



**HAL**  
open science

# Machine learning algorithms for dynamic Internet of Things

Dihia Boulegane

► **To cite this version:**

Dihia Boulegane. Machine learning algorithms for dynamic Internet of Things. Machine Learning [cs.LG]. Institut Polytechnique de Paris, 2021. English. NNT : 2021IPPAT048 . tel-03503316

**HAL Id: tel-03503316**

**<https://theses.hal.science/tel-03503316>**

Submitted on 27 Dec 2021

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT  
POLYTECHNIQUE  
DE PARIS

NNT : 2021IPPAT048

Thèse de doctorat



# Machine Learning Algorithms for Dynamic Internet of Things

Thèse de doctorat de l'Institut Polytechnique de Paris  
préparée à Télécom Paris

École doctorale n°626 École doctorale de l'Institut Polytechnique de  
Paris (EDIPParis)  
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Palaiseau, le 1 Septembre 2021, par

**DIHIA BOULEGANE**

Composition du Jury :

Jesse Read Professeur, Ecole Polytechnique (LIX)	Président
Antoine Cornuejols Professeur, AgroParisTech (MMIP)	Rapporteur
Rosanna Verde Professeur, Università della Campania "Luigi Vanvitelli"	Rapporteur
Georges Hebrail Professeur, Télécom Paris	Examineur
Vincent Lemaire Docteur, Orange Labs	Examineur
Raja Chiky Professeur, ISEP Paris (LISITE)	Examineur
Albert Bifet Professeur, Télécom Paris (LTCl)	Directeur de thèse
Giyarpuram MADHUSUDAN Ingénieur de Recherche, Orange Labs	Encadrant de thèse

# Machine Learning Algorithms for Dynamic Internet of Things

Thesis submitted for the Degree of Doctor of Philosophy :

Télécom Paris  
Institut Polytechnique de Paris

&

Orange Labs



**INSTITUT  
POLYTECHNIQUE  
DE PARIS**



Dihia BOULEGANE

September 2021



*“Ulach win izegrene assif our yebzig” - provèrbe berbère*  
(Nul ne traverse de rivière sans se mouiller)



# Abstract

With the rapid growth of Internet-of-Things (IoT) devices and sensors, sources that are continuously releasing and curating vast amount of data at high pace in the form of stream have spread over various domains of application such as healthcare, energy, or infrastructure monitoring. The ubiquitous data streams are essential for data driven decision-making in different business sectors and organizations using Artificial Intelligence (AI) and Machine Learning (ML) techniques in order to extract valuable knowledge and turn it to appropriate actions. Besides, the data being collected is often associated with a temporal indicator, referred to as temporal data stream that is a potentially infinite sequence of observations captured over time at regular intervals, but not necessarily.

*Forecasting* is one of the most challenging tasks in the field of AI and aims at understanding the process generating the observations over time based on past data in order to accurately predict future behavior. However, the infinite and evolving nature of data streams raises new challenges that learning techniques must handle such as concept drift. Stream Learning is the emerging research field which focuses on learning from infinite and evolving data streams. In this context of evolving data streams, dynamic model combination has emerged as a very promising approach and achieves competitive results. The rationale is that different learning models exhibit different strengths and limitations along the input data stream and thus selecting the best ones only enhances overall predictive performance. However, despite their superior performance, ensemble methods suffer from their exorbitant computational costs in terms of memory and time compared to individual models. This issue is particularly inconvenient in data streams applications where resources are often limited.

The first part of the thesis surveys the current state-of-the-art of dynamic forecast combination and ensemble selection for the stream setting, and provides an extensive and updated review of the research area. The research goal of the thesis can be divided into two main parts: (i) dynamically selecting and combining models within a heterogeneous ensemble to forecast future uni-variate numeric observations in streaming data; (ii) compressing highly complex ensemble into faster and more compact individual model while retaining competitive predictive performance.

The first research goal is split into two steps. Initially, we study how to estimate the predictive performance of individual forecasting models according to the data and contribute by introducing novel windowing and meta-learning based methods to cope with evolving data streams. Subsequently, we propose different selection methods that aim to constituting a committee of models that are as accurate and as diverse as possible for each test instance. The predictions of these models are then weighted and aggregated. We empirically investigate the advantage of the proposed methods against existing methods, using both synthetic and real temporal data streams from different domains of application.

The second part addresses model compression in the streaming setting in order to overcome the computational drawbacks in terms of memory and time of dynamic ensemble methods. We aim at building single model to mimic the behavior of a highly performing complex ensemble while drastically reducing its complexity and coping with concept drift. Finally, we present the first streaming competition "Real-time Machine Learning Competition on Data Streams", at the IEEE Big Data 2019 conference, using the new *SCALAR* platform. The competition data is released in the form of a stream and participants continuously submit their predictions in real-time.

**Keywords:** machine learning; data streams; time series; forecasting; dynamic ensemble selection; model compression

## Résumé

La croissance rapide et vertigineuse de l'Internet des Objets (IdO) ainsi que la prolifération des objets connectés et des capteurs ont donné lieu à diverses sources de données qui génèrent continuellement de grandes quantités de données et à une grande vitesse. Ces données touchent différents domaines d'application tels que la santé, les énergies ou encore la supervision d'infrastructures. Les flux de données sont omniprésents et essentiels dans le processus de prise de décision dans différents secteurs d'activité et organisations en utilisant les techniques d'Intelligence Artificielle (IA) et d'apprentissage automatique (ML pour Machine Learning en anglais) afin d'extraire des connaissances précieuses et les transformer en actions pertinentes. Par ailleurs, les données collectées sont souvent associées à un indicateur temporel, appelé flux de données temporel qui est défini comme étant une séquence d'observations potentiellement infinie capturée au fil du temps à intervalles réguliers, mais pas nécessairement.

La prévision est l'une des tâches les plus complexes dans le domaine de l'IA et vise à comprendre le processus générant les observations au fil du temps sur la base d'un historique de données afin de prédire le comportement futur. Cependant, la nature infinie et changeante des flux de données soulève de nouveaux défis que les techniques d'apprentissage doivent relever, notamment la dérive conceptuelle. L'apprentissage incremental et adaptatif est le domaine de recherche émergent dédié à l'analyse des flux de données. Dans ce contexte évolutif de flux de données, les méthodes d'ensemble qui fusionnent de manière dynamique plusieurs modèles prédictifs accomplissent des résultats très prometteurs et compétitifs. L'intuition est que différents modèles d'apprentissage manifestent différentes forces et limites tout au long du flux et ce en fonction des données présentées en entrée. De ce fait, sélectionner les meilleurs a pour but d'améliorer les performances prédictives globales. Cependant, malgré des performances supérieures, les méthodes d'ensemble souffrent cruellement de leur coût exorbitant en matière de mémoire et de temps de calcul par rapport aux modèles individuels. Cet aspect est particulièrement contraignant dans les applications de flux de données où les ressources sont souvent limitées.

La première partie de la thèse expose l'état de l'art des techniques de fusion et de sélection dynamiques de modèles prédictifs sur des flux de données. Les objectifs de recherche de la thèse peuvent être divisés en deux principaux axes: (i) sélectionner et fusionner de manière dynamique des modèles au sein d'un ensemble de différents modèles afin de prédire les valeurs numériques futures d'un flux de données univarié; (ii) compresser un ensemble hautement complexe en un modèle individuel plus rapide et plus compact tout en conservant des performances prédictives compétitives.

Le premier objectif de recherche est scindé en deux étapes à son tour. Dans un premier temps, nous étudions différentes méthodes pour estimer la performance de chaque modèle

de prévision individuel compris dans l'ensemble en fonction des données en introduisant de nouvelles méthodes basées sur le fenêtrage et le méta-apprentissage pour gérer des flux de données changeants. Par la suite, nous proposons différentes méthodes de sélection qui visent à constituer un comité de modèles aussi précis et aussi divers que possible pour chaque observation. Les prédictions de ces modèles sont ensuite pondérées et agrégées. Nous étudions de manière empirique l'avantage des méthodes proposées relativement aux méthodes de l'état de l'art, en utilisant de données temporelles synthétiques et réelles provenant de différents domaines. La deuxième partie de la thèse traite de la compression des méthodes d'ensemble dynamiques sur des flux de données afin de surmonter les inconvénients relatifs à leur complexité. Nous visons à produire un modèle individuel afin d'imiter le comportement d'un ensemble hautement complexe tout en réduisant de manière drastique leur coût en termes de mémoire et temps de calcul et en gérant les problèmes liés à la dérive conceptuelle. Pour finir, nous présentons "Real-Time Machine Learning Competition on Data Streams", dans le cadre de BigDataCup Challenge de la conférence IEEE Big Data 2019. L'idée est de délivrer les données de la compétition en temps réel à tous les participants et de les engager à soumettre leurs prédictions en temps réel via la plate-forme dédiée SCALAR.

**Mots clés:** apprentissage machine; flux de données, séries temporelles; prévision; selection dynamic d'ensemble; compression de modèle

# Contents

<b>Abstract</b> .....	<b>iii</b>
<b>Résumé</b> .....	<b>v</b>
<b>Contents</b> .....	<b>vii</b>
<b>List of Figures</b> .....	<b>xi</b>
<b>List of Tables</b> .....	<b>xv</b>
<b>1 Introduction</b> .....	<b>1</b>
1.1 <b>Context and motivation</b>	<b>1</b>
1.2 <b>Contributions</b>	<b>5</b>
1.3 <b>Publications</b>	<b>6</b>
1.4 <b>Outline</b>	<b>7</b>
<b>2 Background and Related Work</b> .....	<b>9</b>
<b>2.1 Forecasting in streaming setting</b>	<b>10</b>
2.1.1 Data Streams and Time Series .....	10
2.1.2 Forecasting with temporal streaming data .....	11
2.1.3 Incremental learning .....	12
2.1.4 Concept Drift in evolving data streams .....	13
2.1.5 Forecasting methods for evolving data streams .....	16
2.1.6 Forecasting with supervised learning regression .....	18
2.1.7 Evaluating forecasting models .....	20

<b>2.2</b>	<b>Dynamic Ensembles of forecasters for streaming data</b>	<b>21</b>
2.2.1	The premise of Ensemble methods	22
2.2.2	Dynamic Combination of Forecasters	26
2.2.3	Dynamic ensembles for evolving temporal data streams	27
<b>2.3</b>	<b>Dynamic Ensemble Selection (DES) for temporal streams</b>	<b>31</b>
2.3.1	On Dynamic Ensemble Selection (DES)	31
2.3.2	DES for temporal data streams	33
<b>2.4</b>	<b>Summary</b>	<b>37</b>
<b>3</b>	<b>Streaming Dynamic Ensembles Selection</b>	<b>39</b>
<b>3.1</b>	<b>Individual models' competence estimation for DES</b>	<b>40</b>
3.1.1	Windowing-based DES	41
3.1.2	Meta-Learning based DES	43
<b>3.2</b>	<b>Experts selection: Select or not select, that is the question</b>	<b>47</b>
3.2.1	Trimming (TRIM)	47
3.2.2	Abstaining (ABS)	47
3.2.3	Randomized Selection	52
3.2.4	Performance-diversity trade-off	54
<b>3.3</b>	<b>Empirical Experiments</b>	<b>55</b>
3.3.1	Experimental design	56
3.3.2	Comparing windowing methods	58
3.3.3	Comparing Meta-learning based methods	64
3.3.4	Comparing SW, SLOPE and meta-learning based trimming DES	69
3.3.5	Evaluating experts selection	72
<b>3.4</b>	<b>Summary</b>	<b>77</b>
<b>4</b>	<b>Dynamic Ensemble using Multi-Target Regression</b>	<b>79</b>
<b>4.1</b>	<b>DES using multi-output learning</b>	<b>80</b>
4.1.1	DES via Multi-Label Classification	80
4.1.2	DES via Multi-Target Regression	81
<b>4.2</b>	<b>Streaming Multi-Target Regression</b>	<b>82</b>
4.2.1	Problem Transformation methods	82
4.2.2	Algorithm Adaptation	83
<b>4.3</b>	<b>Empirical Experiments</b>	<b>84</b>
4.3.1	Experimental design	84
4.3.2	Comparing meta-layer	85
4.3.3	Comparing MTR-based DES methods	86
4.3.4	Computational cost	89
4.3.5	Discussion	90
<b>4.4</b>	<b>Summary</b>	<b>91</b>

<b>5</b>	<b>Model Compression</b>	<b>93</b>
<b>5.1</b>	<b>Compressing dynamic ensembles for streaming data</b>	<b>94</b>
5.1.1	Model Compression for batch learning	94
5.1.2	Compression for streaming data	95
<b>5.2</b>	<b>Adaptive Model compression for data streams forecasting</b>	<b>96</b>
5.2.1	Teaching data	96
5.2.2	Student model teaching approach	97
5.2.3	Concept drift adaptation for streaming MC	98
<b>5.3</b>	<b>Experiments</b>	<b>100</b>
5.3.1	Experimental setup	102
5.3.2	Adaptive Compression	109
<b>5.4</b>	<b>Summary</b>	<b>110</b>
<b>6</b>	<b>Real-time Machine Learning Competitions with SCALAR</b>	<b>113</b>
<b>6.1</b>	<b>Real-time Machine Learning Competition on Data Streams</b>	<b>114</b>
6.1.1	Competition Protocol and task	114
6.1.2	Competition Data	115
6.1.3	Winning solutions	117
6.1.4	Competition Results	119
<b>6.2</b>	<b>SCALAR Platform</b>	<b>119</b>
6.2.1	Stream Server and communication	121
6.2.2	Web application	123
<b>6.3</b>	<b>Summary</b>	<b>126</b>
<b>7</b>	<b>Conclusions and Future Work</b>	<b>127</b>
<b>7.1</b>	<b>Conclusions</b>	<b>127</b>
<b>7.2</b>	<b>Future directions</b>	<b>129</b>
7.2.1	Handling delayed labels in temporally evolving data streams	129
7.2.2	Behavioral data and Exceptional (dis)agreement	129
7.2.3	Change detection for multi-variate data	130
7.2.4	Model compression for streaming data	130
	<b>Appendices</b>	<b>131</b>
<b>A</b>	<b>Datasets and Forecasters</b>	<b>131</b>
<b>A.1</b>	<b>Temporal data streams</b>	<b>131</b>
<b>A.2</b>	<b>Forecasters</b>	<b>133</b>
<b>A.3</b>	<b>Blocked Prequential training</b>	<b>134</b>
<b>A.4</b>	<b>Online Ensembles</b>	<b>135</b>
	<b>References</b>	<b>137</b>



