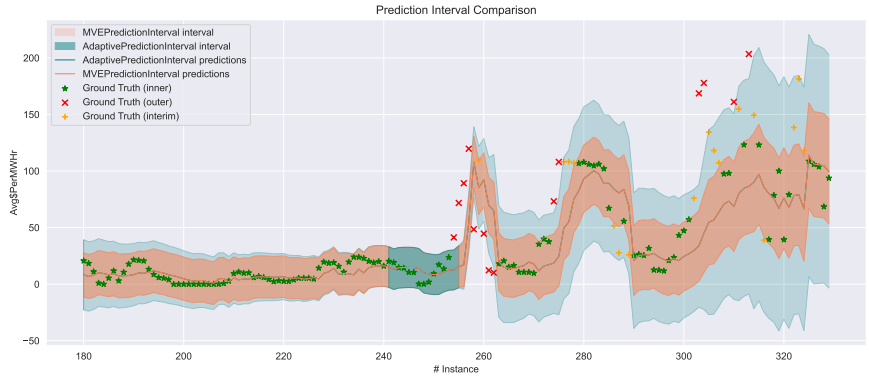
Figure 4.6b depicts the opposite condition on part of the Bike dataset. It can be seen that the adaptive method expands the `MVE` to capture more ground



(a) Abalone Data: Adaptive PI constantly smaller than MVE



(b) Bike Data: Adaptive PI constantly larger than MVE



(c) NZ Energy Price Data: Adaptive PI fluctuates between different modes

Figure 4.6: MVE Area (coral) and Adaptive PI approach (green), ground truths covered by both areas (green "★"), ground truths covered by neither areas (red "×"), and ground truths covered by wider area but not narrower area (orange "+")

truths, indicating the coverage for MVE approach is always below the confidence level.

Figure 4.6c demonstrates a more complicated situation on NZ Energy Price dataset. At the beginning of the figure, the adaptive model is shrinking MVE as the coverage is higher than desired. However, in the middle of the model, the ground truths suddenly change pattern (possibly due to a drift). The coverage naturally decreases for MVE, and the adaptive model starts to expand the interval to capture more ground truths. Consequently, on the right side of Figure 4.6c, it switches to expansion mode so that the green area (adaptive) covers the coral area (MVE). Notably, after switching to expansion mode, the adaptive prediction interval covers more data points that are missed by the basic MVE, as indicated by the orange "+" marks on the right side of the figure.

4.6.5 Coverage Results Comparison

One of the primary objectives of PI is to quantify uncertainty in regression problems, with coverage serving as the key metric for evaluation. That is also the reason for us to assign higher priority to coverage in the CING evaluation (see Section 4.3.2.2). Although an evaluation methodology that considers both coverage and interval width is proposed as CING, it is also important to show that the adaptive prediction interval approaches outperform MVE solely in terms of coverage metric.

We present visualizations images comparing coverage in this section. Figure 4.7 is a scattergram demonstrating the coverage results between MVE and all adaptive approaches (with different limits) for 95% confidence level.

The black lines denote the desired confidence level. The edges of the green area in Figure 4.7 indicate the values where the Coverage Error (CE) of MVE equals to CE of adaptive approaches, i.e. $CE_{MVE} = CE_{AdaPI}$. Therefore, the green area represent the area where $CE_{MVE} < CE_{AdaPI}$. Points will only be covered in the green area when the adaptive approaches achieve a closer-to-confidence-level coverage than the plain MVE under the same setting and

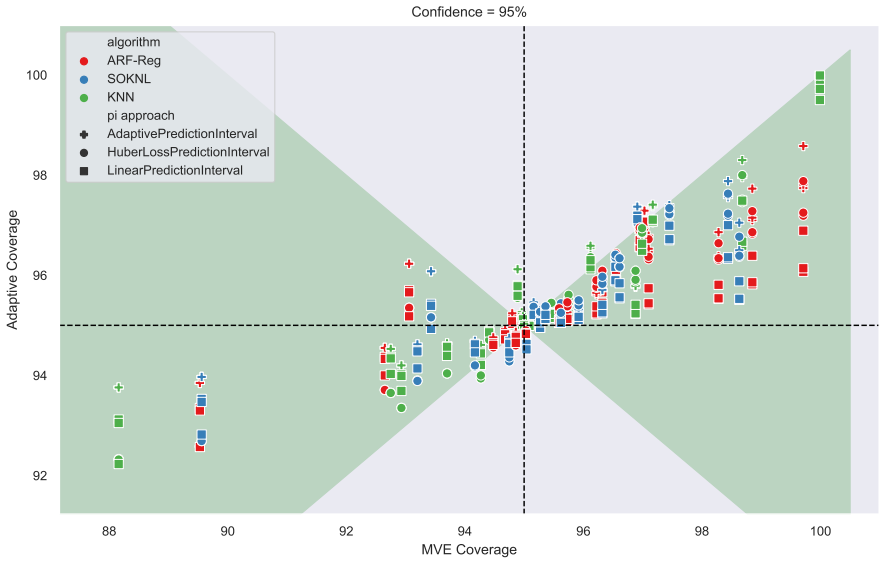


Figure 4.7: Scattergram for $\mathcal{L} = 95\%$.

The x-axis is the coverage value for MVE, and the y-axis shows the coverage value for adaptive approaches, black lines denote the 95% confidence level for both axes, green area highlights the space in which the adaptive approaches outperform MVE in terms of coverage, different colours represent different base learners, and different shapes distinguish different adaptive approaches.

dataset. Figure 4.7 conveys an overall impression of how the adaptive prediction interval approaches perform, attempting to include all relevant information for $\mathcal{L} = 95\%$.

It can be observed that most of the result points land in the green area, indicating that the proposed adaptive approaches generally push the MVE's coverage towards the confidence level. Exceptions can be also seen in the graph, and most of them are located around the cross point (black lines and green edges). This means that most the exceptions happen when MVE's coverage gets close to the confidence level. In these cases, the MVE's coverage pivots around the confidence level, resulting in frequent switches between expanding and shrinking modes in the adaptive approaches. This unstable behaviour causes slightly worse coverage results in some cases.

It is worth mentioning that most of the markers are reasonably far from the edges of the green area, indicating significant improvements of AdaPI with respect to coverage. Note that only the figure for 95% confidence level is shown here. Images regarding other confidence level values can be found in Appendix A.3.

Chapter 5

New Zealand Energy Pricing

As we state in the challenge section (Section 1), regression for SL faces the lack of benchmarking datasets. This chapter introduces and explores novel datasets capturing New Zealand's real-time energy pricing data. These datasets were curated to address a critical gap in resources for streaming regression research. By leveraging data stream analysis techniques, this study investigates the evolving nature of energy prices, identifies anomalies, and evaluates predictive algorithms under varying conditions of concept drift and time-dependence.

In this chapter, we present a newly introduced real-time streaming dataset, depicting the energy prices in New Zealand, maintained by the Electricity Market Information (EMI) website and provided by the New Zealand government. Extensive analysis and experiments are also conducted on the obtained datasets, affirming their usefulness.

5.1 Background and Related Work

On 1 November 2022, New Zealand introduced a real-time electricity pricing system, replacing the ex-post final prices system that had been operating since 1 October 1996. This system results in a new price series – Dispatch Energy Prices, which are usually derived from the SPD (Scheduling, Pricing, and Dispatch) model. The prices are summarized and then announced by the pricing manager.

Dispatch Energy Prices are produced throughout trading periods, each of which lasts for 30 minutes. During each trading period, multiple prices are announced and last for one to several minutes. At the end of the trading period, an interim price is calculated as a weighted sum of the produced prices in that period, where the weights are the duration of each price. If no price errors are claimed, the interim price is declared as the final price at 2 PM on the next business day. For more details, please refer to the EMI website.

Due to the significance of energy and electricity prices to the energy market, research concerning forecasting the energy price has drawn significant attention and produced many solutions. As summarized in [129, 91], numerous techniques and methodologies have been applied to energy price forecasting tasks over the past decades, many of which are machine learning-based.

The ML models utilized in this work are introduced as follows. Sliding Window KNN, Random AMRules [3], Adaptive Random Forest Regressor (ARF-Reg)[46], and Self-Optimising K-Nearest Leaves (SOKNL)[118], are advanced algorithms designed for data stream processing and adaptive learning. Sliding Window KNN uses a moving window to handle changing data distributions efficiently. Random AMRules improves rule-based learning by adapting to evolving data patterns. ARF-Reg leverages ensemble learning with random subspace sampling to provide robust regression predictions in dynamic environments. SOKNL integrates KNN and ARF techniques, using leaf centroids for maintaining relevant historical data and adapting to concept drift. The Adaptive Prediction Interval for Data Stream Regression by Sun [119] dynamically adjusts prediction intervals for accuracy in evolving data streams, using adaptive learning and ensemble techniques to ensure robust predictions. The Half-Space Trees (HST)[121] is a method for anomaly detection in data streams, leveraging a forest of trees built through random half-space partitions to identify low-density regions indicative of anomalies.

5.2 Data and Preprocessing

Although EMI provides plenty of different types of data, this work focuses on the Dispatch Energy Prices themselves, leaving other data for future work.

5.2.1 Brief Data introduction

The daily dispatch energy prices are available on the website as CSV files. The files contain the following features:

- `Trading Date`

  the date that the trading takes place in the format of yyyy-mm-dd.

- `Trading Period`

  the trading period indices, usually 1 to 48.

- `Publish Date Time`

  ISO 8601 formatted date and time in NZ of the prices published by the pricing manager.

- `PointOfConnection (PoC)`

  the point of connection on the grid, possesses more than 200 unique values, more details on the Network supply points (NSP) table website.

- `Island`

  binary values indicating the North Island or South Island of NZ.

- `IsProxyPriceFlag`

  boolean values indicating whether the price is a proxy price, which happens when the PoC is disconnected from the grid due to transmission outage.

- `DollarsPerMegawattHour`

  the published dispatch energy price in dollars per megawatt-hour.

All the files are downloadable from the Dispatch Energy Prices site.

5.2.2 Data Preprocessing

As detailed in Section 5.2.1, most of the attributes included in the data are either time or location related, which are not directly correlated with the energy prices. In this work, we propose an example of preprocessing the provided data source for data stream research. We abstract from 1 Nov. 2022 to 30 Apr. 2024 (18 months) in this work.

The EMI updates the data files on a daily basis, and it is straightforward to acquire desired information for specified periods utilizing our scripts. All the associated code is located at our Github repository.

Figure 5.1: PoCs Overview

5.2.2.1 Regionalization

Our first focus is the `PoC` attribute. There are over 200 Point of Connections throughout New Zealand. The energy demands and prices are based on the different locations themselves meaning that being close in a geographical sense does not necessarily correlate with similar energy prices. Therefore, our preprocessing treats each location (PoC) as a separate case.

Figure 5.1 shows an overview of the PoCs. It is apparent that the PoCs are more dense in the North Island, aligning with the population distribution. This work chooses six PoCs, highlighted as purple markers in Figure 5.1: Albany, Auckland (`ALB0331`); Central Hamilton (`HAM0331`); Wilton, Wellington (`WIL0331`), Islington, Christchurch (`ISL0661`), South Dunedin (`SDN0331`), Stoke, Nelson (`STK0331`). These PoCs are in the main cities in both the North

and South Islands of New Zealand, thus representing more of the population. Datasets for other locations are also easily obtainable via the provided code.

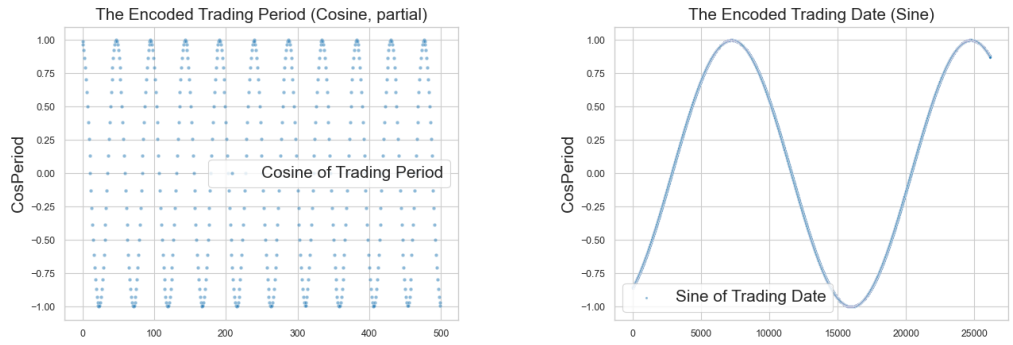
5.2.2.2 Trading Date and Period

Trading date and period represent the periodic information in a year and a day, respectively. We convert the date and period index information into numeric values using trigonometric functions. Specifically, the trading date and period are normalized into $[0, 2\pi]$ using Equation 5.1:

$$x_i = 2\pi \cdot \frac{X_i - X_{min}}{X_{max} - X_{min}} \tag{5.1}$$

where x_i is the i_{th} normalized value, X_i is the i_{th} raw value, and X_{max} and X_{min} are the maximum and minimum values in the series, respectively. Then, the trigonometric functions, Sine and Cosine, are applied to the normalized series. Four new columns are appended to the datasets in this manner: two for the trading period and two for the trading date.

Note that this approach might remove the seasonal information in the original data; therefore, the original information is also retained. Users can select whether they are needed according to the usage context. Figure 5.2a and 5.2b illustrate the trading period and date after encoding.



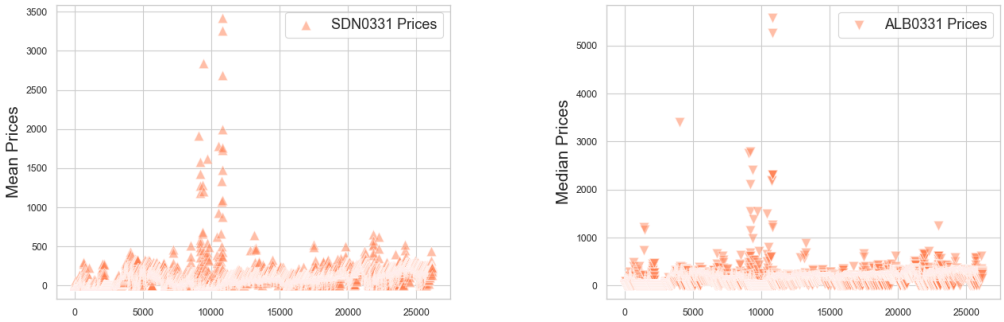
(a) Trading period, showing daily pattern

(b) Trading date, showing yearly pattern

Figure 5.2: Showcase of the encoded periodical information, PoC: `HAM0331`.

5.2.2.3 Define Targets and Relevant Features

Under the current NZ energy system, multiple price announcements are made during a single trading period, and the quantity varies for different periods. As a consequence, in our preprocessing phase, we take either the **mean** or the **median** value to represent a certain trading period as the target.



(a) Mean prices from `SDN0331` PoC. (b) Median prices from `ALB0331` PoC.

Figure 5.3: Demonstration of target values as mean and median prices.

Considering the availability of the previously published prices and the high frequency of updates to the database, it is reasonable to include the previous ground truths in the datasets. In our approach, we include both the mean and median values from the past in the feature space. Our scripts allow users to specify how many trading periods need to be predicted in advance, which is defined as **Delay** in this article as it is a simulation of the delay of the target values' arrival. We include four groups of datasets, predicting the energy prices 0.5, 4, 6, and 24 hours ahead. By adding these features, we include the time dependence information into the data [14]. In this chapter, we focus on data stream abstraction, leaving other scenarios, such as time-series and auto-regression, for further study.

As observed in Figure 5.3, the target values, both in mean and median aspects, occasionally reach extremely high levels. These can be defined as outliers or anomalies. Consequently, we also propose datasets for anomaly detection tasks based on the data, where the anomalies are the highest cer-

tain percent of values in the datasets. The ratio of the anomalous targets is determined by the users.

5.3 Experiments

This section introduces experimental settings conducted on the extracted datasets.

Regression. As the main purpose and most general use case of this project, regression tasks regarding energy prices are the first assessment in this work. Four state-of-the-art streaming regression algorithms are tested: Sliding Window kNN, Random AMRules (RAMRules)[3], Adaptive Random Forest for Regression (ARF-Reg)[46], and Self-Optimising K Nearest Leaves (SOKNL)[118]. Please refer to Chapter 2 for details. The parameter settings for these algorithms follow the defaults in the MOA [13].

Prediction Interval. The prediction interval technique is particularly suitable for energy price forecasting. A predicted range of future prices provides adequate information for decision-making regarding the energy market [111, 134]. In this work, we conduct PI tasks with the first proposed, fully incremental, online prediction interval method – Adaptive Prediction Interval (AdaPI) [119]. AdaPI uses the default settings defined in CapyMOA [49]. Only SOKNL is used as the base learner in this work. See more details in Chapter 2.

Drifts. Another challenging yet important issue in stream learning is concept drift. In this work, we use Page-Hinckley [**?**] and ADWIN [11] detectors on the Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) metrics to demonstrate the drifts in the proposed datasets. Both detectors are available on MOA, CapyMOA, and River [96].

Anomaly Detection. As supervised anomaly detection is excessively studied [18], in this paper, we only consider unsupervised streaming anomaly detection tasks. The Half-Space Trees [121] algorithm is presented here as a demonstration of the datasets' usefulness. We use the implementation on the

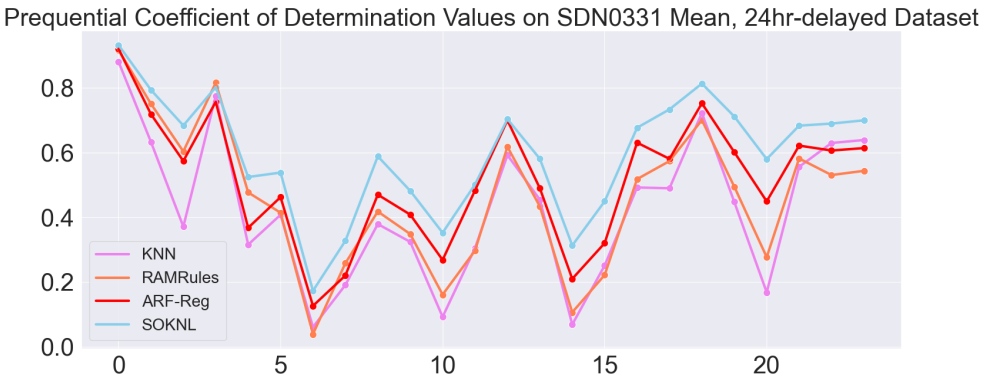


Figure 5.4: SDN0331 PoC: Prequential R^2 Score with 1000 Window Size

River platform. All experiments are also repeated 10 times with different random seeds.

5.4 Results

5.4.1 Regression

Table 5.1 illustrates the overall Coefficient of Determination (R^2) results from the extensive regression experiments. Ten runs with different random seeds were executed on all datasets and algorithms, and the average R^2 values are exhibited in the table. However, all the standard deviation values were smaller than 0.01, so they are omitted in the table.

Key takeaways of Table 5.1: (a) ISL0661, SDN0331, STK0331, and WIL0331 demonstrate relatively stable R^2 values across different algorithms, while ALB0331 and HAM0331 show more fluctuation. (b) SOKNL consistently produces better R^2 results for all PoCs. RAMRules performs less competitively in most cases. (c) As expected, the delay length has a negative impact on the performance of all algorithms. Still, the performance drops are at an acceptable level.

Figure 5.4 illustrates an example exhibition of prequential R^2 results. Due to the space limit, only SDN0331 PoC with mean values as the targets is displayed here. The performance for four algorithms fluctuates with a similar tendency, seemingly affirming several concept drifts in the data.

Table 5.1: Coefficient of Determination Values by Algorithms, PoCs, and Delays

Alg.	ARF-Reg		KNN		RAMRules		SOKNL	
PoC	Mean	Median	Mean	Median	Mean	Median	Mean	Median
Delay = 30 Minutes								
ALBO331	0.56	0.54	0.6	0.57	0.2	0.22	**0.61**	**0.58**
HAMO331	0.57	0.54	0.6	0.58	0.25	0.31	**0.61**	**0.59**
ISLO661	0.63	0.62	0.65	0.64	0.59	0.59	**0.67**	**0.65**
SDNO331	0.64	0.64	0.67	0.66	0.6	0.61	**0.69**	**0.67**
STKO331	0.62	0.61	0.65	0.64	0.55	0.57	**0.67**	**0.65**
WILO331	0.61	0.59	0.64	0.62	0.44	0.37	**0.66**	**0.63**
Delay = 4 Hours								
ALBO331	0.5	0.48	0.5	0.47	0.4	0.38	**0.57**	**0.54**
HAMO331	0.51	0.49	0.51	0.47	0.42	0.41	**0.58**	**0.55**
ISLO661	0.56	0.55	0.57	0.54	0.49	0.48	**0.63**	**0.62**
SDNO331	0.58	0.57	0.58	0.56	0.52	0.5	**0.65**	**0.63**
STKO331	0.56	0.54	0.56	0.54	0.48	0.46	**0.63**	**0.61**
WILO331	0.55	0.53	0.55	0.51	0.43	0.43	**0.62**	**0.59**
Delay = 6 Hours								
ALBO331	0.51	0.48	0.49	0.45	0.34	0.24	**0.57**	**0.54**
HAMO331	0.51	0.48	0.5	0.46	0.39	0.25	**0.58**	**0.55**
ISLO661	0.56	0.55	0.56	0.54	0.5	0.49	**0.63**	**0.61**
SDNO331	0.58	0.57	0.58	0.56	0.52	0.51	**0.65**	**0.63**
STKO331	0.56	0.54	0.55	0.53	0.49	0.48	**0.63**	**0.6**
WILO331	0.55	0.53	0.54	0.5	0.38	0.32	**0.62**	**0.59**
Delay = 24 Hours								
ALBO331	0.53	0.49	0.46	0.43	0.43	0.4	**0.58**	**0.56**
HAMO331	0.53	0.5	0.47	0.44	0.47	0.41	**0.59**	**0.57**
ISLO661	0.57	0.56	0.53	0.51	0.57	0.55	**0.63**	**0.63**
SDNO331	0.59	0.58	0.55	0.52	0.55	0.56	**0.65**	**0.65**
STKO331	0.57	0.56	0.53	0.5	0.57	0.56	**0.63**	**0.62**
WILO331	0.57	0.54	0.51	0.48	0.51	0.5	**0.63**	**0.61**

5.4.2 Prediction Interval

Table 5.2 presents the PI results with 80% and 95% confidence level.

Overall, the coverage values are all higher than the desired confidence level, and the NMPIW values are at a low level. This means that for these datasets, AdaPI can generate relatively compact intervals that cover more-than-required ground truths. This can greatly benefit decision-making in the energy market.

Table 5.2: Coverage and NMPIW values for different PoCs and Delays

Delay	30 Minutes		4 Hours		6 Hours		24 Hours	
PoC	Coverage	NMPIW	Coverage	NMPIW	Coverage	NMPIW	Coverage	NMPIW
Confidence Level = 95%								
ALB0331	98.03	4.91	97.90	5.18	97.90	5.18	97.87	5.10
HAM0331	98.02	4.88	97.86	5.16	97.82	5.20	97.82	5.12
ISL0661	97.26	4.97	97.12	5.28	97.14	5.26	97.18	5.21
SDN0331	97.20	5.04	97.03	5.36	96.99	5.34	97.12	5.30
STK0331	97.33	4.91	97.16	5.22	97.14	5.19	97.24	5.18
WIL0331	97.32	4.88	97.12	5.21	97.18	5.22	97.14	5.12
Confidence Level = 80%								
ALB0331	93.60	2.98	92.68	3.18	92.58	3.18	92.56	3.12
HAM0331	93.48	2.97	92.52	3.17	92.36	3.20	92.44	3.14
ISL0661	90.66	3.01	89.72	3.21	89.70	3.22	89.80	3.19
SDN0331	90.52	3.04	89.57	3.26	89.36	3.25	89.53	3.24
STK0331	90.91	2.98	89.81	3.16	89.78	3.18	89.98	3.16
WIL0331	91.65	2.96	90.34	3.18	90.46	3.19	90.44	3.13

For each PoC, both Coverage and NMPIW values remain relatively stable across different delays, indicating that the model's prediction accuracy and interval width do not significantly fluctuate with the delay. There is also a clear trade-off between coverage and NMPIW, i.e., higher coverage usually requires wider NMPIW. Among the PoCs, ALB0331 and HAM0331 generally show higher coverage and relatively lower NMPIW compared to other PoCs.

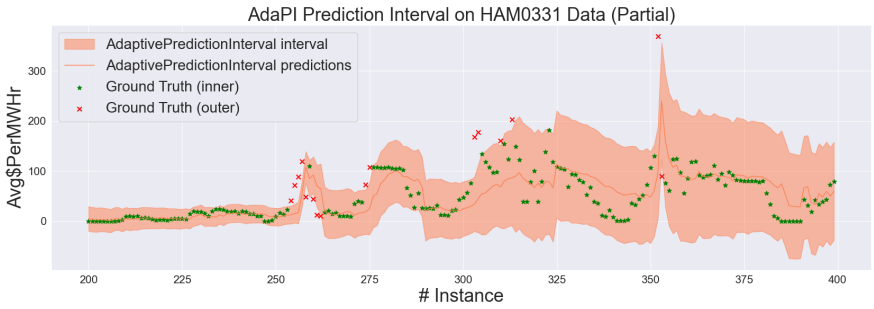
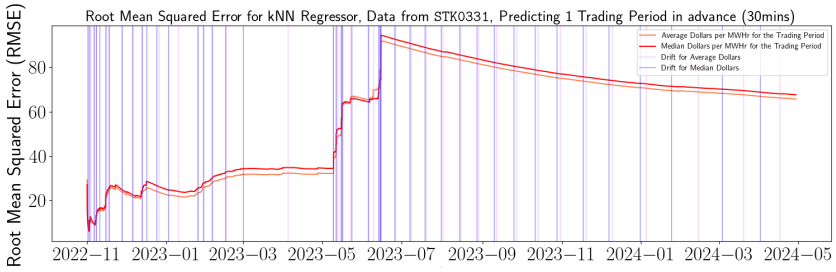


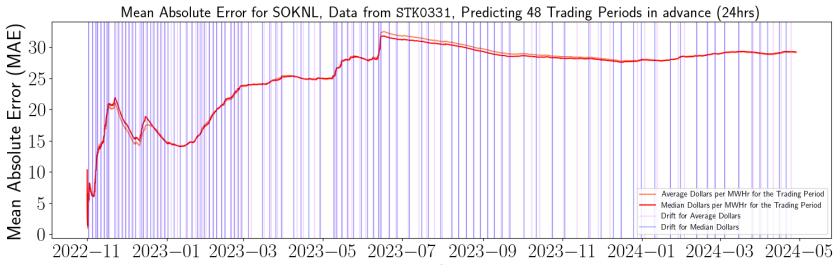
Figure 5.5: Visualized Example of Prediction Intervals Over Time

Figure 5.5 is a visualization of the real-time generated prediction intervals. Only a small portion of the whole dataset is presented here. It can be seen that the data starts fluctuating around 250 instances, and AdaPI adapts to it by enlarging the PI area to protect the coverage from decreasing dramatically.

5.4.3 Drifts



(a) RMSE from KNN Regression by Page-Hinckley with 30 Minutes Delay



(b) MAE from SOKNL by ADWIN with 24 Hours Delay

Figure 5.6: Drift Analysis Showcases on Data from `STK0331`

Figure 5.6 illustrates examples of drift analysis on data from the `STK0331` PoC. Among them, Figure 5.6a includes the drifts detected by the Page-Hinckley test based on the RMSE values from the KNN regressor. Figure 5.6b, on the other hand, highlights the drifts detected by an ADWIN detector using MAE from the SOKNL algorithm. Both median and average targets are involved in the figures. The orange line and violet vertical lines represent the performance and the detected drifts on the average dataset, respectively. Similarly, the red line and blue vertical lines represent the errors and drifts on the median dataset.

One can see that drifts may happen quite frequent;y in these energy prices. This justifies studying and analyzing these datasets using data stream techniques instead of traditional batch learning ones.

There are more drifts reported in Figure 5.6b than in Figure 5.6a because ADWIN is more sensitive than the Page-Hinckley test. There are two apparent

error increases between 2023-05 and 2023-07, which align with the winter season in New Zealand. Intuitively, they are related; however, more investigation is needed for confirmation.

5.4.4 Anomaly Detection

Table 5.3: Half-Space Trees AUC Scores by Anomaly Ratios, PoCs, and Delays

Ratio_A	0.1%		0.5%		1%		2%	
PoC	Mean	Median	Mean	Median	Mean	Median	Mean	Median
`Delay = 30 Minutes`								
ALB0331	0.597	0.668	0.472	0.446	0.429	0.421	0.454	0.452
HAM0331	0.598	0.669	0.47	0.45	0.425	0.419	0.442	0.445
ISL0661	0.573	0.629	0.468	0.486	0.504	0.519	0.5	0.503
SDN0331	0.541	0.627	0.484	0.497	0.501	0.507	0.518	0.525
STK0331	0.572	0.628	0.471	0.461	0.527	0.541	0.496	0.5
WIL0331	0.611	0.65	0.469	0.462	0.433	0.451	0.456	0.454
`Delay = 4 Hours`								
ALB0331	0.388	0.446	0.382	0.375	0.38	0.381	0.428	0.435
HAM0331	0.388	0.446	0.383	0.378	0.378	0.379	0.418	0.428
ISL0661	0.342	0.386	0.378	0.423	0.461	0.486	0.474	0.484
SDN0331	0.313	0.386	0.404	0.436	0.461	0.474	0.492	0.506
STK0331	0.343	0.386	0.386	0.402	0.485	0.505	0.471	0.482
WIL0331	0.389	0.428	0.377	0.391	0.386	0.413	0.429	0.437
`Delay = 6 Hours`								
ALB0331	0.387	0.444	0.384	0.377	0.377	0.383	0.424	0.433
HAM0331	0.388	0.444	0.384	0.379	0.375	0.38	0.413	0.426
ISL0661	0.340	0.380	0.378	0.428	0.465	0.491	0.472	0.483
SDN0331	0.312	0.379	0.402	0.438	0.461	0.475	0.49	0.504
STK0331	0.34	0.38	0.383	0.406	0.482	0.504	0.468	0.48
WIL0331	0.39	0.426	0.378	0.393	0.383	0.413	0.427	0.435
`Delay = 24 Hours`								
ALB0331	0.565	0.571	0.416	0.405	0.414	0.414	0.439	0.443
HAM0331	0.565	0.571	0.416	0.403	0.414	0.414	0.433	0.44
ISL0661	0.524	0.513	0.463	0.491	0.526	0.543	0.498	0.511
SDN0331	0.494	0.512	0.488	0.503	0.542	0.554	0.524	0.54
STK0331	0.523	0.513	0.468	0.47	0.522	0.534	0.496	0.501
WIL0331	0.562	0.546	0.404	0.404	0.426	0.461	0.458	0.464

Table 5.3 exhibits the Receiver Operating Characteristic - Area Under the Curve (ROC AUC) scores from Half-Space Trees algorithm executed on the

processed datasets with different anomaly ratios.

Overall, these datasets appear quite challenging for Half-Space Trees as most of the AUC scores are under 0.5. Due to the temporal characteristics of the datasets, we did not apply a shuffle procedure to the data. Additionally, in most of the datasets, the anomalies (high values of price) are not distributed in a consistent pattern. As illustrated in Figure 5.3, the outliers are concentrated in a specific time period (possibly May – July 2023, taking into account Figure 5.6). This may cause difficulty for the streaming anomaly detection algorithms.

More specifically, the performance of Half-Space Trees is influenced by both the delay and the anomaly ratio. It performs best with shorter delays and lower anomaly ratios. However, with longer delays, it adapts and performs better with higher anomaly ratios. The regression tasks can benefit from the anomaly detection to determine if an "abnormal" target value should be treated as an anomaly or the start of a drift.

Chapter 6

Dynamic Ensemble Member Selection (DEMS)

In Chapter 3, we address the regression tasks by sorting the ensemble members and utilizing the top-K members for the final prediction, where the "top-K" is determined using a dynamic selection manner based on historical performance information. As this approach is proven effective by SOKNL, we attempt to adapt it to classification algorithms. This chapter, as the result of the adaptation, presents a methodology for streaming classifier – Dynamic Ensemble Member Selection (DEMS).

6.1 Introduction

Research has proposed various methods to enhance the performance of ensemble learners, including techniques like Feature Selection [83] and different voting techniques [85]. Classifier Selection (CS) [107] offers an alternative way to aggregate ensemble predictions. Dynamic Classifier Selection (DCS) or Dynamic Ensemble Selection (DES) [43, 24] are particularly powerful among other selection approaches. However, there has been less attention given to DCS or DES in the context of streaming data, which partially motivates the focus of this work. Furthermore, Ensemble Pruning [125] is also an approach to filter "not useful" base learners out from the ensemble. Nonetheless, Ensemble

pruning is commonly employed to reduce computational resource consumption. Current research also seldom addresses streaming ensemble pruning [34].

6.2 Dynamic Ensemble Member Selection

In this chapter, we introduce the Dynamic Ensemble Member Selection (DEMS) framework, a fully online technique that, in theory, can be applied to all ensemble-based streaming classifiers. However, experimental results indicate that it is less effective for Boosting algorithms, as discussed in Section 6.4.

6.2.1 Background

Ensemble techniques often leverage randomization to induce diversity among learners. Due to randomization, for any particular instance, some of the ensemble members will perform better than others. Another characteristic in ensembles is that members typically generate probabilities for each class instead of a single class prediction. In some cases, when the highest probability is almost the same value as the second highest, then this prediction clearly lacks confidence.

These considerations suggest two factors for evaluating the relevance of ensemble members: classifier accuracy (to assess the proficiency of a base learner, denoted as $\mathcal{A}$) and "predictive margin" (to gauge the confidence of a base learner in a particular prediction, denoted as $\mathcal{M}$). While classifier accuracy is straightforward, the definition of "predictive margin" differs from the common margin concept that can be found in [9, 58]. Our defined "predictive margin" is the difference between the first-ranking probability and the second-ranking one in the vote, and can be expressed as Equation 6.1:

$$\mathcal{M} = \frac{v_f - v_s}{\sum_{c=1}^{N} v_c} \tag{6.1}$$

where v_f and v_s are the first and second-ranking vote respectively, c and v_c is the index of each class in the target value and the associated vote, and N

denotes the total class number. The intuition behind this is explained later in Section 6.2.4. Note that in the common cases where the outputs are in soft-label format, the denominator in Equation 6.1 should equal to one. However, some learners generate predictions in different ranges. Therefore, we express it in a more general way, as in Equation 6.1.

This definition is akin to the "margin" concept introduced in [9]. However, in stream learning problems, specifically in the test-then-train fashion, the "correct" labels are not available during the prediction stage. Therefore, unlike in [9], DEMS can only employ this alternative approach. A consequence of this difference is that the margin in [9] is a number in the range $[-1, 1]$, but the predictive margin $\mathcal{M}$ in this work is always a non-negative value.