# List of Figures

1.1	Stream supervised learning scenario. The predictive model toggles between two states: train and predict then back to ready. Labeled data is used to update the model whereas unlabeled instances are used to predict their corresponding target value. . . . .	3
2.1	Different types of concept drift . . . . .	15
2.2	A common ensemble architecture . . . . .	23
2.3	Multiple Classifier Systems (MCS) taxonomy [26] . . . . .	32
2.4	A generic workflow for Dynamic Ensemble Selection (DES) showing the selection and combination steps for each instance in the data stream. . . . .	33
3.1	General Dynamic Ensemble Selection pipeline for data streams . . . . .	40
3.2	Meta-learning pipeline for streaming data . . . . .	43
3.3	Streaming arbitrated meta learning architecture . . . . .	46
3.4	Heatmaps illustrating the average rank (3.4a) and respective standard deviation (3.4b) of SW-DES for varying $w$ and $\alpha$ parameter values. Darker tiles mean higher values . . . . .	58
3.5	Heatmaps illustrating the average rank (left) and respective standard deviation (right) of SLOPE for varying $w$ , $k$ and $\alpha$ parameters. Darker tiles mean higher values . . . . .	60
3.6	Boxplot illustrating the distribution of the rank and respective standard deviation of SW and SLOPE trimming based DES methods in terms of RMSE . . . . .	61

3.7	Boxplot illustrating the relative difference expressed in percentage achieved by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) against other windowing DES methods. Values below (resp. above) the zero line represent improvement (resp. loss) in the predictive performance. . . . .	62
3.8	Proportion of probability of SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) winning/drawing/losing against other windowing DES methods according to the Bayes sign test . . . . .	63
3.9	Heatmaps illustrating the average rank (3.9a) and respective standard deviation (3.9b) of STADE for varying meta-features set and $\alpha$ parameters. Darker tiles mean higher values. . . . .	65
3.10	Boxplot illustrating the distribution of the percentual difference in terms of RMSE achieved by ORI_STA_LAG_LAN against other STADE variants using different meta-features for corresponding selection ratio $\alpha$ . . . . .	66
3.11	Boxplots illustrating the distribution of the rank and respective standard deviation of the best performing STADE trimming DES methods . . . . .	66
3.12	Boxplot illustrating the distribution of the percentual difference in terms of RMSE achieved by the best STADE. . . . .	67
3.13	Proportion of probability of best STADE variant winning/drawing/losing according to the Bayes sign test . . . . .	68
3.14	Boxplot illustrating the distribution of the ratio between the resource spent by best STADE in terms of memory and time relative to other variants. Values below (resp. above) the value 1 stand for lower (resp. higher) computational resource requirement. . . . .	68
3.15	Boxplot illustrating the distribution of the rank and respective standard deviation of trimming based windowing and meta-learning based DES methods in terms of RMSE. . . . .	69
3.16	Boxplot illustrating the percentual/relative difference expressed in percentage achieved by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) compared to other DES methods.70	
3.17	Proportion of probability of best trimming DES method winning/drawing/losing according to the Bayes sign test. . . . .	71
3.18	Boxplot illustrating the distribution of the ratio of the memory (left) and time (right) required by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) relative to STADE and state-of-the-art dynamic ensembles. . . . .	71
3.19	Boxplots illustrating the distribution of the rank and respective standard deviation of trimming and abstaining (ABS) based DES methods for SW, SLOPE and STADE approaches in terms of RMSE. . . . .	74
3.20	Boxplots illustrating the distribution of the rank and respective standard deviation of trimming and randomized based DES methods for SW, SLOPE and STADE approaches in terms of RMSE. . . . .	75
3.21	Boxplots illustrating the distribution of the rank and respective standard deviation of trimming and MMR based DES methods for SW, SLOPE and STADE approaches in terms of RMSE. . . . .	76
4.1	Regressor Chain . . . . .	83

4.2	Boxplot illustrating the distribution of the rank and respective of MTR based meta-layer in terms of aRMSE. . . . .	86
4.3	Boxplot illustrating the distribution of the percentual difference and respective of of the best performing MTR based meta-layer in average (STADE_STA_LAG_LAN) in terms of aRMSE. . . . .	87
4.4	Boxplot illustrating the distribution of the rank and respective standard deviation of MTR-based DES methods and state of the art methods in terms of RMSE . . . . .	88
4.5	Boxplot illustrating the distribution of the percentual difference and respective of of the best performing MTR based (STADE_STA_LAG in terms of RMSE . . . . .	88
4.6	Proportion of probability of STADE_STA_LAG_LAN winning/drawing/losing against other MTR based DES and state of the art methods according to the Bayes sign test . . . . .	89
4.7	Boxplot illustrating the distribution of the ratio of the memory (left) and time (right) required STADE using ORI_STA_LAG relative to other MTR methods. . . . .	90
5.1	Student-Teacher general framework . . . . .	95
5.2	Boxplots illustrating the distribution of the rank and respective standard deviation of compressed student relative to ARF and RHT in terms of RMSE grouped by their respective compression scenario (SIMON, SIMON-SMILE, and SIMON-PELE) . . . . .	103
5.3	Boxplots illustrating the distribution of the rank and respective standard deviation of compressed student relative to BLAST and RHT in terms of RMSE grouped by their respective compression scenario (SIMON, SIMON-SMILE, and SIMON-PELE) . . . . .	104
5.4	Boxplots illustrating the distribution of the rank and respective standard deviation of compressed student relative to STACKING and RHT in terms of RMSE grouped by their respective compression scenario (SIMON, SIMON-SMILE, and SIMON-PELE) . . . . .	105
5.5	Boxplots illustrating the distribution of the rank and respective standard deviation of all compressed student relative to all teachers and RHT with the best performing compression scenario (SIMON-PELE) in terms of RMSE . . . . .	105
5.6	Boxplots illustrating the distribution of the average percentual improvement (expressed in percentage) and respective standard deviation of compressed student relative to teacher (right) and RHT (right) grouped by their respective compression scenario (SIMON, SIMON-SMILE, SIMON-PELE) in terms of RMSE. . . . .	108
5.7	Boxplots illustrating the distribution of the ratio and respective standard deviation of compressed student relative to RHT (left) and respective teacher (right) grouped by compression scenario (SIMON, SIMON-SMILE, SIMON-PELE) in terms of model size. . . . .	109

5.8	Boxplots illustrating the distribution of the ratio and respective standard deviation in terms of total running rime of compressed student relative to respective teacher (left) and RHT (right) grouped by compression scenario (SIMON, SIMON-SMILE, SIMON-PELE). . . . .	109
5.9	Boxplots of the distribution of the rank and respective standard deviation in terms of RMSE of the compressed students relative to teachers and RHT by scenario. . . . .	112
6.1	Competition streaming workflow . . . . .	114
6.2	Competition temporal data stream . . . . .	116
6.3	Windows with distance equal to stream period . . . . .	117
6.4	Platform architectures and services . . . . .	121
6.5	SCALAR data stream server workflow . . . . .	122
6.6	<i>.proto</i> file example . . . . .	123
6.7	SCALAR web interface to create a stream . . . . .	124
6.8	SCALAR web interface to create a competition . . . . .	124
6.9	SCALAR web interface with subscription information . . . . .	124
6.10	Live results on SCALAR web application . . . . .	125
6.11	Participants ranking on SCALAR web application . . . . .	125
A.1	Blocked prequential training workflow . . . . .	134

# List of Tables

3.1	Meta-Features grouped by category and their respective description . . . . .	45
3.2	Average ratio and respective standard deviation of the memory and time required by SLOPE compared to SW counterparts for similar $w$ and $\alpha$ and varying values of $k$ . . . . .	63
4.1	Multi-Target Regression methods . . . . .	84
5.1	Terminology (tags) and respective description for compression approaches . . .	102
5.2	Average rank and respective standard deviation of compression variants with different teachers (ARF, BLAST and STACKING) in terms of RMSE for 30 temporal streams using SIMON-PELE compression scenario. . . . .	106
6.1	Real-time Competition top results . . . . .	119
A.1	Data streams and respective summary . . . . .	132



# Introduction

## 1.1 CONTEXT AND MOTIVATION

Recent years have witnessed a tremendous surge of the *Internet of Things* (IoT) that can be defined as a large network of physical devices (sensors) that connect, interact, and generate data. These devices are continuously creating data at high frequency in the form of stream. The fields of application for IoT-based technologies are numerous and extend to all areas of everyday life such as health, transportation to name but a few of a vast number. In the healthcare area, IoT addresses cases where constant monitoring of health indicators is required for patients suffering from chronic disease such as diabetes. In the smart home area, companies like Orange, have started to introduce numerous IoT-based intelligent systems to help improving the quality of life for the people living in the house as well as their safety <sup>1</sup>. For example, smart energy applications focus on smart and efficient electricity, gas and water consumption. IoT applications also address industrial issues under a variety of names including Industry 4.0 or Industrial Internet of Thing (IIoT). The goal is to improve the way factories and workplaces operate making them safer, more efficient, and more environmentally friendly. In the smart cities, IoT applications contribute in reshaping the way people travel and use public transportation by means of on-demand app-based mobility services. The IoT technology contributes to improving the effectiveness of public transportation with real-time information and improve the customer experience. Other applications include parking space search in crowded cities like Paris or vehicle fleet tracking.

The ubiquitous data is at the core of decision-making in different business sectors and organizations. In fact, politicians as well as business practitioners increasingly rely on data-driven decision-making by extracting valuable insights in order to make relevant decisions. However, as the amount of data and its complexity continues to grow, its analysis becomes unfeasible for humans. *Artificial Intelligence* (AI) is the field of study that aims to produce

---

<sup>1</sup><https://www.orange.fr/maison>

machines that can replicate human's intelligence. Machine Learning (ML) is a sub-field of AI that is based on the idea that machines can learn from data that represents a phenomenon to build a mathematical model which learns to automatically identify patterns and make predictions.

In many cases, the data being collected is in the form of a set of observations captured over time associated with a temporal indicator, denoted as time-series. Time-series (TS) are used to model the dynamics of many real-world phenomena (health, stock market, infrastructure monitoring). More formally, a time-series  $Y$  as a temporally ordered sequence of values  $Y = \{y_1, \dots, y_t, \dots, y_n\}$  collected over a specific time interval (e.g., minutes, hours) but not necessarily, in which  $y_t$  is the value observed at time  $t$  and  $n$  is the number of observations [19]. Instances, are very likely to be temporally dependent that involves the impact of previous behavior on current and future behavior. One of the most challenging tasks in the field of AI is to model the process generating the observations over time based on past behavior in order to accurately predict future behavior. *Forecasting* denotes the process of estimating the future values  $Y$ , for example predicting at time  $t$  the value  $y_{t+1}$ . In fact, meticulously understanding the dynamics of time-series is a key component for decision-makers as it provides foresight and allows proactive decision-making.

Analyzing time-series has been an active research area over the last decades and considerable literature has accumulated in the field of TS analysis and forecasting [19, 27, 44, 72]. Traditional offline approaches for static datasets rely on massive memory storage and multiple passes on the same data. Traditionally, a training set, deemed to be the best representation of the studied phenomenon, is selected in order to train a predictive model. The model is then tested on unseen data to estimate its competence. Adjustments can be brought and the process can be repeated offline as many times as needed until a "good" model is rendered. Henceforth, the final learning model is deployed and used to make new predictions. However, the described traditional approaches have considerable limitations when applied on dynamic data streams.

A Data Stream  $S = \{y_1, y_2, \dots, y_t \dots\}$  (DS) is a potentially infinite sequence of observations that arrive one by one or in small batches over time and often at high rate. Each instance is associated with a timestamp and therefore provides a temporal order as defined for time-series. Besides, data streams are very likely to experience a variety of changes over time referred to as *concept drift*. The infinite and evolving nature of data streams raises new challenges that traditional AI/ML techniques are not designed to handle. In this context, it is crucial that predictive models are able to keep the pace with newly released data, not only in terms of volume but also the speed at which it is generated and processed. Stream Learning is the emerging research field which focuses on learning from infinite and evolving data streams. Therefore, stream mining algorithms must meet a set of requirements where it must i) process each instance only once at most and move to the next instance ii) use limited time and memory iii) cope and adapt to changes with the different sources of non-stationary variation [14, p. 11].

We present in Figure 1.1 a simplified presentation of everlasting training and prediction process of stream learning. For each new instance, a stream mining model is ready to do one of these two actions:

- Receive an unlabeled instance and therefore make a prediction based on the current

model.

- Receive the true value of the target for a previously seen instance and use it to update the current model (training)

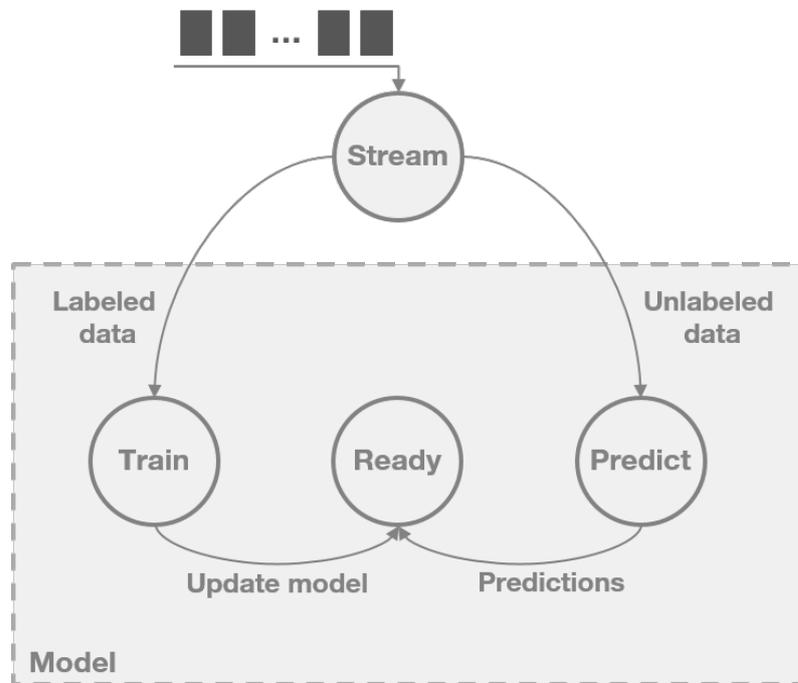


Figure 1.1: Stream supervised learning scenario. The predictive model toggles between two states: train and predict then back to ready. Labeled data is used to update the model whereas unlabeled instances are used to predict their corresponding target value.

Notwithstanding, stream mining methods often assume that instances in the flow are independently and identically distributed (i.i.d.). The assumption does not hold for time-series like data as it is the case for IoT applications, where instances are very likely to be temporally dependent. Read et al. argue that DS and TS fields are complementary since DS mining could take advantage of the strong theoretical background of TS as well as TS practitioners could adopt some of the DS techniques such as concept drift detection methods to enhance TS analysis field. In fact, in the context of our thesis, time series data arrive in the form of continuous and infinite flow, and consequently be treated as a data stream. Besides, data streams often exhibit temporal dependence (serial correlation or auto-correlation), where the previous values of the stream hold valuable information that experts can use to learn and predict future behavior [174].

Considerable literature, since the seminal work of Bates and Granger [8], reported that forecast accuracy can be substantially improved through the combination of multiple individual forecasts. In fact, the *No Free Lunch* theorem for supervised learning states that no learning algorithm is the best for all tasks. One of the most natural rationales for model combination are human experience in everyday life decision-making. Surowiecki in his famous book *“The Wisdom of Crowds ”* [146] addressed the problem of information aggregation in groups, resulting in decisions that, are often better than could have been

made by any single member of the group. The premise of *wisdom-of-the-crowds* was applied to predictive models in the area of Artificial Intelligence and Machine Learning. This has introduced the research field of ensemble methods, also called Multiple Classifier Systems (MCS) or committee. An ensemble is a collection of multiple smaller learners whose predictions are combined to form a single prediction [14, p. 129]. Ensembles, have become a major learning paradigm especially for data streams thanks to their ability to handle stream-specific challenges such as concept drift [94]. A good ensemble is generally believed to contain models that are as accurate as possible, and as diverse as possible. Diversity refers to the difference among the individual learners' behavior [98]. As a matter of fact, if models in the ensemble make the same predictions there would be no benefit from combining them.

Aiolfi and Timmermann [4] further add that different forecasting methods have varying levels of predictive performance at different times of the stream and hence the "best" models changes over time. Dynamic Ensemble Selection (DES) extends ensemble methods and aims at selecting on the fly for each incoming test instance, a subset of the most competent models only and combine their predictions. The weights of each individual forecast in the combination can be computed in such a way that higher weight is granted to better performing forecasters. The performance of forecasting models can be achieved by tracking their behavior in the recent past [115], referred to as windowing methods. Other DES methods are based on meta-learning to learn and predict the performance of individual forecasting models using characteristics extracted from the data, referred to as meta-features. DES methods can be categorized into two groups, namely *individual-based* and *group-based* [26]. In *individual-based* forecasters' behavior is modeled separately whereas *group-based* methods tend to explicitly capture their dependencies.

Despite their superior predictive performance, DES methods lead to exorbitant computational costs in terms of memory compared to individual models. This issue is particularly fastidious in IoT, edge computing and embedded systems applications where the model is deployed on devices with very limited resources. Likewise, data are released at high rate in the stream which imposes temporal constraints for the sake of real-time responsiveness. However, DES methods are time-consuming as they have to frequently update models' weights. Finally, ensemble methods behave like "black boxes", which creates serious challenges in interpreting their decision [99]. In fact, understanding predictive models and the mechanism behind their decisions is becoming a crucial issue addressed by the emerging research field of Explainable Artificial Intelligence.

The computational complexity and lack of interpretability issues of DES methods motivated Model Compression (MC). Model compression consists in training a single predictive model (Student) to mimic the behavior of a complex ensemble (Teacher) [31]. The student model has comparable predictive performance compared to the teacher while reducing the time and memory complexity. This approach is referred to as the Student-Teacher (ST) framework and has gained significant attention over the last years. ST was often applied for neural networks where a more compact single neural network was obtained with comparable predictive performance relative to the complex ensemble of neural networks [76]. MC methods were extensively studied for the classification task, whereas a few studies address the regression tasks that deal with continuous target values. More recently, model compression was investigated for DES methods on time-series forecasting task [39]. They show that

dynamic forecast combination methods can be compressed into an individual model that has comparable predictive performance while being drastically faster and more compact. Nevertheless, existing methods are mainly designed for offline learning and comprises two stages. First, the teacher is trained on a predefined training set and then used to predict on validation set. The predictions are then used to train the student in order to mimic the function learned by the teacher and then deployed to predict on all the upcoming instances. However, this framework is limited when applied to dynamic temporal data streams. In fact, when concept drift occurs, the collective behavior between the teacher and student is likely to be altered and thus loose the advantage of compressing a highly performing complex teacher.

## 1.2 CONTRIBUTIONS

The main research direction of the thesis addresses the aforementioned stakes about forecasting for evolving temporal data streams. This thesis contributes to the stream mining field and ensemble methods by introducing and exploring novel approaches that dynamically select and combine individual models to boost overall predictive performance and tackle stream specific challenges such as concept drift. In this section we summarize the main contributions.

In the context of Dynamic Ensemble Selection (DES) methods for temporal data streams forecasting (Chapter 3):

- We introduce a novel individual models performance evaluation that falls within the category of windowing based DES methods. We harness the notion of local performance computed on a set of the nearest neighbors queried over a sliding window. The assumption is that the forecaster is very likely to behave the same way on upcoming instances that fall within the local region.
- We investigate meta-learning based DES methods to predict future error of individual models in the pool using different categories of meta-features. In fact, we take advantage of general meta-features to characterize the data at hand as well as the behavior of other learning models. Besides, we leverage stream-specific meta-features in order to tackle concept drift that inevitably impacts the performance of forecasting models in the ensemble.
- We introduce several selection methods that enhance diversity among selected experts to a certain extent. The goal is to select a committee of models that are as accurate as possible and as diverse as possible in order to combine their predictions.

In the context of group-based DES (Chapter 4):

- We propose to take advantage of the potential dependencies within base-models' by formulating the DES task as a Multi-Target Regression (MTR) problem. We show that explicitly considering models' dependencies while learning their behavior improves overall predictive performance.
- We overview different adaptive MTR methods where the task is to simultaneously learn and predict several continuous valued targets and explicitly capturing their dependencies.

- To the best of our knowledge, this work is the first to use adaptive MTR methods for DES to model the behavior of each component in an ensemble of forecasters on data streams.

In the context of Model Compression (MC) (Chapter 5):

- We show the applicability of model compression for ensembles of forecasters in the streaming setting. We demonstrate that Student-Teacher (ST) based model compression can be effectively adapted to the stream setting to address uni-variate temporal streams forecasting.
- We propose a model compression approach that leverages both teacher's predictions as well as true values of the target.
- We introduce an adaptive model compression to cope with concept drift that may alter models' predictive performance. We monitor the loss of the student model and replace it with a new one when it becomes obsolete due to concept drift.

In the context of streaming machine learning competition (Chapter 6):

- We introduce the first streaming competition "Real-time Machine Learning Competition on Data Streams", at the IEEE Big Data 2019 conference. The competition falls within the network activity monitoring application using multi-step ahead point *forecasting* to predict the upcoming values to be observed in the stream based on historical data.
- We present SCALAR, the first platform dedicated to host streaming machine learning competitions where data are released in the form of stream and participants continuously submit their predictions in the form of stream as well.

### 1.3 PUBLICATIONS

The research work presented in this thesis has been submitted and published on scientific venues in the corresponding field. In the following we provide a list of the publications:

- Dihia Boulegane, Albert Bifet, and Giyyarpuram Madhusudan, "Arbitrated Dynamic Ensemble with Abstaining for Time-Series Forecasting on Data Streams." In: *2019 IEEE International Conference on Big Data, BigData 2019, Los Angeles, CA, USA, December 9-12, 2019. pp.1040-1045*
- Dihia Boulegane, Nedeljko Radulovic, Albert Bifet, Ghislain Fievet, Jimin Sohn, Yeonwoo Nam, Seojeong Yu and Dong-Wan Choi, "Real-Time Machine Learning Competition on Data Streams at the IEEE Big Data 2019." In: *2019 IEEE International Conference on Big Data, BigData 2019, Los Angeles, CA, USA, December 9-12, 2019. pp.3493-3497*
- Dihia Boulegane, Albert Bifet, Haytham Elghazel, and Giyyarpuram Madhusudan, "Streaming Time Series Forecasting using Multi-Target Regression with Dynamic Ensemble Selection." In: *2020 IEEE International Conference on Big Data, BigData 2020, Virtually, December 10-13, 2020. pp.2170-2179*

- Dihia Boulegane, Vitor Cerqueira Albert Bifet, and Giyyarpuram Madhusudan, "Adaptive Model Compression for Evolving Data Streams." In: *Submitted to ECML-PKDD 2021*.

Finally, another publication related to the work on the data stream competition is the following:

- Nedeljko Radulovic, Dihia Boulegane, and Albert Bifet, "SCALAR - A Platform for Real-time Machine Learning Competitions on Data Streams." In: *Journal of Open Source Software, December 2020. 5(56), 2676*

## 1.4 OUTLINE

The structure of this thesis is as follows:

- **Chapter 1:** We introduce the motivation of the topic of the thesis on dynamic ensembles or forecasters for streaming data. We highlight the main goals and challenges addressed. Finally, we summarize the contributions of the thesis.
- **Chapter 2:** We introduce the fundamental concepts related to the streaming setting and temporal data mining. We detail the commonalities between the fields of data stream mining and a well-established field that is time series analysis. We highlight data streams and cover the challenges imposed by their infinite and evolving nature that learning models must cope with. We define the problem of temporal data streams forecasting and dynamic selection and combination of predictive models. We survey existing methods in the literature state-of-the-art for dynamic ensemble selection and combination to cope with evolving data streams
- **Chapter 3:** We propose a new streaming Dynamic Ensemble Selection (DES) approaches. We investigate several methods to evaluate base-models predictive performance that are capable of coping with both concept drift and concept evolution. Besides, we propose different selection methods that promote both predictive performance and diversity while selecting a committee of experts whose predictions will be aggregated. The proposed DES methods tackle the problem of concept drift and meet the computational constraints imposed by the streaming setting.
- **Chapter 4:** We formulate the meta-learning based DES as a Multi-Target Regression (MTR) problem in order to leverage the dependency information among base-models in order to improve the predictive performance of the meta-layer. An overview of MTR approaches is depicted with an emphasis on methods tailored for the streaming setting.
- **Chapter 5:** We introduce a streaming model compression approach for dynamic ensembles to tackle the evolving temporal data streams forecasting task.
- **Chapter 7:** We conclude with a summary of the results achieved in this thesis, and a discussion on future developments.



## Background and Related Work

In this chapter, we provide a general overview of the background related to the topic of our research in the thesis namely data stream mining, ensemble methods and dynamic ensemble selection. In Section 2.1, we address the principle of forecasting task in the streaming setting with temporal dependency under the statistical supervised learning framework. A particular attention is given to evolving data streams that experience changes and how learning algorithms handle concept drift. Section 2.2 details ensemble approaches that dynamically combine several forecasts to improve overall predictive performance and discuss state-of-the-art methods. Section 2.3 introduces the Dynamic Ensemble Selection (DES) approach that aims at selecting for each instance in the stream the most competent models in the pool and combine their predictions. The rationale is that different learning models exhibit different levels of predictive performance along the input data stream and thus selecting the best ones only positively impacts ensemble's performance. Finally, Section 2.4 concludes the chapter with a summary on dynamic ensemble methods for forecasting on evolving data streams.

## 2.1 FORECASTING IN STREAMING SETTING

Recent years have witnessed the spread of a large number of data sources and particularly sources where the data are continuously collected at high frequency (fast rates) in the form of stream (flow). With the increasing ubiquity of data streams in applications such as sensor networks, Internet of Things (IoT), system monitoring, electricity demand, data stream mining has emerged as a crucial topic in the Machine Learning (ML) and Artificial Intelligence (IA) fields [2, 14, 95]. However, the emergence of streaming data introduced new challenges that traditional learning methods could not handle.