In DEMS, both $\mathcal{A}$ and $\mathcal{M}$ are utilized as the selection criterion. Specifically, since $\mathcal{A}$ and $\mathcal{M}$ are all probability-like and can only range from zero to one, the geometric mean of these two metrics is used to rank the ensemble members. Henceforth, the combination of these two metrics is referred to as the "Confidence of the base learner", denoted by $\mathcal{C}$, i.e. $\mathcal{C} = \sqrt{\mathcal{A} \times \mathcal{M}}$.

6.2.2 Dynamic Ensemble Member Selection

In this section, we present an overview of how DEMS operates. During the prediction stage, DEMS queries all ensemble members for their accuracy ($\mathcal{A}$) and vote, from which the predictive margin ($\mathcal{M}$) can be computed. Subsequently, all ensemble members are ranked according to the combined confidence ($\mathcal{C}$). Only a leading portion of the base learners will participate in the final prediction, i.e., only $\mathcal{K}$ out of all the ensemble learners are selected. The optimal $\mathcal{K}$ is determined through historical information and updated during the training session. The final prediction comprises the aggregation of the vote from the top-ranking $\mathcal{K}$ members.

In the training stage, alongside the standard learning process, DEMS incorporates a simulated prediction phase, during which DEMS acquires predictions for all possible $\mathcal{K}$ values. These predictions are assessed against the

ground truth, and the performance of each $\mathcal{K}$ value is recorded and updated accordingly. Subsequently, DEMS uses the recorded performance to decide the optimal $\mathcal{K}$ for the next prediction. This $\mathcal{K}$ selection mechanism is similar to the one in the SOKNL [117], known there as the "self-optimising" mechanism.

The $\mathcal{K}$ selection can be performed in either a global manner or a sliding windowed manner. In the global approach, the performance of all possible $\mathcal{K}$ values is assessed and recorded from the start of the data stream to the prediction moment. Conversely, in the windowed manner, $\mathcal{K}$ values are selected based on the most recent window of the data stream.

6.2.3 Pseudo-Code

The pseudo-code of DEMS is detailed in Algorithm 2. It is important to note that this is a fully supervised algorithm employing a test-then-train approach. In this manner, examples from the data stream acquire the ground truths immediately after the testing procedure and are then used for model training.

6.2.4 Theoretical Insights

In this section, we discuss the potential reasons why DEMS has the capability to improve upon existing ensemble methods.

As an example of Dynamic Ensemble Selection (DES) techniques, following the taxonomy in [24], a DES consists of three parts: Ensemble Generation, Ensemble Selection, and Ensemble Aggregation. Some algorithms primarily focus on the generation step, such as Streaming Random Patches [51], where effectiveness largely depends on ensemble generation and diversity induction. Ensemble aggregation is how the results from different learners are combined. For example, Adaptive Random Forest [44] uses weighted majority votes, with the weights being the individual accuracy of the base learners.

DEMS, as a framework capable of working with all streaming ensemble methods, relies on the ensemble itself to construct the first stage. For ensemble selection, DEMS chooses the best-performing $\mathcal{K}$ learners in the ensemble

Algorithm 2: Dynamic Ensemble Member Selection

Notations: $\mathcal{E}$: ensemble; $N_{\mathcal{E}}$: ensemble number; $\mathcal{K}$: K values; $\mathcal{K}_{best}$: the best K value ; $\mathcal{L}$: base learners; E: evaluator ; $\mathbf{e}$: example; $\mathcal{A}$: base learner accuracy; $\mathcal{C}$: confidence value (Sec 6.2.1); $\mathbb{R}$: list of the number of correctly classified examples for each $\mathcal{K}_i$.

Data: Data Streams : $\mathcal{D}$

Result: Predictions : $\mathcal{P}$

`/* Initialization */`

$\mathcal{K}_{best} \leftarrow N_{\mathcal{E}}$;

for $i < N_{\mathcal{E}}$ **do**

    build $\mathcal{L}_i$;

    build E_i;

`/* Execution */`

while $\mathcal{D}$ *has more examples* **do**

    `/* test example e */`

    for $i < N_{\mathcal{E}}$ **do**

        $\mathcal{M}_i \leftarrow \mathcal{L}_i$.getMargin($\mathbf{e}$);

        $\mathcal{A}_i \leftarrow E_i$.getAccuracy();

        $\mathcal{C}_i \leftarrow \mathcal{M}_i \times \mathcal{A}_i$;

        $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_i$

    sort $\mathcal{E}$ by $\mathcal{C}$;

    `/* predict by top` $\mathcal{K}_{best}$ `learners */`

    for $i < \mathcal{K}_{best}$ **do**

        $\mathcal{P}_i \leftarrow \mathcal{L}_i$.predict($\mathbf{e}$);

    return $\mathcal{P} \leftarrow \sum_{i=0}^{\mathcal{K}} \mathcal{P}_i$;

    `/* train with example e */`

    for $i < N_{\mathcal{E}}$ **do**

        evaluate $\mathcal{L}_i$; `/* update` E_i `&` $\mathcal{A}_i$ `*/`

        evaluate $\mathcal{K} = i$; `/* update` $\mathbb{R}$ `*/`

        $\mathcal{L}_i$.train($\mathbf{e}$); `/* do regular train */`

    $\mathcal{K}_{best} \leftarrow \underset{index}{argmin}(\mathbb{R}) + 1$; `/* update` $\mathcal{K}_{best}$ `*/`

based on base learner accuracy $\mathcal{A}$ and the predictive margin $\mathcal{M}$ defined in Equation 6.1. One can easily observe that selecting $\mathcal{K}$ members is similar to the pruning technique for batch learning. The regular margin is frequently employed for pruning traditional ML ensemble algorithms [59, 72, 133], demonstrating the effectiveness of the regular margin as a pruning criterion. However, the regular margin is defined as in Equation 6.2:

$$\text{Margin} = \frac{v_t - v_{w_1}}{\sum_{c=1}^{N} v_c} \tag{6.2}$$

where v_t is the vote for the true class, and the v_{w_1} is the first ranking wrong vote.

Nonetheless, unlike the batch learning protocol, supervised streaming learning cannot access the class label during the prediction step. Therefore, Equation 6.2 is not applicable in the streaming scenario. This is the reason why we replace the regular margin with our predictive margin $\mathcal{M}$.

Although the label information is not available in the prediction phase, each individual learner $\mathcal{L}_i$ in the ensemble can be assigned an incremental evaluator E_i. Using the evaluator, the accuracy $\mathcal{A}_i$ can be assessed, which can also be regarded as the possibility of the highest vote in the predictive result being correct. This can be expressed as in Equation 6.3:

$$\mathcal{A} = \mathbb{P}(v_f = v_t) \tag{6.3}$$

where v_f and v_t are defined in Equation 6.1 and 6.2, respectively.

Thus, we have found a way to simulate the margin without knowing the correct class label. The effectiveness of this simulation depends largely on the individual accuracy $\mathcal{A}$. Only when $\mathcal{A}$ is adequately high can the predictive margin more closely approximate the regular margin criterion. As a consequence, both accuracy $\mathcal{A}$ and predictive margin $\mathcal{M}$ are considered as the sorting criteria in DEMS. It is noteworthy that when all single component classifiers are rather inaccurate, DEMS may not be capable of significantly improving performance, as observed in the results in Section 6.4.

The choice of the geometric mean for combining $\mathcal{M}$ and $\mathcal{A}$ instead of using their sum (or weighted sum) is based on two main reasons. Firstly, the summation strategy is typically used for independent values. However, $\mathcal{M}$ and $\mathcal{A}$ are not independent in our context. Summation is appropriate when either variable being high is sufficient, whereas the geometric mean is more suitable when both variables need to be large. As discussed in Section 6.2.2, ensemble members are included in the final prediction process only when they are both accurate and confident, as measured by $\mathcal{A}$ and $\mathcal{M}$, respectively. Secondly, the weighted sum approach requires a weight vector, adding extra parameters to the algorithm. This complexity hinders further modifications to DEMS, such as incorporating additional sorting and evaluation criteria like tree depths, node weights, etc. Consequently, we chose the geometric mean to represent the effectiveness of the base learners, considering these factors.

DEMS possesses two other important properties. Firstly, DEMS can be universally implemented for existing streaming ensembles as long as the individual learners' $\mathcal{A}$ can be assessed in real-time. Secondly, a base learner with a built-in drift detection and adaptation mechanism will have better accuracy ($\mathcal{A}$), that in turn might help DEMS to also yield larger improvements.

6.2.5 Diversity Evaluation

As widely discussed in the literature ([29, 132, 98, 44, 53], etc.), diversity is an essential aspect of ensemble learning. DEMS's member selection strategy may cause concerns regarding a possible reduction of diversity. As a consequence, we provide a diversity analysis in this work One of the most commonly used approaches – entropy-based measurement [25, 88] – is utilized as the diversity criterion.

The proof for entropy increasing with ensemble diversity can be demonstrated through analysis of extreme cases and intermediate scenarios. Consider an ensemble with m classifiers predicting probabilities over n classes. In the case of perfect agreement, all classifiers predict identical probabilities:

$$p_i = 1 \text{ for some class } i \text{ and } p_j = 0 \text{ for all } j \neq i$$

This results in entropy:

$$H(p) = -(1\log_2(1) + 0\log_2(0)) = 0 \tag{6.4}$$

Conversely, in the case of maximum disagreement, classifiers predict a uniform distribution:

$$p_i = \frac{1}{n} \text{ for all classes } i$$

yielding entropy:

$$H(p) = -\sum_{i=1}^{n} \frac{1}{n} \log_2(\frac{1}{n}) = \log_2(n) \tag{6.5}$$

For any intermediate case, we can apply Jensen's inequality due to the concavity of entropy:

$$H(\sum_{i=1}^{m} w_i p_i) \geq \sum_{i=1}^{m} w_i H(p_i)$$

where w_i are weights summing to 1. This inequality proves that entropy increases with prediction diversity.

In summary, entropy across the probability distribution of the predictions provided by the ensemble can be regarded as a measure of diversity. Evaluating the entropy within the votes from the full ensemble as well as the selected subset offers insights into the changes in diversity. According the Equation 6.4 and 6.5, higher entropy indicates greater diversity within the ensemble.

6.3 Experiments Settings

Our experimental comparison comprises 18 common benchmark datasets from the data streams literature. Half of them are real-world datasets, half of them are synthetic. Detailed introduction of the datasets can be found in 2.3.2.

6.3.1 Algorithm Parametrization

In this section, the parametrization for our experiments is presented for clarity and reproducibility.

We apply DEMS to five ensemble algorithms and compare them to their original models. These five ensembles have already been mentioned in Section 2: Online Accuracy Updated Ensemble (OAUE) [16], Online Smooth Boosting (OSBoost) [19], Leveraging Baging (LVB) [12], Adaptive Random Forest (ARF) [44], and Streaming Random Patches (SRP) [51].

Hoeffding Naive Bayes Trees (HNBT) [70], a variant of the original Hoeffding Tree [76], serve as the base learner for all ensemble learners. Although other base learners like NaiveBayes, Perceptron, and Rule-based algorithms were considered, HNBT exhibited significant improvements over those alternatives. As a result, only tree-based ensemble learners are employed in this work. The number of base learners is capped at 100 for all ensemble algorithms. OAUE builds new base learners every 500 instances (window length). In OSBoost, the γ parameter for updating the weights to the base learners is set to 0.1. The subspace size of ARF and SRP is 60% of the entire feature space. All experiments are conducted 10 times with different random seeds.

Noticeably, as mentioned in Section 6.2.2, the $\mathcal{K}$ selection can be global or windowed. Experiments with a global approach, and with window sizes of 5000 and 1000 have been conducted, showing only slight differences. Due to limited space we only present results for a window size of 5000 here.

For comparison purposes, we included the `BLAST` algorithm in our experiments. The parameter settings for `BLAST` followed the default settings presented on the MOA platform.

We also attempted to include the Dynamic Ensemble Selection for Imbalanced Data Streams with Concept Drift (DES-ICD) algorithm [81]. The corresponding code can be found on GitHub. However, due to the design of the DES-ICD algorithm, categorical feature values are not processable. Additionally, DES-ICD can only provide binary predictions. Therefore, we conducted

experiments for DES-ICD only on one-hot-encoded datasets with two class labels. However, only nine of the 18 datasets were suitable, leading us to exclude DES-ICD from our main experimental results. The completed tests for DES-ICD performed worse than the other approaches, indicating that algorithms specifically tailored for imbalanced data are likely to perform worse than the more general models.

6.4 Experimental Results and Discussion

This section illustrates experiments results in multiple perspectives and relevant discussion.

6.4.1 Overall Accuracy

Table 6.1 presents the overall accuracy of all the algorithms. The results are presented in the format of "Average Accuracy$_{(SD)}$", where the SD (standard deviation) is obtained from ten executions with different random seeds. Since OAUE, OSBoost, and `BLAST` are not randomizable, the standard deviation is omitted for them and their variants. The table includes 18 datasets and 11 algorithms. The best values in each row (dataset) are in bold, and underlines highlight cases where the DEMS variants improve upon the original methods.

`BLAST` achieves the best results in two datasets and outperforms OAUE and OSBoost in terms of average accuracy. However, its performance on other datasets is less competitive.

One can see that in Table 6.1, there are 74 out of 90 cases where DEMS has a positive impact on the original algorithms. Furthermore, in cases where the original algorithms produce higher accuracy than DEMS, the differences are rarely larger than 0.05%. On the contrary, the improvements from DEMS range from 0.012% to 16.963%, with an average increased accuracy of 2.162%. It is worth noting that these improvements are built on top of state-of-the-art ensemble algorithms, whose performance is already extremely competitive.

Table 6.1: Test-Then-Train Accuracy (%)

Algorithms	ARF	DEMS_{ARF}	SRP	DEMS_{SRP}	LVB	DEMS_{LVB}	OAUE	$\text{DEMS}_{\text{OAUE}}$	OSBoost	$\text{DEMS}_{\text{OSBoost}}$	BLAST
Datasets						Synthetic					
AGR_a	$88.618_{(0.268)}$	$90.331_{(0.245)}$	$92.907_{(0.177)}$	$\mathbf{94.176}_{(0.058)}$	$89.744_{(0.282)}$	$91.036_{(0.261)}$	90.879	92.16	90.956	91.85	82.483
AGR_g	$83.59_{(0.228)}$	$85.15_{(0.351)}$	$89.786_{(0.149)}$	$\mathbf{91.426}_{(0.041)}$	$86.088_{(0.073)}$	$86.658_{(0.055)}$	85.764	87.9	87.876	88.48	79.748
LED_a	$74.029_{(0.013)}$	$73.989_{(0.015)}$	$\mathbf{74.038}_{(0.013)}$	$74.002_{(0.018)}$	$73.892_{(0.022)}$	$73.917_{(0.015)}$	73.459	73.65	72.856	73.02	73.446
LED_g	$73.211_{(0.008)}$	$73.178_{(0.011)}$	$\mathbf{73.246}_{(0.013)}$	$73.201_{(0.019)}$	$73.222_{(0.013)}$	$73.184_{(0.014)}$	72.594	72.71	72.488	72.5	72.853
RBF_f	$77.471_{(0.018)}$	$79.488_{(0.018)}$	$76.381_{(0.025)}$	$77.037_{(0.032)}$	$59.798_{(0.042)}$	$71.733_{(0.084)}$	57.787	74.75	42.721	55.17	**84.031**
RBF_m	$87.444_{(0.024)}$	$87.605_{(0.024)}$	$86.041_{(0.016)}$	$86.076_{(0.014)}$	$85.119_{(0.037)}$	$86.854_{(0.02)}$	84.197	86.45	66.081	77.8	**89.356**
RTG_a	$83.102_{(0.156)}$	$\mathbf{84.501}_{(0.183)}$	$79.845_{(0.527)}$	$83.979_{(0.352)}$	$74.019_{(0.649)}$	$79.247_{(0.334)}$	73.791	77.19	67.237	68.94	69.428
SEA_a	$\mathbf{89.413}_{(0.014)}$	$89.387_{(0.02)}$	$88.761_{(0.151)}$	$89.276_{(0.021)}$	$87.86_{(0.037)}$	$89.054_{(0.022)}$	88.129	88.41	88.35	88.33	87.658
SEA_g	$\mathbf{89.002}_{(0.018)}$	$88.982_{(0.018)}$	$88.179_{(0.19)}$	$88.863_{(0.036)}$	$88.281_{(0.056)}$	$88.708_{(0.027)}$	87.833	87.97	88.101	88.08	87.352
Avg. Acc	82.86	83.534	83.332	**84.241**	79.828	82.218	79.409	82.367	75.155	78.229	80.706
Avg. Rank	4	4	3.67	**3**	6.67	5.11	8.56	6.11	8.89	8.11	7.89
						Real					
Airlines	$65.087_{(0.173)}$	$62.416_{(0.22)}$	$\mathbf{68.561}_{(0.032)}$	$68.005_{(0.089)}$	$63.509_{(0.016)}$	$63.488_{(0.014)}$	65.485	66.56	65.743	65.78	67.160
CovT_S	$85.407_{(0.073)}$	$\mathbf{88.273}_{(0.078)}$	$84.684_{(0.21)}$	$87.871_{(0.172)}$	$78.816_{(0.067)}$	$82.006_{(0.066)}$	75.785	77.58	71.486	71.44	71.421
CovT	$94.885_{(0.024)}$	$95.651_{(0.023)}$	$95.339_{(0.018)}$	$\mathbf{96.056}_{(0.011)}$	$93.366_{(0.029)}$	$94.575_{(0.04)}$	92.665	93.72	86.93	89.5	92.222
ElecN	$90.646_{(0.046)}$	$\mathbf{91.797}_{(0.055)}$	$89.725_{(0.131)}$	$91.274_{(0.095)}$	$89.173_{(0.05)}$	$91.065_{(0.105)}$	86.818	88.68	87.509	88.47	81.707
Elec	$88.048_{(0.072)}$	$\mathbf{90.074}_{(0.102)}$	$86.334_{(0.227)}$	$88.788_{(0.119)}$	$85.036_{(0.054)}$	$88.816_{(0.118)}$	83.106	86.62	83.901	85.25	81.179
KDD	$99.974_{(0.0)}$	$99.957_{(0.001)}$	$99.98_{(0.0)}$	$99.958_{(0.002)}$	$99.958_{(0.001)}$	$\mathbf{99.975}_{(0.0)}$	2.457	6.85	99.882	99.96	99.969
Nomao	$97.249_{(0.038)}$	$97.397_{(0.051)}$	$97.366_{(0.037)}$	$\mathbf{97.452}_{(0.044)}$	$96.336_{(0.049)}$	$96.861_{(0.064)}$	57.714	58.13	95.462	95.48	95.973
Sensor	$93.234_{(0.063)}$	$\mathbf{95.322}_{(0.065)}$	$91.96_{(0.083)}$	$94.733_{(0.029)}$	$89.73_{(0.015)}$	$91.974_{(0.008)}$	87.502	91.67	76.861	83.95	59.025
WISDM	$84.932_{(0.208)}$	$85.269_{(0.233)}$	$85.567_{(0.268)}$	$\mathbf{86.622}_{(0.258)}$	$80.491_{(0.194)}$	$82.154_{(0.281)}$	76.394	77.85	79.66	79.51	74.862
Avg. Acc	89.829	89.573	88.835	**90.084**	86.268	87.879	69.769	71.961	83.048	84.370	80.391
Avg. Rank	4	3.11	3.33	**2.11**	6.44	4.44	8.89	6.78	8.33	7.56	7.94
Avg. Acc	85.845	86.553	86.083	**87.163**	83.048	85.048	74.589	77.164	79.101	81.3	80.549
Avg. Rank	4	3.67	3.17	**2.33**	6.5	4.67	8.44	6.11	8.5	7.61	7.92

One can also observe that the best results are located in the DEMS modified algorithms in 13 out of 18 datasets, affirming the DEMS's ability to push the ensembles' performance upwards. In particular, DEMS_{SRP} achieves the best accuracy and the best ranking for synthetic data, real-world data, and overall across all datasets.

As explained in Section 2, Adaptive Random Forest (ARF), Streaming Random Patches (SRP), and Leveraging Bagging (LVB) construct their entire base learner group from the beginning of the streams, and the base learners are independent of each other. It is evident that ARF, SRP, and LVB exhibit significant improvements in accuracy. Online Smooth Boosting initializes all learners at once but updates them sequentially in an ordered manner. As DEMS includes a base learner sorting stage, its enhancement atop OSBoost is relatively constrained. On the other hand, Online Accuracy Updated Ensemble (OAUE) builds its ensemble members sequentially. The evaluation array of the performance of each possible K value in DEMS, i.e., the $\mathbb{R}$ in Algorithm 2, expands as the ensemble size grows for OAUE and similar algorithms. The accuracy of OAUE also experiences a significant enhancement. Therefore, this affirms that the capability of DEMS to improve on different types of ensemble strategies highlights the usability of the proposed methodology.

6.4.2 Critical Difference Diagram

We apply the Friedman test [38] to determine if there are any significant differences between the mean values of the populations with significance level of α = 0.1. Differences between populations are significant, if the difference of the mean rank is greater than the critical distance CD=3.292 of the Nemenyi test [27]. Therefore, we assume that there is a statistically significant difference between the median values of the populations. It is apparent that all the DEMS-modified methods outperform their original counterparts. SRP, ARF, and their variants group together and exhibit a significant difference from other algorithms, with DEMS_{SRP} yielding the best results and achieving

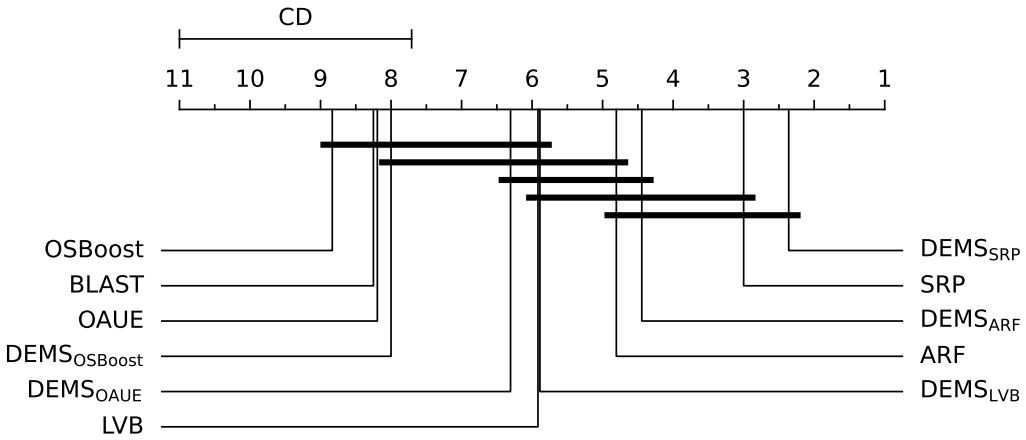


Figure 6.1: The Critical Difference Diagram across Five Methods, Associated DEMS Variants, and `BLAST`

a considerable advantage over the rest of the competitors.

6.4.3 Kappa Statistics

Table 6.2 presents the kappa statistic (See Section 2.2.3.1) results in a format similar to Table 6.1. In the table, the best results for each dataset are highlighted in bold, while improvements made by DEMS over the original models are underlined.

The kappa statistic results mirror those of accuracy. In 70 out of 90 cases, DEMS outperforms the original classifiers. Moreover, DEMS achieves the highest kappa values in 13 out of 18 datasets.

An intriguing observation from the kappa statistics is that none of the classifiers significantly outperform a random classifier on the Airlines dataset. This suggests that the inherent challenges of the Airlines dataset may be causing difficulties for all algorithms tested.

6.4.4 $\mathcal{K}$ Value Analysis

In this section, we illustrate the real-time results of the selected $\mathcal{K}$ values alongside the prequential accuracy. The results for all five original algorithms and their DEMS versions are included. Due to space limitations, only the

Table 6.2: Kappa Statistics (%)

Algorithms	ARF	DEMS_{ARF}	SRP	DEMS_{SRP}	LVB	DEMS_{LVB}	OAUE	$\text{DEMS}_{\text{OAUE}}$	OSBoost	$\text{DEMS}_{\text{OSBoost}}$
Datasets					Synthetic					
AGR_a	$77.195_{(0.535)}$	$\underline{80.556}_{(0.492)}$	$85.772_{(0.354)}$	$\underline{\mathbf{88.307}}_{(0.117)}$	$79.443_{(0.563)}$	$\underline{82.013}_{(0.526)}$	81.709	$\underline{84.26}$	81.842	$\underline{83.66}$
AGR_g	$67.039_{(0.456)}$	$\underline{70.074}_{(0.711)}$	$79.486_{(0.3)}$	$\underline{\mathbf{82.777}}_{(0.083)}$	$72.061_{(0.149)}$	$\underline{73.189}_{(0.112)}$	71.37	$\underline{75.67}$	75.662	$\underline{76.89}$
LED_a	$71.144_{(0.014)}$	$71.099_{(0.017)}$	$\mathbf{71.153}_{(0.014)}$	$71.114_{(0.02)}$	$70.991_{(0.024)}$	$\underline{71.019}_{(0.017)}$	70.509	$\underline{70.72}$	69.84	$\underline{70.02}$
LED_g	$70.234_{(0.009)}$	$70.198_{(0.012)}$	$\mathbf{70.273}_{(0.015)}$	$70.224_{(0.021)}$	$70.247_{(0.014)}$	$70.204_{(0.016)}$	69.548	$\underline{69.67}$	69.431	$\underline{69.45}$
RBF_f	$70.336_{(0.024)}$	$\underline{\mathbf{73.106}}_{(0.024)}$	$68.838_{(0.033)}$	$\underline{69.796}_{(0.042)}$	$47.069_{(0.056)}$	$\underline{63.055}_{(0.11)}$	43.706	$\underline{67.05}$	22.394	$\underline{41.5}$
RBF_m	$83.653_{(0.031)}$	$\underline{\mathbf{83.868}}_{(0.031)}$	$81.79_{(0.021)}$	$\underline{81.844}_{(0.019)}$	$80.632_{(0.049)}$	$\underline{82.916}_{(0.026)}$	79.456	$\underline{82.41}$	55.49	$\underline{71.21}$
RTG_a	$73.442_{(0.266)}$	$\underline{\mathbf{75.769}}_{(0.313)}$	$67.932_{(0.824)}$	$\underline{75.081}_{(0.589)}$	$60.376_{(0.897)}$	$\underline{67.675}_{(0.526)}$	59.704	$\underline{64.56}$	50.113	$\underline{52.08}$
SEA_a	$\mathbf{76.496}_{(0.032)}$	$76.462_{(0.042)}$	$74.98_{(0.347)}$	$\underline{76.218}_{(0.047)}$	$72.663_{(0.092)}$	$\underline{75.678}_{(0.047)}$	73.499	$\underline{74.26}$	73.987	73.94
SEA_g	$\mathbf{75.54}_{(0.042)}$	$\underline{\mathbf{75.54}}_{(0.039)}$	$73.586_{(0.451)}$	$\underline{75.288}_{(0.086)}$	$73.711_{(0.139)}$	$\underline{74.869}_{(0.061)}$	72.793	$\underline{73.21}$	73.404	73.36
					Real					
Airlines	$28.106_{(0.342)}$	$20.49_{(0.594)}$	$\mathbf{35.124}_{(0.064)}$	$33.702_{(0.216)}$	$24.969_{(0.032)}$	$24.919_{(0.024)}$	27.359	$\underline{30.34}$	29.825	29.75
CovT_S	$76.143_{(0.118)}$	$\underline{\mathbf{81.057}}_{(0.127)}$	$74.807_{(0.354)}$	$\underline{80.377}_{(0.283)}$	$65.161_{(0.108)}$	$\underline{70.897}_{(0.11)}$	60.066	$\underline{63.52}$	54.281	54.23
CovT	$91.762_{(0.039)}$	$\underline{93.014}_{(0.038)}$	$92.501_{(0.029)}$	$\underline{\mathbf{93.663}}_{(0.017)}$	$89.341_{(0.046)}$	$\underline{91.291}_{(0.064)}$	88.213	$\underline{89.94}$	79.073	$\underline{83.2}$
ElecN	$80.784_{(0.093)}$	$\underline{\mathbf{83.28}}_{(0.111)}$	$78.863_{(0.268)}$	$\underline{82.184}_{(0.19)}$	$77.685_{(0.102)}$	$\underline{81.732}_{(0.214)}$	72.917	$\underline{76.89}$	74.328	$\underline{76.46}$
Elec	$75.399_{(0.151)}$	$\underline{\mathbf{79.534}}_{(0.213)}$	$71.721_{(0.477)}$	$\underline{76.879}_{(0.25)}$	$69.031_{(0.114)}$	$\underline{76.941}_{(0.246)}$	65.006	$\underline{72.45}$	66.802	$\underline{69.73}$
KDD	$99.955_{(0.001)}$	$99.927_{(0.002)}$	$99.967_{(0.001)}$	$99.928_{(0.004)}$	$99.929_{(0.002)}$	$\underline{99.957}_{(0.001)}$	1.974	$\underline{5.48}$	99.798	$\underline{\mathbf{99.93}}$
Nomao	$93.252_{(0.093)}$	$\underline{93.652}_{(0.122)}$	$93.54_{(0.091)}$	$\underline{\mathbf{93.785}}_{(0.108)}$	$90.909_{(0.128)}$	$\underline{92.312}_{(0.156)}$	25.24	$\underline{26.03}$	88.822	$\underline{88.9}$
Sensor	$93.089_{(0.064)}$	$\underline{\mathbf{95.221}}_{(0.066)}$	$91.787_{(0.084)}$	$\underline{94.62}_{(0.03)}$	$89.51_{(0.015)}$	$\underline{91.802}_{(0.008)}$	87.234	$\underline{91.49}$	76.367	$\underline{83.61}$
WISDM	$79.005_{(0.296)}$	$\underline{79.547}_{(0.328)}$	$79.832_{(0.389)}$	$\underline{\mathbf{81.409}}_{(0.377)}$	$72.642_{(0.278)}$	$\underline{75.187}_{(0.402)}$	66.438	$\underline{68.64}$	71.647	71.41

AGR_g dataset is exhibited in this article.

In Figure 6.2 – 6.6, the green lines present the $\mathcal{K}$ values over time, while the orange and blue lines depict the original ensembles and their DEMS variants' prequential accuracy, respectively.

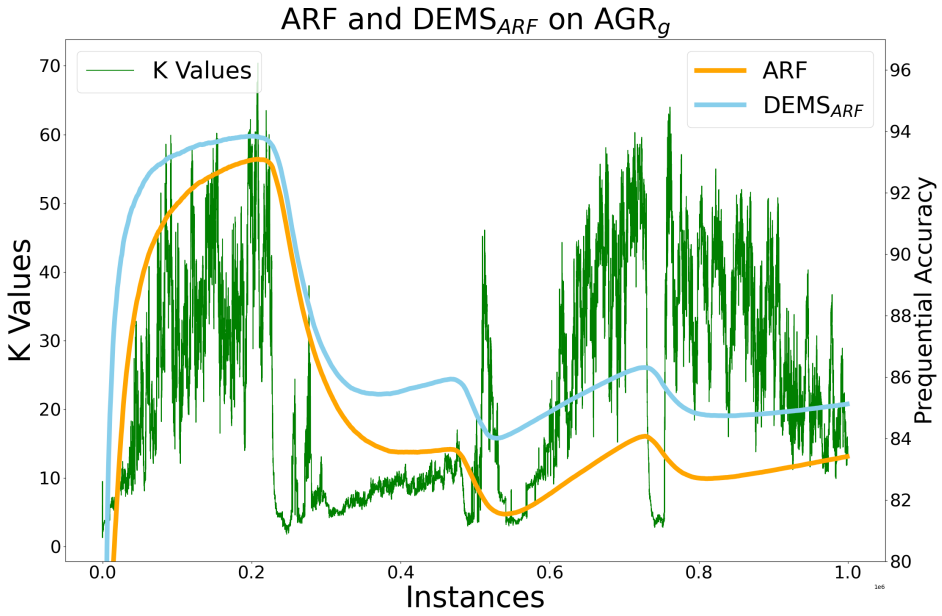


Figure 6.2: AGR_g dataset: prequential accuracy for ARF and $DEMS_{ARF}$, and best K values over time

In each of these five figures, three accuracy drops are evident, always aligning with the simulated concept drifts. Interestingly, the $\mathcal{K}$ values also tend to respond to these concept drifts. This leads to a quicker and more effective recovery for DEMS in terms of accuracy.