In this section, we lay the basis for our work on the thesis. We overview the data stream mining field and highlight its challenges compared to batch learning. Besides, we focus on the forecasting task in temporally evolving environments and overview existing methods throughout the literature especially ensemble methods.

### 2.1.1 DATA STREAMS AND TIME SERIES

**Definition 2.1.1 — Data Stream.** A data stream  $S$  is a potentially infinite sequence of observations (instances) that arrive one by one or in small batches over time. Each instance has a timestamp and therefore provides a temporal order for the stream.

As described in Definition 2.1.1, Data Streams (DS) have the characteristic of being an infinite flow where observations are continuously generated at high rate [57]. Besides, thanks to the timestamp associated to each instance, a natural temporal order comes up, which suggests that there might exist a relationship between the notion of DS and Time Series (TS). We define below the notion of time-series.

**Definition 2.1.2 — Time series.** A time-series  $Y$  is a temporally ordered sequence of values  $Y = \{y_1, \dots, y_t, \dots, y_n\}$  collected over a specific time interval (but not only), in which  $y_t$  is the value of the series at time  $t$  and  $n$  is the number of observations.

Time series, as the name suggests, are temporally ordered sequences where observations are generally captured in successive and equal time intervals (e.g., minutes, hours). Yet, some time series are recorded at irregular time intervals.

Hereafter, we discuss the commonalities and differences of DS and TS. Definitions 2.1.1 and 2.1.2, of DS and TS respectively, suggest that the two notions have much in common and can potentially be unified. However, conversely to time series, where data points are likely to be temporally dependent, data streams assume that instances are independently and identically distributed (i.i.d.). Despite this significant difference, the relationship between TS and DS has been questioned in several works. One can think about the relationship between DS and TS in two different ways: time series data may arrive in the form of continuous and infinite flow, and consequently be treated as a data stream. On the other hand, data streams can exhibit temporal dependence and thus be considered as time series [70]. Temporal dependence (serial correlation or auto-correlation) is often encountered in traditional time series analysis [19], where the previous values of the series are often the only predictive information that experts can use. Žliobaitė et al. [174] have demonstrated that many of the benchmark data sets used in data streams mining show temporal dependence. More recently, Read et al. [128] have drawn in more details the relationship between the two notions and

concluded that DS is a special case from TS. Indeed, DS mining can be presented as a TS analysis extension where the data are unbound flow continuously captured over time. Read et al. argue that DS and TS fields are complementary since DS mining could take advantage of the strong theoretical background of TS as well as TS practitioners could adopt some of the DS techniques such as concept drift detection methods to enhance TS analysis field. Nonetheless, DS mining raises new challenges such as single pass and incremental learning. These challenges will be discussed in more details throughout the thesis.

The evolution of *Internet of Things* (IoT), has spawned several data sources such as sensors that are continuously recording measurements in different areas (environment, health ...) or consumption data such as electricity. Temporal dependence is very common in this kind of data streams and therefore joins the assumption of time series data [174]. One of the most popular goals behind temporal data analysis is to predict the future behavior of the data. This is more commonly referred to as *forecasting*. In fact, forecasting is required in many application domains for the decision-making process. In this context, it should be noted that Orange is one of leading providers of a LoRa based network <sup>1</sup> in France that enables long-range transmissions with low power consumption for connected devices. Business practitioners at the company pay particular attention to the quality of service provided to consumers. Real-time analysis and prediction of network traffic behavior is a proactive approach to ensure secure, reliable and qualitative network communication, and hence increase customer satisfaction. Analysis is performed on on infinite streams of data characterizing the activity of the network in order to model and forecast future behavior and potentially trigger abnormal events and faults. In this context, forecasting is an important part for data-driven decision-making based on accurate prediction of the control measures that reflect the reliability and condition of the network infrastructure.

In our work and the remainder of the thesis, we consider a *temporal data stream* as a continuous and infinite flow of time series data, this means that samples are not i.i.d sampled and the flow is supposedly endless. There exists an extensive literature in the field of TS analysis and forecasting [19, 27, 44, 72] that often rely on memory storage and multiple passes on the same data. However, the DS mining scenario brings new challenges such as: (i) the single-pass algorithm by instance; (ii) the incremental update of learning models as new data arrive; (iii) the reduction in memory requirements since it is impossible to maintain all data in memory; (iv) the minimization of time complexity so that the algorithm is capable of updating models before the arrival of new data. Consequently, some of the TS forecasting tools cannot be directly applied on a streaming data scenarios yet, they provide a fundamental background for the forecasting approaches in DS field.

In the next section, we will discuss in more details the forecasting tasks and related challenges in the streaming scenario.

### 2.1.2 FORECASTING WITH TEMPORAL STREAMING DATA

Temporal data streams as presented in Definition 2.1.1 are a suitable abstraction to support real-time analytics in order to extract valuable knowledge and train predictive models. One of the most important tasks of temporally dependent data streams analysis is to predict

---

<sup>1</sup><https://pro.orange.fr/actualites/la-technologie-lora-pour-les-pros-CNT00000JCIVP.html>

the future evolution of the data based on the past and present behavior. This is commonly referred to as *forecasting*. Forecasting is involved in many applications domains such as weather, health, electricity demand, telecommunication and infrastructure monitoring. The use of temporal data for understanding the past and predicting future is a fundamental part of business decisions in every sector mentioned above.

Formally, let  $S = \{y_1, y_2, \dots, y_t \dots\}$  be the observed temporal stream, we intend to forecast future values  $\hat{y}_{t+h}$  at time  $t$  where  $h$  is a positive integer that stands for the forecasting horizon (lead time) and  $t$  is the forecast origin. It is important to specify the time  $t$  at which the forecast  $\hat{y}_{t+h}$  is made as well as the forecasting horizon  $h$  as they determine the information available for the forecaster. The goal is to learn a model or function to estimate the future behavior of the stream

$$\hat{y}_{t+h} = f(\mathcal{I}_t, h) \quad (2.1)$$

where  $\mathcal{I}_t$  is the historical and predictive information and the function  $f$  is the forecasting method capturing the temporal stream model. We can distinguish between two different forecasting approaches based on the number of periods for which a forecast is desired, known as the forecasting horizon  $h$ . *One-step* ahead forecasting methods ( $h = 1$ ) predict a single future value at time  $t$ . *Multi-step* ahead forecasting consists in predicting, at time  $t$ , the next  $h$  values  $\{\hat{y}_{t+1}, \dots, \hat{y}_{t+h}\}$  where  $h > 1$  [147]. Multi-step ahead forecasting is more challenging than one-step ahead due to error accumulation and reduced accuracy [151]. In fact, higher forecasting horizons lead to a more difficult predictive task as uncertainty increases [162].

Likewise, we can distinguish between two types of forecasting tasks depending on the number of variables that are captured over time *univariate* and *multivariate*. *Univariate* temporal streams are one-dimensional, i.e, they comprise a single varying target. For example, data collected from a sensor measuring the temperature of a room every second. Univariate temporal streams are based on the assumption that the future behavior depends only on the historical data of the target variable itself and not any other effect. On the other hand, *multivariate* temporal streams involve multiple time-dependent variables. Each variable depends not only on its historical data but also on the other variables. This means that multivariate temporal streams can be decomposed into several vectors of univariate streams. Multi-variate processes are of great interest when several related univariate time series are observed simultaneously over time and may co-vary with each other. For example, in economics, we may be interested in the joint evolution of interest rates, money supply and unemployment [131].

Finally, we distinguish between point and interval forecasts [44]. The *point forecast* aims at predicting a single numeric value. On the other hand, the *forecast interval* gives a lower and upper bound within which we expect the true future value to lie. In our work we focus on numerical uni-variate temporal data streams only and point forecast, meaning that the target value  $y_t \in \mathbb{R}, \forall y \in S$ . Everything related to interval-forecast and multi-variate series will be beyond the scope of the thesis.

### 2.1.3 INCREMENTAL LEARNING

The infinite and evolving nature of data streams raises new challenges that batch traditional forecasting techniques are not designed to handle. We discuss here the general framework of

stream learning tailored to continuous flows of data. Data stream mining algorithms must meet the following requirements [14]:

- **Process one instance at a time:** Inspect each instance only once at most.
- **Use a limited amount of memory:** Data streams are assumed to be infinite, thus, storing data for further processing is impractical.
- **Work in a limited amount of time:** Use a limited amount of time to process each instance.
- **Be ready to predict at any point:** Conversely to offline models, stream models are continuously updated and should be ready to predict at any point in time.
- **Adapt to temporal changes:** Temporal data streams are likely to evolve over time which is a major challenge. In this context, learning algorithms should be able to cope with the different sources of non-stationary variation.

Basically, the goal of stream mining is to build models that are able to learn incrementally as new data are released in the stream. Besides, time and memory are important constraints that have to be considered while processing data. In fact, IoT related applications are often Moreover, data streams may evolve over time and experience considerable changes that algorithms have to cope with. These changes are referred to as concept drift and will be discussed in further details here after.

#### 2.1.4 CONCEPT DRIFT IN EVOLVING DATA STREAMS

Temporally-dependent data streams are likely to experience a variety of changes over time [143, 163]. In practice time-evolving data usually exhibit seasonal patterns [1] and are consequently said "non-stationary". The notion of time-series stationarity and concept drift are closely related to one another. In fact, when sources of non-stationary variation are at play, it is said that a concept drift occurs [41]. We emphasize on the definition of concept and concept drift for temporal data streams as presented in [128].

##### Concept

Every temporal data stream is generated by a set of functions called generating process [7, 88]. A sequence of observations produced by the same generating process over time constitutes a *concept*. Following the definition provided in [128], let  $P_t(\cdot)$  be the probability distribution playing at time  $t$  and generating observations  $y_t$ , i.e.,  $y_t \sim P_t(\cdot)$ . The behavior produced by  $P_t(\cdot)$ , such as time dependencies and trends is referred to as a *concept*. Assuming that  $S_c = \{P_{-\infty}, \dots, P_t, \dots, P_0\}$  is a set of probability distributions happening from a distant past to the present, if  $P_k = P_t$ , this means that same data distribution is used and, consequently, the same concept is generated.

##### Concept Drift

Any change in the underlying generating process echos through the entire data observations. This phenomenon is called *concept drift*. Concept drift is one of the main challenges when learning from time-dependent and evolving data streams. It complicates the learning process

where algorithms must be able to cope with the different sources of change [62]. Particularly, predictive models should be able to detect and adapt to concept drift while being robust to noise. Noise can be described as observations that do not fall within typical behavior determined by the current concept. Noise can be seen as fluctuations that may mislead the model and make it believe a concept drift occurred.

#### 2.1.4.1 Types of concept drift

Although changes in the temporal stream may be arbitrary and may occur anywhere in the stream, we can distinguish the main types of concept drift. We name below the four types of concept drift and illustrate them in Figure 2.1. We must concede that different naming may exist throughout the literature and that the one we adopt is not exclusive:

- **Abrupt:** the concept  $P_t$  has remained unchanged for a long time and then changes to a new concept  $P_{t+1}$  that is completely different from  $P_t$ . In this case, transition between the two concepts is sudden, and thus, adaptation time is vital since the old concept is no longer valid.
- **Gradual:** the data observed in the stream is generated from both the old  $P_t$  and the new concept  $P_{t+1}$  alternately during the transition period. The generating process alternates between the old and the new concept before settling down to the new concept exclusively.
- **Incremental:** the transition from old concept to the new concept happens with a smooth progression such that, at each step we experience tiny changes. The distance from the old concept increases whereas the distance to the new concept decreases.
- **Recurring:** the concepts that have appeared in the past tend to reappear in the future. This is particularly interesting with temporal streams due to seasonality and periodicity patterns.

#### 2.1.4.2 Concept drift adaptation

Learning algorithms must include an adaptation strategy to cope with changes in the presence of concept drift. Adaptation strategies can be divided into two main approaches: *blind* and *informed* [62]. We may find other naming in the literature for the two approaches where passive (resp. active) stands for the blind (resp. informed) method [52].

- **Blind (Passive):** blind adaptation strategies do not use any explicit change detection mechanism. Rather, the model is continuously updated as new data are released in the stream whether a drift happened or no.
- **Informed (Active):** informed methods explicitly use a concept drift detection mechanism to flag any change in the stream. If a change signal is detected, an adaptation operation is triggered to update the model and consolidate the new concept.

Another strategy for concept drift adaptation is based on *ensemble* approach. Basically, one or several models are called at different times or subsets of the data streams [14]. The ensemble is equipped with a manager that contains rules for creating, pruning, revising and

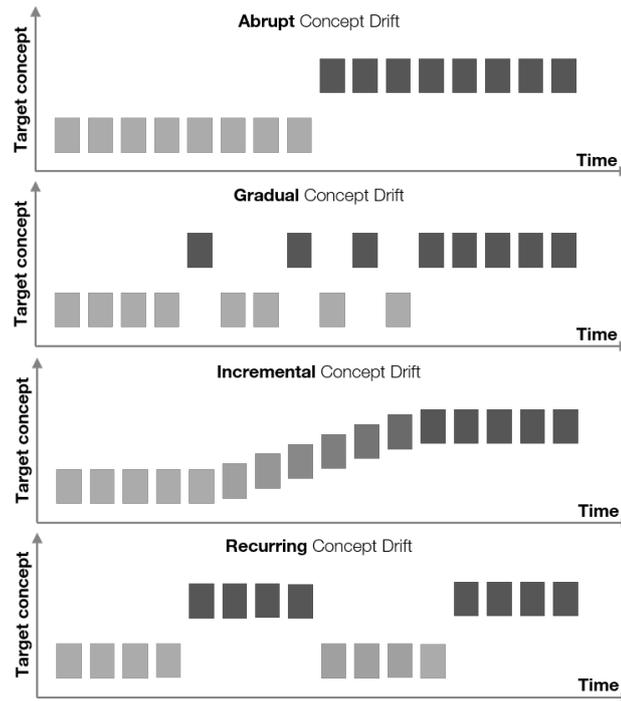


Figure 2.1: Different types of concept drift

combining individual models. The ensemble approach will be addressed in further details in Section 2.2. We refer to the work of Gama et al. [62] for a more detailed survey on concept drift adaptation.