For instance, in Figure 6.4, before the first drift (around 250K instances), the $\mathcal{K}$ values are close to the ensemble size, leading to similar performance of the LVB and $DEMS_{LVB}$. When the first drift occurs, the accuracy for both approaches drops, and the $\mathcal{K}$ values also decrease suddenly. This phenomenon indicates that the selection mechanism detects the drift based on $\mathcal{A}$ and/or $\mathcal{M}$, and thus starts predicting with only a small subset of the ensemble members that are robust to this drift. The blue line ($DEMS_{LVB}$ accuracy) stops dropping after a short period, while the orange line (LVB accuracy) continues

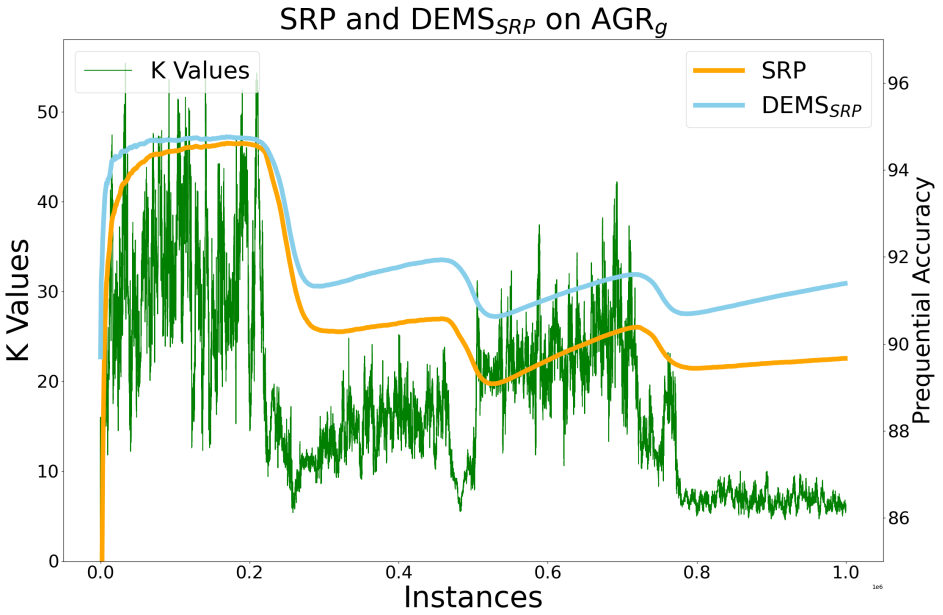


Figure 6.3: AGR$_g$ dataset: prequential accuracy for SRP and DEMS$_{\text{SRP}}$, and best K values over time

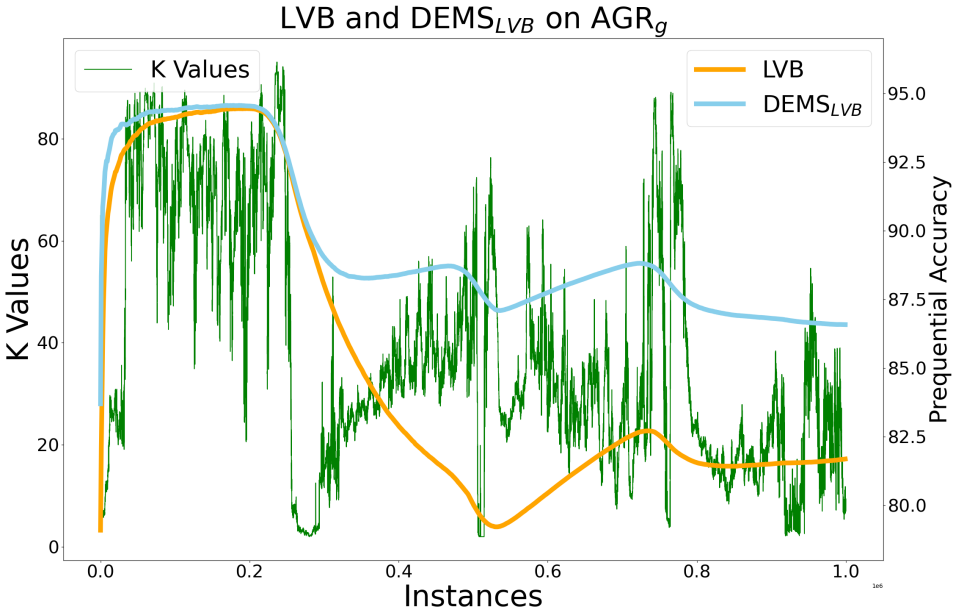


Figure 6.4: AGR$_g$ dataset: prequential accuracy for LVB and DEMS$_{\text{LVB}}$, and best K values over time

to diminish for a much longer time. A similar trend can also be observed in Figure 6.3. After the third drift (around 750K instances), the SRP accuracy

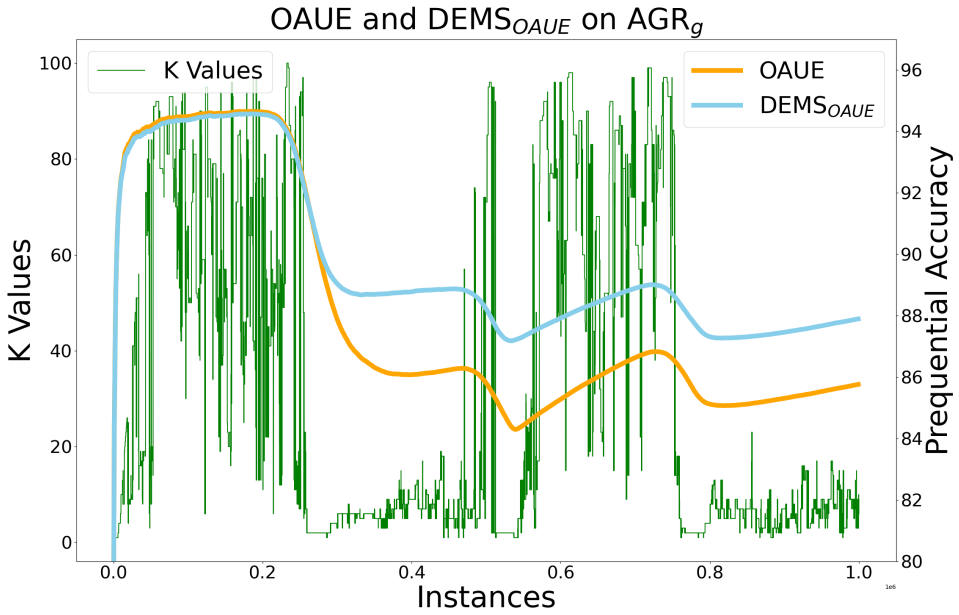


Figure 6.5: AGR$_g$ dataset: prequential accuracy for OAUE and DEMS$_{\text{OAUE}}$, and best K values over time

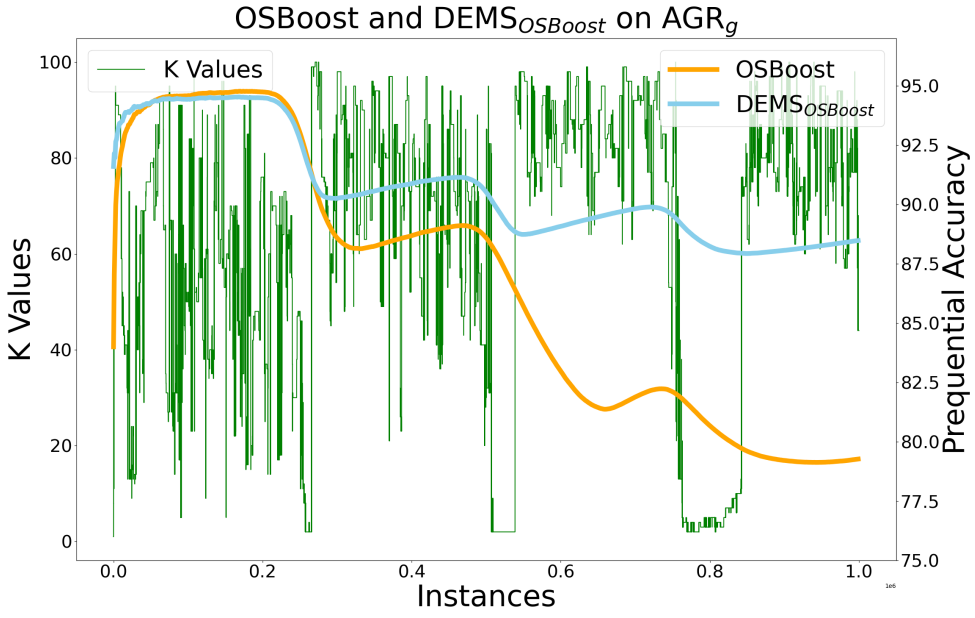


Figure 6.6: AGR$_g$ dataset: prequential accuracy for OSBoost and DEMS$_{\text{OSBoost}}$, and best K values over time

(orange line) remains at a similar level until the end of the stream, whereas the DEMS$_{\text{SRP}}$ performance (blue line) recovers from the drift and starts improving

by obtaining predictions from a tiny subset (fewer than 10) of the ensemble members.

Consequently, it can be concluded that DEMS is capable of enhancing the performance of ensemble learners, especially in the presence of concept drifts.

6.4.5 Diversity Analysis

This section provides an in-depth analysis of ensemble diversity.

6.4.5.1 Fixed Ensemble Size

In order to gain a more profound understanding of the impact of diversity on ensemble learning, we conducted an extra set of experiments to observe and analyze the effect of diversity on model performance. We computed the average $\mathcal{K}$ values selected by DEMS per algorithm per dataset, and used these averages as the fixed ensemble size for running the experiments. This configuration will henceforth be referred to as the "Fixed" set.

Due to the space limitations, we only present a comparison between the fixed set, the full-size set, and the DEMS set for ARF, SRP, and OSBoost algorithms in Table 6.3. We chose these three algorithms, as ARF and SRP were the best-performing ones according to Table 6.1, whereas OSBoost showed a lot more variation in behaviour. The $N_{\mathcal{E}}$ columns contain the ensemble size of the fixed experiment sets. Following that, accuracy from the fixed-size, full-size, and DEMS version are presented sequentially. The highest accuracies on each dataset within the same algorithm (ARF, SRP, or OSBoost) are highlighted in bold.

In the ARF and SRP sections, almost all selected $N_{\mathcal{E}}$ values range from 1 to 40, indicating that DEMS tends to favor a small subset of the ensemble. On the contrary, for OSBoost, DEMS often chooses larger subset sizes. Another observation is that, in most cases, DEMS outperforms the fixed-size and full-size setting. Similarly, the full-size results are generally better than the fixed-size ones. Even though there are exceptions, such as for the KDD or the SEA-

Table 6.3: Comparison among Fixed-size, Full-size, and DEMS variants of ARF, SRP, and OSBoost

Datasets	ARF				SRP				OSBoost		
	$N_{\mathcal{E}}$	Fixed-Size	Full-Size	DEMS	$N_{\mathcal{E}}$	Fixed-Size	Full-Size	DEMS	$N_{\mathcal{E}}$	Fixed-Size	Full-Size	DEMS
AGR_a	21	87.92$_{(0.55)}$	88.62$_{(0.27)}$	**90.03$_{(0.25)}$**	21	92.03$_{(1.00)}$	92.91$_{(0.18)}$	**94.18$_{(0.06)}$**	63	88.80	90.96	**91.85**
AGR_g	23	82.87$_{(0.68)}$	83.59$_{(0.23)}$	**85.15$_{(0.35)}$**	21	88.49$_{(0.52)}$	89.79$_{(0.15)}$	**91.43$_{(0.04)}$**	64	85.04	87.88	**88.48**
Airlines	30	64.91$_{(0.20)}$	**65.09$_{(0.17)}$**	62.42$_{(0.22)}$	35	**68.20$_{(0.06)}$**	**68.56$_{(0.03)}$**	68.01$_{(0.09)}$	42	**66.24**	65.74	65.78
$CovT_S$	14	84.85$_{(0.15)}$	85.41$_{(0.07)}$	**88.27$_{(0.08)}$**	12	83.57$_{(0.58)}$	84.68$_{(0.21)}$	**87.87$_{(0.17)}$**	97	**71.58**	71.49	71.44
CovT	20	94.89$_{(0.04)}$	94.98$_{(0.02)}$	**95.65$_{(0.02)}$**	19	95.55$_{(0.02)}$	95.34$_{(0.02)}$	**96.06$_{(0.01)}$**	17	84.64	86.93	**89.50**
ElecN	14	90.23$_{(0.12)}$	90.65$_{(0.05)}$	**91.80$_{(0.06)}$**	13	88.98$_{(0.29)}$	89.73$_{(0.73)}$	**91.27$_{(0.10)}$**	12	83.76	87.51	**88.47**
Elec	7	87.12$_{(0.18)}$	88.05$_{(0.07)}$	**90.07$_{(0.10)}$**	7	84.54$_{(0.91)}$	86.33$_{(0.23)}$	**88.79$_{(0.12)}$**	19	80.00	83.90	**85.25**
KDD	2	**99.97$_{(0.00)}$**	**99.97$_{(0.00)}$**	99.96$_{(0.00)}$	2	99.97$_{(0.00)}$	**99.98$_{(0.00)}$**	99.96$_{(0.00)}$	4	99.91	99.88	**99.96**
LED_a	34	**74.02$_{(0.01)}$**	**74.03$_{(0.01)}$**	73.99$_{(0.02)}$	40	74.00$_{(0.02)}$	**74.04$_{(0.01)}$**	74.00$_{(0.02)}$	44	72.31	72.86	**73.02**
LED_g	34	73.19$_{(0.01)}$	**73.21$_{(0.01)}$**	73.18$_{(0.01)}$	42	73.21$_{(0.02)}$	**73.25$_{(0.01)}$**	73.20$_{(0.02)}$	51	72.21	72.49	**72.50**
Nomao	13	97.15$_{(0.07)}$	97.25$_{(0.04)}$	**97.40$_{(0.05)}$**	14	97.30$_{(0.04)}$	97.37$_{(0.04)}$	**97.45$_{(0.04)}$**	64	95.41	95.46	**95.48**
RBF_f	24	75.89$_{(0.04)}$	77.47$_{(0.02)}$	**79.49$_{(0.02)}$**	30	75.42$_{(0.04)}$	76.38$_{(0.03)}$	**77.04$_{(0.03)}$**	4	36.11	42.72	**55.17**
RBF_m	40	87.22$_{(0.03)}$	87.44$_{(0.02)}$	**87.61$_{(0.02)}$**	47	**86.10$_{(0.02)}$**	86.04$_{(0.02)}$	86.08$_{(0.01)}$	4	54.32	66.08	**77.80**
RTG_a	23	82.97$_{(0.26)}$	83.10$_{(0.16)}$	**84.50$_{(0.18)}$**	15	79.15$_{(0.95)}$	79.85$_{(0.53)}$	**83.98$_{(0.35)}$**	46	61.17	67.24	**68.94**
SEA_a	56	**89.41$_{(0.01)}$**	**89.41$_{(0.01)}$**	89.39$_{(0.02)}$	38	88.86$_{(0.07)}$	88.76$_{(0.15)}$	**89.28$_{(0.02)}$**	49	88.27	**88.35**	88.33
SEA_g	51	**89.00$_{(0.02)}$**	**89.00$_{(0.02)}$**	88.98$_{(0.02)}$	36	87.90$_{(0.55)}$	88.18$_{(0.19)}$	**88.86$_{(0.04)}$**	44	88.06	**88.10**	88.08
Sensor	24	94.65$_{(0.16)}$	93.23$_{(0.06)}$	**95.32$_{(0.07)}$**	19	92.18$_{(0.13)}$	91.96$_{(0.08)}$	**94.73$_{(0.03)}$**	17	70.10	76.86	**83.95**
WISDM	19	85.23$_{(0.39)}$	84.93$_{(0.21)}$	**85.27$_{(0.23)}$**	18	86.34$_{(0.28)}$	85.57$_{(0.27)}$	**86.62$_{(0.26)}$**	70	78.55	**79.66**	79.51

based datasets, these differences in performance are marginal. In summary, DEMS generally outperforms the original full-size algorithms, which again outperform the respective fixed-size versions.

6.4.5.2 Entropy

We now proceed to present example plots of the entropy analysis. Due to space constraints, only a select number of plots are displayed, as the remaining plots exhibit similar patterns to those presented. The rest of the figures can be seen in Appendix B. For consistence, we include entropy results for ARF, SRP, and OSBoost on the AGR_g dataset, which was chosen for the analysis of $\mathcal{K}$ values in Section 6.4.4. Similar results on the CovT dataset are also presented here as an example for the real-world datasets. The results are illustrated in Figures 6.7 to 6.12.

An entropy value is calculated for each prediction from the different algorithms (origin, DEMS, and fixed), and an average entropy within a fixed-size window is computed. As mentioned earlier in Section 6.2.5, a higher entropy value indicates a greater diversity of votes.

In this set of figures, the x-axis represents the window indices, while the y-axis shows the average entropy calculated from the predictive votes. The original algorithms, fixed ensemble size algorithms, and DEMS algorithms are shown in red, blue, and green, respectively.

An apparent observation is that the entropy results for the fixed-size and full-size ensembles are quite similar, indicating that reducing the ensemble size has only a limited impact on the diversity of the ensemble. Whereas, the performance of the full-size is evidently better than fixed ones. This suggests that diversity is not a determining factor for better performance.

However, since the DEMS strategy filters the ensemble based on the predictive margin (see Section 6.2.2), the entropy for the DEMS version is predictably lower than that of the original algorithms.

The impact of DEMS on diversity can be summarized as follows: it main-

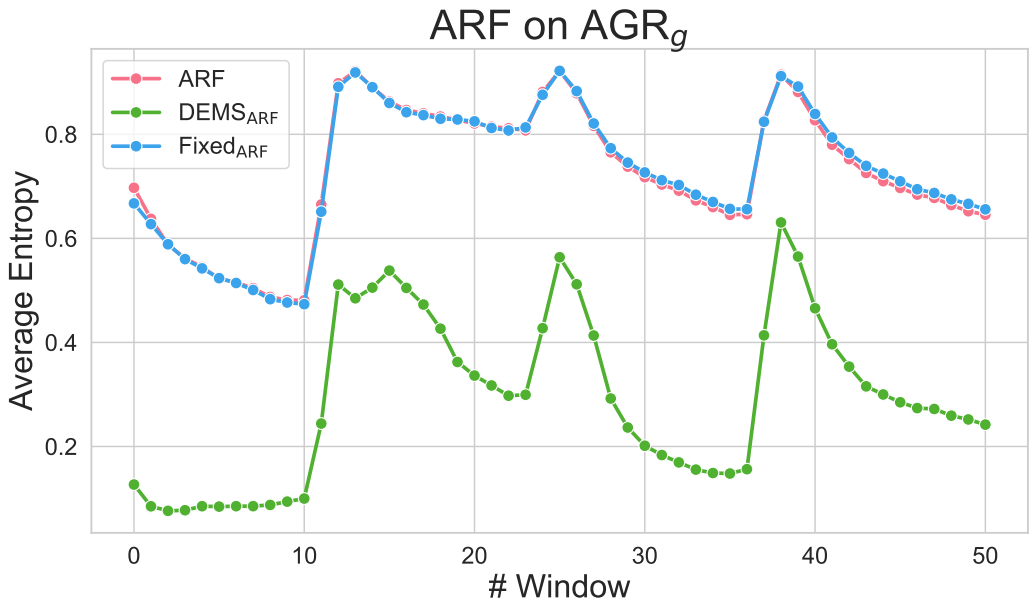


Figure 6.7: Windowed Entropy of Predictions for ARF related algorithms on AGR$_g$ Dataset

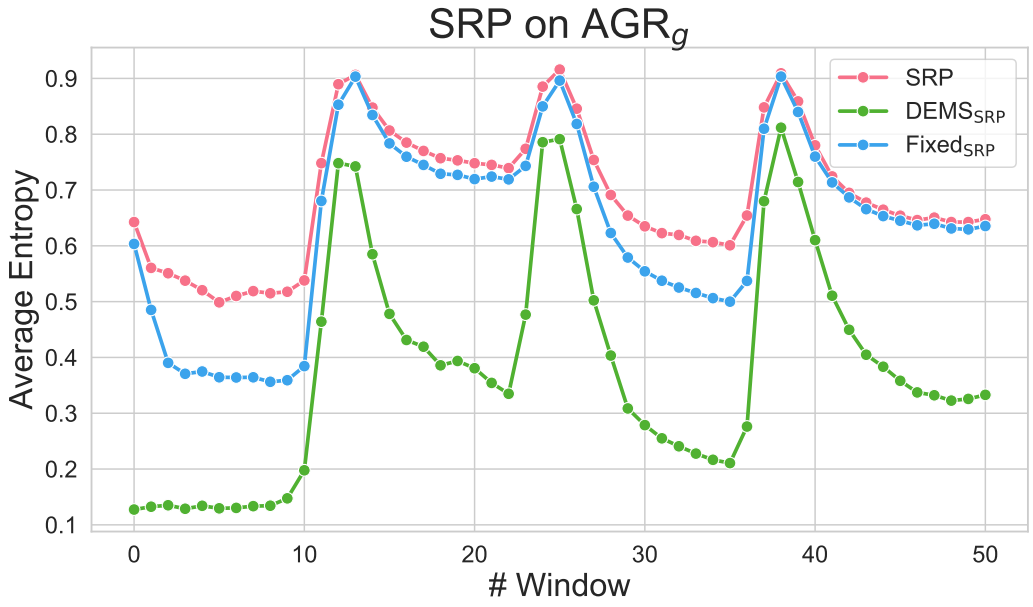


Figure 6.8: Windowed Entropy of Predictions for SRP related algorithms on AGR$_g$ Dataset

tains diversity at the ensemble growth stage, as it does not alter the process of building ensemble members. However, it reduces diversity at the prediction stage by selecting more "concentrated" members with higher accuracy,

Figure 6.9: Windowed Entropy of Predictions for OSBoost related algorithms on AGR$_g$ Dataset

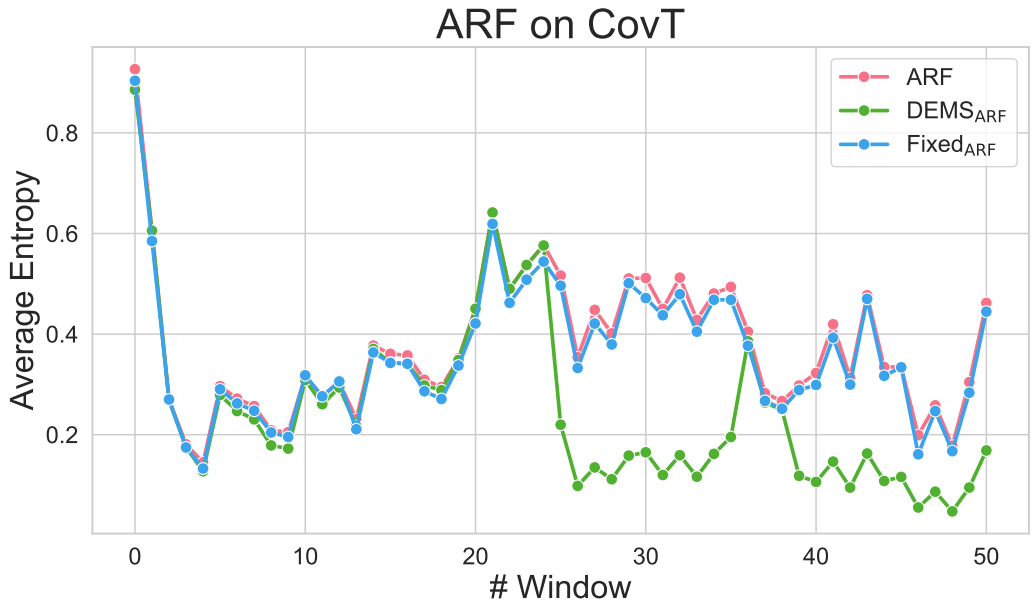


Figure 6.10: Windowed Entropy of Predictions for ARF related algorithms on CovT Dataset

resulting in more "reliable" predictions.

For example, focusing on the ARF algorithm applied to the AGR$_g$ dataset, we observe that the full-size ARF achieves an accuracy of 88.62%, while the

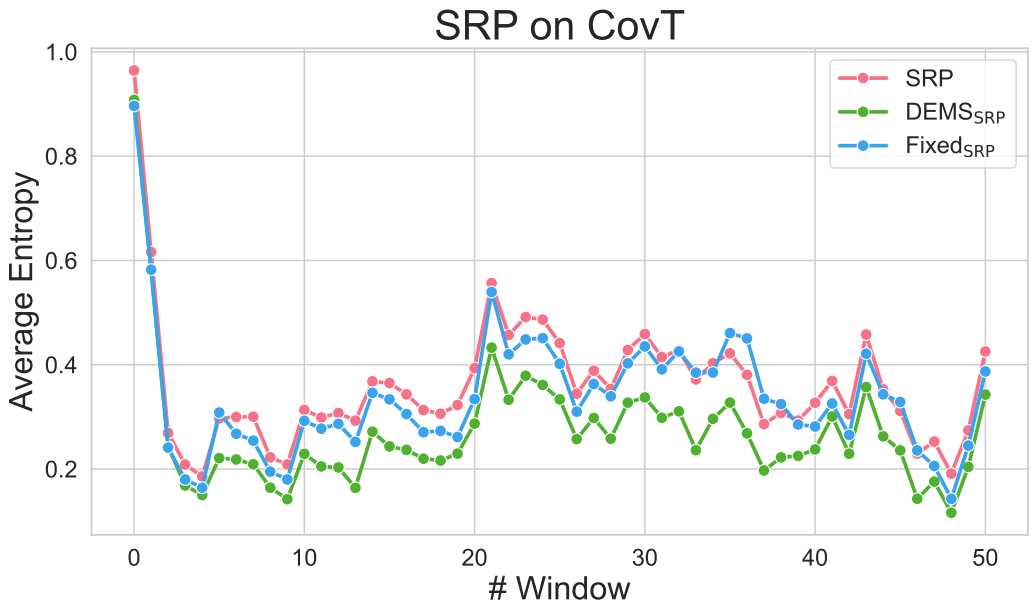


Figure 6.11: Windowed Entropy of Predictions for SRP related algorithms on CovT Dataset

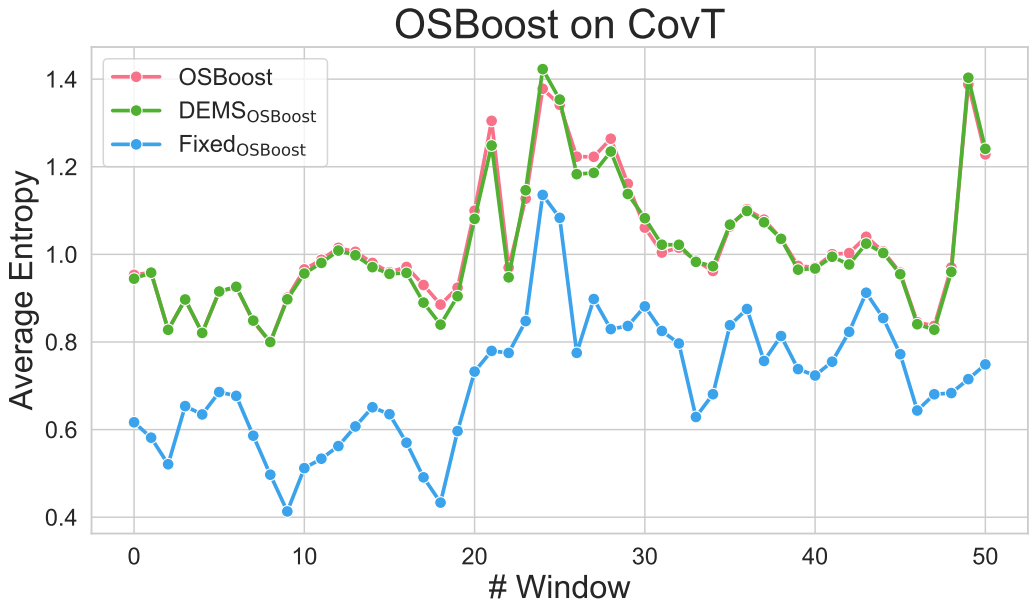


Figure 6.12: Windowed Entropy of Predictions for OSBoost related algorithms on CovT Dataset

fixed setting with an ensemble size of 21 reaches 87.92%. In comparison, DEMS$_{ARF}$ achieves a higher accuracy of 90.33%. The full and fixed settings show minimal performance differences in entropy. However, DEMS outper-

forms both by reducing entropy at the prediction level, contributing to improved accuracy.

We can also observe behavior while dealing with concept drifts. At each drift point, entropy rises rapidly due to the uncertain predictions made by the models when facing a new concept. Once the models adapt to the new concept, entropy decreases as expected, indicating that the ensemble members have learned the new concept and are now more "confident" in their predictions.

In Figure 6.7, it is evident that the entropy for DEMS_{ARF} (green line) decreases more dramatically. Using the first drift as an example and coordinating with Figure 6.2, we see that during the first accuracy drop, DEMS starts selecting a much smaller subset of ensemble members, significantly reducing entropy. Our assumption is that most ensemble members lose accuracy during the drift, while only a small subset has structures better suited to learning the new concept. Thus, using smaller $\mathcal{K}$ values helps recover accuracy more quickly, as confirmed by Figure 6.2.

The results for OSBoost do not necessarily show the same trends. Figure 6.9 shows that the entropy values from the full, DEMS, and fixed versions of the OSBoost algorithm are rather similar. Figure 6.12 shows yet another pattern, where the fixed-size setting yields the lowest entropy values.

In Figure 6.9, the fixed-size setting utilizes a relatively high ensemble size (64), whereas Figure 6.12 employs a much smaller size (17). Notably, the entropy of the fixed-size setting in Figure 6.9 closely aligns with that of the full-size version, while in Figure 6.12, the entropy of the fixed-size is significantly lower than the full-size. This suggests that a larger ensemble size in OSBoost contributes to greater diversity.

Furthermore, the dynamic selection of DEMS introduces a degree of randomness when applied to algorithms like OSBoost, where the ensemble members are not independent. Consequently, no significant differences in entropy between DEMS and the full-size approaches are observed on the CovT and AGR_g datasets. Additional entropy results for OSBoost, which are not in-

cluded in this paper, similarly showed no consistent patterns or uniform behaviour.

6.4.6 Runtime

Table 6.4 presents the runtime for all experiments. Similar to Table 6.1, the values in the cells represent the mean and SD from 10 executions, following the format "$\text{Mean}_{(\text{SD})}$." The bottom row indicates the "average overhead" (Avg. Overhead) for runtime comparison between the original algorithms and their DEMS variants.

This overhead is calculated as the extra runtime of the DEMS (T_{DEMS}) compared to the original algorithms runtime (T_{origin}), i.e.,

$$\text{Overhead} = \frac{T_{\text{DEMS}} - T_{\text{origin}}}{T_{\text{origin}}}$$

The average overhead is the mean value calculated across 18 datasets for each algorithm.

The results indicate that the average extra runtime that DEMS adds to original methods ranges from 4.7% (OSBoost) to 23.6% (SRP). For streaming learning, this increase in runtime remains within an acceptable range. Moreover, the current DEMS implementation still leaves some room for further efficiency improvements.

Table 6.4: Test-Then-Train Runtime (seconds)

Datasets	ARF	DEMS_{ARF}	SRP	DEMS_{SRP}	LVB	DEMS_{LVB}	OAUE	$\text{DEMS}_{\text{OAUE}}$	OSBoost	$\text{DEMS}_{\text{OSBoost}}$
AGR_a	$1819_{(37)}$	$1858_{(51)}$	$2756_{(63)}$	$3956_{(182)}$	$814_{(22)}$	$1016_{(64)}$	$478_{(4)}$	$601_{(23)}$	$330_{(20)}$	$469_{(21)}$
AGR_g	$2083_{(37)}$	$2355_{(78)}$	$3231_{(70)}$	$4304_{(127)}$	$883_{(16)}$	$1041_{(33)}$	$596_{(17)}$	$667_{(9)}$	$354_{(11)}$	$446_{(27)}$
LED_a	$1925_{(39)}$	$1995_{(262)}$	$2909_{(78)}$	$3571_{(271)}$	$1609_{(58)}$	$1519_{(101)}$	$1408_{(77)}$	$2203_{(91)}$	$1351_{(101)}$	$1363_{(124)}$
LED_g	$1898_{(44)}$	$2259_{(299)}$	$3000_{(120)}$	$3970_{(316)}$	$1577_{(66)}$	$1694_{(99)}$	$1482_{(89)}$	$1585_{(121)}$	$1429_{(144)}$	$1473_{(137)}$
RBF_f	$1639_{(23)}$	$1819_{(143)}$	$3107_{(53)}$	$3915_{(173)}$	$955_{(8)}$	$888_{(64)}$	$661_{(45)}$	$847_{(33)}$	$655_{(66)}$	$727_{(65)}$
RBF_m	$1646_{(34)}$	$1612_{(189)}$	$2834_{(50)}$	$3431_{(261)}$	$1291_{(25)}$	$1381_{(163)}$	$949_{(66)}$	$1231_{(90)}$	$717_{(80)}$	$671_{(21)}$
RTG_a	$312_{(6)}$	$303_{(8)}$	$531_{(11)}$	$578_{(47)}$	$247_{(3)}$	$236_{(34)}$	$125_{(31)}$	$151_{(11)}$	$138_{(22)}$	$192_{(10)}$
SEA_a	$84_{(2)}$	$90_{(13)}$	$138_{(6)}$	$171_{(27)}$	$42_{(3)}$	$54_{(12)}$	$21_{(1)}$	$23_{(2)}$	$15_{(0)}$	$14_{(0)}$
SEA_g	$85_{(1)}$	$93_{(13)}$	$140_{(2)}$	$176_{(25)}$	$41_{(2)}$	$54_{(11)}$	$21_{(1)}$	$20_{(0)}$	$15_{(0)}$	$14_{(0)}$
Airlines	$3237_{(82)}$	$3830_{(53)}$	$2623_{(39)}$	$3633_{(72)}$	$2716_{(41)}$	$2965_{(78)}$	$577_{(31)}$	$717_{(44)}$	$334_{(20)}$	$475_{(21)}$
CovT_S	$2436_{(110)}$	$2397_{(189)}$	$2558_{(46)}$	$3053_{(95)}$	$1565_{(91)}$	$1638_{(171)}$	$1149_{(92)}$	$1161_{(78)}$	$954_{(66)}$	$931_{(90)}$
CovT	$2030_{(55)}$	$2166_{(275)}$	$3267_{(69)}$	$4123_{(492)}$	$1332_{(22)}$	$1326_{(137)}$	$1047_{(103)}$	$911_{(75)}$	$926_{(78)}$	$799_{(56)}$
ElecN	$79_{(2)}$	$85_{(5)}$	$124_{(5)}$	$147_{(4)}$	$37_{(1)}$	$40_{(1)}$	$10_{(0)}$	$10_{(1)}$	$12_{(1)}$	$13_{(1)}$
Elec	$67_{(3)}$	$67_{(2)}$	$135_{(6)}$	$154_{(6)}$	$26_{(1)}$	$26_{(0)}$	$8_{(0)}$	$8_{(0)}$	$10_{(0)}$	$10_{(0)}$
KDD	$3549_{(79)}$	$3384_{(68)}$	$7358_{(107)}$	$8488_{(233)}$	$3780_{(211)}$	$4376_{(245)}$	$3711_{(243)}$	$4081_{(389)}$	$9524_{(228)}$	$8674_{(442)}$
Nomao	$189_{(8)}$	$217_{(26)}$	$326_{(14)}$	$403_{(48)}$	$151_{(3)}$	$197_{(35)}$	$43_{(4)}$	$84_{(7)}$	$75_{(9)}$	$77_{(8)}$
Sensor	$7677_{(117)}$	$7450_{(358)}$	$11640_{(293)}$	$14566_{(533)}$	$7750_{(162)}$	$6339_{(471)}$	$5581_{(81)}$	$4387_{(103)}$	$5860_{(170)}$	$4845_{(156)}$
WISDM	$31_{(1)}$	$33_{(3)}$	$43_{(1)}$	$48_{(4)}$	$24_{(0)}$	$23_{(1)}$	$1_{(0)}$	$1_{(0)}$	$16_{(1)}$	$12_{(1)}$
Avg. Overhead	5.8%		23.6%		9.0%		15.6%		4.7%	

Chapter 7

Conclusions

This thesis has contributed novel methodologies for addressing some of the challenges in streaming learning. The Self-Optimising K Nearest Leaves (SOKNL) algorithm successfully improved regression accuracy by dynamically combining the strengths of `kNN` and ARF-Reg and efficiently abstracting information from the processed data points from the data streams.

The Adaptive Prediction Interval (`AdaPI`) framework introduced a versatile solution for quantifying uncertainty, ensuring robust and accurate `PI` boundaries under evolving data streams. In addition, a novel evaluation framework for Prediction Interval tasks, namely Coverage Interval width in Non-dominated Group (CING), is proposed to tackle the difficulty in balancing the two necessary metrics in `PI` evaluation using Multi-Objective Optimization techniques.

The introduction of the NZEP datasets has provided valuable resources for real-world energy analytics research. Our research includes a comprehensive analysis of these datasets, leading to two key conclusions: (1) the datasets are highly suitable for streaming learning (SL) scenarios, making them strong candidates for benchmarking datasets for streaming research; and (2) current SL algorithms are capable of generating accurate predictions on this real-world, constantly evolving data, further confirming the effectiveness of SL.

Furthermore, the Dynamic Ensemble Member Selection (DEMS) approach

offered significant advancements in stream classification by adaptively selecting ensemble members, showcasing resilience to concept drift while maintaining computational efficiency.

Through extensive empirical validation, this work demonstrated the practical utility and robustness of the proposed methods across multiple domains, paving the way for further research into adaptive, real-time data analytics.

7.1 Future Work

In this section, we outline several potential directions for future research.

7.1.1 Self-Optimising K Nearest Leaves

Exploring similar KNN integration with other streaming learners presents an interesting avenue for research. Furthermore, SOKNL currently disregards nominal features during node splitting, inherited from ARF-Reg. Implementing a nominal feature split function could enhance regression performance.

7.1.2 Adaptive Prediction Interval

Firstly, the same adaptation principles could be applied to other static prediction interval algorithms, although the practical impact of these modifications would require thorough evaluation. Secondly, the factor curve presented in this paper appears relatively moderate and conservative; a more aggressive and radical scaling strategy might yield even better results.

7.1.3 New Zealand Energy Pricing

For NZEP, we will focus on improving preprocessing techniques, exploring additional machine learning models, and enhancing the dataset's applicability to other regions and energy markets. Introducing other information source, like meteorological, into the current datasets would be another intriguing attempt.

Research in time-series abstraction of the NZEP data is also a promising direction.

7.1.4 Dynamic Ensemble Member Selection

Future work for DEMS include: 1. adding different ensemble learners to the experiments, actually pruning less frequently used ensemble members to achieve better results and runtime; 2. exploiting active drift detection and adaptation; and, 3. devising order-preserving DEMS variants for boosting-based ensembles.

Appendix A

Chapter 4 Additional Results

A.1 Full Coverage and NMPIW Results

We present the full coverage and NMPIW results in this appendix.

Table A.1: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 80\%$ and Limit $\mathcal{M} = 0.05$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	85.83	20.99	85.79	19.17	85.15	20.95
HLI	85.63	20.84	85.63	18.98	85.19	21.05
Linear	83.2	19.01	**83.0**	**17.45**	83.28	19.57
MVE	87.98	22.87	88.13	20.86	86.3	22.11
Abalone						
Logarithm	84.2	24.0	91.69	14.11	93.38	13.9
HLI	84.14	23.78	89.94	12.67	91.2	12.06
Linear	**81.53**	22.76	86.74	11.12	88.31	**10.24**
MVE	89.5	26.51	94.95	17.65	96.92	19.84
CPU Activity						
Logarithm	87.84	7.17	99.66	0.68	99.45	0.91
HLI	87.24	6.65	99.51	0.29	98.16	0.52
Linear	**83.28**	6.27	99.12	**0.28**	96.82	0.5
MVE	89.24	8.66	99.99	4.43	99.99	4.46
Naval Propulsion Plant						
Logarithm	86.04	18.09	84.72	17.82	83.31	14.44
HLI	85.85	17.98	84.65	17.82	83.46	14.56
Linear	83.24	16.38	82.18	16.38	**81.75**	**13.79**
MVE	87.7	19.37	86.44	18.77	84.24	14.91
Ailerons						
Logarithm	92.77	16.26	92.71	15.41	92.85	13.78
HLI	92.24	15.29	92.31	14.59	92.21	12.97
Linear	**90.36**	13.23	90.57	12.7	90.37	**11.19**
MVE	94.33	19.3	94.09	18.04	94.39	16.34
Miami Housing						
Logarithm	91.5	2.68	92.93	2.27	90.86	2.42
HLI	91.07	2.55	92.12	2.07	90.44	2.32
Linear	89.13	2.18	90.2	**1.74**	**88.3**	2.02
MVE	93.11	3.08	94.75	2.75	92.36	2.75
NZ Energy Price						
Logarithm	92.11	3.42	92.39	3.4	91.05	3.98
HLI	90.61	3.14	90.96	3.11	89.78	3.71
Linear	87.52	2.68	88.08	**2.66**	**86.85**	3.19
MVE	94.62	18.12	94.82	4.14	93.38	4.69
Elevators						
Logarithm	80.7	18.73	79.67	17.48	**79.79**	28.97
HLI	80.62	18.75	78.73	**16.87**	76.6	26.38
Linear	80.25	18.51	79.04	17.13	78.22	27.74
MVE	80.83	18.88	78.92	16.91	75.49	25.56
Bike						
Logarithm	83.64	28.22	83.44	24.74	82.65	27.5
HLI	83.75	28.4	83.68	24.84	82.79	27.65
Linear	82.16	26.47	82.03	**23.33**	**81.26**	26.11
MVE	84.82	29.76	84.87	26.1	83.88	28.92
California Housing						
Logarithm	87.68	29.48	87.61	29.08	89.32	24.35
HLI	87.42	29.3	87.34	28.79	89.22	23.97
Linear	86.15	26.9	**86.08**	26.5	87.59	**21.49**
MVE	88.19	30.55	87.9	29.96	90.28	25.95
Superconductivity						

Table A.2: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 80% and Limit $\mathcal{M} = 0.05$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	80.92	50.38	**79.87**	51.42	79.79	46.52
HLI	81.4	50.67	78.65	50.34	78.37	**45.36**
Linear	80.34	50.01	79.56	51.09	79.29	46.16
MVE	81.79	50.94	78.34	50.02	78.08	45.13
Health Insurance						
Logarithm	90.86	16.17	90.59	15.68	89.54	17.95
HLI	89.91	14.9	89.55	14.61	88.83	16.96
Linear	87.55	12.47	86.95	**12.43**	**86.37**	14.48
MVE	92.94	19.61	92.93	18.85	91.5	20.93
House8L						
Logarithm	81.02	59.92	80.76	58.0	80.06	70.95
HLI	81.35	60.33	80.87	58.07	79.34	69.9
Linear	80.49	59.3	80.35	**57.55**	**79.89**	70.72
MVE	81.66	60.66	80.95	58.17	79.21	69.74
MetroTraffic						
Logarithm	88.52	15.17	90.33	11.92	92.58	11.71
HLI	86.26	14.64	90.3	11.69	92.13	10.87
Linear	86.81	14.03	89.25	10.07	90.48	**9.04**
MVE	**86.13**	14.71	90.82	12.92	93.29	13.68
Diamonds						
Logarithm	89.04	11.03	89.79	6.52	89.8	6.39
HLI	88.56	10.6	89.09	6.14	89.23	6.04
Linear	**86.46**	8.96	86.86	5.16	87.18	**5.04**
MVE	90.49	12.45	91.46	7.6	91.24	7.36
Video Transcoding						
Logarithm	79.84	14.99	**79.97**	3.86	80.04	3.67
HLI	78.19	14.68	79.96	3.8	80.04	3.61
Linear	78.94	14.76	79.13	3.72	78.98	**3.5**
MVE	79.12	14.93	82.57	4.23	82.41	4.02
Wave Energy						
Logarithm	80.9	24.1	81.59	17.71	80.22	21.76
HLI	81.25	24.31	81.93	17.86	80.37	21.84
Linear	80.28	23.78	80.63	**17.32**	**80.07**	21.68
MVE	81.77	24.61	82.89	18.28	80.51	21.9
FriedmanGra						
Logarithm	80.73	24.09	81.49	17.7	80.2	21.76
HLI	81.04	24.28	81.84	17.84	80.29	21.81
Linear	80.2	23.81	80.6	**17.32**	**80.09**	21.71
MVE	81.5	24.57	82.73	18.25	80.41	21.87
FriedmanGsg						
Logarithm	**79.94**	19.36	81.93	**14.1**	79.54	18.67
HLI	80.21	19.48	82.23	14.21	78.63	18.19
Linear	79.52	19.17	80.52	13.62	78.92	18.33
MVE	80.73	19.72	83.54	14.7	78.72	18.22
FriedmanLea						
Logarithm	81.72	18.32	82.19	10.5	80.58	11.23
HLI	81.92	18.43	82.47	10.59	80.85	11.3
Linear	80.97	17.96	81.04	**10.17**	**80.16**	11.11
MVE	82.25	18.58	83.5	10.91	81.08	11.37
Hyper_a						

Table A.3: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 90% and Limit $\mathcal{M} = 0.05$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	91.92	27.66	92.2	25.04	91.5	27.8
HLI	91.96	27.97	92.16	25.3	91.42	27.98
Linear	**90.84**	26.29	91.02	**23.69**	91.0	27.12
MVE	92.75	29.36	93.33	26.77	92.0	28.37
Abalone						
Logarithm	94.14	29.23	95.58	19.01	96.55	18.68
HLI	93.51	28.48	94.79	17.7	95.41	16.35
Linear	**91.5**	27.18	93.15	15.34	93.79	**14.0**
MVE	98.21	34.02	97.28	22.66	98.21	25.46
CPU Activity						
Logarithm	94.0	9.26	99.86	0.83	99.78	1.1
HLI	93.39	8.67	99.81	0.34	99.34	0.56
Linear	**91.34**	8.25	99.69	**0.33**	98.74	0.54
MVE	94.93	11.11	99.99	5.69	99.99	5.73
Naval Propulsion Plant						
Logarithm	92.08	23.97	91.9	23.34	90.95	18.91
HLI	92.08	24.13	91.91	23.51	90.99	19.04
Linear	91.02	22.64	90.78	22.02	**90.36**	**18.53**
MVE	92.62	24.86	92.44	24.1	91.17	19.14
Ailerons						
Logarithm	95.64	22.09	95.58	20.87	95.68	18.74
HLI	95.43	21.45	95.49	20.19	95.51	18.15
Linear	94.8	19.03	94.96	18.01	**94.77**	**16.21**
MVE	96.21	24.77	96.05	23.16	96.22	20.97
Miami Housing						
Logarithm	95.13	3.58	95.79	3.06	94.7	3.23
HLI	95.03	3.48	95.53	2.87	94.59	3.18
Linear	94.09	3.05	94.5	2.46	93.61	2.84
MVE	95.65	3.96	96.56	3.53	95.38	3.53
NZ Energy Price						
Logarithm	95.96	4.48	96.04	4.48	95.3	5.28
HLI	95.17	4.19	95.24	4.16	94.59	5.02
Linear	93.5	3.65	93.62	**3.62**	**92.9**	4.43
MVE	97.09	5.3	97.27	5.32	96.35	6.02
Elevators						
Logarithm	**89.29**	28.58	89.19	25.34	89.04	41.54
HLI	87.39	25.63	87.04	23.06	86.47	36.67
Linear	88.52	27.26	88.27	24.34	87.82	39.03
MVE	86.33	24.24	85.64	**21.7**	83.91	32.81
Bike						
Logarithm	90.13	39.03	90.33	33.43	**90.01**	37.73
HLI	89.61	37.94	90.08	33.08	89.44	36.8
Linear	89.96	38.49	90.16	**32.96**	89.79	37.23
MVE	89.78	38.2	90.28	33.5	89.61	37.12
California Housing						
Logarithm	93.52	37.81	93.43	37.42	93.96	32.53
HLI	93.43	37.63	93.31	37.14	93.75	32.06
Linear	92.62	34.78	**92.59**	34.25	93.1	**29.63**
MVE	93.87	39.21	93.62	38.46	94.04	33.31
Superconductivity						

Table A.4: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 90% and Limit $\mathcal{M} = 0.05$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	92.72	60.69	91.14	62.52	90.05	57.69
HLI	92.67	60.71	91.44	62.99	90.18	57.88
Linear	91.02	58.85	90.39	61.86	**89.97**	**57.56**
MVE	95.71	65.38	92.45	64.2	90.28	57.93
Health Insurance						
Logarithm	94.05	22.16	94.19	21.26	93.23	24.6
HLI	93.75	21.43	93.84	20.38	93.08	24.32
Linear	92.5	18.82	92.43	**18.12**	**91.86**	21.81
MVE	95.02	25.17	95.37	24.2	94.0	26.87
House8L						
Logarithm	91.26	75.4	90.93	72.86	90.57	88.22
HLI	91.51	75.98	91.19	73.44	90.84	88.81
Linear	90.51	73.72	90.36	**71.59**	**90.19**	87.42
MVE	92.2	77.85	91.79	74.66	91.2	89.51
MetroTraffic						
Logarithm	93.7	21.35	93.67	18.8	93.97	17.43
HLI	92.11	20.01	92.71	17.05	93.96	16.92
Linear	92.84	19.98	93.08	17.02	93.65	**15.79**
MVE	**90.86**	18.87	92.41	16.58	94.07	17.56
Diamonds						
Logarithm	92.8	15.11	93.13	9.03	93.06	8.88
HLI	92.81	15.14	93.03	8.94	93.02	8.86
Linear	**91.8**	13.87	91.92	**7.95**	92.0	**7.95**
MVE	93.3	15.98	93.88	9.75	93.59	9.45
Video Transcoding						
Logarithm	91.12	17.98	**89.81**	5.03	89.65	4.85
HLI	91.31	18.01	89.44	4.89	88.91	4.68
Linear	90.4	17.63	89.48	4.85	89.16	**4.67**
MVE	93.65	19.17	90.84	5.43	90.13	5.16
Wave Energy						
Logarithm	90.45	30.85	90.85	22.62	90.03	27.91
HLI	90.65	31.06	91.04	22.77	90.14	28.0
Linear	90.11	30.55	90.31	**22.22**	**89.98**	27.88
MVE	91.15	31.59	91.86	23.46	90.26	28.11
FriedmanGra						
Logarithm	90.39	30.84	90.57	22.67	90.07	27.94
HLI	90.58	31.03	90.75	22.8	90.09	27.96
Linear	90.1	30.58	90.2	**22.36**	**90.02**	27.89
MVE	91.03	31.53	91.49	23.43	90.21	28.07
FriedmanGsg						
Logarithm	89.88	25.03	90.42	18.06	89.32	24.45
HLI	89.72	24.91	90.57	18.15	88.14	23.45
Linear	89.63	24.84	**89.87**	**17.71**	88.83	24.01
MVE	90.18	25.32	91.56	18.86	88.09	23.38
FriedmanLea						
Logarithm	90.43	23.46	90.13	13.67	89.79	14.76
HLI	90.57	23.58	90.22	13.71	89.44	14.56
Linear	90.12	23.18	**89.98**	**13.57**	89.66	14.69
MVE	90.89	23.84	90.68	14.0	89.49	14.59
Hyper_a						

Table A.5: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 95\%$ and Limit $\mathcal{M} = 0.05$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	94.88	35.09	95.26	30.86	95.26	34.32
HLI	94.72	36.13	95.36	32.29	**94.98**	35.18
Linear	94.82	35.18	95.1	**30.68**	95.16	34.37
MVE	94.68	34.98	95.62	31.9	94.96	33.81
Abalone						
Logarithm	97.74	33.71	97.56	23.63	97.96	24.32
HLI	97.19	33.32	97.19	23.33	97.46	22.9
Linear	**96.07**	31.23	96.33	20.61	96.59	**19.12**
MVE	99.71	40.54	98.44	27.0	98.68	30.34
CPU Activity						
Logarithm	96.47	11.34	99.93	0.96	99.9	1.28
HLI	96.32	10.93	99.93	**0.38**	99.72	0.6
Linear	**95.42**	10.48	99.87	**0.38**	99.5	0.58
MVE	97.1	13.24	99.99	6.78	99.99	6.82
Naval Propulsion Plant						
Logarithm	95.19	30.32	95.38	28.95	94.92	23.4
HLI	95.05	30.42	95.37	29.19	94.71	23.3
Linear	**95.04**	29.97	95.08	28.29	94.88	23.35
MVE	94.8	29.62	95.16	28.71	94.41	**22.8**
Ailerons						
Logarithm	97.28	29.54	97.37	27.33	97.38	24.51
HLI	96.96	28.41	97.02	26.81	97.06	24.0
Linear	97.04	26.91	97.18	24.92	97.07	**22.37**
MVE	97.03	29.52	**96.91**	27.59	97.17	24.99
Miami Housing						
Logarithm	96.86	4.53	97.27	3.87	96.79	4.0
HLI	96.88	4.53	97.2	3.82	96.84	4.02
Linear	96.56	4.15	96.7	**3.44**	**96.49**	3.71
MVE	96.95	4.72	97.45	4.2	96.99	4.2
NZ Energy Price						
Logarithm	97.68	5.61	97.82	5.58	97.37	6.54
HLI	97.39	5.39	97.48	5.33	97.11	6.35
Linear	96.55	4.79	96.57	**4.73**	**96.25**	5.76
MVE	97.98	6.31	98.17	6.34	97.66	7.17
Elevators						
Logarithm	93.85	39.56	**93.97**	33.39	93.76	53.74
HLI	92.58	35.61	92.69	30.48	92.33	49.38
Linear	93.35	37.65	93.53	32.15	93.12	51.75
MVE	89.53	28.88	89.56	**25.86**	88.16	39.1
Bike						
Logarithm	94.55	51.58	**94.61**	44.54	94.53	49.53
HLI	93.71	49.28	93.9	42.42	93.65	47.29
Linear	94.37	51.46	94.5	44.27	94.37	49.32
MVE	92.65	45.52	93.2	**39.92**	92.75	44.23
California Housing						
Logarithm	96.31	46.24	96.33	44.43	96.54	41.04
HLI	96.34	45.35	96.34	44.46	96.31	39.84
Linear	95.89	42.68	**95.88**	41.71	96.12	**37.64**
MVE	96.55	46.72	96.54	45.82	96.12	39.69
Superconductivity						

Table A.6: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} =$ 95% and Limit $\mathcal{M} = 0.05$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	97.1	68.82	96.46	70.45	95.41	67.74
HLI	96.83	68.25	96.37	70.48	95.53	68.56
Linear	95.82	**65.89**	95.51	68.86	**95.15**	67.29
MVE	98.85	77.9	98.63	76.5	95.75	69.03
Health Insurance						
Logarithm	95.94	28.52	96.18	27.06	95.37	31.43
HLI	95.97	28.83	96.16	27.21	95.43	31.93
Linear	95.43	26.69	95.54	**25.02**	**95.15**	30.89
MVE	96.32	29.99	96.61	28.83	95.46	32.01
House8L						
Logarithm	95.62	90.01	95.7	86.17	95.77	102.25
HLI	95.75	90.74	95.82	86.89	95.88	102.78
Linear	**95.22**	88.51	95.24	**84.67**	95.23	100.54
MVE	96.22	92.77	96.32	88.96	96.88	106.66
MetroTraffic						
Logarithm	96.23	29.22	96.08	27.47	96.12	26.89
HLI	95.35	27.21	95.16	25.11	95.55	23.57
Linear	95.7	28.28	95.43	26.35	95.8	25.04
MVE	93.06	22.49	93.43	**19.75**	**94.89**	20.92
Diamonds						
Logarithm	95.14	19.3	95.33	11.53	95.17	11.59
HLI	95.1	19.18	95.37	11.59	95.09	11.42
Linear	95.09	19.18	95.12	11.3	95.12	11.54
MVE	94.96	19.04	95.36	11.62	**94.98**	**11.26**
Video Transcoding						
Logarithm	96.33	20.62	94.73	6.16	94.6	5.92
HLI	96.31	20.55	94.28	5.98	93.94	**5.7**
Linear	95.53	19.96	94.65	6.01	94.47	5.78
MVE	98.28	22.84	**94.75**	6.47	94.27	6.14
Wave Energy						
Logarithm	95.25	36.74	95.3	26.98	95.01	33.38
HLI	95.38	36.98	95.4	27.13	95.01	33.42
Linear	95.09	36.4	95.11	**26.7**	**95.0**	33.36
MVE	95.73	37.64	95.92	27.96	95.09	33.5
FriedmanGra						
Logarithm	95.16	36.76	95.15	27.12	95.02	33.48
HLI	95.27	36.97	95.16	27.17	94.93	33.38
Linear	95.05	36.5	95.05	**26.94**	**95.0**	33.47
MVE	95.59	37.57	95.62	27.92	**95.0**	33.44
FriedmanGsg						
Logarithm	94.85	29.99	94.78	22.01	94.2	29.72
HLI	94.59	29.76	94.55	**21.79**	93.35	28.46
Linear	94.79	29.88	94.64	21.84	94.02	29.43
MVE	94.86	30.17	**95.04**	22.48	92.93	27.86
FriedmanLea						
Logarithm	94.9	28.18	94.68	17.11	94.64	18.27
HLI	94.86	28.13	94.18	**16.63**	94.04	17.7
Linear	94.84	28.09	94.63	17.03	94.58	18.21
MVE	**95.03**	28.41	94.17	16.68	93.7	17.39
Hyper_a						

Table A.7: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 99\%$ and Limit $\mathcal{M} = 0.05$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	98.49	52.77	98.55	45.64	98.55	51.39
HLI	98.49	68.58	98.55	61.34	98.51	67.03
Linear	98.65	63.65	**98.71**	55.68	98.69	59.91
MVE	97.57	45.97	98.03	**41.93**	97.61	44.43
Abalone						
Logarithm	99.68	40.9	99.32	**33.19**	99.08	40.82
HLI	99.52	48.89	99.28	41.5	99.1	51.24
Linear	99.32	42.76	99.1	36.35	**99.06**	44.97
MVE	99.93	53.28	99.57	35.48	**99.06**	39.88
CPU Activity						
Logarithm	98.74	17.14	99.97	1.45	99.97	2.12
HLI	98.72	18.61	99.98	0.48	99.95	0.74
Linear	**98.84**	18.45	99.98	**0.46**	99.92	0.69
MVE	98.8	17.41	99.99	8.91	99.99	8.97
Naval Propulsion Plant						
Logarithm	98.46	49.83	98.51	47.17	98.61	34.36
HLI	98.46	52.22	98.51	49.35	98.6	36.46
Linear	98.63	53.24	98.68	50.48	**98.81**	36.53
MVE	97.0	38.93	97.38	37.73	97.58	**29.97**
Ailerons						
Logarithm	99.17	49.33	99.2	43.64	99.17	40.22
HLI	99.17	50.1	99.2	45.46	99.18	42.33
Linear	99.2	49.69	99.24	44.79	99.23	41.7
MVE	98.04	38.79	98.13	36.26	**98.16**	**32.85**
Miami Housing						
Logarithm	98.77	7.4	98.78	6.47	98.87	6.22
HLI	98.77	7.53	98.77	6.61	98.87	6.31
Linear	98.87	7.95	98.88	6.95	**98.94**	6.66
MVE	98.12	6.2	98.39	**5.52**	98.41	**5.52**
NZ Energy Price						
Logarithm	99.12	8.81	99.17	8.69	99.11	8.95
HLI	99.11	8.86	99.17	8.69	99.08	10.4
Linear	99.11	8.95	99.14	8.77	99.09	10.6
MVE	98.88	**8.3**	**98.96**	8.33	98.65	9.42
Elevators						
Logarithm	98.29	57.64	98.31	49.07	98.27	75.03
HLI	98.29	58.16	98.34	49.44	98.31	75.61
Linear	98.42	59.09	**98.46**	50.43	98.42	77.27
MVE	93.49	37.96	94.33	**33.99**	93.14	51.38
Bike						
Logarithm	98.42	78.05	98.43	69.09	98.46	75.96
HLI	98.45	84.28	98.45	75.71	98.49	82.17
Linear	98.61	83.81	98.61	75.27	**98.68**	82.81
MVE	96.06	59.82	96.29	**52.46**	96.17	58.13
California Housing						
Logarithm	98.82	62.04	98.82	59.83	98.94	56.19
HLI	98.81	62.15	98.83	60.76	98.96	56.99
Linear	98.87	61.36	98.86	60.18	**98.99**	55.58
MVE	98.79	61.4	98.86	60.22	98.36	**52.17**
Superconductivity						

Table A.8: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 99% and Limit $\mathcal{M} = 0.05$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	99.49	89.87	99.54	86.1	99.17	87.72
HLI	99.42	91.65	99.45	87.95	99.2	91.63
Linear	99.21	85.4	99.22	**82.77**	**99.07**	89.01
MVE	99.79	102.38	99.82	100.54	99.43	90.72
Health Insurance						
Logarithm	98.46	49.63	98.5	46.28	98.42	54.77
HLI	98.47	51.54	98.52	48.36	98.44	56.91
Linear	98.62	53.96	**98.67**	50.35	98.6	58.67
MVE	97.72	39.41	97.91	**37.89**	97.31	42.07
House8L						
Logarithm	99.31	113.16	99.3	109.23	99.35	126.08
HLI	99.27	114.77	99.27	111.11	99.31	127.96
Linear	99.09	111.63	**99.07**	**108.25**	99.12	124.21
MVE	99.76	121.92	99.75	116.91	99.88	140.17
MetroTraffic						
Logarithm	**98.98**	43.63	98.84	43.52	98.51	56.32
HLI	98.86	45.68	98.79	44.77	98.62	56.0
Linear	98.82	48.64	98.69	47.69	98.48	57.06
MVE	95.51	29.56	94.96	**25.96**	95.96	27.49
Diamonds						
Logarithm	98.37	35.24	98.29	20.9	98.33	21.05
HLI	98.41	35.47	98.33	21.06	98.36	21.28
Linear	**98.56**	37.06	98.49	22.09	98.5	22.18
MVE	96.91	25.03	97.07	15.27	96.84	**14.8**
Video Transcoding						
Logarithm	99.47	25.5	98.65	8.87	98.62	8.24
HLI	99.39	25.58	98.65	9.0	98.63	8.34
Linear	99.19	24.72	**98.83**	9.19	98.81	8.46
MVE	99.93	30.02	98.05	8.51	98.15	**8.07**
Wave Energy						
Logarithm	99.04	48.05	98.91	**36.25**	98.95	44.19
HLI	99.06	48.59	98.89	36.53	98.93	44.42
Linear	99.03	48.04	**98.98**	36.64	**98.98**	44.47
MVE	99.2	49.47	98.92	36.74	98.93	44.03
FriedmanGra						
Logarithm	99.02	48.21	98.88	**36.44**	98.93	44.33
HLI	**99.01**	48.71	98.86	36.74	98.9	44.53
Linear	99.02	48.29	98.97	36.93	98.98	44.86
MVE	99.13	49.38	98.83	36.69	98.85	43.95
FriedmanGsg						
Logarithm	98.8	40.11	98.56	32.07	98.41	40.86
HLI	98.78	40.35	98.58	32.56	98.43	41.25
Linear	**98.94**	41.18	98.74	33.89	98.63	42.37
MVE	98.61	39.64	97.92	**29.54**	97.31	36.62
FriedmanLea						
Logarithm	98.84	38.07	98.61	24.64	98.63	25.56
HLI	98.8	38.0	98.61	24.73	98.63	25.65
Linear	**98.93**	38.48	98.81	25.47	98.82	26.39
MVE	98.65	37.34	98.05	**8.51**	97.67	22.85
Hyper_a						

Table A.9: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 80% and Limit $\mathcal{M} = 0.1$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	85.94	21.05	85.92	19.22	85.21	20.99
HLI	85.73	20.89	85.73	19.03	85.23	21.07
Linear	83.32	19.11	**83.14**	**17.46**	83.34	19.63
MVE	87.98	22.87	88.13	20.86	86.3	22.11
Abalone						
Logarithm	84.34	24.05	91.85	14.22	93.59	14.13
HLI	84.22	23.82	90.16	12.8	91.42	12.18
Linear	**81.58**	22.77	87.01	11.18	88.61	**10.38**
MVE	89.5	26.51	94.95	17.65	96.92	19.84
CPU Activity						
Logarithm	87.87	7.22	99.82	0.78	99.53	0.95
HLI	87.34	6.66	99.89	**0.45**	98.64	0.6
Linear	**83.46**	6.29	99.84	**0.45**	97.74	0.58
MVE	89.24	8.66	99.99	4.43	99.99	4.46
Naval Propulsion Plant						
Logarithm	86.09	18.14	84.82	17.86	83.37	14.46
HLI	85.94	18.02	84.73	17.85	83.49	14.58
Linear	83.36	16.46	82.28	16.46	**81.83**	**13.81**
MVE	87.7	19.37	86.44	18.77	84.24	14.91
Ailerons						
Logarithm	92.86	16.37	92.74	15.53	92.93	13.87
HLI	92.33	15.41	92.37	14.7	92.32	13.08
Linear	**90.55**	13.37	90.7	12.85	90.59	**11.32**
MVE	94.33	19.3	94.09	18.04	94.39	16.34
Miami Housing						
Logarithm	91.55	2.7	93.05	2.29	90.95	2.43
HLI	91.13	2.57	92.23	2.09	90.51	2.33
Linear	89.25	2.21	90.38	**1.77**	**88.44**	2.04
MVE	93.11	3.08	94.75	2.75	92.36	2.75
NZ Energy Price						
Logarithm	92.24	3.45	92.48	3.43	91.13	4.01
HLI	90.78	3.17	91.09	3.15	89.91	3.74
Linear	87.86	2.72	88.37	**2.7**	**87.12**	3.23
MVE	94.62	4.13	94.82	4.14	93.38	4.69
Elevators						
Logarithm	80.71	18.75	79.66	17.47	**79.79**	28.97
HLI	80.63	18.76	78.74	**16.87**	76.6	26.38
Linear	80.26	18.5	79.03	17.12	78.17	27.67
MVE	80.83	18.88	78.92	16.91	75.49	25.56
Bike						
Logarithm	83.68	28.27	83.5	24.79	82.69	27.55
HLI	83.78	28.44	83.72	24.88	82.81	27.71
Linear	82.22	26.53	82.06	**23.39**	**81.3**	26.16
MVE	84.82	29.76	84.87	26.1	83.88	28.92
California Housing						
Logarithm	87.73	29.54	87.65	29.14	89.36	24.43
HLI	87.45	29.36	87.39	28.85	89.28	24.05
Linear	86.28	27.06	**86.22**	26.67	87.72	**21.67**
MVE	88.19	30.55	87.9	29.96	90.28	25.95
Superconductivity						

Table A.10: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 80% and Limit $\mathcal{M} = 0.1$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	80.94	50.4	**79.87**	51.42	79.79	46.52
HLI	81.42	50.68	78.65	50.34	78.37	45.36
Linear	80.36	50.03	79.53	51.07	79.27	46.14
MVE	81.79	50.94	78.34	50.02	78.08	**45.13**
Health Insurance						
Logarithm	90.94	16.31	90.73	15.8	89.62	18.07
HLI	90.03	15.05	89.66	14.74	88.92	17.08
Linear	87.76	12.64	87.21	**12.58**	**86.57**	14.66
MVE	92.94	19.61	92.93	18.85	91.5	20.93
House8L						
Logarithm	81.05	59.94	80.77	58.01	**80.06**	70.95
HLI	81.36	60.34	80.87	58.07	79.34	69.9
Linear	80.51	59.32	80.36	**57.59**	79.89	70.69
MVE	81.66	60.66	80.95	58.17	79.21	69.74
MetroTraffic						
Logarithm	88.53	15.2	90.35	11.97	92.62	11.8
HLI	**86.26**	14.65	90.32	11.75	92.2	10.98
Linear	86.79	14.08	89.32	10.18	90.62	**9.21**
MVE	86.13	14.71	90.82	12.92	93.29	13.68
Diamonds						
Logarithm	89.11	11.09	89.86	6.57	89.88	6.43
HLI	88.64	10.67	89.18	6.19	89.32	6.09
Linear	**86.63**	9.07	87.06	5.23	87.36	**5.11**
MVE	90.49	12.45	91.46	7.6	91.24	7.36
Video Transcoding						
Logarithm	79.83	14.99	80.04	3.87	80.09	3.68
HLI	78.2	14.68	**80.0**	3.81	80.1	3.61
Linear	78.9	14.75	79.12	3.72	78.98	**3.5**
MVE	79.12	14.93	82.57	4.23	82.41	4.02
Wave Energy						
Logarithm	80.93	24.12	81.63	17.73	80.23	21.76
HLI	81.27	24.32	81.96	17.87	80.37	21.84
Linear	80.3	23.79	80.67	**17.32**	**80.08**	21.69
MVE	81.77	24.61	82.89	18.28	80.51	21.9
FriedmanGra						
Logarithm	80.76	24.1	81.53	17.72	80.2	21.77
HLI	81.06	24.29	81.86	17.85	80.29	21.81
Linear	80.21	23.82	80.62	**17.33**	**80.09**	21.71
MVE	81.5	24.57	82.73	18.25	80.41	21.87
FriedmanGsg						
Logarithm	**79.94**	19.36	81.98	14.12	79.54	18.67
HLI	80.23	19.49	82.27	14.22	78.63	18.19
Linear	79.51	19.17	80.56	**13.63**	78.9	18.32
MVE	80.73	19.72	83.54	14.7	78.72	18.22
FriedmanLea						
Logarithm	81.74	18.33	82.23	10.51	80.6	11.23
HLI	81.94	18.43	82.5	10.6	80.86	11.3
Linear	81.01	17.98	81.08	**10.19**	**80.17**	11.11
MVE	82.25	18.58	83.5	10.91	81.08	11.37
Hyper_a						

Table A.11: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 90% and Limit $\mathcal{M} = 0.1$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	91.94	27.74	92.24	25.1	91.54	27.84
HLI	91.94	27.97	92.22	25.37	91.42	28.02
Linear	**90.88**	26.39	91.08	**23.76**	91.02	27.12
MVE	92.75	29.36	93.33	26.77	92.0	28.37
Abalone						
Logarithm	94.29	29.33	95.63	19.15	96.63	18.9
HLI	93.59	28.44	94.87	17.85	95.53	16.47
Linear	**91.59**	27.18	93.3	15.47	93.95	**14.16**
MVE	98.21	34.02	97.28	22.66	98.21	25.46
CPU Activity						
Logarithm	94.08	9.33	99.99	1.0	99.82	1.14
HLI	93.47	8.66	99.99	**0.57**	99.59	0.67
Linear	**91.39**	8.26	99.99	**0.57**	99.21	0.66
MVE	94.93	11.11	99.99	5.69	99.99	5.73
Naval Propulsion Plant						
Logarithm	92.1	24.02	91.93	23.39	90.97	18.93
HLI	92.07	24.15	91.91	23.52	90.99	19.05
Linear	91.05	22.69	90.81	22.09	**90.33**	**18.53**
MVE	92.62	24.86	92.44	24.1	91.17	19.14
Ailerons						
Logarithm	95.65	22.2	95.59	20.98	95.69	18.83
HLI	95.47	21.57	95.51	20.3	95.54	18.26
Linear	**94.83**	19.24	95.01	18.19	94.84	**16.34**
MVE	96.21	24.77	96.05	23.16	96.22	20.97
Miami Housing						
Logarithm	95.16	3.6	95.84	3.07	94.73	3.24
HLI	95.05	3.5	95.57	2.9	94.61	3.19
Linear	94.13	3.08	94.58	**2.49**	**93.67**	2.86
MVE	95.65	3.96	96.56	3.53	95.38	3.53
NZ Energy Price						
Logarithm	96.01	4.52	96.1	4.51	95.36	5.31
HLI	95.25	4.22	95.32	4.19	94.68	5.05
Linear	93.65	3.7	93.78	**3.67**	**93.02**	4.47
MVE	97.09	5.3	97.27	5.32	96.35	6.02
Elevators						
Logarithm	**89.29**	28.57	89.19	25.35	89.05	41.54
HLI	87.39	25.63	87.04	23.05	86.47	36.67
Linear	88.47	27.2	88.2	24.3	87.74	38.87
MVE	86.33	24.24	85.64	**21.7**	83.91	32.81
Bike						
Logarithm	90.13	39.0	90.33	33.46	**90.01**	37.73
HLI	89.62	37.94	90.09	33.1	89.45	36.82
Linear	89.96	38.41	90.16	**32.98**	89.77	37.22
MVE	89.78	38.2	90.28	33.5	89.61	37.12
California Housing						
Logarithm	93.54	37.88	93.44	37.49	93.97	32.58
HLI	93.46	37.69	93.34	37.19	93.76	32.11
Linear	92.7	34.95	**92.66**	34.44	93.16	**29.84**
MVE	93.87	39.21	93.62	38.46	94.04	33.31
Superconductivity						