#### 2.1.4.3 Concept drift detection

Informed concept drift adaptation approaches rely on specialized detection methods to explicitly identify any drift that occurs in the data stream. The goal of a drift detection method is to reliably report changes in the data characteristics while being robust to *noise* and *outliers*. Efficient drift detection is an important trade-off between reacting quickly to changes and having few false alarms. Besides, change detectors have to meet the stream mining requirements addressed in Section 2.1.3. We describe below one of the most popular and stream-friendly change detection methods, namely ADWIN. We point the reader to the survey of Gama et al. [62] for more information in drift detectors where a plethora of techniques have been proposed.

##### The ADWIN change detector

The ADWIN, for **AD**aptive **WIN**dowing, is a change detection method that presents theoretical guarantees [12]. ADWIN efficiently keeps a variable-length window of recently seen items in the stream (bits or real-valued variables). The window has the maximal length such that the following hypothesis holds: "There has been no change in the data distribution". The algorithm divides the window into two sub-windows ( $W_0, W_1$ ) and runs a test  $T(W_0, W_1, \delta)$  using their respective averages to determine if the two sub-windows are likely to come from the same distribution or no, i.e, a drift has occurred. The value  $\delta \in (0, 1)$  is a user defined parameter that stands for the confidence value. If a change is triggered,  $W_0$  is replaced by  $W_1$

and  $W_1$  is reset to a new window.

### 2.1.5 FORECASTING METHODS FOR EVOLVING DATA STREAMS

Read et al.[128] and Gomes et al. [70] have presented the data stream (DS) mining as an extension of the time series (TS) field, characterized by the infinite flow of data where instances are continuously released over time and potentially at high frequency. As discussed in Section 2.1.3, the infinite and evolving nature of data streams have introduced a list of requirements that batch methods are not designed to handle. In fact, the DS mining scenario brings new challenges such as: (i) the single-pass algorithm (at most) by instance; (ii) the incremental update of learning models as new data arrive; (iii) the reduction in memory requirements since it is impossible to maintain all data in memory; (iv) the minimization of time complexity so that the algorithm is capable of updating models before the arrival of new data. Consequently, some of the TS forecasting tools cannot be directly applied on a streaming data scenarios yet, they provide a fundamental background for the forecasting approaches in DS field.

There exists an extensive literature in the field of TS analysis [19, 27, 44, 72] that often rely on memory storage and multiple passes on the same data. These methods cannot easily be used in the streaming scenario, thus we focus on the methods that are tailored to the streaming setting requirements only. The forecasting methods described here can be updated when new instances are released. We assume immediate feedback from the environment, i.e. true values of the series are immediately released at time  $t$ . For convenience, we describe here the uni-variate temporal streams and one-step ahead point forecasts only where we output at time  $t$  a prediction  $\hat{y}_{t+1} \in \mathbb{R}$ .

#### 2.1.5.1 Simple forecasting methods

Several forecasting models have been proposed in the literature, some of them, although very simple, have demonstrated effective predictive performance in practice.

- **Average:** the next value of the series is predicted according to the historical mean:

$$\hat{y}_{t+1} = \bar{y} = \frac{1}{t} \sum_{i=1}^t y_i \quad (2.2)$$

The historical mean can include forgetting mechanism to cope with the evolving data streams. In fact, it is assumed that recent instances are more important, thus a fading factor or a sliding window can be used to emphasize on recent behavior.

- **Naive:** also known as the random walk forecast, predicts the next value as being the value of the latest observation:

$$\hat{y}_{t+1} = y_t \quad (2.3)$$

- **Drift:** a variation of the *Naive* method that leaves some room to an increase or decrease over time. The forecast value is equal to last value plus a drift value that stands for the average change seen in the historical data.

$$\hat{y}_{t+1} = y_t + \frac{1}{t-1} \sum_{i=1}^t (y_i - y_{i-1}) \quad (2.4)$$

All of the *Average*, *Naive* and *Drift* forecasting methods are suitable for adaptive incremental learning as they are based on means that are incremental operations.

### 2.1.5.2 Exponential Smoothing

The exponential smoothing approach is based on the idea that the future values of the series can be modeled using a linear combination (weighted average) of its past observations. The Simple Exponential Smoothing (SES) [30] produces a weighted average of past values where the weight decays exponentially as the observations get older. The one-step-ahead forecast is defined by:

$$\hat{y}_{t+1} = \alpha y_t + \alpha(1 - \alpha)y_{t-1} + \alpha(1 - \alpha)^2 y_{t-2} + \dots \quad (2.5)$$

where  $0 \leq \alpha \leq 1$  is the smoothing parameter that controls the rate at which the weights decrease. We describe an alternative representation of SES that better highlights the incremental implementation of Equation 2.5 for streaming data. Let  $y_1$  be the first value observed in the stream at time  $t = 1$  and  $s_t$  be the output of the SES that can be considered at time  $t$  as the best estimate of the next  $y_{t+1}$ . The representation comprises a *forecast* (Equation 2.6) and a *smoothing* equation (Equation 2.7).

$$\text{Forecast equation: } \hat{y}_{t+1} = s_t \quad (2.6)$$

$$\text{Smoothing equation: } s_t = \alpha y_t + (1 - \alpha)s_{t-1} \quad (2.7)$$

The SES method is suitable for forecasting temporal data with no clear trend or seasonal pattern, i.e. no periodic and repetitive variations at regular intervals (weekly, monthly, or quarterly). The importance given for past and current observation vary depending on the value of  $\alpha$ . For any  $\alpha \in (0, 1)$ , the weights attached to the observations decrease exponentially as observations get older. If  $\alpha$  is close to 0, higher weights are attributed to older observations. If  $\alpha$  is close to 1, more weight is given to more recent observations. If  $\alpha = 1$ , the forecast is equivalent to the *Naive* method as  $\hat{y}_{t+1} = y_t$ . There are several methods to achieve exponential smoothing, we refer the reader to the work of Hyndman and Athanasopoulos [81] for a more complete survey.

### 2.1.5.3 ARIMA

The ARMA (Auto-Regressive Moving Average) is one of the most commonly used methods to model uni-variate time series.  $ARMA(p, q)$  combines two components:  $AR(p)$ , and  $MA(q)$

Auto-Regressive (AR) model aims at forecasting the future values using a linear combination of past values of the target variable of interest. Therefore, an auto-regressive model of order  $p$  (referred to as  $AR(p)$ ) can be written a

$$\hat{y}_t = c + \sum_{i=1}^p \phi_i y_{t-i} + \varepsilon_t \quad (2.8)$$

where  $\varepsilon_t$  is white noise. This can be seen as a multiple regression with lagged values of  $y_t$  as features.

Similarly, the  $MA(q)$  model uses past errors to model the series:

$$\hat{y}_t = \mu + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t \quad (2.9)$$

where  $\mu$  is the mean of the observations,  $\theta_i, \forall i \in \{1, \dots, q\}$  are the parameters of the models and  $q$  stands for the order. The  $MA(q)$  methods models the time series according to random errors that occurred in the previous  $q$  lags [44].

The model  $ARMA(p, q)$  is defined for stationary time-series and combines the model  $AR(p)$  and the  $MA(q)$  models as follows:

$$\hat{y}_t = c + \sum_{i=1}^p \phi_i y_{t-i} + \sum_{i=1}^q \theta_i \varepsilon_{t-i} + \varepsilon_t \quad (2.10)$$

However, real-world time-series often exhibit a non-stationary dynamic including trend and seasonality.  $ARIMA(p, q, d)$  addresses this issue by introducing an integration parameter of order  $d$ . This means that  $ARIMA$  first applies  $d$  differencing transformations to the time-series in order to make it stationary and then it applies  $ARMA(p, q)$ .

#### 2.1.5.4 Forecasting based on regression

The forecasting task can be cast as a regression problem that falls within the supervised learning framework. The basic idea is that we forecast the future values ( $\hat{y}_i$ ) of temporal data stream assuming that it has a relationship with other variables comprised in the feature vector  $x_i$  such that

$$\hat{y}_i = f(x_i) + \varepsilon_i \quad (2.11)$$

where  $f$  is the unknown regression function and  $\varepsilon_i$ s are the random errors. Our work focuses on uni-variate temporal streams only, therefore, the feature vector  $x_{t+1}$  contains information about past observations of the variable of interest  $y$ . Section 2.1.6 will address regression methods to perform forecasting task on streaming data. More particularly, it focuses on ensembles methods and will highlight state-of-the-art methods that have proven competitive predictive performance.

### 2.1.6 FORECASTING WITH SUPERVISED LEARNING REGRESSION

The forecasting problem can be cast as a regression task that falls within the supervised learning approach according to the Statistical Learning Theory (SLT) framework designed by Vapnik [159]. This is motivated by the large quantity of data generated in the context of IoT applications. It relies on three main components: an input space  $\mathcal{X}$ , an output space  $\mathcal{Y}$  (in our case  $\mathcal{Y} \subseteq \mathbb{R}$ ) and a model  $f : \mathcal{X} \rightarrow \mathcal{Y}$  to represent the relationship between inputs and their respective outputs. The goal is to learn the function  $f$  from labeled data  $(X, Y)$  sampled according to some Joint Probability Distribution  $JPD(\mathcal{X} \times \mathcal{Y})$ . Each data sample  $(X_i, y_i) \in (X, Y)$  is comprised of the feature vector  $X_i \in \mathbb{R}^d$  of dimension  $d$  so that  $X_i = \langle x_i^1, x_i^2, \dots, x_i^d \rangle$  and the corresponding output value  $y_i \in \mathbb{R}$ . The function  $f$  is then used on unseen instances  $X'$  where  $\{X \cup X'\} = \emptyset$ , to predict their respective target values, that is:

$$\hat{y}'_k = f(X'_k) \quad (2.12)$$

The validity of the regression results as described above depends on two main assumptions: (i) the distribution of the data  $\text{JPD}(\mathcal{X} \times \mathcal{Y})$  must be stationary, i.e., it cannot change over time, and (ii) the data observations are sampled following the independent and identically distributed (i.i.d) approach. Nevertheless, these assumptions are contradictory to the assumptions made when dealing with streaming data. In fact, the evolving nature of data streams, where concept drift is likely to happen at any time, contradicts the assumption made about the stationarity of the distribution. Besides, data streams often exhibit temporal dependence [128], which negates the second assumption above stating the i.i.d sampling procedure.

The two issues addressed above make the supervised learning model deficient for streaming data, and therefore needs to encompass mechanism to overcome these challenges. The stationarity constraint is handled using concept drift detection and adaptation methods detailed in Section 2.1.4. Regarding the controversy of data independence, it can be handled provided some data transformation process. This can be achieved using the auto-regressive model based on Takens' time delay embedding [149].

### 2.1.6.1 Auto-regressive model

Auto-regressive (AR) approaches are popular in the forecasting problem where the series is projected into an Euclidean space to unfold data dependency. Suppose that at time  $t$ , we need to make a prediction for the next value  $y_{t+1}$ , by then we will have already seen observations  $\{y_1, y_2, \dots, y_{t-1}, y_t\}$ . We assume immediate arrival of true target values after each prediction (no delay in labels)<sup>2</sup>. Following the supervised learning terminology presented above, each data instance  $(X_t, y_t)$  is modelled based on the past  $p$  values, i.e, the vector of features takes the form of  $X_t = \langle y_{t-1}, y_{t-2}, \dots, y_{t-p} \rangle$ . The time delay embedding requires the parameter  $p$  that controls the time lag, also referred to as the embedding dimension. This determines the number of historical observations to be included as explanatory variables, the higher the value of  $p$  the further the AR process looks back in the past.

The AR approach leads to a multiple regression task where the temporal dependency in the data stream is modelled by incorporating past observations as explanatory variables and therefore unfold data dependence that constitutes a major obstacle for the application of supervised regression models in temporally dependent data streams. In other words, in a multiple regression model, we forecast the variable of interest using a combination of the different predictors. In an AR model, we forecast the variable of interest using a combination of a finite set of its past values. This is a common approach to model uni-variate temporal data and can be found at the core of several state-of-the-art forecasting models such as ARIMA [19] discussed in Section 2.1.5.3 and time-lagged neural networks [55].

### 2.1.6.2 Regression methods for streaming data

Having set out the temporal stream forecasting as supervised learning task involving regression algorithms, we can now discuss a set of existing regression algorithms that were proposed to handle streaming data requirements and challenges. We discuss below the most renowned ones throughout the literature.

<sup>2</sup>This assumption is very common in the data stream mining field, yet sometimes it is not realistic

### K-nearest neighbors

The  $K$ -Nearest Neighbor (KNN) is a regression method that keeps track of the last training samples over a sliding window  $w$  and predictions are obtained following a two-stage procedure: (i) find the closest  $k$  neighbors to the test sample in the data window, (ii) aggregate the target values of the neighbors to predict the target value for the test sample at hand.