Table A.12: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 90% and Limit $\mathcal{M} = 0.1$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	92.79	60.8	91.17	62.57	**90.05**	57.69
HLI	92.71	60.77	91.45	63.01	90.18	57.89
Linear	91.06	58.89	90.39	61.88	89.97	**57.58**
MVE	95.71	65.38	92.45	64.2	90.28	57.93
Health Insurance						
Logarithm	94.13	22.26	94.26	21.36	93.25	24.7
HLI	93.79	21.55	93.89	20.49	93.12	24.41
Linear	92.57	18.97	92.53	**18.25**	**91.92**	21.9
MVE	95.02	25.17	95.37	24.2	94.0	26.87
House8L						
Logarithm	91.29	75.48	90.95	72.91	90.59	88.24
HLI	91.52	76.04	91.21	73.47	90.85	88.84
Linear	90.53	73.8	90.38	**71.6**	**90.19**	87.44
MVE	92.2	77.85	91.79	74.66	91.2	89.51
MetroTraffic						
Logarithm	93.7	21.37	93.67	18.82	93.97	17.46
HLI	92.1	20.01	92.71	17.06	93.97	16.95
Linear	92.8	20.02	93.07	17.04	93.67	15.87
MVE	**90.86**	18.87	92.41	**16.58**	94.07	17.56
Diamonds						
Logarithm	92.84	15.15	93.17	9.06	93.09	8.91
HLI	92.83	15.17	93.07	8.97	93.05	8.88
Linear	**91.84**	13.94	91.98	8.0	92.06	**7.99**
MVE	93.3	15.98	93.88	9.75	93.59	9.45
Video Transcoding						
Logarithm	91.18	18.0	**89.81**	5.04	89.65	4.86
HLI	91.35	18.03	89.46	4.89	88.93	4.68
Linear	90.42	17.63	89.47	4.84	89.13	**4.66**
MVE	93.65	19.17	90.84	5.43	90.13	5.16
Wave Energy						
Logarithm	90.47	30.86	90.87	22.63	90.03	27.91
HLI	90.67	31.06	91.06	22.77	90.14	28.0
Linear	90.12	30.55	90.33	**22.22**	**89.98**	27.88
MVE	91.15	31.59	91.86	23.46	90.26	28.11
FriedmanGra						
Logarithm	90.4	30.85	90.59	22.68	90.07	27.94
HLI	90.59	31.04	90.77	22.8	90.09	27.96
Linear	90.1	30.58	90.21	**22.36**	**90.01**	27.89
MVE	91.03	31.53	91.49	23.43	90.21	28.07
FriedmanGsg						
Logarithm	**89.88**	25.03	90.45	18.07	89.32	24.45
HLI	89.73	24.92	90.59	18.16	88.14	23.45
Linear	89.62	24.83	89.87	**17.71**	88.81	23.98
MVE	90.18	25.32	91.56	18.86	88.09	23.38
FriedmanLea						
Logarithm	90.44	23.47	90.14	13.67	89.79	14.76
HLI	90.58	23.59	90.23	13.72	89.45	14.56
Linear	90.13	23.19	**89.98**	**13.57**	89.66	14.69
MVE	90.89	23.84	90.68	14.0	89.49	14.59
Hyper_a						

Table A.13: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} =$ 99% and Limit $\mathcal{M} = 0.1$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	98.49	52.77	98.55	45.65	98.55	51.4
HLI	98.49	68.58	98.55	61.36	98.51	67.04
Linear	98.65	63.04	**98.73**	55.35	98.71	60.0
MVE	97.57	45.97	98.03	**41.93**	97.61	44.43
Abalone						
Logarithm	99.69	41.56	99.33	33.32	99.08	40.88
HLI	99.54	45.38	99.28	35.74	99.08	45.51
Linear	99.34	41.11	99.08	**33.34**	**99.05**	41.89
MVE	99.93	53.28	99.57	35.48	99.06	39.88
CPU Activity						
Logarithm	98.74	17.13	99.99	1.57	99.97	1.74
HLI	98.72	17.69	99.99	**0.89**	99.97	0.93
Linear	**98.84**	17.73	99.99	**0.89**	99.97	0.92
MVE	98.8	17.41	99.99	8.91	99.99	8.97
Naval Propulsion Plant						
Logarithm	98.46	49.83	98.51	47.17	98.61	34.36
HLI	98.47	52.22	98.52	49.35	98.6	36.46
Linear	98.62	53.25	98.69	50.39	**98.8**	36.39
MVE	97.0	38.93	97.38	37.73	97.58	**29.97**
Ailerons						
Logarithm	**99.17**	49.36	99.2	43.68	**99.17**	40.26
HLI	99.18	50.18	99.2	45.25	99.19	42.33
Linear	99.2	49.71	99.25	44.5	99.22	41.68
MVE	98.04	38.79	98.13	36.26	98.16	**32.85**
Miami Housing						
Logarithm	98.77	7.4	98.78	6.47	98.87	6.22
HLI	98.77	7.53	98.77	6.61	98.87	6.31
Linear	98.87	7.92	98.87	6.92	**98.94**	6.61
MVE	98.12	6.2	98.39	**5.52**	98.41	**5.52**
NZ Energy Price						
Logarithm	99.12	8.81	99.18	41.34	99.07	10.39
HLI	99.12	8.86	99.17	8.69	99.08	10.4
Linear	99.122	8.95	99.15	8.78	99.07	10.64
MVE	98.88	**8.3**	**98.96**	8.33	98.65	9.42
Elevators						
Logarithm	98.29	57.64	98.31	49.07	98.27	75.03
HLI	98.29	58.16	98.34	49.44	98.31	75.61
Linear	98.38	58.81	**98.43**	50.19	98.39	76.86
MVE	93.49	37.96	94.33	**33.99**	93.14	51.38
Bike						
Logarithm	98.42	78.05	98.43	69.1	98.46	75.96
HLI	98.45	84.28	98.45	75.71	98.5	82.17
Linear	98.59	83.39	98.6	74.91	**98.66**	82.22
MVE	96.06	59.82	96.29	**52.46**	96.17	58.13
California Housing						
Logarithm	98.82	62.08	98.82	59.82	98.94	56.2
HLI	98.8	61.92	98.82	60.5	98.96	56.73
Linear	98.87	61.24	98.84	60.11	**98.98**	55.42
MVE	98.79	61.4	98.86	60.22	98.36	**52.17**
Superconductivity						

Table A.14: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 99% and Limit $\mathcal{M} = 0.1$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	99.51	90.11	99.55	86.42	99.17	87.66
HLI	99.43	91.62	99.45	87.93	99.2	91.68
Linear	99.21	85.32	99.23	**82.79**	**99.07**	89.04
MVE	99.79	102.38	99.82	100.54	99.43	90.72
Health Insurance						
Logarithm	98.46	49.63	98.5	46.28	98.42	54.78
HLI	98.47	51.54	98.52	48.36	98.44	56.91
Linear	98.6	53.49	**98.66**	50.26	98.6	58.32
MVE	97.72	39.41	97.91	**37.89**	97.31	42.07
House8L						
Logarithm	99.32	113.31	99.3	109.22	99.36	126.4
HLI	99.28	115.06	99.28	111.12	99.32	128.12
Linear	**99.09**	111.73	**99.09**	**108.25**	99.12	124.06
MVE	99.76	121.92	99.75	116.91	99.88	140.17
MetroTraffic						
Logarithm	**98.98**	43.63	98.84	43.52	98.51	56.32
HLI	98.86	45.68	98.79	44.77	98.62	56.0
Linear	98.8	48.51	98.66	47.56	98.44	56.56
MVE	95.51	29.56	94.96	**25.96**	95.96	27.49
Diamonds						
Logarithm	98.37	35.24	98.29	20.9	98.33	21.05
HLI	98.41	35.47	98.33	21.06	98.36	21.28
Linear	**98.55**	36.85	98.47	21.99	98.49	22.06
MVE	96.91	25.03	97.07	15.27	96.84	**14.8**
Video Transcoding						
Logarithm	99.49	25.61	98.65	8.88	98.62	8.25
HLI	99.4	25.67	98.65	8.79	98.62	8.16
Linear	99.2	24.75	**98.82**	9.03	98.8	8.34
MVE	99.93	30.02	98.05	8.51	98.15	**8.07**
Wave Energy						
Logarithm	99.05	48.07	98.91	**36.24**	98.95	44.19
HLI	99.06	48.56	98.89	36.47	98.93	44.38
Linear	99.03	47.96	98.97	36.61	**98.98**	44.42
MVE	99.2	49.47	98.92	36.74	98.93	44.03
FriedmanGra						
Logarithm	99.02	48.25	98.88	**36.42**	98.93	44.33
HLI	**99.01**	48.66	98.86	36.69	98.9	44.49
Linear	99.02	48.23	98.96	36.89	98.98	44.81
MVE	99.13	49.38	98.83	36.69	98.85	43.95
FriedmanGsg						
Logarithm	98.8	40.1	98.56	32.05	98.41	40.86
HLI	98.78	40.32	98.58	32.51	98.43	41.21
Linear	**98.94**	41.09	98.72	33.7	98.62	42.27
MVE	98.61	39.64	97.92	**29.54**	97.31	36.62
FriedmanLea						
Logarithm	98.84	38.07	98.61	24.64	98.63	25.56
HLI	98.8	38.01	98.62	24.73	98.63	25.65
Linear	**98.92**	38.47	98.8	25.44	98.81	26.34
MVE	98.65	37.34	98.05	**8.51**	97.67	22.85
Hyper_a						

Table A.15: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 80% and Limit $\mathcal{M} = 0.5$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	86.82	21.66	86.64	19.77	85.73	21.4
HLI	86.48	21.49	86.38	19.59	85.75	21.38
Linear	84.61	19.97	84.45	**18.19**	**84.29**	20.25
MVE	87.98	22.87	88.13	20.86	86.3	22.11
Abalone						
Logarithm	85.75	24.72	93.24	15.47	95.14	16.03
HLI	85.23	24.25	91.93	14.03	93.64	13.89
Linear	**82.6**	23.22	89.92	**12.68**92.03	12.71	
MVE	89.5	26.51	94.95	17.65	96.92	19.84
CPU Activity						
Logarithm	88.14	7.7	99.99	2.4	99.99	2.42
HLI	87.87	7.19	99.99	**2.22**	99.99	2.23
Linear	**85.87**	6.58	99.99	**2.22**	99.99	2.23
MVE	89.24	8.66	99.99	4.43	99.99	4.46
Naval Propulsion Plant						
Logarithm	86.71	18.62	85.45	18.25	83.73	14.65
HLI	86.57	18.48	85.31	18.17	83.64	14.69
Linear	84.84	17.26	83.46	17.16	**82.52**	**14.09**
MVE	87.7	19.37	86.44	18.77	84.24	14.91
Ailerons						
Logarithm	93.45	17.48	93.28	16.49	93.53	14.79
HLI	93.01	16.68	92.96	15.8	93.12	14.07
Linear	**91.98**	15.17	92.0	14.4	92.1	**12.77**
MVE	94.33	19.3	94.09	18.04	94.39	16.34
Miami Housing						
Logarithm	92.18	2.85	93.83	2.46	91.49	2.55
HLI	91.82	2.74	93.18	2.31	91.21	2.47
Linear	90.47	2.49	91.93	**2.08**	**89.7**	2.26
MVE	93.11	3.08	94.75	2.75	92.36	2.75
NZ Energy Price						
Logarithm	93.36	3.7	93.53	3.7	92.21	4.27
HLI	92.36	3.48	92.57	3.47	91.3	4.06
Linear	90.63	3.15	90.97	**3.14**	**89.53**	3.69
MVE	94.62	4.13	94.82	4.14	93.38	4.69
Elevators						
Logarithm	80.78	18.82	79.67	17.47	**79.79**	28.97
HLI	80.68	18.8	78.81	**16.88**	76.6	26.39
Linear	80.39	18.57	78.88	17.04	77.52	27.15
MVE	80.83	18.88	78.92	16.91	75.49	25.56
Bike						
Logarithm	84.1	28.81	84.05	25.23	83.06	28.03
HLI	84.12	28.77	84.09	25.16	83.11	28.02
Linear	82.84	27.3	82.78	**23.93**	**81.91**	26.69
MVE	84.82	29.76	84.87	26.1	83.88	28.92
California Housing						
Logarithm	87.97	30.03	87.86	29.6	89.76	25.03
HLI	87.76	29.86	**87.6**	29.32	89.71	24.8
Linear	87.12	28.49	87.07	28.0	88.75	**23.24**
MVE	88.19	30.55	87.9	29.96	90.28	25.95
Superconductivity						

Table A.16: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} =$ 80% and Limit $\mathcal{M} = 0.5$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	81.19	50.56	**79.87**	51.43	79.79	46.52
HLI	81.53	50.77	78.65	50.34	78.38	45.36
Linear	80.57	50.15	79.35	50.89	79.02	45.96
MVE	81.79	50.94	78.34	50.02	78.08	**45.13**
Health Insurance						
Logarithm	91.81	17.57	91.65	16.94	90.42	19.13
HLI	91.05	16.49	90.9	15.97	89.83	18.33
Linear	89.78	14.75	89.3	**14.38**	**88.41**	16.49
MVE	92.94	19.61	92.93	18.85	91.5	20.93
House8L						
Logarithm	81.23	60.17	80.88	58.12	**80.06**	70.95
HLI	81.46	60.43	80.89	58.1	79.34	69.89
Linear	80.73	59.56	80.5	**57.72**	79.77	70.51
MVE	81.66	60.66	80.95	58.17	79.21	69.74
MetroTraffic						
Logarithm	88.55	15.41	90.55	12.37	92.98	12.58
HLI	86.32	14.74	90.53	12.23	92.73	12.0
Linear	86.48	14.47	89.95	11.21	91.9	**10.69**
MVE	**86.13**	14.71	90.82	12.92	93.29	13.68
Diamonds						
Logarithm	89.68	11.65	90.54	6.97	90.44	6.81
HLI	89.33	11.32	90.02	6.67	90.05	6.55
Linear	**88.06**	10.18	88.76	5.97	88.87	**5.85**
MVE	90.49	12.45	91.46	7.6	91.24	7.36
Video Transcoding						
Logarithm	**79.84**	15.01	80.74	3.96	80.79	3.76
HLI	78.4	14.73	80.56	3.89	80.61	**3.7**
Linear	78.47	14.69	79.3	3.77	79.3	3.56
MVE	79.12	14.93	82.57	4.23	82.41	4.02
Wave Energy						
Logarithm	81.19	24.28	82.02	17.9	80.3	21.8
HLI	81.43	24.41	82.22	17.98	80.41	21.86
Linear	80.51	23.9	81.0	**17.47**	**80.12**	21.71
MVE	81.77	24.61	82.89	18.28	80.51	21.9
FriedmanGra						
Logarithm	80.99	24.25	81.91	17.89	80.23	21.79
HLI	81.19	24.37	82.1	17.97	80.33	21.83
Linear	80.38	23.91	80.94	**17.47**	**80.11**	21.72
MVE	81.5	24.57	82.73	18.25	80.41	21.87
FriedmanGsg						
Logarithm	**80.15**	19.47	82.5	14.31	79.54	18.67
HLI	80.38	19.56	82.64	14.36	78.67	18.2
Linear	79.61	19.2	81.12	**13.82**	78.74	18.24
MVE	80.73	19.72	83.54	14.7	78.72	18.22
FriedmanLea						
Logarithm	81.87	18.4	82.65	10.64	80.75	11.27
HLI	82.05	18.48	82.78	10.68	80.93	11.32
Linear	81.33	18.14	81.55	**10.32**	**80.35**	11.16
MVE	82.25	18.58	83.5	10.91	81.08	11.37
Hyper_a						

Table A.17: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} =$ 90% and Limit $\mathcal{M} = 0.5$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	92.26	28.26	92.67	25.6	91.7	28.06
HLI	92.22	28.05	92.57	25.32	91.54	27.78
Linear	**91.38**	26.54	91.54	**24.11**	91.18	27.1
MVE	92.75	29.36	93.33	26.77	92.0	28.37
Abalone						
Logarithm	95.53	30.56	96.35	20.41	97.34	21.35
HLI	94.56	29.46	95.68	19.03	96.59	18.74
Linear	**92.65**	27.92	94.64	17.26	95.7	**16.9**
MVE	98.21	34.02	97.28	22.66	98.21	25.46
CPU Activity						
Logarithm	94.25	9.87	99.99	3.09	99.99	3.11
HLI	94.08	9.19	99.99	**2.84**	99.99	2.86
Linear	**92.37**	8.62	99.99	**2.84**	99.99	2.86
MVE	94.93	11.11	99.99	5.69	99.99	5.73
Naval Propulsion Plant						
Logarithm	92.35	24.35	92.18	23.73	91.13	19.07
HLI	92.28	24.36	92.14	23.68	91.02	19.04
Linear	91.53	23.24	91.29	22.69	**90.55**	**18.68**
MVE	92.62	24.86	92.44	24.1	91.17	19.14
Ailerons						
Logarithm	95.89	23.15	95.73	21.84	95.89	19.65
HLI	95.74	22.65	95.71	21.31	95.79	19.17
Linear	**95.36**	20.87	**95.36**	19.86	95.43	**17.78**
MVE	96.21	24.77	96.05	23.16	96.22	20.97
Miami Housing						
Logarithm	95.35	3.75	96.12	3.25	94.96	3.35
HLI	95.28	3.67	95.92	3.11	94.86	3.31
Linear	94.63	3.37	95.31	**2.81**	**94.17**	3.06
MVE	95.65	3.96	96.56	3.53	95.38	3.53
NZ Energy Price						
Logarithm	96.56	4.82	96.71	4.82	95.94	17.52
HLI	96.06	4.56	96.14	4.55	94.86	**3.31**
Linear	95.1	4.17	95.16	4.16	**94.36**	4.93
MVE	97.09	5.3	97.27	5.32	97.66	7.17
Elevators						
Logarithm	**89.29**	28.55	89.2	25.36	89.05	41.55
HLI	87.39	25.63	87.04	23.05	86.47	36.67
Linear	87.98	26.39	87.62	23.7	86.98	37.44
MVE	86.33	24.24	85.64	**21.7**	83.91	32.81
Bike						
Logarithm	90.15	39.03	90.36	33.53	**90.0**	37.77
HLI	89.65	37.87	90.13	33.08	89.52	36.76
Linear	89.83	38.1	90.11	**33.01**	89.65	36.93
MVE	89.78	38.2	90.28	33.5	89.61	37.12
California Housing						
Logarithm	93.64	38.46	93.59	38.01	94.08	33.1
HLI	93.65	38.3	93.46	37.72	93.91	32.62
Linear	93.16	36.57	**93.15**	36.02	93.61	**31.3**
MVE	93.87	39.21	93.62	38.46	94.04	33.31
Superconductivity						

Table A.18: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 90\%$ and Limit $\mathcal{M} = 0.5$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	93.64	61.87	91.56	62.98	90.12	57.74
HLI	93.34	61.54	91.71	63.27	90.2	57.91
Linear	91.7	59.53	90.65	62.12	**90.0**	**57.57**
MVE	95.71	65.38	92.45	64.2	90.28	57.93
Health Insurance						
Logarithm	94.53	23.35	94.69	22.41	93.56	25.5
HLI	94.25	22.6	94.39	21.66	93.48	25.15
Linear	93.39	20.68	93.46	**19.71**	**92.59**	23.24
MVE	95.02	25.17	95.37	24.2	94.0	26.87
House8L						
Logarithm	91.6	76.19	91.21	73.44	90.78	88.62
HLI	91.7	76.53	91.37	73.77	90.97	89.02
Linear	90.81	74.41	90.59	**72.08**	**90.33**	87.68
MVE	92.2	77.85	91.79	74.66	91.2	89.51
MetroTraffic						
Logarithm	93.7	21.55	93.69	18.98	94.02	17.65
HLI	92.1	20.04	92.71	17.12	94.02	17.21
Linear	92.33	20.06	92.91	17.08	93.81	**16.52**
MVE	**90.86**	18.87	92.41	16.58	94.07	17.56
Diamonds						
Logarithm	93.06	15.5	93.49	9.33	93.31	9.14
HLI	93.03	15.48	93.38	9.24	93.25	9.09
Linear	**92.31**	14.54	92.58	8.53	92.59	**8.45**
MVE	93.3	15.98	93.88	9.75	93.59	9.45
Video Transcoding						
Logarithm	91.74	18.25	**89.89**	5.1	89.67	4.92
HLI	91.7	18.18	89.75	4.99	89.14	4.76
Linear	90.6	17.72	89.35	4.88	88.95	**4.68**
MVE	93.65	19.17	90.84	5.43	90.13	5.16
Wave Energy						
Logarithm	90.66	31.07	91.15	22.87	90.07	27.96
HLI	90.78	31.2	91.25	22.94	90.16	28.03
Linear	90.22	30.64	90.49	**22.34**	**89.98**	27.88
MVE	91.15	31.59	91.86	23.46	90.26	28.11
FriedmanGra						
Logarithm	90.58	31.04	90.84	22.88	90.08	27.96
HLI	90.7	31.17	90.93	22.95	90.11	27.99
Linear	90.18	30.66	90.31	**22.44**	**90.0**	27.89
MVE	91.03	31.53	91.49	23.43	90.21	28.07
FriedmanGsg						
Logarithm	89.92	25.07	90.78	18.3	89.32	24.46
HLI	89.84	25.02	90.82	18.33	88.15	23.46
Linear	89.53	24.77	**90.02**	**17.79**	88.5	23.75
MVE	90.18	25.32	91.56	18.86	88.09	23.38
FriedmanLea						
Logarithm	90.56	23.57	90.28	13.76	89.79	14.76
HLI	90.66	23.66	90.33	13.78	89.46	14.57
Linear	90.25	23.31	**90.01**	**13.6**	89.58	14.64
MVE	90.89	23.84	90.68	14.0	89.49	14.59
Hyper_a						

Table A.19: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} =$ 95% and Limit $\mathcal{M} = 0.5$ (Part I)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	94.92	35.31	95.38	31.3	95.26	34.32
HLI	94.74	35.07	95.44	31.43	**94.98**	34.04
Linear	94.72	34.59	95.08	**30.21**	95.16	33.76
MVE	94.68	34.98	95.62	31.9	94.96	33.81
Abalone						
Logarithm	98.58	35.14	97.88	24.9	98.3	26.77
HLI	97.88	34.03	97.63	23.84	98.0	24.28
Linear	**96.89**	32.42	97.0	**22.05**	97.49	22.08
MVE	99.71	40.54	98.44	27.0	98.68	30.34
CPU Activity						
Logarithm	96.79	11.93	99.99	3.68	99.99	3.7
HLI	96.72	11.19	99.99	**3.39**	99.99	3.41
Linear	**95.74**	10.74	99.99	**3.39**	99.99	3.41
MVE	97.1	13.24	99.99	6.78	99.99	6.82
Naval Propulsion Plant						
Logarithm	95.24	30.44	95.46	29.17	94.91	23.44
HLI	**95.03**	30.21	95.37	29.01	94.71	23.07
Linear	95.08	30.2	95.21	28.57	94.86	23.18
MVE	94.8	29.62	95.16	28.71	94.41	**22.8**
Ailerons						
Logarithm	97.29	30.16	97.37	28.08	97.41	25.06
HLI	97.01	29.03	97.05	27.31	97.12	24.53
Linear	97.07	28.06	97.12	26.27	97.11	**23.69**
MVE	97.03	29.52	**96.91**	27.59	97.17	24.99
Miami Housing						
Logarithm	96.95	4.62	97.4	4.01	96.95	4.09
HLI	96.95	4.61	97.34	3.95	96.94	4.08
Linear	96.7	4.37	96.99	**3.68**	**96.63**	3.88
MVE	96.95	4.72	97.45	4.2	96.99	4.2
NZ Energy Price						
Logarithm	97.89	5.92	98.05	5.9	97.65	6.83
HLI	97.73	5.74	97.88	5.68	97.45	6.67
Linear	97.27	5.26	97.36	**5.2**	**96.93**	6.21
MVE	97.98	6.31	98.17	6.34	97.66	7.17
Elevators						
Logarithm	93.85	39.56	93.97	33.39	93.76	53.72
HLI	92.58	35.61	92.69	30.47	92.32	49.37
Linear	92.57	35.58	**92.82**	30.67	92.23	48.88
MVE	89.53	28.88	89.56	**25.86**	88.16	39.1
Bike						
Logarithm	94.55	51.71	**94.62**	44.58	94.53	49.58
HLI	93.71	48.69	93.89	41.83	93.65	46.72
Linear	94.0	49.62	94.14	42.94	94.03	47.68
MVE	92.65	45.52	93.2	**39.92**	92.75	44.23
California Housing						
Logarithm	96.42	46.85	96.41	45.04	96.59	41.52
HLI	96.44	45.88	96.41	45.03	96.36	40.05
Linear	**96.11**	44.2	96.17	43.3	96.3	**39.12**
MVE	96.55	46.72	96.54	45.82	96.12	39.69
Superconductivity						

Table A.20: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} =$ 95% and Limit $\mathcal{M} = 0.5$ (Part II)

	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	97.73	71.11	97.05	71.61	95.54	68.18
HLI	97.28	69.83	96.77	71.33	95.61	68.83
Linear	96.39	**66.97**	95.88	69.35	**95.23**	67.45
MVE	98.85	77.9	98.63	76.5	95.75	69.03
Health Insurance						
Logarithm	96.09	29.1	96.35	27.82	95.41	31.68
HLI	96.09	29.15	96.34	27.75	95.45	31.86
Linear	95.66	27.56	95.84	**25.98**	**95.23**	31.08
MVE	96.32	29.99	96.61	28.83	95.46	32.01
House8L						
Logarithm	95.82	90.83	95.9	87.1	96.09	103.43
HLI	95.9	91.11	95.97	87.33	96.09	103.49
Linear	**95.38**	89.1	95.41	**85.17**	95.41	101.06
MVE	96.22	92.77	96.32	88.96	96.88	106.66
MetroTraffic						
Logarithm	96.23	29.31	96.08	27.54	96.12	26.97
HLI	95.35	27.21	95.16	25.11	95.55	23.58
Linear	95.18	27.15	94.93	24.72	95.59	23.74
MVE	93.06	22.49	93.43	**19.75**	**94.89**	20.92
Diamonds						
Logarithm	95.16	19.32	95.37	11.61	95.18	11.6
HLI	95.1	19.18	95.38	11.62	95.09	11.43
Linear	95.09	19.22	95.21	11.39	95.12	11.52
MVE	94.96	19.04	95.36	11.62	**94.98**	11.26
Video Transcoding						
Logarithm	96.86	21.09	**94.75**	6.23	94.61	5.98
HLI	96.64	20.83	94.36	6.03	94.0	5.75
Linear	95.81	20.19	94.46	6.0	94.21	**5.73**
MVE	98.28	22.84	**94.75**	6.47	94.27	6.14
Wave Energy						
Logarithm	95.37	36.98	95.46	27.2	**95.01**	33.38
HLI	95.46	37.13	95.5	27.28	95.02	33.42
Linear	95.13	36.48	95.17	**26.77**	**94.99**	33.34
MVE	95.73	37.64	95.92	27.96	95.09	33.5
FriedmanGra						
Logarithm	95.26	36.96	95.23	27.26	95.02	33.47
HLI	95.34	37.1	95.25	27.31	94.94	33.38
Linear	95.08	36.57	95.05	**26.96**	94.97	33.4
MVE	95.59	37.57	95.62	27.92	**95.0**	33.44
FriedmanGsg						
Logarithm	94.87	30.08	94.83	22.09	94.2	29.71
HLI	94.65	29.84	94.65	21.91	93.35	28.45
Linear	94.65	29.72	94.52	**21.71**	93.69	28.88
MVE	94.86	30.17	**95.04**	22.48	92.93	27.86
FriedmanLea						
Logarithm	94.91	28.22	94.69	17.14	94.64	18.27
HLI	94.9	28.19	94.2	**16.65**	94.04	17.69
Linear	94.83	28.09	94.46	16.87	94.39	18.02
MVE	**95.03**	28.41	94.17	16.68	93.7	17.39
Hyper_a						

Table A.21: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L} = 99\%$ and Limit $\mathcal{M} = 0.5$ (Part I)

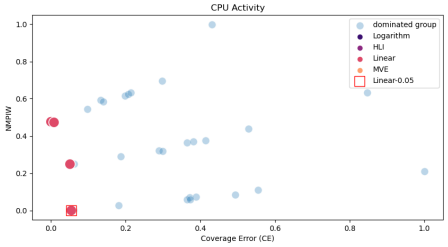
	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	98.49	52.78	98.57	45.72	98.51	51.35
HLI	98.49	58.89	98.55	51.78	98.51	57.33
Linear	98.43	54.84	**98.61**	47.98	98.53	53.26
MVE	97.57	45.97	98.03	**41.93**	97.61	44.43
Abalone						
Logarithm	99.83	44.56	99.4	34.26	99.11	41.28
HLI	99.68	44.75	99.35	36.56	99.1	45.95
Linear	99.55	40.89	99.19	**33.5**	**99.06**	42.81
MVE	99.93	53.28	99.57	35.48	**99.06**	39.88
CPU Activity						
Logarithm	98.74	17.08	99.99	4.83	99.99	4.86
HLI	98.73	16.6	99.99	**4.45**	99.99	4.48
Linear	98.76	16.5	99.99	**4.45**	99.99	4.48
MVE	**98.8**	17.41	99.99	8.91	99.99	8.97
Naval Propulsion Plant						
Logarithm	98.45	49.82	98.5	47.14	98.61	34.33
HLI	98.46	49.75	98.52	46.89	98.6	**33.97**
Linear	98.47	50.25	98.56	47.06	**98.68**	34.22
MVE	97.0	38.93	97.38	37.73	97.58	29.97
Ailerons						
Logarithm	99.15	48.34	99.2	43.88	99.17	40.5
HLI	99.17	49.12	99.2	44.11	99.17	41.19
Linear	**99.05**	48.68	99.15	43.39	99.07	40.95
MVE	98.04	38.79	98.13	36.26	98.16	**32.85**
Miami Housing						
Logarithm	98.77	7.4	98.78	6.47	98.88	6.2
HLI	98.77	7.47	98.77	6.56	98.87	6.26
Linear	98.8	7.44	98.81	6.51	**98.89**	6.4
MVE	98.12	6.2	98.39	**5.52**	98.41	**5.52**
NZ Energy Price						
Logarithm	99.12	8.84	99.18	8.74	99.07	10.41
HLI	99.12	8.86	99.17	8.71	99.08	10.39
Linear	99.12	8.79	99.17	8.62	99.06	10.45
MVE	98.88	**8.3**	**98.96**	8.33	98.65	9.42
Elevators						
Logarithm	98.29	57.64	98.31	49.07	98.27	75.03
HLI	98.29	58.04	**98.33**	49.25	98.31	75.49
Linear	97.96	56.76	98.1	48.34	98.04	74.09
MVE	93.49	37.96	94.33	**33.99**	93.14	51.38
Bike						
Logarithm	98.42	78.02	98.43	69.15	98.46	75.98
HLI	98.45	78.72	98.45	69.5	**98.49**	76.04
Linear	98.37	78.46	98.37	68.52	98.44	75.45
MVE	96.06	59.82	96.29	**52.46**	96.17	58.13
California Housing						
Logarithm	98.82	62.17	98.84	60.09	98.94	56.25
HLI	98.8	61.5	98.81	60.11	98.95	56.03
Linear	98.84	61.16	98.82	59.85	**98.97**	56.05
MVE	98.79	61.4	98.86	60.22	98.36	**52.17**
Superconductivity						

Table A.22: Coverage (%) and NMPIW (%) Results: Confidence Level $\mathcal{L}$ = 99% and Limit $\mathcal{M} = 0.5$ (Part II)

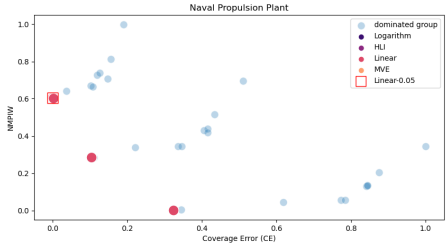
	ARF-Reg		SOKNL		KNN	
	coverage	NMPIW	coverage	NMPIW	coverage	NMPIW
Logarithm	99.61	93.17	99.68	89.43	99.23	88.26
HLI	99.52	94.43	99.56	90.35	99.24	91.87
Linear	99.35	86.84	99.39	**83.1**	**99.1**	88.22
MVE	99.79	102.38	99.82	100.54	99.43	90.72
Health Insurance						
Logarithm	98.45	49.54	98.5	46.28	98.42	54.76
HLI	98.47	50.2	98.52	47.02	98.44	55.57
Linear	98.46	49.66	**98.54**	46.65	98.42	54.48
MVE	97.72	39.41	97.91	**37.89**	97.31	42.07
House8L						
Logarithm	99.45	114.99	99.42	110.85	99.49	128.57
HLI	99.37	114.33	99.35	110.15	99.4	127.61
Linear	99.16	111.6	**99.15**	**108.03**	99.2	124.09
MVE	99.76	121.92	99.75	116.91	99.88	140.17
MetroTraffic						
Logarithm	**98.98**	43.63	98.84	43.52	98.51	56.32
HLI	98.86	45.67	98.79	44.77	98.62	56.0
Linear	98.47	46.51	98.27	44.83	98.07	50.83
MVE	95.51	29.56	94.96	**25.96**	95.96	27.49
Diamonds						
Logarithm	98.37	35.24	98.29	20.9	98.33	21.05
HLI	**98.41**	35.47	98.33	21.06	98.36	21.28
Linear	98.32	34.6	98.27	20.61	98.29	20.67
MVE	96.91	25.03	97.07	15.27	96.84	**14.8**
Video Transcoding						
Logarithm	99.61	26.28	98.65	8.92	98.63	8.25
HLI	99.51	25.82	98.65	8.8	98.63	8.14
Linear	99.31	24.92	**98.7**	8.84	98.68	8.16
MVE	99.93	30.02	98.05	8.51	98.15	**8.07**
Wave Energy						
Logarithm	99.07	48.26	98.92	**36.35**	98.95	44.18
HLI	99.08	48.69	98.89	36.52	98.93	44.38
Linear	**99.03**	47.96	98.94	36.46	**98.97**	44.33
MVE	99.2	49.47	98.92	36.74	98.93	44.03
FriedmanGra						
Logarithm	99.03	48.37	98.89	**36.58**	98.93	44.33
HLI	99.03	48.73	98.87	36.74	98.9	44.48
Linear	**99.01**	48.24	98.92	36.74	98.96	44.58
MVE	99.13	49.38	98.83	36.69	98.85	43.95
FriedmanGsg						
Logarithm	98.8	40.13	98.57	**32.04**	98.41	40.86
HLI	98.78	40.33	98.58	32.43	98.43	41.19
Linear	**98.86**	40.56	98.57	32.34	98.42	41.07
MVE	98.61	39.64	97.92	29.54	97.31	36.62
FriedmanLea						
Logarithm	98.84	38.09	98.61	24.65	98.63	25.56
HLI	98.8	37.95	98.62	24.65	98.63	25.58
Linear	**98.88**	38.29	98.67	24.91	98.7	25.86
MVE	98.65	37.34	98.05	**8.51**	97.67	22.85
Hyper_a						

A.2 Full CING Results

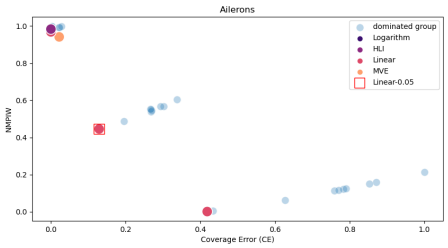
This appendix includes the supplementary figures for CING results.