The streaming KNN method requires 4 parameters (i)  $k$ : the number of nearest neighbors to search for (ii)  $w$  the size of the sliding window to be maintained online, (iii) the distance metric to be used that highly impact the predictive performance of the method. The Euclidean distance is appropriate for dense instances with numeric features which corresponds to the regression based forecasting task [14, p. 145] and, (iv) the aggregation method could be the average, median or a weighted average, the higher the distance the lower the weight.

### Decision Trees

Fast Incremental Model Tree with Drift Detection (FIMT-DD) [84] is an any time incremental decision tree model for time-changing data streams with explicit drift detection. FIMT-DD is able to learn very fast and the only memory it requires is for storing sufficient statistics at tree leaves where predictions are computed using linear models. It effectively maintains an up-to-date model to cope with different types of concept drifts. The algorithm enables local change detection and adaptation (using Page-Hinkley [121]), avoiding the costs of re-growing the whole tree when only local changes are necessary. When a sub-tree is under-performing, an alternative tree grows with new incoming instances and replaces the original tree if it performs better.

### Ensemble methods

Ensemble methods are combinations of several small models whose individual predictions are aggregated to output a final prediction. Ensembles tend to improve overall predictive performance at the expense of additional time and memory complexity. Ensembles are even more attractive for the streaming data setting as they allow adaptation to concept drift and cope with changes that may occur in the temporal stream. There exist a plethora of ensemble methods tailored to the problem of temporal data streams forecasting that will be discussed in further details in Section 2.2

## 2.1.7 EVALUATING FORECASTING MODELS

Mining data streams poses new challenges that have to be considered when evaluating models. In batch learning, the evaluation procedure determines which examples are used for training and which are used for testing the model. This is often achieved by splitting the data into disjoint subsets (training and test data set)<sup>3</sup>. However, in the streaming setting, the data flow is assumed to be infinite and therefore, algorithm evaluation must be adapted in order to assess learning models' performance.

### 2.1.7.1 Prequential evaluation

The prequential (*interleaved test-then-train*) evaluation is a prevalent performance evaluation method in the stream learning literature [49]. In the prequential evaluation, each individual example is first used to test the model *before* it is used for updating it. The performance

<sup>3</sup>when data is limited, cross-validation is preferred

measure can be incrementally updated along the stream. Besides, the prequential evaluation implements the idea that more recent examples are more important using a sliding window or a decaying factor. The size of the window or the value of the fading factor are user defined parameters [14, p. 88].

### 2.1.7.2 Evaluation metrics

To evaluate the performance of the proposed forecasting algorithms, we rely on a set of evaluation metrics to quantify models' loss. We measure the difference between the predicted values by the model and the true values that are observed in the stream. Several evaluation metrics have been depicted in the literature:

- MSE: Mean Squared Error

$$\text{MSE} = \frac{1}{n} \sum_{t=1}^n (\hat{y}_t - y_t)^2 \quad (2.13)$$

- RMSE: Root Mean Squared Error

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n} \sum_{t=1}^n (\hat{y}_t - y_t)^2} \quad (2.14)$$

- MAE: Mean Absolute Error

$$\text{MAE} = \frac{1}{n} \sum_{t=1}^n |\hat{y}_t - y_t| \quad (2.15)$$

- SMAPE: Symmetric Mean Absolute Percentage Error

$$\text{SMAPE} = \begin{cases} 0, & \text{if } y_t, \hat{y}_t = 0. \\ \frac{100}{n} * \sum_{t=1}^n 2 \times \frac{|\hat{y}_t - y_t|}{|\hat{y}_t| + |y_t|} & \text{otherwise.} \end{cases} \quad (2.16)$$

The factor 2 can be removed to make the  $\text{sMAPE}$  values  $\in [0, 100]/[0, 1]$  to ease its interpretation as a percentage value without loss of generality.

The evaluation metrics help comparing the relative predictive performance of different forecasting methods on the same data. The goal is make these forecast errors as small as possible over the stream.

## 2.2 DYNAMIC ENSEMBLES OF FORECASTERS FOR STREAMING DATA

An ensemble can be described as a collection of multiple smaller learners whose predictions are combined to form a single final prediction [14]. The goal of ensemble methods is to improve predictive performance compared to any single learner in the pool. One of the most natural rationales for ensemble methods are human experience in everyday life decision making. Surowiecki in his famous book “*The Wisdom of Crowds*” [146] addressed the problem of information aggregation in groups, resulting in decisions that, are often better

than could have been made by any single member of the group. Authors have presented a plethora of examples and, amongst them, point estimation of a continuous quantity. We refer to the experience conducted by the statistician Francis Galton at the 1906 country fair in the city of Plymouth where nearly 800 people participated in a contest where the goal is to guess the weight of a slaughtered and dressed ox. The person who guessed closest to the real weight of the ox won a prize. He observed that the median guess, 1207 pounds, was accurate within 1% of the true weight of 1198 pounds. This collective guess was not only better than the actual winner of the contest but also better than the guesses made by cattle experts at the fair. In fact, he found that if the same question was asked to enough people, they might come up with better answers than even experts. Later, the choice between the median and the mean was discussed and it turned out that the mean would have been a better approximation for the ox's weight compared to the median [160]. This early example of the benefits of forecast combination has opened the doors to new ensemble methods.

### 2.2.1 THE PREMISE OF ENSEMBLE METHODS

The premise of *wisdom-of-the-crowds* was applied to predictive models in the area of machine learning and computational intelligence. This has given rise to the field of ensemble methods. In both batch and streaming scenarios, ensemble methods tend to improve overall predictive performance over any of their components at the cost of more time and memory resources. Ensemble methods have often demonstrated very promising results particularly for the streaming setting. In fact, if there were an expert model, referred to as an oracle, whose predictions were always true, there would be no need to any other decision and thus, there would be no need for ensemble methods. However, no such oracle exists and every model has some defects.

The original goal for combining several opinions in our daily lives is to improve our confidence that we are making the right decision [125]. The reason is no different in the machine-learning field by weighing various predictions, and efficiently combining to obtain a final decision to boost predictive performance. In this section, we provide a background on ensemble methods, particularly for streaming data. Next, we discuss the fundamental aspects of any ensemble approach, that is, diversity, generating and training ensemble members, and finally combining their predictions. We will then address a new challenge in the field, that is ensemble or model selection, that aims at selecting the most appropriate components only before combining their predictions.

Ensemble methods, also called Multiple Classifier Systems (MCS), committee or mixture of experts, have been at the center of the machine-learning and AI research directions in recent years especially for data streams [94]. We describe more formally the notion of ensemble methods in Definition 2.2.1. We will refer to this category of approaches as *ensemble* and to individual models comprising the ensemble as *experts* or *base-models* interchangeably in the remainder of the thesis.

**Definition 2.2.1 — Ensemble.** An ensemble is a set of learners whose individual decisions are combined in some way to predict on new examples [51].

It is not clear when the history of ensemble methods started but we can detail three domains that can be affiliated to the early stages of the current research area of ensemble

methods. Zhou [172, p. 16] have attributed the early contributions for ensemble methods to three main fields: combining classifiers, ensembles of weak learners and mixture of experts. Combining classifiers was mostly studied in the pattern recognition community and aims at designing powerful rules to combine strong classifiers. Ensembles of weak learners was mostly studied in the machine learning community where researchers often work on weak learners and try to design powerful algorithms to boost the performance from weak to strong learner. Mixture of experts was mostly studied in the neural networks community and adopt the *divide-and-conquer* strategy to cover different input space regions using different learners. Then a function is in charge of switching between experts, i.e. which learner to use according to the data at hand.

Ensemble methods have become a major learning paradigm and are particularly popular in the data stream context. They offer many advantages over single-model approaches and they can be used to handle data-stream-specific challenges such as concept drift. Figure 2.2 details a generic approach for ensemble methods designed for streaming data. Let  $M$  be an ensemble of size  $m$ , i.e, there are  $m$  individual models in the ensemble. For each incoming test instance  $x_t$  in the stream, each individual model  $M^i \in M$  provides its prediction  $\hat{y}_t^i$  and all the predictions are then combined according to some aggregation function defined in the combiner  $g$  to output a single final prediction  $\hat{y}_t$ .

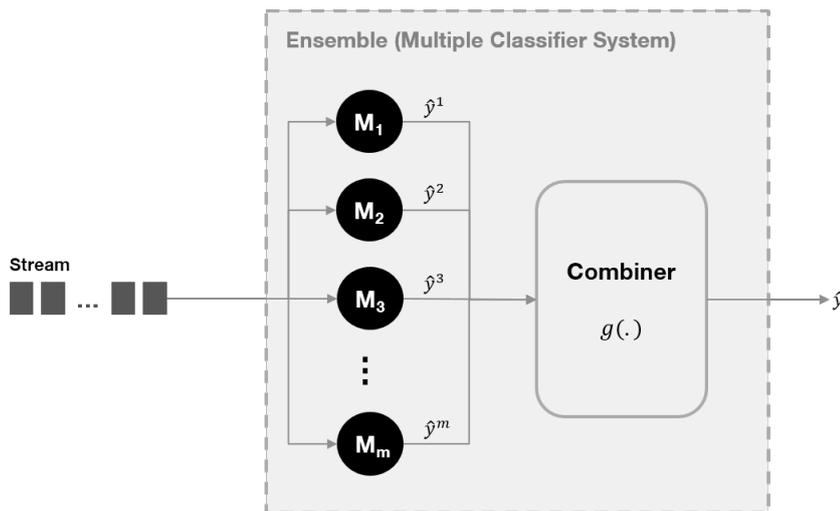


Figure 2.2: A common ensemble architecture

### 2.2.1.1 Why combination works ?

Building a strong single model is a difficult task whereas a set of "weak" learners is relatively easy to obtain and can be effectively learned and combined into a single model to boost overall predictive performance [56]. In fact, according to the "*No free lunch theorem*" formulated by Wolpert [165], there is not a single model that is appropriate for all tasks, rather, each learner has varying level of expertise according to the data at hand. As stated in the review article published by Ho [78]:

*Instead of looking for the best set of features and the best classifier, now we look for the best set of classifiers and then the best combination method.*

Dietterich [51] has attributed the benefits of ensemble methods combination to three main reasons, that is statistical, computational, and representational.

- **Statistical:** the set of models is too large to explore and there may be several different algorithms giving the same predictive performance. Choosing a single model includes a risk that a mistakenly chosen one could perform very poorly on future data. Therefore, combining several models reduces the risk of choosing a wrong one.
- **Computational:** learning algorithms often perform local search that may get stuck in local optima. In cases where there is enough training data, it may still be computationally difficult for the learning algorithms to find the best hypothesis. An ensemble constructed by running the local search from many different starting points may provide a better approximation to the true unknown function than any of the individual model.
- **Representational:** often, the true unknown hypothesis could not be represented by any hypothesis in the hypothesis space. Combining the hypotheses may expand the space of representable functions, and thus the learning algorithm may be able to approximate more accurately the true unknown hypothesis.

A learning algorithm that suffers from the statistical issue is generally said to have a high "*variance*", a learning algorithm that suffers from the computational issue can be described as having a high "*computational variance*", and a learning algorithm that suffers from the representational issue is generally said to have a high "*bias*". Therefore, through combination, the variance as well as the bias of learning algorithms may be reduced [172]. This is discussed in more detail in Section 2.2.1.3. That being said, it is important to mention that combining several models does not necessarily lead to a predictive performance that is guaranteed to be better than the best component model in the ensemble. However, it reduces the risk of choosing a model with poor performance [172].

### 2.2.1.2 Ensemble diversity

One of the key elements required for the success of ensemble methods is “diversity” [28]. Ensemble diversity is the disagreement (difference) among the individual learners outputs [98]. It is a fundamental notion in ensemble methods as it is closely related to its predictive performance. Models in the ensemble should not make the same errors otherwise there would be no benefit from combining them. To get a good ensemble, it is generally believed that the base learners should be as accurate as possible, and as diverse as possible. In fact, the individual outputs may compensate for each other’s discrepancies. However, generating diverse individual learners, is not easy since they are trained for the same task and from the same data. This, in addition to the fact that individual models must not be too weak, usually leads to highly correlated models.

Basically, the success of an ensemble lies in achieving a good trade-off between individual models performance and diversity. Ensembles comprising accurate models only is often worse than ensembles combining some accurate models together with some relatively weak ones. In fact, complementarity is more important than pure accuracy [172]. It is crucial to have individual models that are as unique as possible, particularly with regards to predictions

errors [124]. However, despite the fact that diversity is crucial, there is still a lack of understanding of how diversity impacts predictive performance throughout the literature.

There exists several methods to enhance the diversity of an ensemble. Some methods rely on input data manipulation (bagging and boosting [22]), using different parameters for the base models or even using different algorithms. The rationale behind using different algorithms to generate ensemble components is that different inductive biases lead to individual models that cover different regions of the hypothesis space. Diversity enhancing methods will be addressed in more details in Section 2.2.3. We refer the reader to the book of Kuncheva [98] and the survey of Brown et al. [28] for further reading about ensemble diversity and its impact on predictive performance.

### 2.2.1.3 The Bias-Variance-Covariance decomposition

Error decomposition is an important tool for analyzing the performance of learning algorithms. Geman, Bienenstock, and Doursat [63] have presented the bias-variance quadratic loss decomposition of a single predictive model that breaks the error into three components which are intrinsic noise, bias and variance. The intrinsic noise is a lower bound on the expected error of any learning algorithm on the target; the bias measures how closely the average estimate of the learning algorithm is able to approximate the target; the variance measures how much the estimate of the learning approach fluctuates for different training sets of the same size [172]. The intrinsic noise is difficult to estimate, it is often subsumed into the bias term. Thus, the generalization error is broken into the bias and the variance terms only. These two terms are usually inversely related, reducing the bias leads to an increase in variance, and vice versa. Let  $y$  be the target and  $M$  the learning model, the Mean Squared Error (MSE) can be decomposed

$$\begin{aligned} \text{MSE}(h) &= \mathbb{E}[(M - y)^2] \\ &= (\mathbb{E}[M] - y)^2 + \mathbb{E}[(M - \mathbb{E}[M])^2] \\ &= \text{bias}^2(M) + \text{variance}(M) \end{aligned} \quad (2.17)$$

where the bias and variance of learner  $h$  is respectively

$$\text{bias}(M) = \mathbb{E}[M] - y \quad (2.18)$$

$$\text{variance}(M) = \mathbb{E}(M - \mathbb{E}[M])^2 \quad (2.19)$$

For an ensemble of  $m$  models  $M_1, \dots, M_m$ , the decomposition of Equation 2.17 can be further expanded, yielding the bias-variance-covariance decomposition [157]. Without loss of generality, we suppose that the individual learners predictions are combined with equal weights (uniformly). The average bias, average variance, and average covariance of the  $m$  individual models comprising the ensemble are defined respectively as

$$\overline{\text{bias}}(M) = \frac{1}{m} \sum_{i=1}^m (\mathbb{E}[M_i] - y) \quad (2.20)$$

$$\overline{\text{variance}}(M) = \frac{1}{m} \sum_{i=1}^m \mathbb{E}(M_i - \mathbb{E}[M_i])^2 \quad (2.21)$$

$$\overline{\text{covariance}}(M) = \frac{1}{m(m-1)} \sum_{i=1}^m \sum_{\substack{j=1 \\ j \neq i}}^m \mathbb{E}(M_i - \mathbb{E}[M_i])\mathbb{E}(M_j - \mathbb{E}[M_j]) \quad (2.22)$$

Therefore, the **bias-variance-covariance** decomposition of the MSE of an ensemble is

$$\text{MSE}(M) = \overline{\text{bias}}^2(M) + \frac{1}{m} \overline{\text{variance}}(M) + \left(1 - \frac{1}{m}\right) \overline{\text{covariance}}(M) \quad (2.23)$$

Equation 2.23 shows that, in addition to the bias and variance of the individual forecasters, the generalization error (MSE) of the ensemble is closely related to the covariance term [29]. The covariance term models the correlation between the individual predictions and consequently, ensemble's diversity. The smaller the covariance, the better the ensemble. If all the learners make similar errors, the covariance will be large, thus it is preferred that the individual learners make different errors. The covariance term shows that diversity is important to improve ensemble predictive performance. The optimum *diversity* is that which optimally balances the components to reduce the overall MSE.

### 2.2.2 DYNAMIC COMBINATION OF FORECASTERS

The forecasting task is no exception, in fact, combining several forecasters has emerged as a very promising approach in the late 1960s since the seminal work of Reid [129, 130] and Bates and Granger [8]. Following Yang [168], the forecast combination methods can be classified into two main categories, depending on the goal of the combination. The first class, denoted *combination for improvement* aims at finding the best linear combination of a set of forecasts. The second class, denoted *combination for adaptation* aims to perform as well as the best individual procedure. It can be interpreted as a dynamic model selection, where the combination tends to rant all the weight to the best available forecaster. The idea of model selection and combination in a non-stationary environment is that the best available forecaster may change over time.

A plethora of both theoretical and experimental studies have demonstrated that forecast accuracy can be considerably improved through the combination of multiple individual forecasts. Many exhaustive surveys and discussions have sketched the contours of forecast combination's future developments and applications [3, 4, 6, 46, 50, 73]. Bates and Granger [8] have encouraged the idea of combination when alternative forecasts are available. Combined forecast should be created, possibly as a weighted average of the individual forecasts. When past performance can be tracked over time, the weights of each individual forecast in the combination can be computed in such a way that most weight is given to the approach that has performed best in the recent past [115]. In fact, simple approaches to forecast combination have often achieved remarkably high predictive performance [105, 106]. A large set of combination approaches have been studied in [46] and results have reported that simple averaging of forecasts performs at least as well as methods where individual weights are computed on the basis of past performance.

### The problem of dynamic forecast combination

We describe more formally the problem of dynamic forecast combination following the definitions presented in [152]. Suppose that we are interested in forecasting a single variable  $y$ , one or multiple steps ahead. Let  $t$  the forecast origin, and  $h \geq 1$  be the forecast horizon. The goal is to predict  $y_{t+h}$  given all the information known up to time  $t$  denoted by  $\mathcal{I}_t$ . For example, if the data generation process is assumed to be auto-regressive of order  $p$  as described in Section 2.1.6.1,  $\mathcal{I}_t$  may contain the past and present observations, that is,  $\mathcal{I}_t = \{y_s | s \leq t\}$ . The information set may also include models' historical behavior in order to track their respective performance.

Let  $M = \{M^1, M^2, \dots, M^m\}$  be a pool of  $m$  forecasters and  $\hat{Y}_{t+h} = \langle \hat{y}_{t+h}^1, \hat{y}_{t+h}^2, \dots, \hat{y}_{t+h}^m \rangle$  be the vector containing their respective predictions. Let  $\hat{y}_{t+h} = g(\hat{Y}_{t+h}, w_{t+h})$  be the combined point forecast as a function of the underlying individual forecasts and the parameters of the combination,  $w_{t+h} \in \mathcal{W}_t$ , where  $\mathcal{W}_t$  is a compact subset of  $\mathbb{R}^m$ . In other words, the combination parameters  $w_{t+h} = \langle w_{t+h}^1, w_{t+h}^2, \dots, w_{t+h}^m \rangle$  will be considered as the combination weights for each individual prediction. The combination parameters may be adapted as  $\mathcal{I}_t$  evolves over time. Point forecasts generally provide insufficient information for a decision maker as one may be interested in the degree of uncertainty surrounding the forecast. Nevertheless, the vast majority of studies on forecast combination has dealt with point forecast. Our work will focus on this case only, and everything related to interval-forecast will be beyond the scope of the thesis.

Forecast combination aims at choosing a time-varying mapping  $g_t$  from the vector of  $m$  individual forecasts  $\hat{Y}_{t+h}$  to a single real valued-target,  $\hat{y}_{t+h} \in \mathbb{R}$  that best approximates the real value  $y_{t+h}$  and consequently minimizes some loss function  $\mathcal{L}$  such as the Mean Squared Error (MSE) or any other evaluation metric discussed in Section 2.1.7.2. Timmermann [152] has provided a set of arguments to explain the advantages of forecast combination in time-varying data as in the case of data streams:

- Individual forecasts may be very differently affected by non-stationarities such as concept drift. Some models will adapt more or less slowly. Thus, combination of several forecasts from models with different degrees of adaptability will outperform forecasts from individual models [4].
- It is very unlikely that a single model dominates all other models at all points in time rather, the best performing model will change over time as the individual models have varying levels of expertise over time. Combining forecasts is a robust way to reduce predictions errors compared to individual forecasts.

### 2.2.3 DYNAMIC ENSEMBLES FOR EVOLVING TEMPORAL DATA STREAMS

Model combination has been widely addressed in the batch-learning environment where the datasets are assumed to be finite, and stationary [98, 167, 172]. Most of the existing work focuses on classification problems. However, these techniques are often not directly applicable to the regression task that was substantially less studied [111, 132]. Ensemble methods are particularly popular in the data stream setting as they can handle challenges related to evolving streaming data. Streaming ensemble methods can cope with concept

drifts and recurrent concepts [68]. Similarly to batch-learning, much research in the ensemble methods literature for streaming data has focused on the classification task [68], however, more recently, more attention is given to data stream regression task that deals with continuous valued outputs, such as the Adaptive Random Forest for regression [69].

Kuncheva [96] presented a broad taxonomy of the main dynamic ensemble methods that have been proposed throughout the literature to cope with changing environments. Online and adaptive ensemble methods can be grouped into five categories:

- **Dynamically combining:** (or horse racing algorithms), the algorithms include ensembles where the individual experts are trained in advance (offline) and the changes in the environment are handled by adapting the combination rule of the individual predictions.
- **Updating training data:** the algorithms in this group rely on using fresh data to introduce updates in the individual models. In this case, the combination rule may or may not change along the stream.
- **Updating the ensemble members:** the ensemble members are incrementally updated or retrained in a batch mode if blocks of data are made available along the stream.
- **Changing structure of the ensemble:** the ensemble tracks individual models' performance and if a concept drift occurs in the stream, the worst performing component is replaced by a new one that is trained on more recent data.
- **Adding new features:** the importance of the features will evolve along the life of the ensemble and the need for new features may emerge without redesigning the entire ensemble.

There are three important aspects to be considered when designing an ensemble method for streaming data [71]:

- **Generation:** defines how the individual models are created, commonly including some mechanism to enhance diversity among the base learners. This will be discussed in more detail in Section 2.2.3.
- **Combination:** describes how individual predictions are aggregated into the single ensemble output.
- **Update dynamics:** defines how and when base models will be reset or updated as new data are released in the stream. This is fundamental when dealing with evolving data streams to cope with changes in the underlying data distribution.

There are two main approaches to ensemble (model combinations) methods based on the learning algorithm they use as base models. *Homogeneous* ensembles, are composed of algorithms of the same kind whereas ensembles combining different kinds of algorithms are so-called *heterogeneous*. Thus, diversity in the ensembles is induced by different means in the two approaches.

### 2.2.3.1 Homogeneous ensembles

In homogeneous ensembles, the base models are generated using the same algorithm. Diversity is achieved by varying the information used to construct each model, such as using different subsets of training data (Bagging[8], Boosting[9]), or using different feature subspaces (Random Subspace Selection [77]). Random Patches (RP) [104] builds each individual model from a random patch of data where each patch is obtained by drawing random subsets of both instances and features from the whole data set. We describe below some of the most popular homogeneous ensemble methods for streaming data.

#### Bagging

The *bagging* method was first proposed for the batch setting by Breiman [21] where the same algorithm is used to infer a set of base models that are potentially different since they are trained on different *bootstrap samples* of the data. Each sample is created by drawing a random sample with replacement from the original training set. However, it is not feasible to draw a sample following this method in the streaming setting due to the infinite size of the data. Rather, Oza [119, 120] proposed Online Bagging which instead of sampling with replacement, gives each incoming instance a weight according to the distribution  $Poisson(1)$ . Later, ADWIN Bagging [13] was proposed to improve online bagging and better cope with changes in the stream. It monitors the error using ADWIN change detector and replaces the worst performing one by a new model as soon as a change is detected (replace the loser).

#### Boosting

Boosting algorithms combine several base models trained with different samples of the input stream. However, conversely to bagging that trains models in parallel, boosting creates models sequentially where each model depends on the performance of the previously constructed models. Boosting gives more weights to examples misclassified by the current model so that the next model in the sequence pays more attention to these examples. Oza [119, 120] proposed *Online Boosting* for streaming data where models are updated with a weight computed depending on the performance of the previous models.

#### Random Forest

Breiman [22] proposed Random Forest (RF) as an extension for bagging using tree models in the ensemble. RF injects more randomness to increase diversity among decision tree base predictors. In addition to the randomization of input data by sampling, each internal node of a tree is based on a small subset of the original feature set selected randomly, often the square root of the total number. Thus, the internal construction of each base tree is randomized. Adaptive Random Forest (ARF) is a streaming implementation of RF that includes an effective resampling method and an adaptation strategy to cope with different types of concept drift [66].

#### Random Patches

The Random Patches (RP) framework follows a very simple, yet effective, ensemble method that builds each individual model of the pool from a random patch of data obtained by drawing random subsets of both instances and features from the whole dataset [104]. Gomes, Read, and Bifet [67] presented the Streaming Random Patches (SRP) method that is an extension of Random Patches for classification tasks and meets the streaming setting requirements.

### 2.2.3.2 Heterogeneous ensembles

Conversely to *homogeneous* ensemble, *heterogeneous* ensemble include a hybrid pool of models. The rationale behind heterogeneous methods is that different models may have different expertise about the input data as they are built on different inductive biases.

Langdon, Barrett, and Buxton [101] proposed a generic method to combine decision trees and artificial neural networks in the pharmaceutical field related to drug discovery. Caruana et al. [36] proposed a framework for heterogeneous based on a large library including different predictors (at the expense of the individual models performances) and efficiently combine and select them to improve predictive performance. Tsoumakas, Angelis, and Vlahavas [154] use statistical procedures to select the best subgroup among different classification algorithms and combine their decisions with simple aggregation methods such as Weighted Voting. More recently, heterogeneous ensemble were proposed for the streaming setting [135, 136] addressing classification tasks only. The methods were proposed for classification in the streaming setting but can be naturally be adapted for continuous valued targets in the streaming setting. Yet, their performance have to be proven. Cerqueira et al. [38] address the forecasting task and proposed the Arbitrated Dynamic Ensemble (ADE) based on meta-learning to adaptively combine models' predictions in the batch setting.

#### Stacking

Stacked generalization, more commonly referred to as *stacking* is the process of learning an ensemble of heterogeneous (but not necessarily so) models whose outputs will serve as features to a meta-model. Stacking model involves a two-levels learning architecture. The first level (base-models) learn on the data stream whereas the second level (meta-model) learns how to best combine the individual predictions. The original feature vector of the data stream can optionally be given to the meta-model Wolpert [164]. Stacking has become a major ensemble method to boost weak learners by naturally taking into account their errors correlations

### 2.2.3.3 Combining forecasting models for streaming data

As mentioned before, a large number of ensemble methods can be found in the literature for solving classification tasks on streams, but only a few exist when dealing with continuous outputs such as in regression tasks [68]. Recent surveys have indexed some streaming ensemble methods for the regression task that are tailored to solve the forecast combination problem [94]. Additive Expert ensemble (AddExp) was proposed for non-stationary data streams where a weight is associated to each base learner [92]. The weight of a base learner is always multiplied by  $\beta^{|\hat{y}-y|}$  and  $0 \leq \beta < 1$ , where  $\hat{y}$  is the prediction given by the base learner and  $y$  is the actual value of the target. AddExp includes a pruning stage to remove unnecessary models from the pool. When a new expert is added and the predefined maximum size of the pool is reached, the expert with the lowest weight is removed before adding the new one.