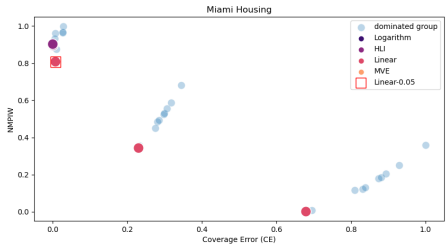
(a) CPU Activity



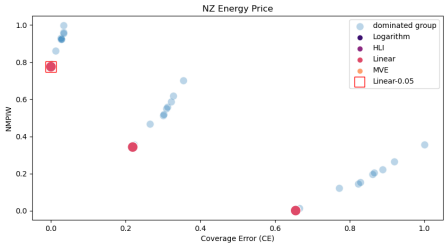
(b) Naval Propulsion Plant



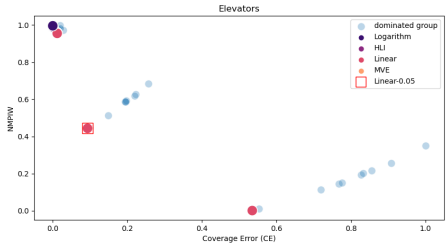
(c) Ailerons



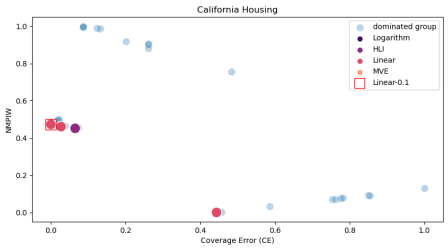
(d) Miami Housing



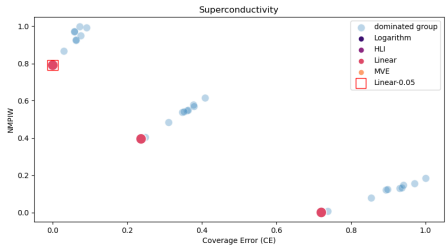
(e) NZ Energy Price



(f) Elevators

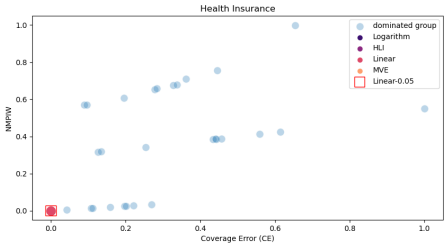


(g) California Housing

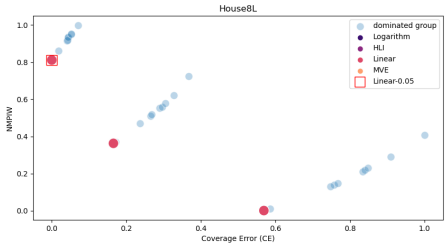


(h) Superconductivity

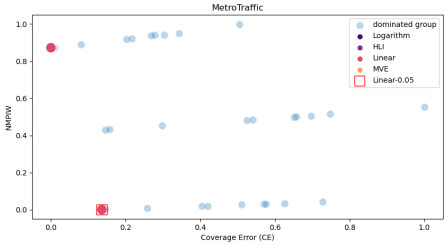
Figure A.1: Supplementary CING Evaluation Results for Different Datasets (Part I)



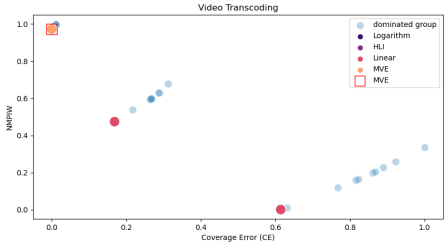
(a) Health Insurance



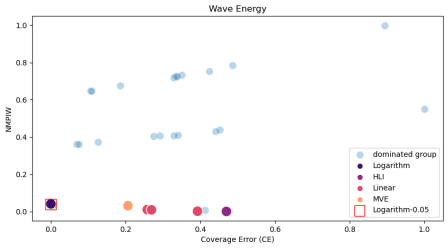
(b) House8L



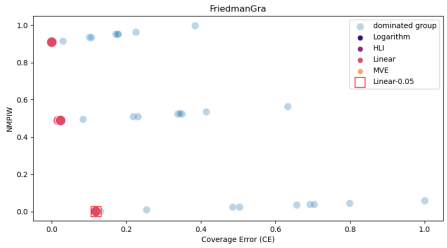
(c) MetroTraffic



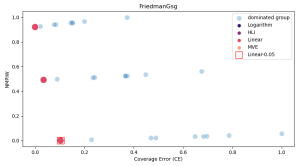
(d) Video Transcoding



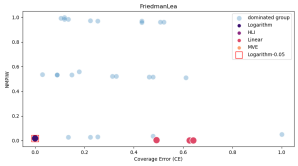
(e) Wave Energy



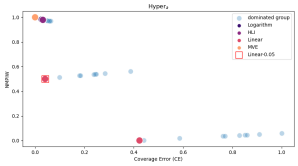
(f) FriedmanGra



(g) FriedmanGsg



(h) FriedmanLea

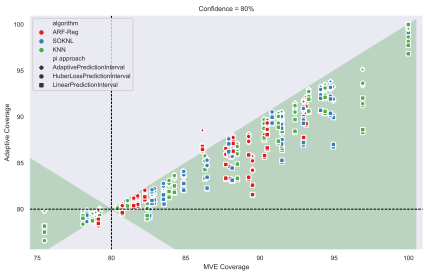


(i) Hyper$_a$

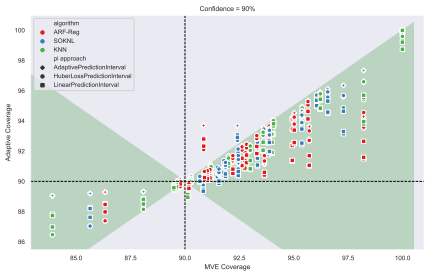
Figure A.2: Supplementary CING Evaluation Results for Different Datasets (Part II)

A.3 Full Coverage Results Comparison

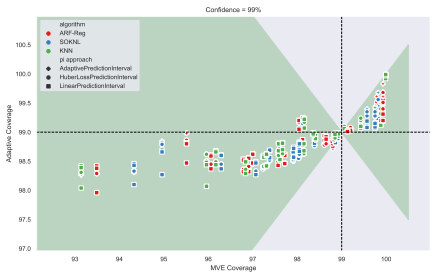
We present scattergram for coverage results comparison for $\mathcal{L} = 80, 90$, and 99 in this appendix.



(a) Scattergram for $\mathcal{L} = 80\%$.



(b) Scattergram for $\mathcal{L} = 90\%$.



(c) Scattergram for $\mathcal{L} = 99\%$.

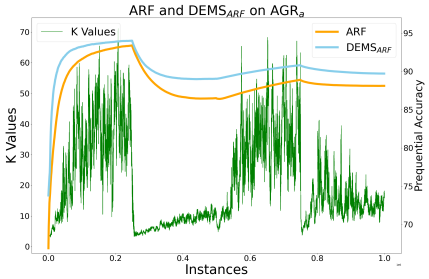
Figure A.3: Scattergrams showing the comparison of coverage values between MVE and adaptive approaches for different confidence levels ($\mathcal{L} = 80\%$, 90%, 99%). The black lines denote the confidence level for both axes, and the green area highlights where adaptive approaches outperform MVE in terms of coverage. Different colors represent different base learners, and different shapes distinguish the adaptive approaches.

Appendix B

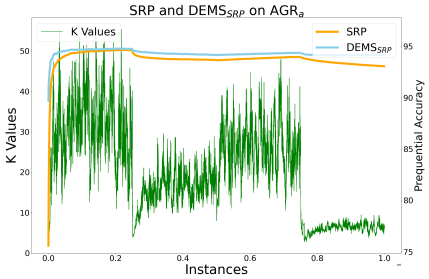
Chapter 6 Additional Results

B.1 Full $\mathcal{K}$ Values Plots

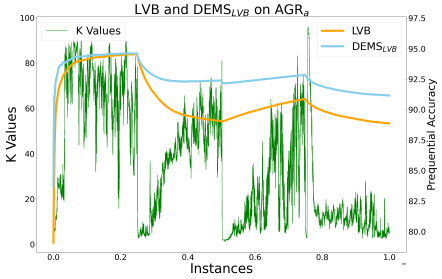
This appendix section exhibits the plots of prequential accuracy and $\mathcal{K}$ values for both original ensemble and the DEMS version algorithms not included in Chapter 6.



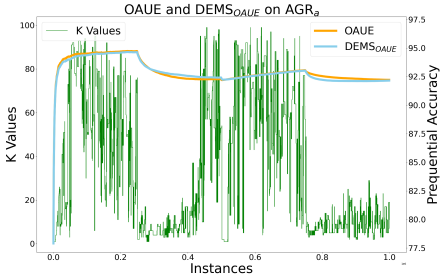
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on AGR$_a$



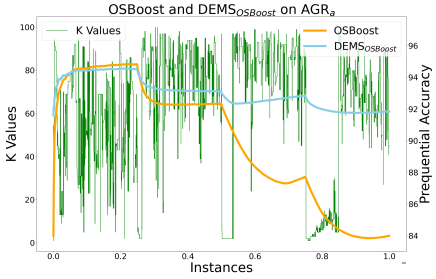
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on AGR$_a$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on AGR$_a$

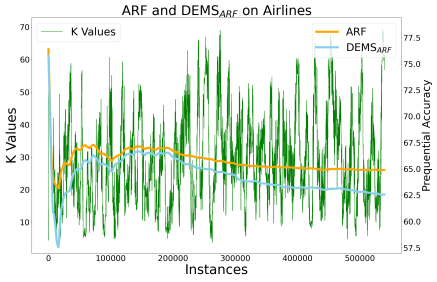


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on AGR$_a$

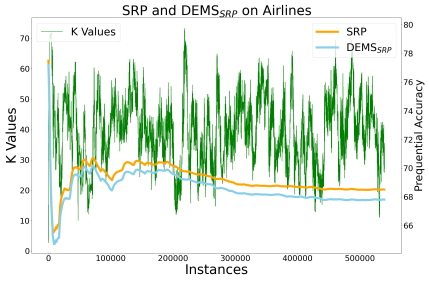


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on AGR$_a$

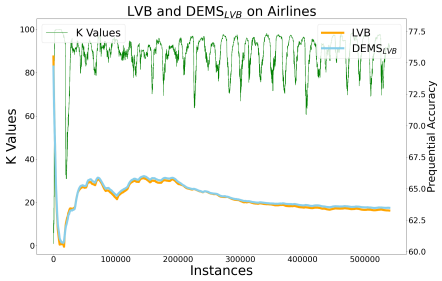
Figure B.1: AGR$_a$ dataset: prequential accuracy and K values over time



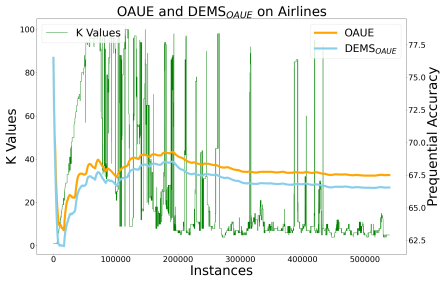
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on Airlines



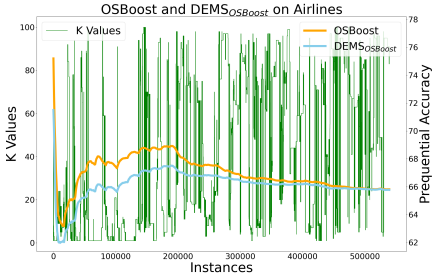
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on Airlines



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on Airlines

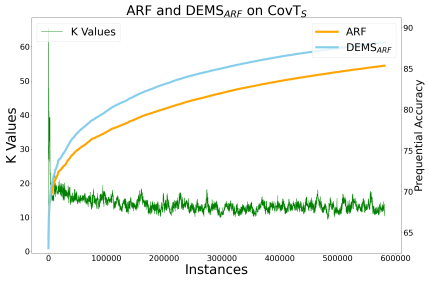


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on Airlines

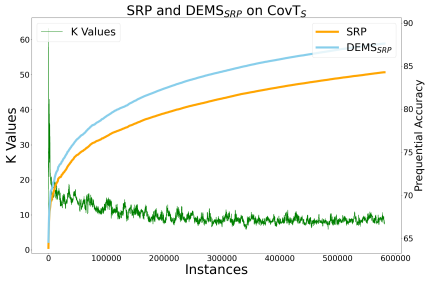


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on Airlines

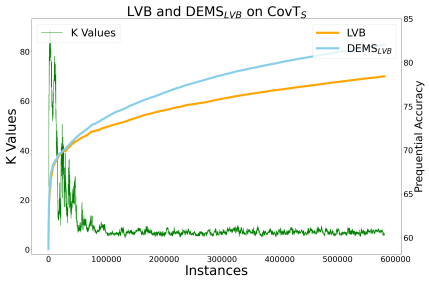
Figure B.2: Airlines dataset: prequential accuracy and K values over time



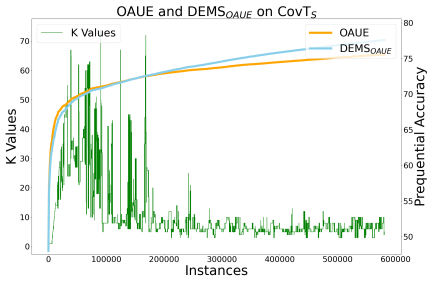
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on CovT$_S$



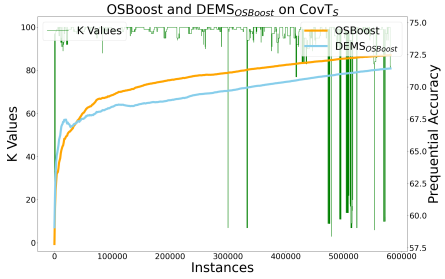
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on CovT$_S$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on CovT$_S$

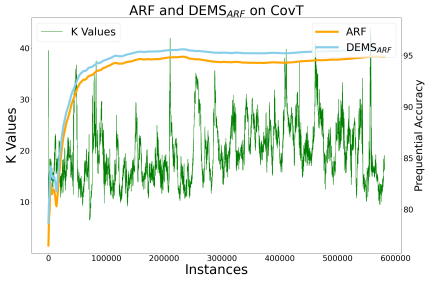


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on CovT$_S$

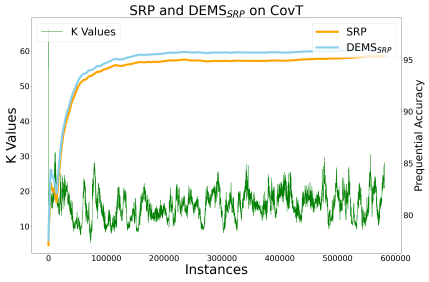


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on CovT$_S$

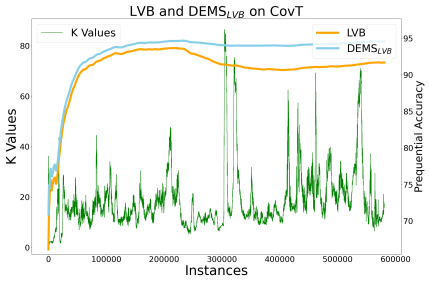
Figure B.3: CovT$_S$ dataset: prequential accuracy and K values over time



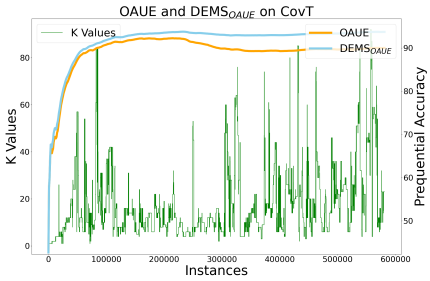
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on CovT



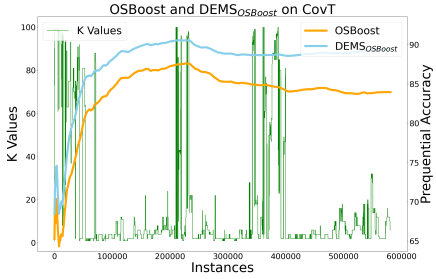
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on CovT



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on CovT

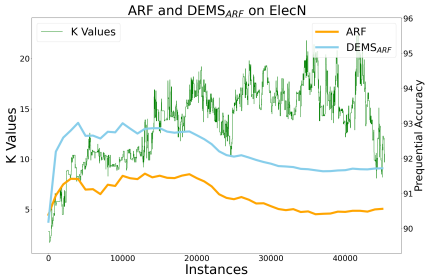


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on CovT

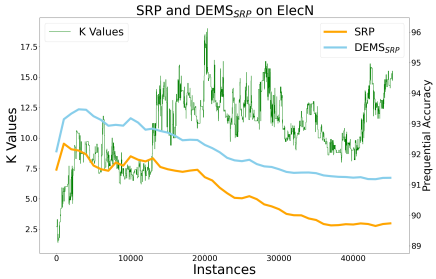


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on CovT

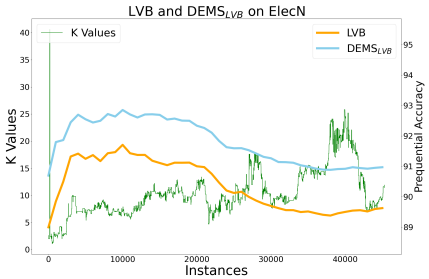
Figure B.4: CovT dataset: prequential accuracy and K values over time



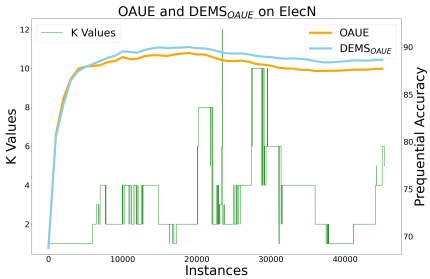
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on ElecN



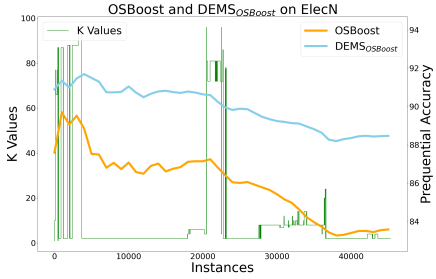
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on ElecN



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on ElecN

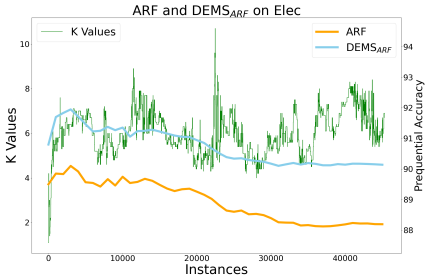


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on ElecN

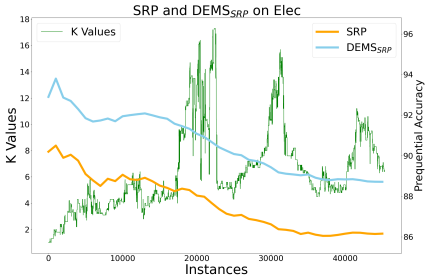


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on ElecN

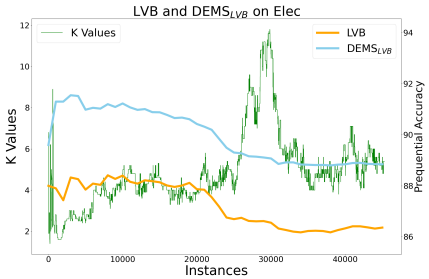
Figure B.5: ElecN dataset: prequential accuracy and K values over time



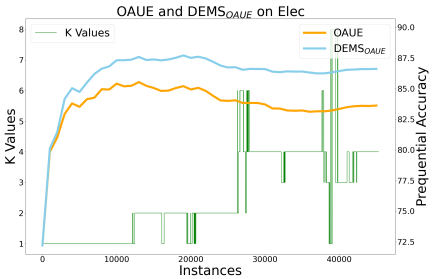
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on Elec



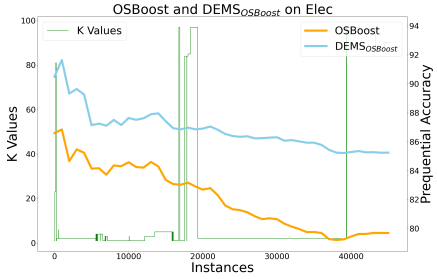
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on Elec



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on Elec

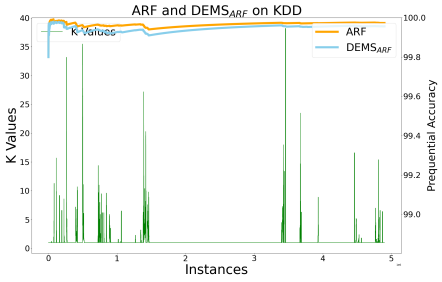


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on Elec

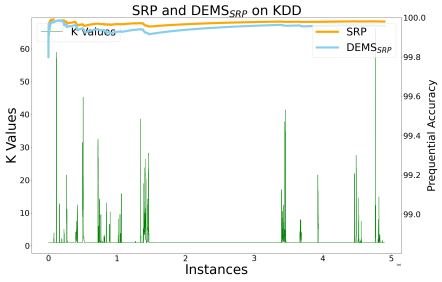


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on Elec

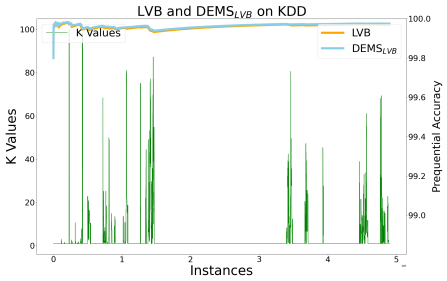
Figure B.6: Elec dataset: prequential accuracy and K values over time



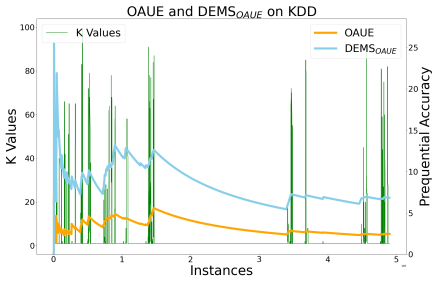
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on KDD



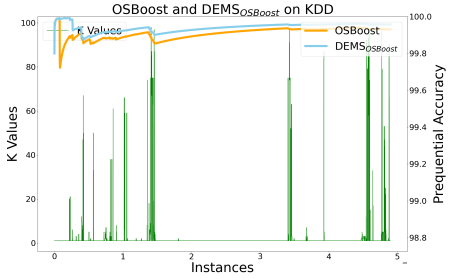
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on KDD



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on KDD

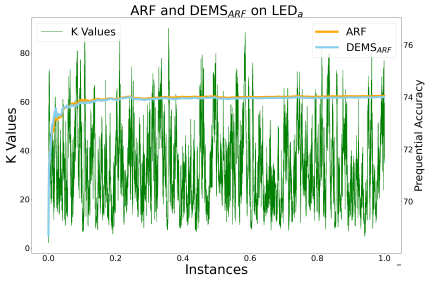


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on KDD

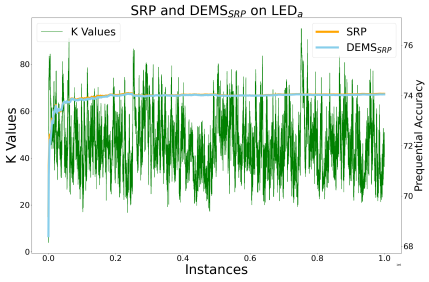


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on KDD

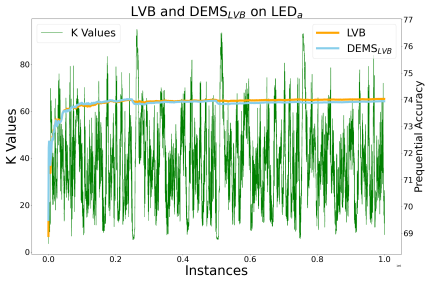
Figure B.7: KDD dataset: prequential accuracy and K values over time



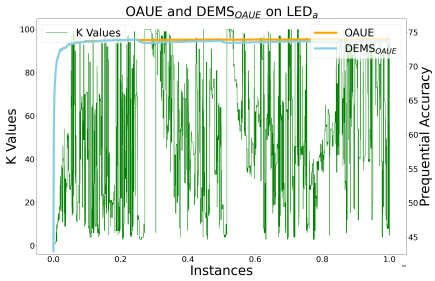
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on LED$_a$



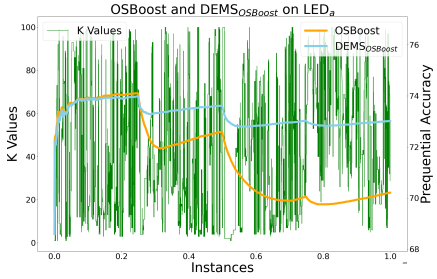
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on LED$_a$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on LED$_a$

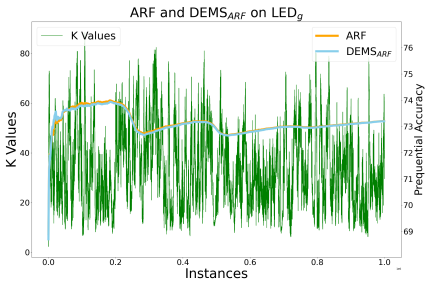


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on LED$_a$

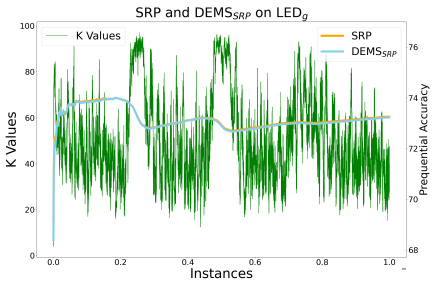


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on LED$_a$

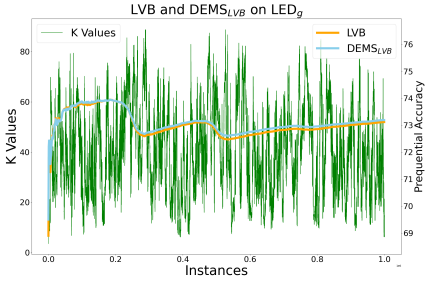
Figure B.8: LED$_a$ dataset: prequential accuracy and K values over time



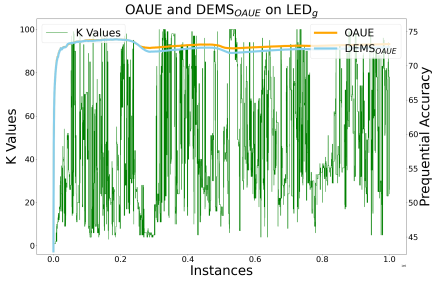
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on LED$_g$



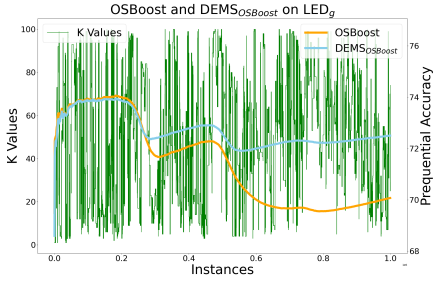
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on LED$_g$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on LED$_g$

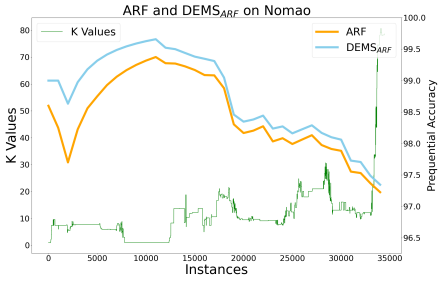


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on LED$_g$

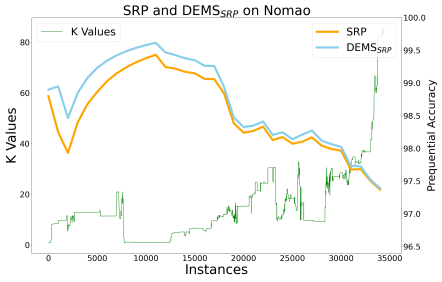


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on LED$_g$

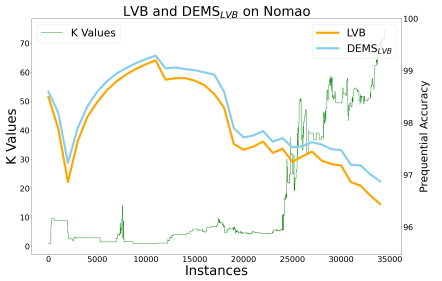
Figure B.9: LED$_g$ dataset: prequential accuracy and K values over time



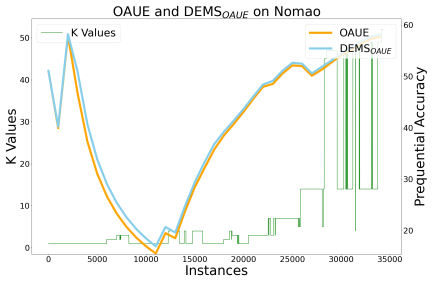
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on Nomao



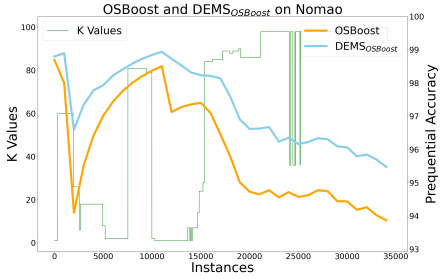
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on Nomao



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on Nomao

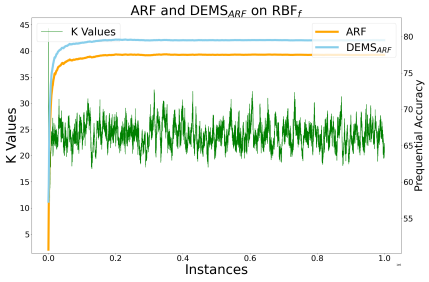


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on Nomao

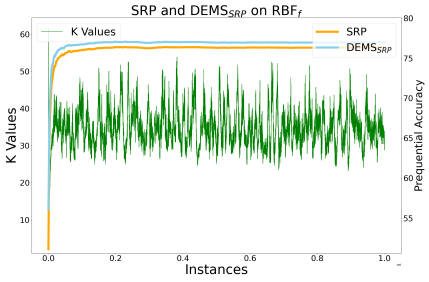


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on Nomao

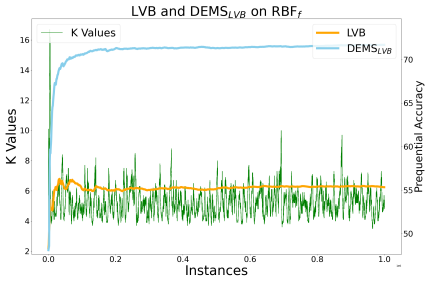
Figure B.10: Nomao dataset: prequential accuracy and K values over time



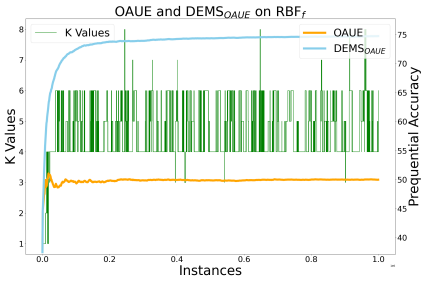
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on RBF$_f$



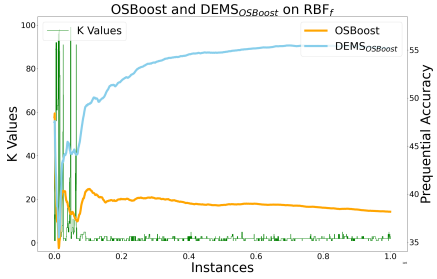
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on RBF$_f$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on RBF$_f$

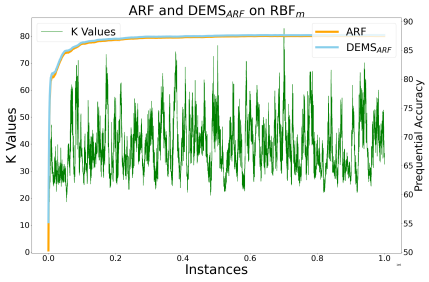


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on RBF$_f$

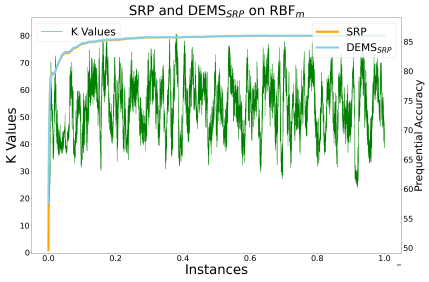


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on RBF$_f$

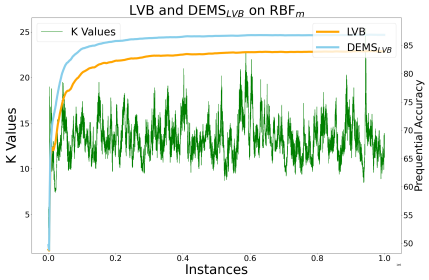
Figure B.11: RBF$_f$ dataset: prequential accuracy and K values over time



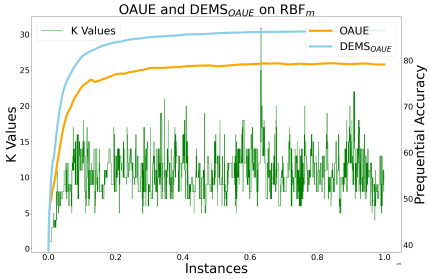
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on RBF$_m$



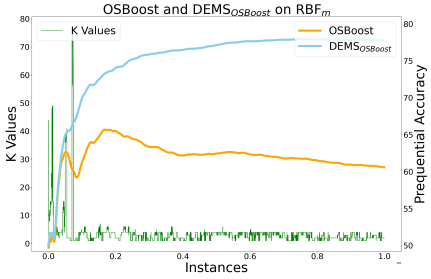
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on RBF$_m$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on RBF$_m$

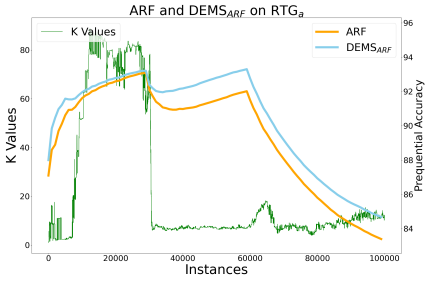


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on RBF$_m$

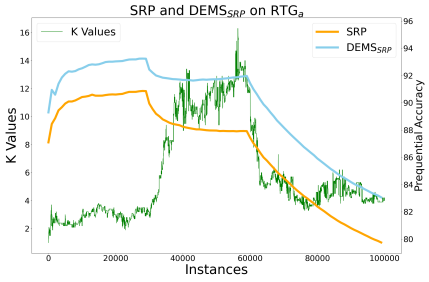


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on RBF$_m$

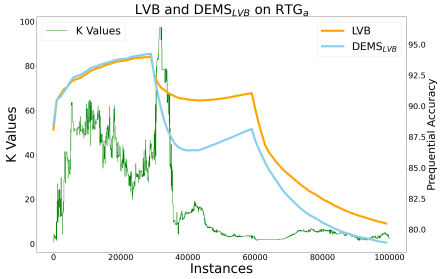
Figure B.12: RBF$_m$ dataset: prequential accuracy and K values over time



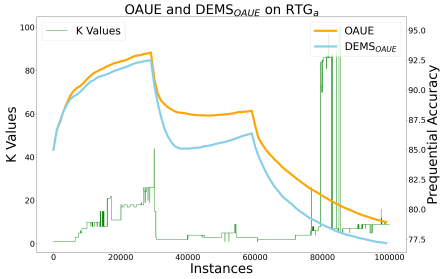
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on RTG$_a$



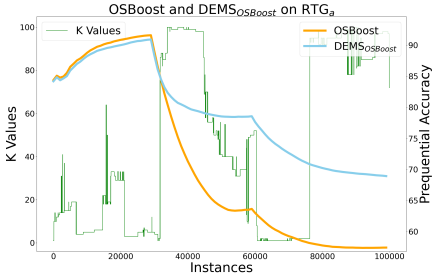
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on RTG$_a$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on RTG$_a$

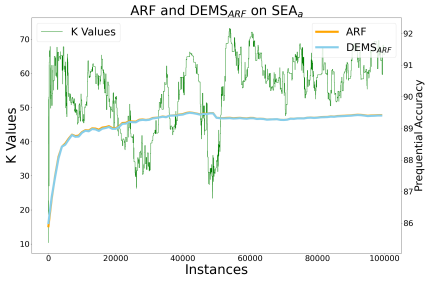


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on RTG$_a$

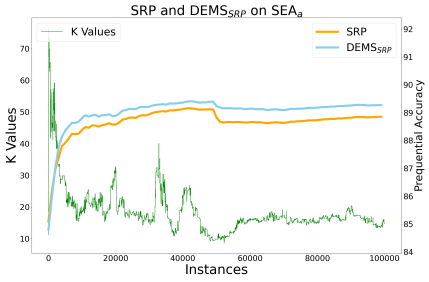


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on RTG$_a$

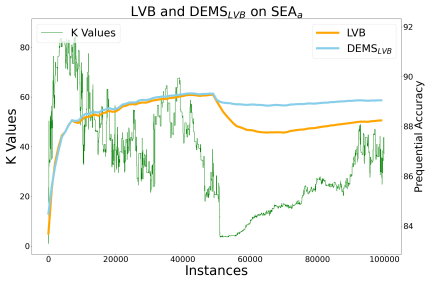
Figure B.13: RTG$_a$ dataset: prequential accuracy and K values over time



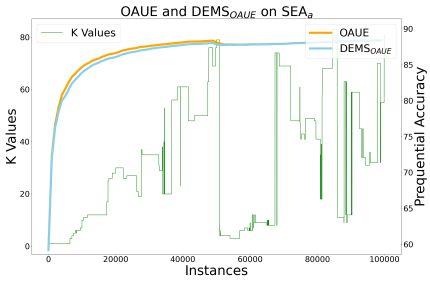
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on SEA$_a$



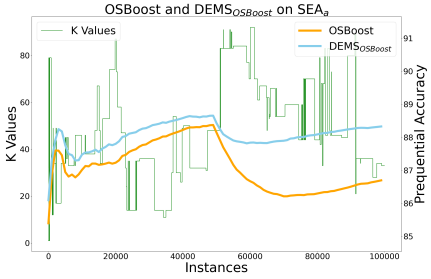
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on SEA$_a$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on SEA$_a$

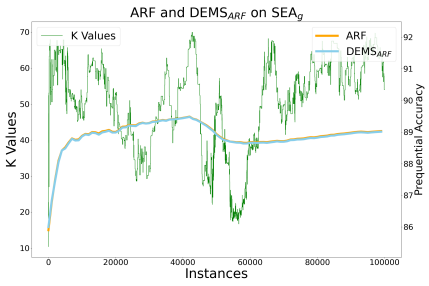


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on SEA$_a$

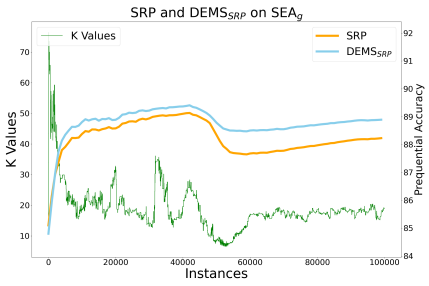


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on SEA$_a$

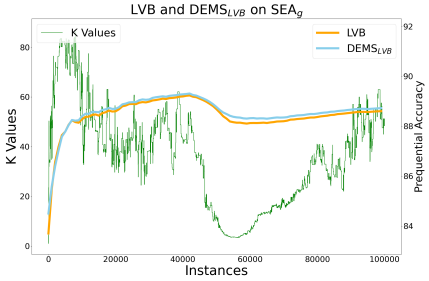
Figure B.14: SEA$_a$ dataset: prequential accuracy and K values over time



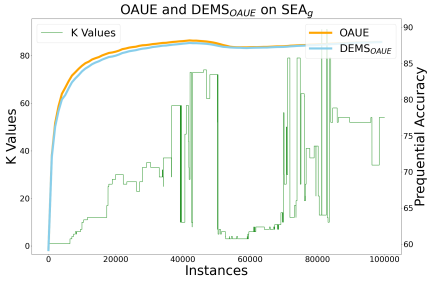
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on SEA$_g$



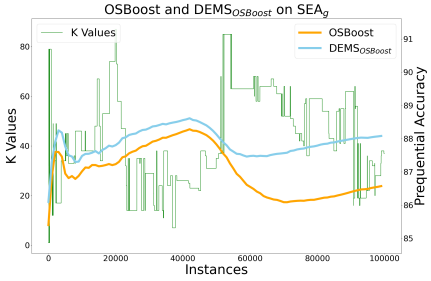
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on SEA$_g$



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on SEA$_g$

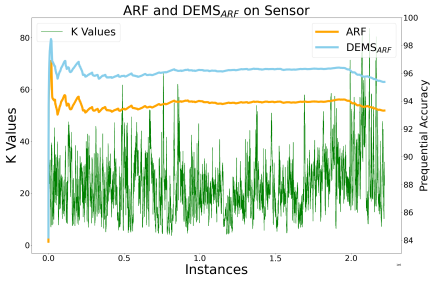


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on SEA$_g$

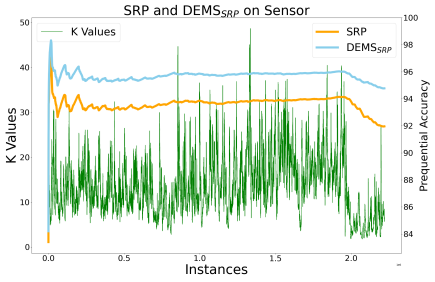


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on SEA$_g$

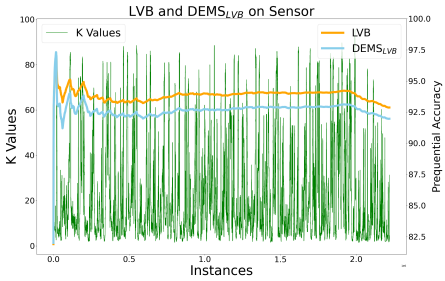
Figure B.15: SEA$_g$ dataset: prequential accuracy and K values over time



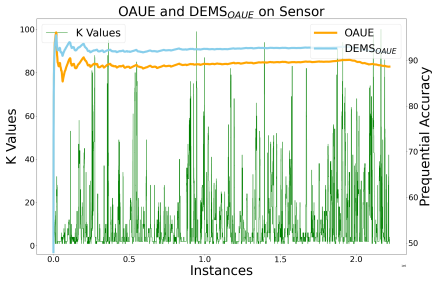
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on Sensor



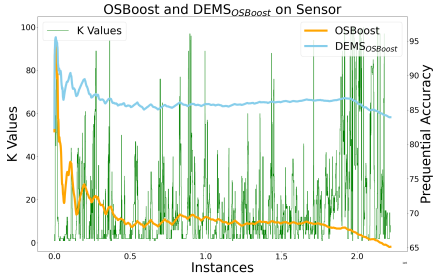
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on Sensor



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on Sensor

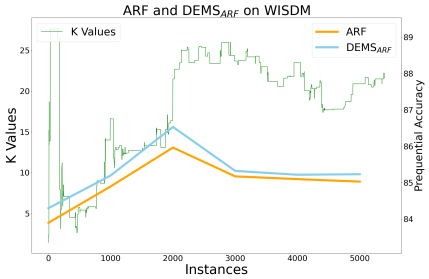


(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on Sensor

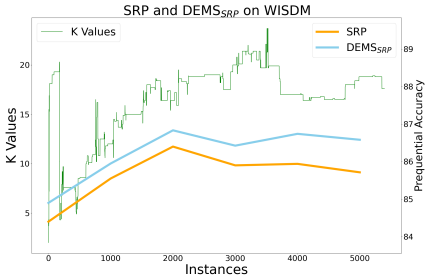


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on Sensor

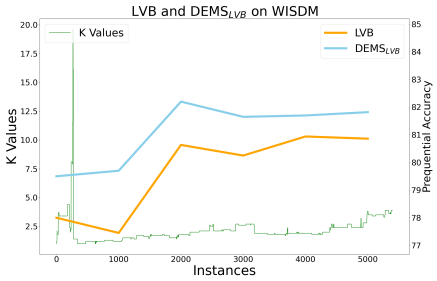
Figure B.16: Sensor dataset: prequential accuracy and K values over time



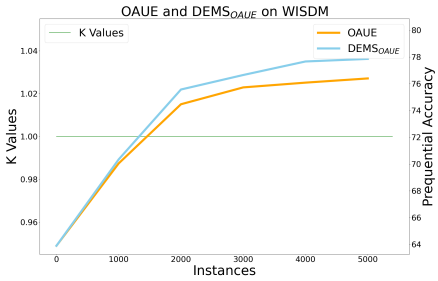
(a) Prequential accuracy for ARF and DEMS$_{ARF}$, and best K values over time on WISDM



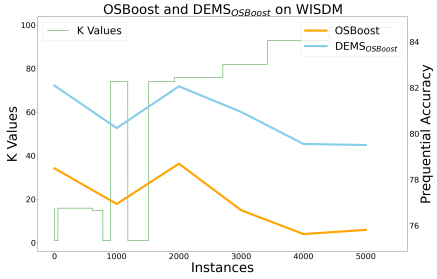
(b) Prequential accuracy for SRP and DEMS$_{SRP}$, and best K values over time on WISDM



(c) Prequential accuracy for LVB and DEMS$_{LVB}$, and best K values over time on WISDM



(d) Prequential accuracy for OAUE and DEMS$_{OAUE}$, and best K values over time on WISDM

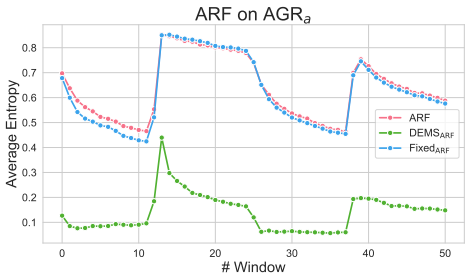


(e) Prequential accuracy for OSBoost and DEMS$_{OSBoost}$, and best K values over time on WISDM

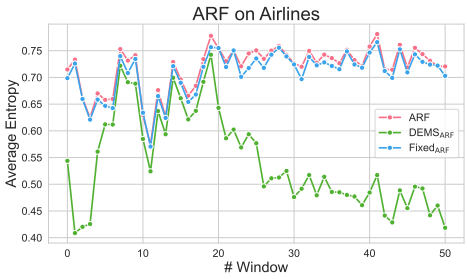
Figure B.17: WISDM dataset: prequential accuracy and K values over time

B.2 Full Entropy Diversity Results

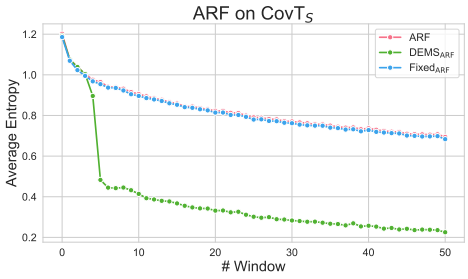
This appendix section provides all the plots of the entropy diversity executed for DEMS and relevant algorithms.



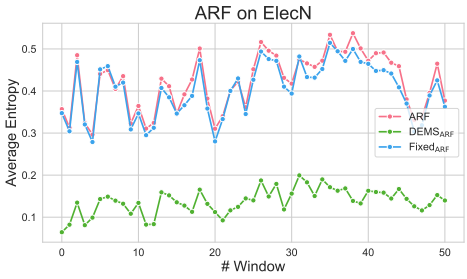
(a) Entropy of ARF on AGR$_a$



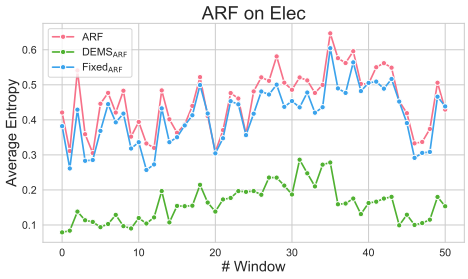
(b) Entropy of ARF on Airlines



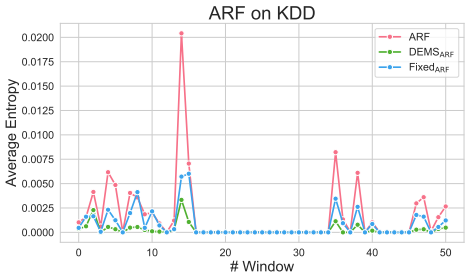
(c) Entropy of ARF on CovT$_S$



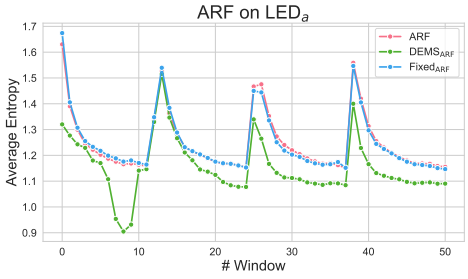
(d) Entropy of ARF on ElecN



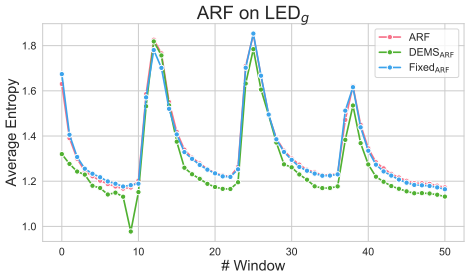
(e) Entropy of ARF on Elec



(f) Entropy of ARF on KDD

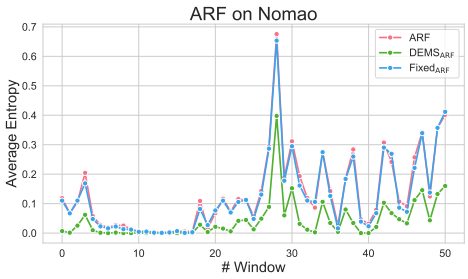


(g) Entropy of ARF on LED$_a$

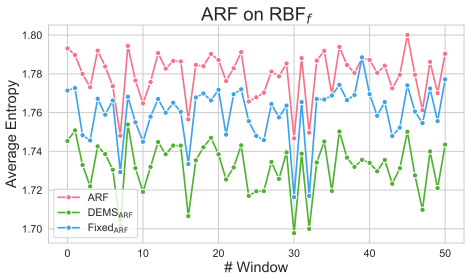


(h) Entropy of ARF on LED$_g$

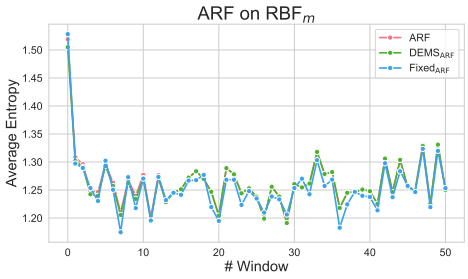
Figure B.18: List of Entropy Plots (Part I)



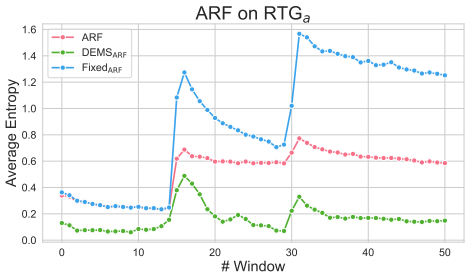
(a) Entropy of ARF on Nomao



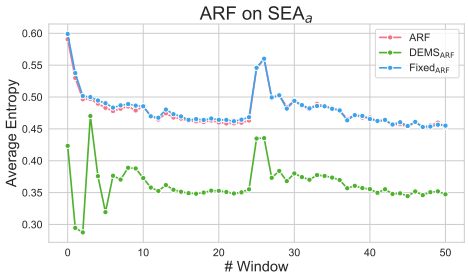
(b) Entropy of ARF on RBF$_f$



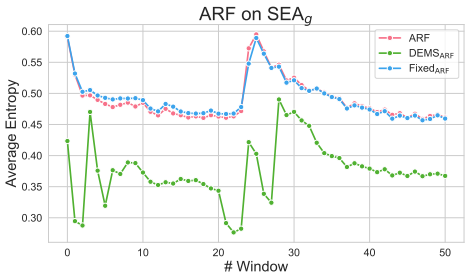
(c) Entropy of ARF on RBF$_m$



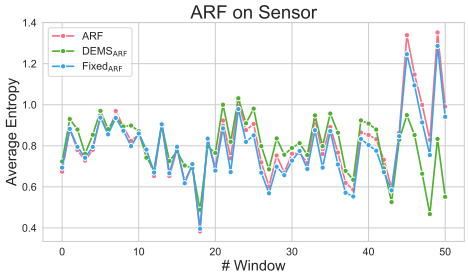
(d) Entropy of ARF on RTG$_a$



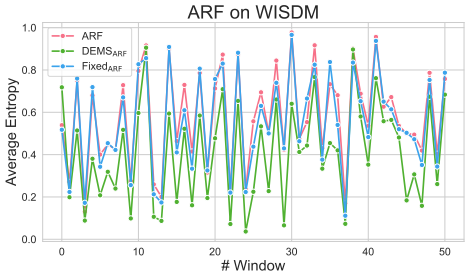
(e) Entropy of ARF on SEA$_a$



(f) Entropy of ARF on SEA$_g$

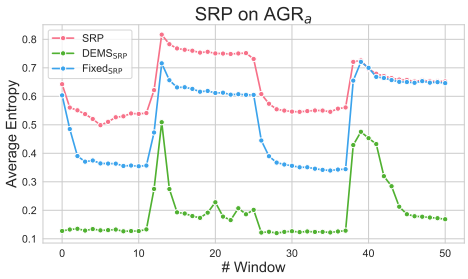


(g) Entropy of ARF on Sensor

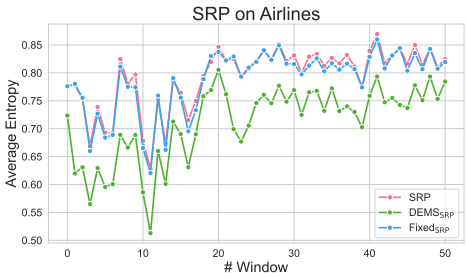


(h) Entropy of ARF on WISDM

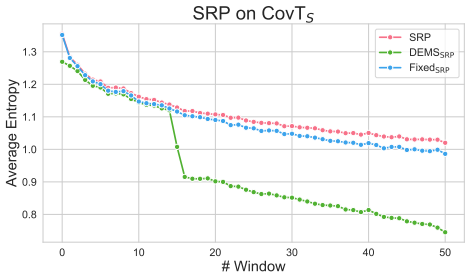
Figure B.19: List of Entropy Plots (Part II)



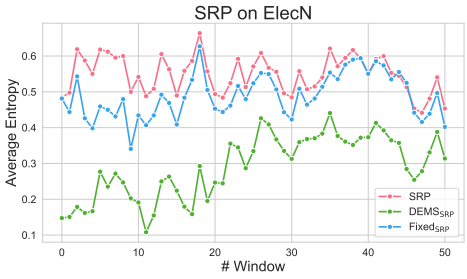
(a) Entropy of SRP on AGR$_a$



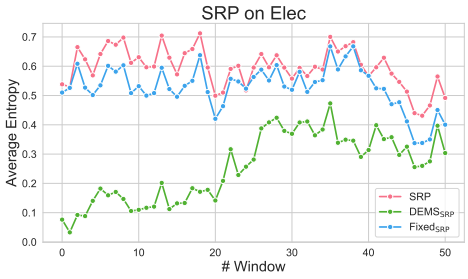
(b) Entropy of SRP on Airlines



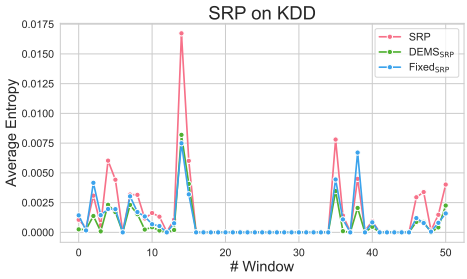
(c) Entropy of SRP on CovT$_S$



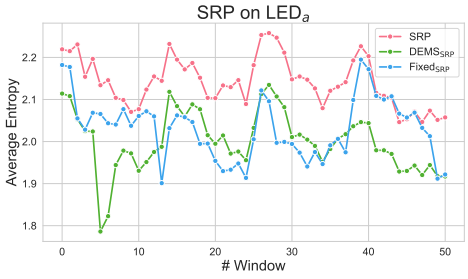
(d) Entropy of SRP on ElecN



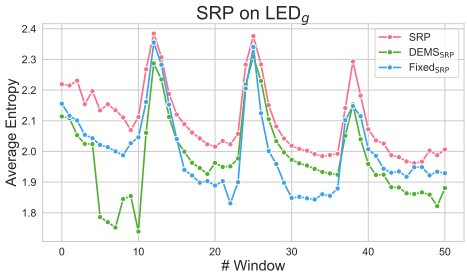
(e) Entropy of SRP on Elec



(f) Entropy of SRP on KDD

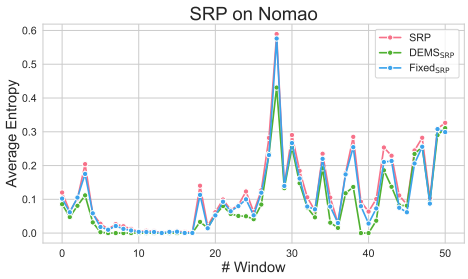


(g) Entropy of SRP on LED$_a$

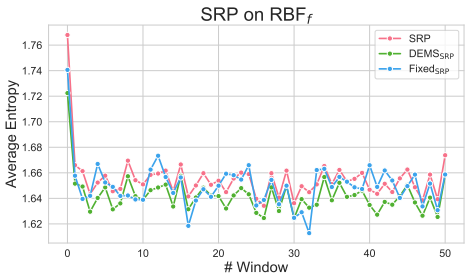


(h) Entropy of SRP on LED$_g$

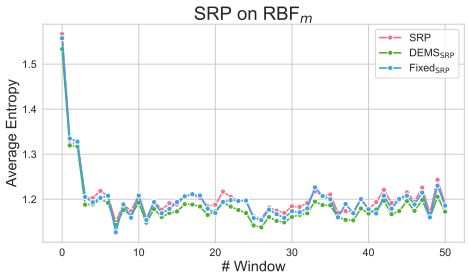
Figure B.20: List of Entropy Plots (Part III)



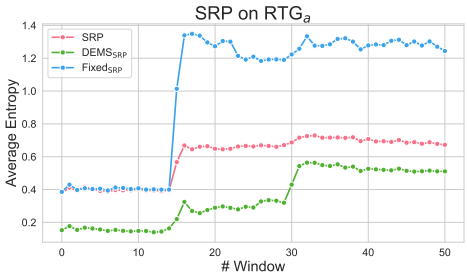
(a) Entropy of SRP on Nomao



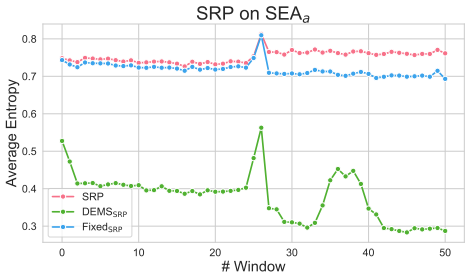
(b) Entropy of SRP on RBF$_f$



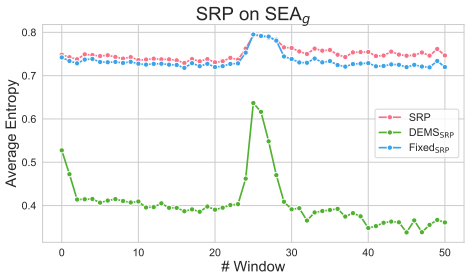
(c) Entropy of SRP on RBF$_m$



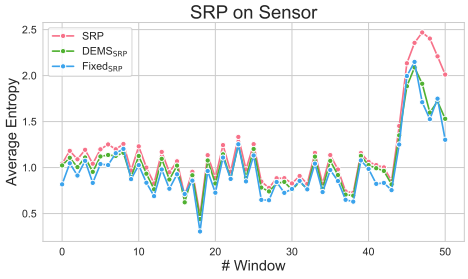
(d) Entropy of SRP on RTG$_a$



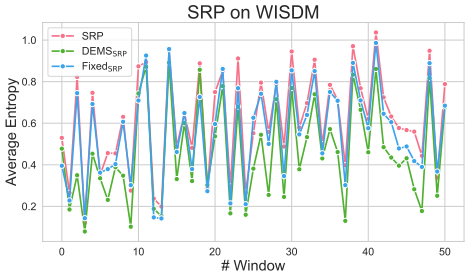
(e) Entropy of SRP on SEA$_a$



(f) Entropy of SRP on SEA$_g$

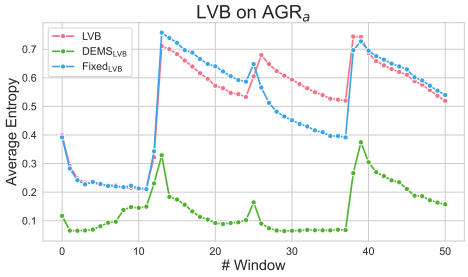


(g) Entropy of SRP on Sensor

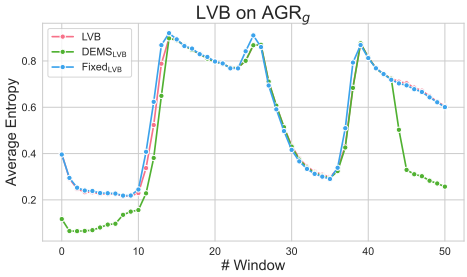


(h) Entropy of SRP on WISDM

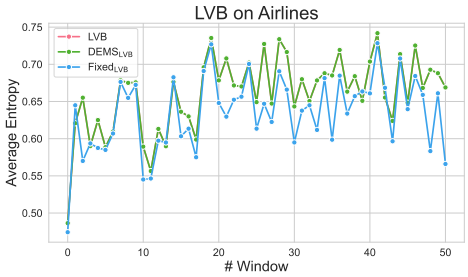
Figure B.21: List of Entropy Plots (Part IV)



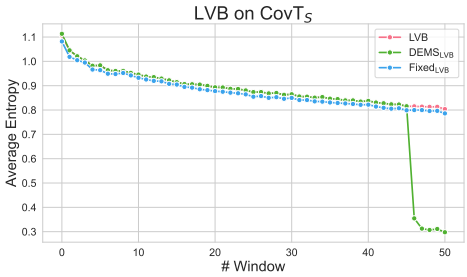
(a) Entropy of LVB on AGR$_a$



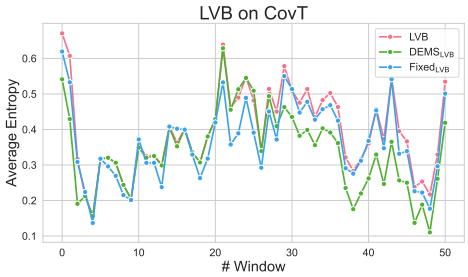
(b) Entropy of LVB on AGR$_g$



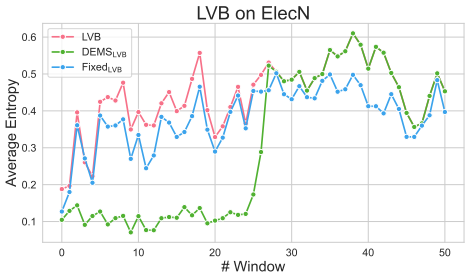
(c) Entropy of LVB on Airlines



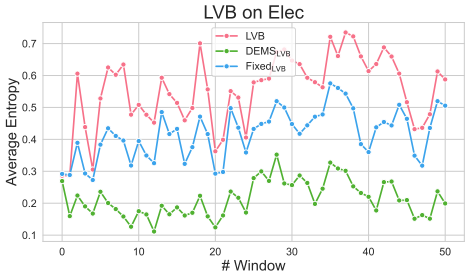
(d) Entropy of LVB on CovT$_S$



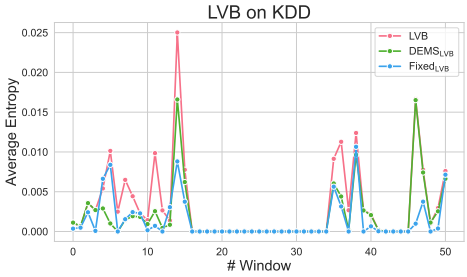
(e) Entropy of LVB on CovT



(f) Entropy of LVB on ElecN

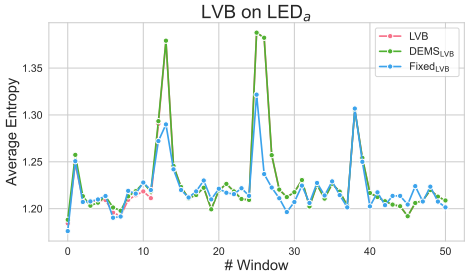


(g) Entropy of LVB on Elec

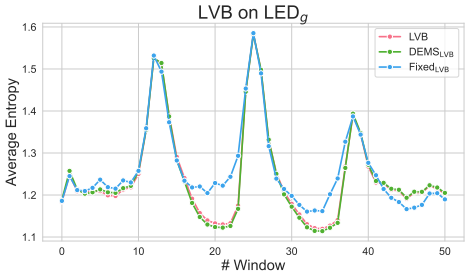


(h) Entropy of LVB on KDD

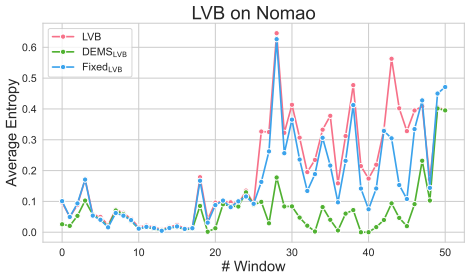
Figure B.22: List of Entropy Plots (Part V)



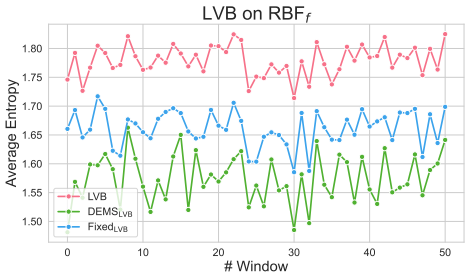
(a) Entropy of LVB on LED$_a$



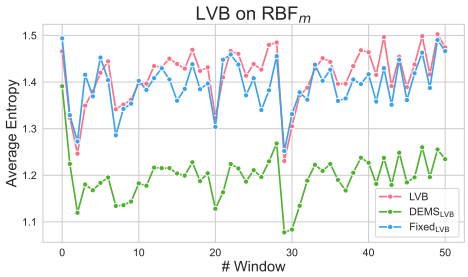
(b) Entropy of LVB on LED$_g$



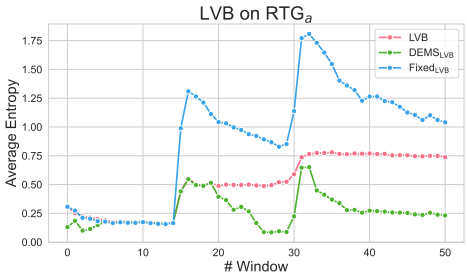
(c) Entropy of LVB on Nomao



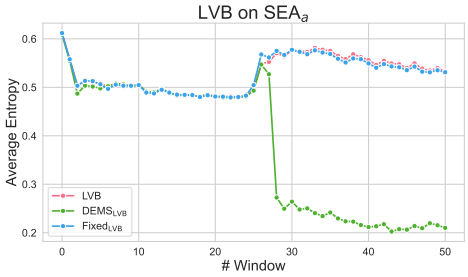
(d) Entropy of LVB on RBF$_f$



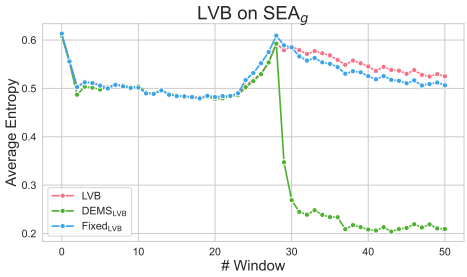
(e) Entropy of LVB on RBF$_m$



(f) Entropy of LVB on RTG$_a$

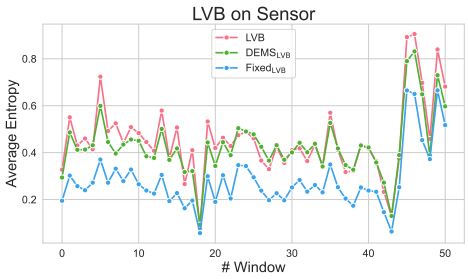


(g) Entropy of LVB on SEA$_a$

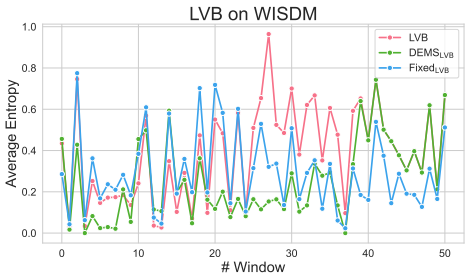


(h) Entropy of LVB on SEA$_g$

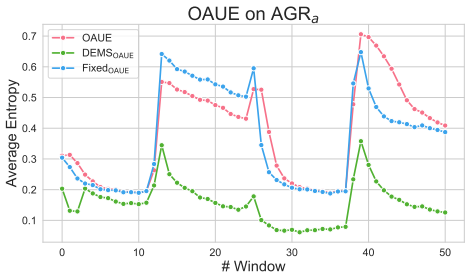
Figure B.23: List of Entropy Plots (Part VI)



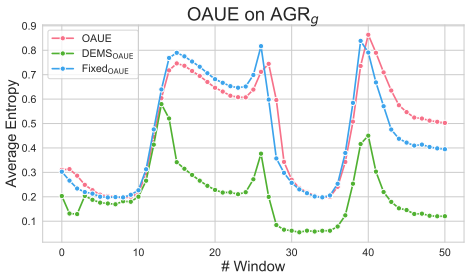
(a) Entropy of LVB on Sensor



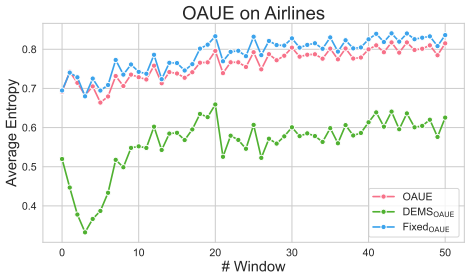
(b) Entropy of LVB on WISDM



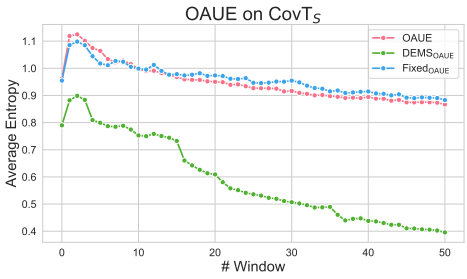
(c) Entropy of OAUE on AGR$_a$



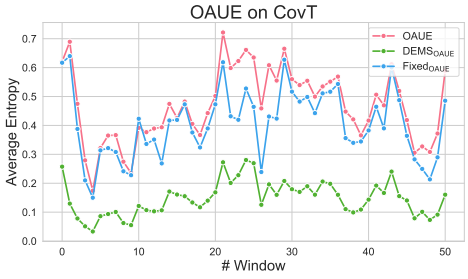
(d) Entropy of OAUE on AGR$_g$



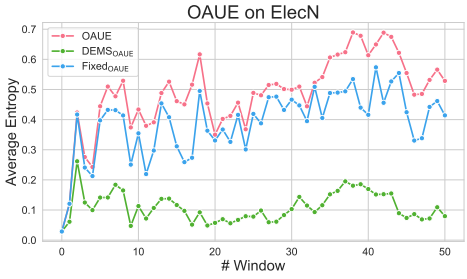
(e) Entropy of OAUE on Airlines



(f) Entropy of OAUE on CovT$_S$

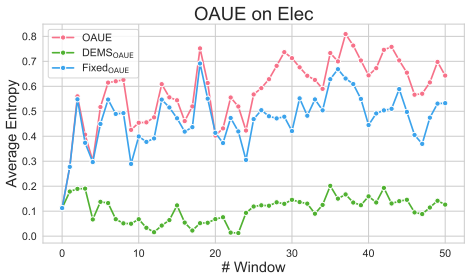


(g) Entropy of OAUE on CovT

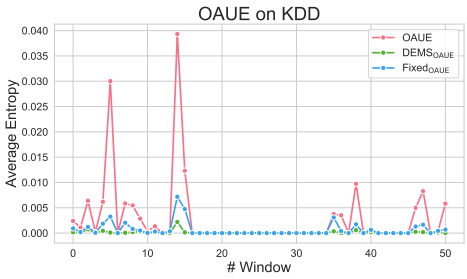


(h) Entropy of OAUE on ElecN

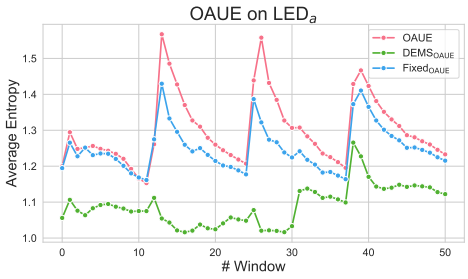
Figure B.24: List of Entropy Plots (Part VII)



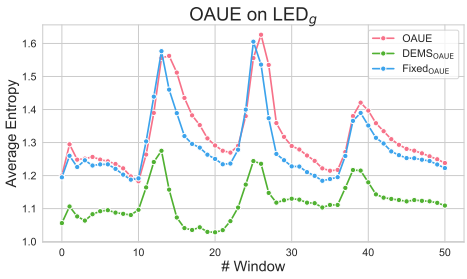
(a) Entropy of OAUE on Elec



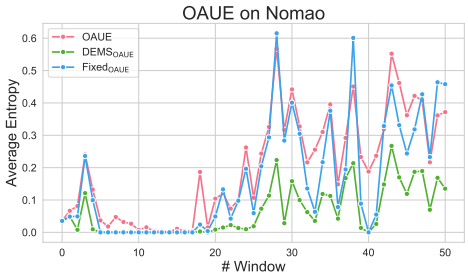
(b) Entropy of OAUE on KDD



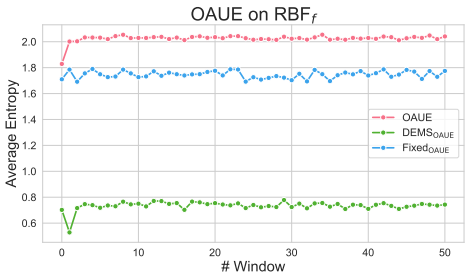
(c) Entropy of OAUE on LED$_a$



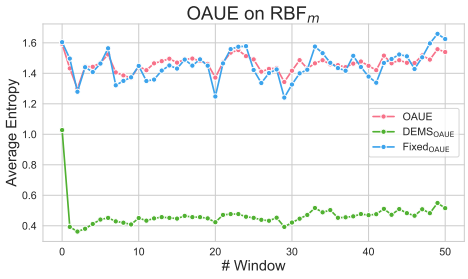
(d) Entropy of OAUE on LED$_g$



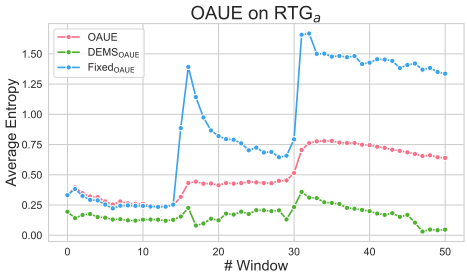
(e) Entropy of OAUE on Nomao



(f) Entropy of OAUE on RBF$_f$

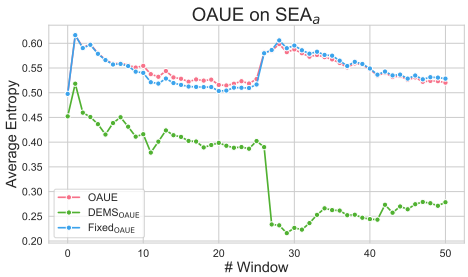


(g) Entropy of OAUE on RBF$_m$

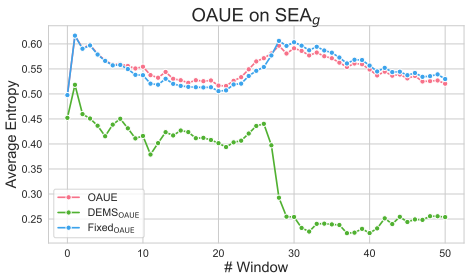


(h) Entropy of OAUE on RTG$_a$

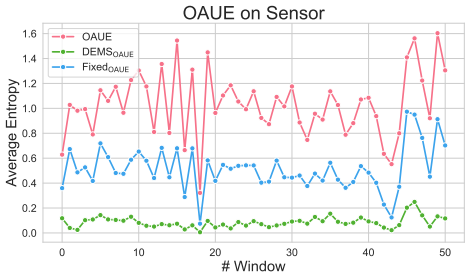
Figure B.25: List of Entropy Plots (Part VIII)



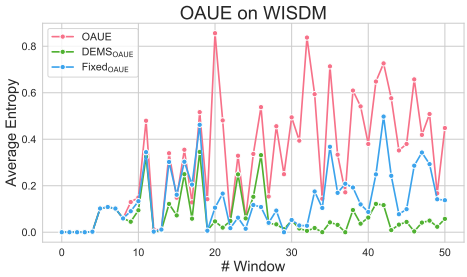
(a) Entropy of OAUE on SEA$_a$



(b) Entropy of OAUE on SEA$_g$

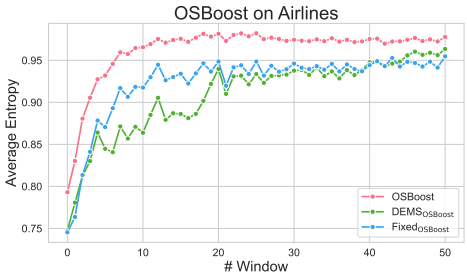


(c) Entropy of OAUE on Sensor

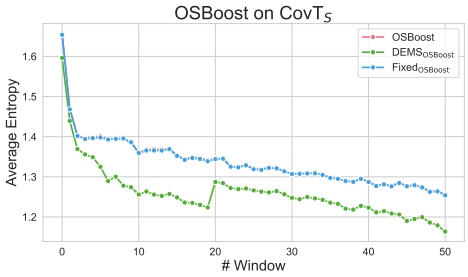


(d) Entropy of OAUE on WISDM

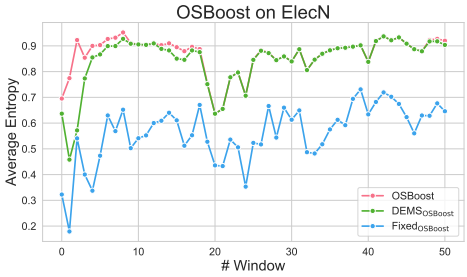
(e) Entropy of OSBoost on AGR$_a$



(f) Entropy of OSBoost on Airlines

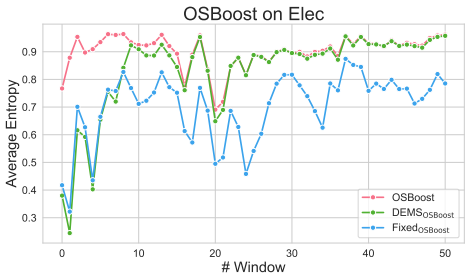


(g) Entropy of OSBoost on CovT$_S$

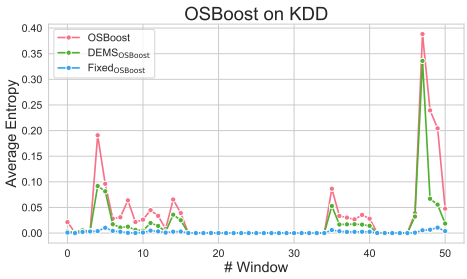


(h) Entropy of OSBoost on ElecN

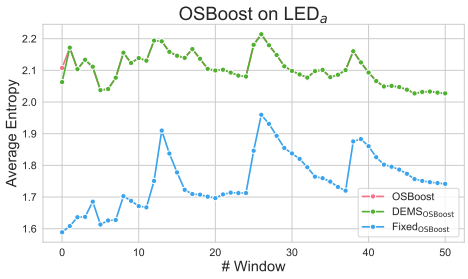
Figure B.26: List of Entropy Plots (Part IX)



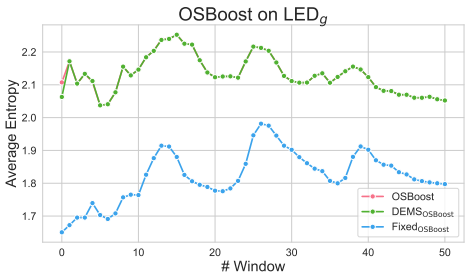
(a) Entropy of OSBoost on Elec



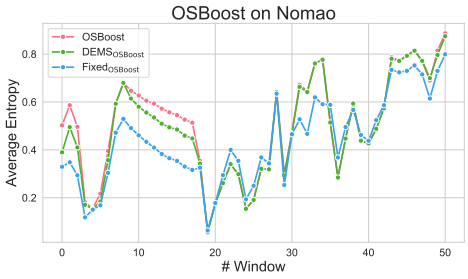
(b) Entropy of OSBoost on KDD



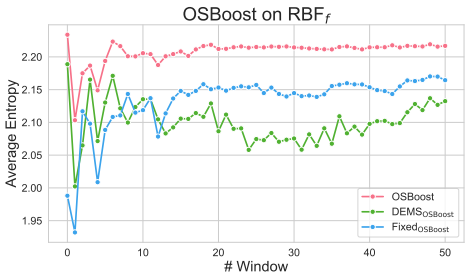
(c) Entropy of OSBoost on LED$_a$



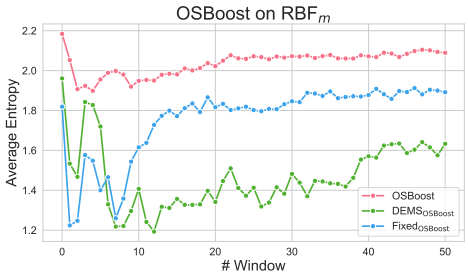
(d) Entropy of OSBoost on LED$_g$



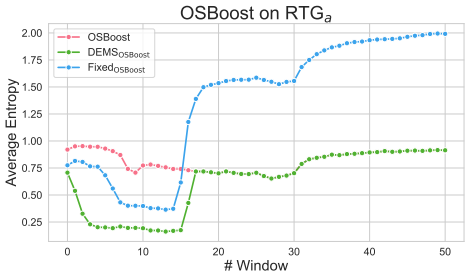
(e) Entropy of OSBoost on Nomao



(f) Entropy of OSBoost on RBF$_f$

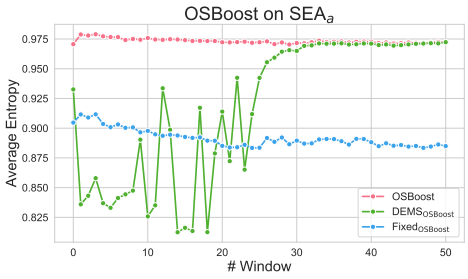


(g) Entropy of OSBoost on RBF$_m$

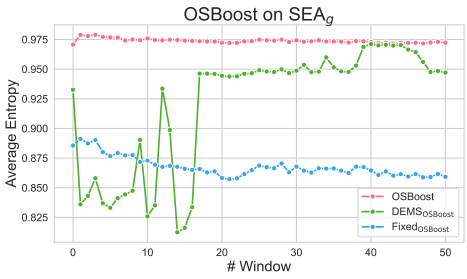


(h) Entropy of OSBoost on RTG$_a$

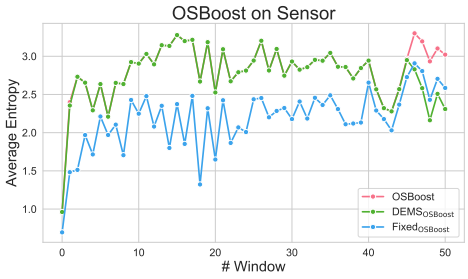
Figure B.27: List of Entropy Plots (Part X)



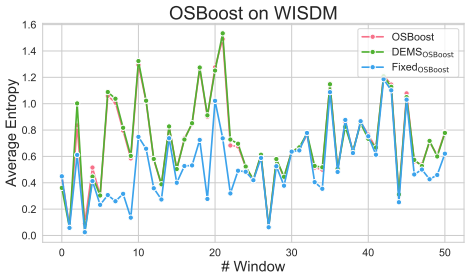
(a) Entropy of OSBoost on SEA$_a$



(b) Entropy of OSBoost on SEA$_g$



(c) Entropy of OSBoost on Sensor



(d) Entropy of OSBoost on WISDM

Figure B.28: List of Entropy Plots (Part XI)

References

[1] Charu C Aggarwal, S Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier, 2003.

[2] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database mining: A performance perspective. *IEEE transactions on knowledge and data engineering*, 5(6):914–925, 1993.

[3] Ezilda Almeida, Carlos Ferreira, and Joao Gama. Adaptive model rules from data streams. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I 13*, pages 480–492. Springer, 2013.

[4] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[5] Elif Selen Babüroğlu, Alptekin Durmuşoğlu, and Türkay Dereli. Concept drift from 1980 to 2020: a comprehensive bibliometric analysis with future research insight. *Evolving Systems*, 15(3):789–809, 2024.

[6] Manuel Baena-Garcıa, José del Campo-Ávila, Raul Fidalgo, Albert Bifet, Ricard Gavalda, and Rafael Morales-Bueno. Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86. Citeseer, 2006.

[7] Maroua Bahri and Albert Bifet. Incremental k-nearest neighbors using

reservoir sampling for data streams. In *Discovery Science: 24th International Conference*, pages 122–137. Springer, 2021.

[8] Maroua Bahri, Albert Bifet, João Gama, Heitor Murilo Gomes, and Silviu Maniu. Data stream analysis: Foundations, major tasks and tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(3):e1405, 2021.

[9] Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E Schapire. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[10] Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 59–68, 2015.

[11] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[12] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I 21*, pages 135–150. Springer, 2010.

[13] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*, pages 44–50. PMLR, 2010.

[14] Albert Bifet, Jesse Read, Indrė Žliobaitė, Bernhard Pfahringer, and Geoff Holmes. Pitfalls in benchmarking data stream classification and

how to avoid them. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part I 13*, pages 465–479. Springer, 2013.

[15] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. Cart. *Classification and regression trees*, 1984.

[16] Dariusz Brzezinski and Jerzy Stefanowski. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, 265:50–67, 2014.

[17] Laurent Candillier and Vincent Lemaire. Design and analysis of the nomao challenge active learning in the real-world. In *Proceedings of the ALRA: active learning in real-world applications, workshop ECML-PKDD*, pages 1–15. Citeseer, 2012.

[18] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[19] Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. An online boosting algorithm with theoretical justifications. *arXiv preprint arXiv:1206.6422*, 2012.

[20] Arnaud Chiolero, Valérie Santschi, Bernard Burnand, Robert W Platt, and Gilles Paradis. Meta-analyses: with confidence or prediction intervals? *European journal of epidemiology*, 27(10):823–825, 2012.

[21] Ajay Choudhary, Preeti Jha, Aruna Tiwari, and Neha Bharill. A brief survey on concept drifted data stream regression. *Soft Computing for Problem Solving: Proceedings of SocProS 2020, Volume 2*, pages 733–744, 2021.

[22] Ajay Choudhary, Preeti Jha, Aruna Tiwari, and Neha Bharill. A brief survey on concept drifted data stream regression. In Aruna Tiwari,

Kapil Ahuja, Anupam Yadav, Jagdish Chand Bansal, Kusum Deep, and Atulya K. Nagar, editors, *Soft Computing for Problem Solving*, pages 733–744, Singapore, 2021. Springer Singapore.

[23] Andrea Coraddu, Luca Oneto, Alessandro Ghio, Stefano Savio, Davide Anguita, and Massimo Figari. Condition Based Maintenance of Naval Propulsion Plants. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5K31K.

[24] Rafael MO Cruz, Robert Sabourin, and George DC Cavalcanti. Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, 41:195–216, 2018.

[25] Padraig Cunningham and John Carney. Diversity versus quality in classification ensembles based on feature selection. In *European Conference on Machine Learning*, pages 109–116. Springer, 2000.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[27] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[28] Tewodros Deneke. Online Video Characteristics and Transcoding Time Dataset. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C9K.

[29] Thomas G Dietterich et al. Ensemble learning. *The handbook of brain theory and neural networks*, 2(1):110–125, 2002.

[30] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000.

[31] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

[32] Joao Duarte, Joao Gama, and Albert Bifet. Adaptive model rules from high-speed data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(3):1–22, 2016.

[33] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2015.

[34] Sanem Elbasi, Alican Büyükçakır, Hamed Bonab, and Fazli Can. On-the-fly ensemble pruning in evolving data streams. *arXiv preprint arXiv:2109.07611*, 2021.

[35] Michael TM Emmerich and André H Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural computing*, 17:585–609, 2018.

[36] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2:113–127, 2014.

[37] Isvani Frias-Blanco, José del Campo-Ávila, Gonzalo Ramos-Jimenez, Rafael Morales-Bueno, Agustin Ortiz-Diaz, and Yailé Caballero-Mota. Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823, 2014.

[38] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[39] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[40] Joao Gama and Mohamed Medhat Gaber. *Learning from data streams: processing techniques in sensor networks*. Springer Science & Business Media, 2007.

[41] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, pages 286–295. Springer, 2004.

[42] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[43] Giorgio Giacinto and Fabio Roli. Dynamic classifier selection. In *International Workshop on Multiple Classifier Systems*, pages 177–189. Springer, 2000.

[44] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106:1469–1495, 2017.

[45] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, 50(2):1–36, 2017.

[46] Heitor Murilo Gomes, Jean Paul Barddal, Luis Eduardo Boiko Ferreira, and Albert Bifet. Adaptive random forests for data stream regression. In *ESANN*, 2018.

[47] Heitor Murilo Gomes and Albert Bifet. Practical machine learning for streaming data. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6418–6419, 2024.

[48] Heitor Murilo Gomes, Maciej Grzenda, Rodrigo Mello, Jesse Read, Minh Huong Le Nguyen, and Albert Bifet. A survey on semi-supervised learning for delayed partially labelled data streams. *ACM Computing Surveys*, 55(4):1–42, 2022.

[49] Heitor Murilo Gomes, Anton Lee, Nuwan Gunasekara, Yibin Sun, Guilherme Weigert Cassales, Justin Jia Liu, Marco Heyden, Vitor Cerqueira, Maroua Bahri, Yun Sing Koh, Bernhard Pfahringer, and Albert Bifet. CapyMOA: Efficient machine learning for data streams in python, 2025.

[50] Heitor Murilo Gomes, Jacob Montiel, Saulo Martiello Mastelini, Bernhard Pfahringer, and Albert Bifet. On ensemble techniques for data stream regression. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[51] Heitor Murilo Gomes, Jesse Read, and Albert Bifet. Streaming random patches for evolving data stream classification. In *2019 IEEE international conference on data mining (ICDM)*, pages 240–249. IEEE, 2019.

[52] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21(2):6–22, 2019.

[53] Heitor Murilo Gomes, Jesse Read, Albert Bifet, and Robert J Durrant. Learning from evolving data streams through ensembles of random patches. *Knowledge and Information Systems*, 63(7):1597–1625, 2021.

[54] Bryce Goodman. Hard choices and hard limits in artificial intelligence. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 112–121, 2021.

[55] Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018.

[56] Nuwan Gunasekara, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Survey on online streaming continual learning. In *IJCAI*, pages 6628–6637, 2023.

[57] Nuwan Amila Gunasekara. *Advanced Adaptive Classifier Methods for Data Streams*. PhD thesis, The University of Waikato, 2023.

[58] Huaping Guo, Hongbing Liu, Ran Li, Changan Wu, Yibo Guo, and Mingliang Xu. Margin & diversity based ordering ensemble pruning. *Neurocomputing*, 275:237–246, 2018.

[59] Huaping Guo, Hongbing Liu, Ran Li, Changan Wu, Yibo Guo, and Mingliang Xu. Margin & diversity based ordering ensemble pruning. *Neurocomputing*, 275:237–246, 2018.

[60] Myrianthi Hadjicharalambous, Marios M Polycarpou, and Christos G Panayiotou. Neural network-based construction of online prediction intervals. *Neural Computing and Applications*, 32(11):6715–6733, 2020.

[61] Myrianthi Hadjicharalambous, Marios M Polycarpou, and Christos G Panayiotou. Neural network-based construction of online prediction intervals. *Neural Computing and Applications*, 32(11):6715–6733, 2020.

[62] Gerald J Hahn. Factors for calculating two-sided prediction intervals for samples from a normal distribution. *Journal of the American Statistical Association*, 64(327):878–888, 1969.

[63] Gerald J Hahn. Additional factors for calculating prediction intervals for samples from a normal distribution. *Journal of the American Statistical Association*, 65(332):1668–1676, 1970.

[64] Gerald J Hahn and Wayne Nelson. A survey of prediction intervals and their applications. *Journal of Quality Technology*, 5(4):178–188, 1973.

[65] Gerald J Hahn and Wayne Nelson. A survey of prediction intervals and their applications. *Journal of Quality Technology*, 5(4):178–188, 1973.

[66] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018.

[67] Michael Harries, New South Wales, et al. Splice-2 comparative evaluation: Electricity pricing. 1999.

[68] Marco Heyden, Heitor Murilo Gomes, Edouard Fouché, Bernhard Pfahringer, and Klemens Böhm. Leveraging plasticity in incremental decision trees. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 38–54. Springer, 2024.

[69] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[70] Geoffrey Holmes, Richard Kirkby, and Bernhard Pfahringer. Stress-testing hoeffding trees. In *Knowledge Discovery in Databases: PKDD 2005: 9th European Conference on Principles and Practice of Knowledge Discovery in Databases, Porto, Portugal, October 3-7, 2005. Proceedings 9*, pages 495–502. Springer, 2005.

[71] Lisha Hu, Wenxiu Li, Yaru Lu, and Chunyu Hu. Scalable concept drift adaptation for stream data mining. *Complex & Intelligent Systems*, pages 1–19, 2024.

[72] Ruihan Hu, Songbin Zhou, Yisen Liu, and Zhiri Tang. Margin-based pareto ensemble pruning: an ensemble pruning algorithm that learns to search optimized ensembles. *Computational intelligence and neuroscience*, 2019, 2019.

[73] J. Huang, J. Rojas, M. Zimmer, H. Wu, Y. Guan, and P. Weng. Hyper-

parameter auto-tuning in self-supervised robotic learning. *IEEE Robotics and Automation Letters*, 6(2):3537–3544, 2021.

[74] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pages 492–518. Springer, 1992.

[75] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

[76] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

[77] Elena Ikonomovska, Joao Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23:128–168, 2011.

[78] Elena Ikonomovska, Joao Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23:128–168, 2011.

[79] Elena Ikonomovska, Joao Gama, Bernard Zenko, and Saso Dzeroski. Speeding-up hoeffding-based regression trees with options. In *ICML*, 2011.

[80] Adriana Sayuri Iwashita and Joao Paulo Papa. An overview on concept drift learning. *IEEE access*, 7:1532–1547, 2018.

[81] Botao Jiao, Yinan Guo, Dunwei Gong, and Qiuju Chen. Dynamic ensemble selection for imbalanced data streams with concept drift. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[82] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[83] Utkarsh Mahadeo Khaire and R Dhanalakshmi. Stability of feature selection algorithm: A review. *Journal of King Saud University-Computer and Information Sciences*, 34(4):1060–1073, 2022.

[84] Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, and Khaled Ghédira. Discussion and review on evolving data streams and concept drift adapting. *Evolving systems*, 9:1–23, 2018.

[85] J Zico Kolter and Marcus A Maloof. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, 8:2755–2790, 2007.

[86] Chrysoula Kosma, Giannis Nikolentzos, Nancy Xu, and Michalis Vazirgiannis. Time series forecasting models copy the past: How to mitigate. In *International Conference on Artificial Neural Networks*, pages 366–378. Springer, 2022.

[87] Georg Krempl, Indre Žliobaite, Dariusz Brzeziński, Eyke Hüllermeier, Mark Last, Vincent Lemaire, Tino Noack, Ammar Shaker, Sonja Sievi, Myra Spiliopoulou, et al. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1):1–10, 2014.

[88] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51:181–207, 2003.

[89] Marília Lima, Manoel Neto, Telmo Silva Filho, and Roberta A. de A. Fagundes. Learning under concept drift for regression—a systematic literature review. *IEEE Access*, 10:45410–45429, 2022.

[90] Viktor Losing, Barbara Hammer, and Heiko Wersing. Tackling heterogeneous concept drift with the self-adjusting memory (sam). *Knowledge and Information Systems*, 54(1):171–201, 2018.

[91] Hongfang Lu, Xin Ma, Minda Ma, and Senlin Zhu. Energy price prediction using data-driven models: A decade review. *Computer Science Review*, 39:100356, 2021.

[92] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[93] Tianyuan Lu, Lei Wang, and Xiaoyong Zhao. Review of anomaly detection algorithms for data streams. *Applied Sciences*, 13(10):6353, 2023.

[94] Gang Luo. A review of automatic selection methods for machine learning algorithms and hyper-parameter values. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 5(1):1–16, 2016.

[95] Chaitanya Manapragada, Geoffrey I Webb, and Mahsa Salehi. Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1953–1962, 2018.

[96] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. River: machine learning for streaming data in python. *Journal of Machine Learning Research*, 22(110):1–8, 2021.

[97] Hayet Mouss, D Mouss, N Mouss, and L Sefouhi. Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In *2004 5th Asian Control Conference (IEEE Cat. No. 04EX904)*, volume 2, pages 815–818. IEEE, 2004.

[98] Giung Nam, Jongmin Yoon, Yoonho Lee, and Juho Lee. Diversity matters when learning from ensembles. *Advances in neural information processing systems*, 34:8367–8377, 2021.

[99] Warwick Nash, Tracy Sellers, Simon Talbot, Andrew Cawthorn, and Wes Ford. Abalone. UCI Machine Learning Repository, 1995. DOI: https://doi.org/10.24432/C55C7W.

[100] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *ICNN*. IEEE, 1994.

[101] R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

[102] Ewan S Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[103] Abu Rayhan, Rajan Rayhan, and Swajan Rayhan. Artificial general intelligence: Roadmap to achieving human-level capabilities, 2023.

[104] Jesse Read, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *Advances in Intelligent Data Analysis XI: 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings 11*, pages 313–323. Springer, 2012.

[105] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[106] Jonathan V Roth. Prediction interval analysis is underutilized and can be more helpful than just confidence interval analysis. *Journal of clinical monitoring and computing*, 23(3):181, 2009.

[107] Dymitr Ruta and Bogdan Gabrys. Classifier selection for majority voting. *Information fusion*, 6(1):63–81, 2005.

[108] Raghavendra Selvan, Julian Schön, and Erik B Dam. Operating critical machine learning models in resource constrained regimes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 325–335. Springer, 2023.

[109] Jaime Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022.

[110] D. L. Shrestha and D. P. Solomatine. Machine learning approaches for estimation of prediction interval for the model output. *Neural networks*, 19(2):225–235, 2006.

[111] Nitin Anand Shrivastava, Abbas Khosravi, and Bijaya Ketan Panigrahi. Prediction interval estimation for electricity price and demand using support vector machines. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3995–4002. IEEE, 2014.

[112] Vinicius MA Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858, 2020.

[113] Salvatore Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip Chan. KDD Cup 1999 Data. UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C51C7N.

[114] W Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382, 2001.

[115] Andrés L Suárez-Cetrulo, David Quintana, and Alejandro Cervantes. A

survey on machine learning for recurring concept drifting data streams. *Expert Systems with Applications*, 213:118934, 2023.

[116] Yibin Sun, Heitor Murilo Gomes, Bernhard Pfahringer, and Albert Bifet. Real-time energy pricing in new zealand: An evolving stream analysis. In Rafik Hadfi, Patricia Anthony, Alok Sharma, Takayuki Ito, and Quan Bai, editors, *PRICAI 2024: Trends in Artificial Intelligence*, pages 91–97, Singapore, 2025. Springer Nature Singapore.

[117] Yibin Sun, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. SOKNL: A novel way of integrating k-nearest neighbours with adaptive random forest regression for data streams. *Data Mining and Knowledge Discovery*, 36(5):2006–2032, 2022.

[118] Yibin Sun, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. SOKNL: A novel way of integrating k-nearest neighbours with adaptive random forest regression for data streams. *Data Mining and Knowledge Discovery*, 2022.

[119] Yibin Sun, Bernhard Pfahringer, Heitor Murilo Gomes, and Albert Bifet. Adaptive prediction interval for data stream regression. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 130–141. Springer, 2024.

[120] Shatskikh SYa and LE Melkumova. Normality assumption in statistical data analysis. In *CEUR Workshop Proceedings*, volume 1638, pages 763–768, 2016.

[121] Swee Chuan Tan, Kai Ming Ting, and Tony Fei Liu. Fast anomaly detection for streaming data. In *Twenty-second international joint conference on artificial intelligence*. Citeseer, 2011.

[122] Hadi Tarazodar, Karamollah Bagherifard, Samad Nejatian, Hamid Parvin, and Razieh Malekhosseini. Mitigating concept drift in data

streams: an incremental decision tree approach. *Soft Computing*, pages 1–30, 2024.

[123] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 10, 2020.

[124] Luís Torgo. Partial linear trees. In *ICML*, pages 1007–1014, 2000.

[125] Grigorios Tsoumakas, Ioannis Partalas, and Ioannis Vlahavas. A taxonomy and short review of ensemble selection. In *Workshop on Supervised and Unsupervised Ensemble Methods and Their Applications*, pages 1–6, 2008.

[126] Bruno Veloso, João Gama, and Benedita Malheiro. Self hyper-parameter tuning for data streams. In *International Conference on Discovery Science*, pages 241–255. Springer, 2018.

[127] János Végh. How deep the machine learning can be, 2020.

[128] Abraham Wald. *Sequential analysis*. Courier Corporation, 2004.

[129] Rafał Weron. Electricity price forecasting: A review of the state-of-the-art with a look into the future. *International journal of forecasting*, 30(4):1030–1081, 2014.

[130] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[131] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101, 1996.

[132] Yanzhao Wu, Ling Liu, Zhongwei Xie, Ka-Ho Chow, and Wenqi Wei. Boosting ensemble accuracy by revisiting ensemble diversity metrics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16469–16477, 2021.

[133] Fan Yang, Wei-hang Lu, Lin-kai Luo, and Tao Li. Margin optimization based pruning for random forest. *Neurocomputing*, 94:54–63, 2012.

[134] Chenxu Zhang and Yong Fu. Probabilistic electricity price forecast with optimal prediction interval. *IEEE Transactions on Power Systems*, 2023.

[135] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. *ACM Sigmod Record*, 25(2):103–114, 1996.

[136] Jun Zhao, Wei Wang, and Chunyang Sheng. Industrial prediction intervals with data uncertainty. In *Data-Driven Prediction for Industrial Processes and Their Applications*, pages 159–222. Springer, 2018.

[137] Zhi-Hua Zhou. Learnability with time-sharing computational resource concerns. *National Science Review*, page nwae204, 2024.

[138] Alaettin Zubaroğlu and Volkan Atalay. Data stream clustering: a review. *Artificial Intelligence Review*, 54(2):1201–1236, 2021.