Adaptive Model Rules (AMRules) [53] is an ensemble method that combines a set of rule-based models to solve regression problem on high-speed data streams. The ensemble encourages diversity by randomizing both the set of instances and the set of attributes in order to reduce the error on regression tasks. Besides, a change detector is used Page-Hinkley (PH) [121] to monitor the evolution of the online error of each rule. When a significant

change is triggered, the correspondent rule is removed from the set of rules.

Ikonomovska, Gama, and Džeroski [85] proposed an Online Random Forest (ORF) and Online Bagging (OBAG) that use the FIMT-DD [84] as a base learner. Adaptive Random Forest for Regression [69] (ARF-REG) is an extension of the Adaptive Random Forest (ARF) that uses the FIMT-DD [84] as base model. ARF-REG is based on a warning and drift detection scheme per tree, such that after a warning has been detected for one of them, a background tree starts learning in parallel and replaces the old one if the warning escalates to a drift conversely to existing methods that simply reset the base model whenever a drift is detected. The method is not bounded to a specific drift detection algorithm.

The Streaming Random Patches [67] (SRP) trains each base learner on a different subset of the features and the instances from the original data, namely a random patch. This diversity enhancing strategy, is similar to the one used in random forests [69] by combining random subspaces and online bagging, yet it is not restricted to using decision trees as base learner. Similarly, Gomes et al. [71] have studied different ensemble techniques while taking into consideration issues related to the context of regression, such as combination and reset strategies to cope with concept drift. They have tested different implicit techniques to enhance diversity within homogeneous ensembles. It relies on manipulating the input training data used to train each base learner. Their experiments include bagging (BAG) [22], the Random Space (RS) and the Random Patches (RP) [104] since they are not bound to a specific algorithm for base learners. On top of that, they proposed different fixed and random techniques to continuously reset base models, and, thus, keep the ensemble up-to-date with the latest concepts. They concluded that this has a positive effect on the predictive performance. Experiments have demonstrated that simpler reset strategies such as periodically replacing members of the ensemble with new models trained on different windows can outperform more reactive strategies that are based on drift detector.

## 2.3 DYNAMIC ENSEMBLE SELECTION (DES) FOR TEMPORAL STREAMS

The rationale behind heterogeneous ensembles is that by having an ensemble composed of different learning base models, it is expected that these models exhibit different levels of predictive performance along the input data stream. As described in Section 2.2.2, we can dynamically combine the available models to manage the strengths and limitations of each one across the temporal data stream. All the ensemble methods previously discussed involve all the base models in the pool to compute the final prediction. However, since models show varying predictive performance at different times (point) of the data sequence, it would be more interesting to perform a selection stage on the ensemble before aggregating their predictions [29]. In fact, instead of combining the outputs of all the available models, we should select the model or subset of models that we trust the most to be accurate on the test sample at hand.

### 2.3.1 ON DYNAMIC ENSEMBLE SELECTION (DES)

The selection task can be achieved either by *Static* or *Dynamic* approaches [90]. Static methods select the same subset of individual models and remains constant for all upcoming test observations. The main limitation of static methods in time-dependent data streams is its

inability to capture the evolving dynamics of the temporal and cope with concept drift. It is more insightful to adopt dynamic selection in which the subset of selected models to be combined is decided for each incoming test instance and thus varies over time [38].

DES methods can further be divided into two sub-categories, namely *individual-based* and *group-based* [26]. In *individual-based* methods, the selection of a subset of models for each test instance is done by estimating the competence level of the base models individually without taking their dependencies into account. On the other hand, *group-based* methods tend to modeling models dependencies when selecting the subset of experts. A comprehensive and detailed survey on ensemble methods (also referred to as MCS) and ensemble selection was provided by Britto Jr, Sabourin, and Oliveira [26] where they detailed a taxonomy on the different methods of ensemble generation and selection discussed so far. The taxonomy is depicted in Figure 2.3.

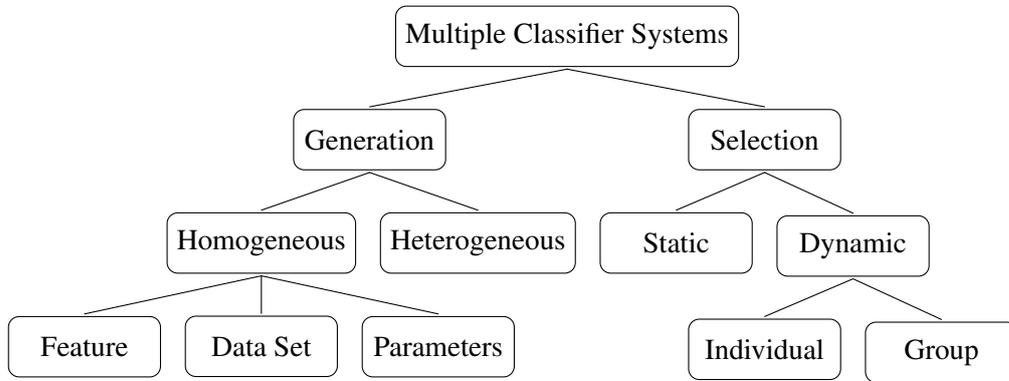


Figure 2.3: Multiple Classifier Systems (MCS) taxonomy [26]

We describe in Figure 2.4 the general workflow to perform Dynamic Ensemble Selection (DES) on data streams. For each incoming test instance  $x$ , DES selects the most appropriate subset of base models and combines their respective predictions to output the final prediction. To construct a Dynamic Ensemble Selection (DES) approach, we have to address the following questions:

- How to build the individual models following the ensemble generation methods detailed in Section 2.2.3 and what would be the best method to constitute the pool of base models to be as accurate and as diverse as possible
- How to evaluate the performance of each base model for a given instance  $x$  before proceeding to its selection ?
- How to select the set of base models to be involved in the aggregation stage for the final prediction once the competence of each base model is computed?

In DES, the "competence" of each base model in the ensemble is estimated in the vicinity of the test instance  $x$  and the most competent members are selected to predict. Kuncheva [98] grouped the competence estimate methods into two main approaches according to [65]:

- Decision-independent ("a priori"): the competence is determined based on the test instance  $x$ , prior to finding out the predictions of each base model.

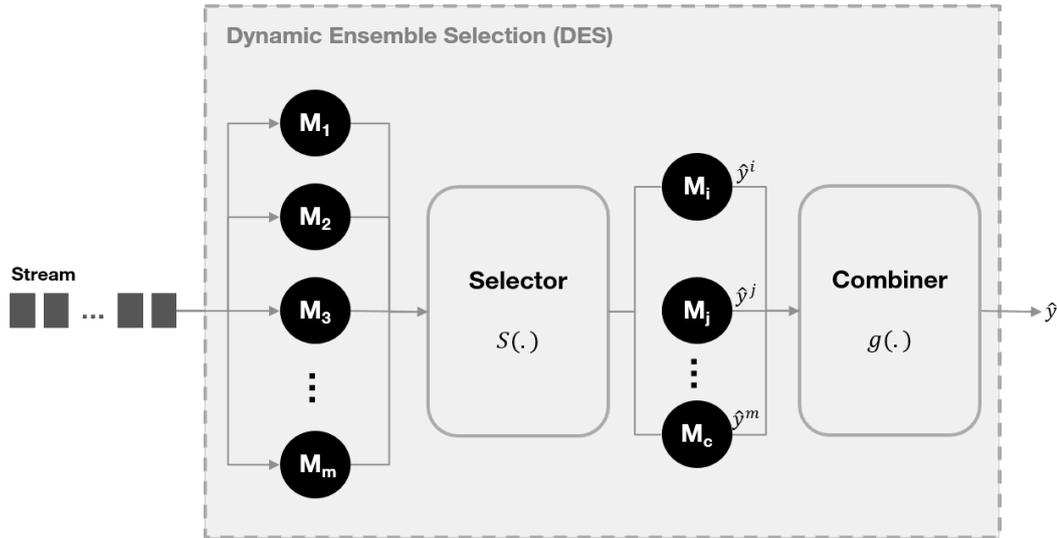


Figure 2.4: A generic workflow for Dynamic Ensemble Selection (DES) showing the selection and combination steps for each instance in the data stream.

- Decision-dependent ("a posteriori"): the competence of each base model is computed knowing the prediction of the latter.

In the remainder of the work, we will be focusing on DES methods to solve the forecasting task on streaming data. There exist a large number of DES methods for other tasks such as classification or other settings such as batch learning, but they are beyond the scope of the thesis.

### 2.3.2 DES FOR TEMPORAL DATA STREAMS

There have been several forecast aggregation methods proposed throughout the ensemble methods literature. Simply averaging all the available experts' predictions (`Simple`) in the pool has demonstrated a good level of predictive performance despite the simplicity of the combination approach [47]. Marcellino [107] proposed to use the median value of all the experts' predictions instead of their average as the ensemble's forecast in order to reduce the negative impact of extreme values.

However, as stated before, experts have varying levels of expertise along the temporal stream. Thus, it is more promising to adapt the weight assigned to each expert according to its expected performance, the higher the expertise, the higher the weight. Nonetheless, estimating models' performance and accordingly computing their weights for each test instance is not trivial. In fact, simple methods such as the simple average (`Simple`) provide competitive results and are sometimes hard to beat. This has led to a challenge named the "*forecasting combination puzzle*" [64].

We detail below state-of-the-art DES methods in time dependent data streams that can be grouped into three categories, namely: windowing, regret minimization and meta-learning methods. Several regret-minimization based strategies have been proposed for aggregating the predictions of forecasting models. Regret is the average error suffered with respect to the best we could have obtained (Oracle). The thesis focuses on windowing and meta-learning

based DES methods. We point the reader to the work of Cesa-Bianchi and Lugosi [42] for further reading on regret minimization based methods.

### 2.3.2.1 Windowing

Windowing DES methods are based on the assumption that the region of competence of base models is temporal. The performance of experts is determined on a window of recent data or using a forgetting mechanism that accentuates the importance of recent data. The rationale is that recent observations are more likely to be similar to the ones we intend to predict in the near future. Thus, good-performing experts on recent observations are more likely to be better on next instances. This is even more valid in the streaming setting where the stream is supposedly infinite and evolving (concept drift). Simple forecasts averages [47] can be improved by computing weighted average of all the available forecasts.

Jose and Winkler [86] point out the important fact that the simple mean aggregation method is sensitive to extreme values and suggest that excluding such values or reducing their extremity might be beneficial. Authors studied the performance of simple yet robust aggregation methods, that is trimmed means. Empirical results suggest that moderate symmetric trimming of 10~30% of the forecasts can provide improved combined predictions. The authors argue that trimmed means can be more robust than the mean, yet not as limited as the median, which ignores valuable information contained in the set of forecasts.

**AEC** (Adaptive Ensemble Combination) is a method for adaptively combining a set of forecasters [141]. It uses an exponential weighting strategy to combine forecasters according to their past performance, including a forgetting factor to give more importance to recent values AEC tends to select the best predictor (weights tend to 0 or 1). It is a time-varying version of (**A**ggregated **F**orecast **T**hrough **E**xponent **R**e-weighting) [175]. Weights depend only on the past forecasts and the past realizations of the temporal series. The weights are sequentially updated after each additional observation.

Moreira-Matias et al. [113] introduced a novel methodology to predict the number of services that will emerge at a given taxi stand in a short-term time horizon using streaming data. They propose three distinct short-term prediction models and a data stream ensemble framework to combine them. Each model is weighted according to its error on the data contained in the sliding time window  $[t - w, t]$  where  $w$  is a user defined parameter standing for the window size. They have used the Symmetric Mean Percentage Error (**sMAPE**) described in Section 2.1.7.2 to quantify models error.

**BLAST** (**B**est **L**AST) [135] was initially proposed for classification tasks based on Online Performance Estimation that selects the best performing base model on a window of recent data to be the only active model for the next instances where  $w$  is a user defined parameter. BLAST can easily be extended to the forecasting task of continuous values by adapting the loss function in the performance evaluation.

**DEMSC** (**D**rift-based **E**nsemble **M**ember **S**election using **C**lustering [140] is a two-stage DES framework that promotes both accuracy and diversity. DEMSC performs an informed selection of base learners for each test instance based on models' performance drift detection mechanism that excludes poorly performing base models and selects the top performing ones. The performance is computed over a sliding window. The clustering stage groups similar models based on their predictions and then selects clusters' representatives only to achieve a

highly accurate and diverse ensemble.

### 2.3.2.2 Meta-Learning

Brazdil et al. [20] describe that meta-learning is a way of modelling the learning process of one or several learning algorithms. The field of meta-learning studies how learning systems can become more effective through experience. The goal is not simply to find a good solution, but to do it with increasing efficiency through time. field of meta-learning addresses the question what machine learning algorithms work well on what data.

**Definition 2.3.1 — Meta Learning.** Meta-learning, or learning to learn, is the science of systematically observing how different machine learning approaches perform on a wide range of learning tasks, and then learning from this experience, or meta-data, to learn new tasks much faster than otherwise possible [158].

The algorithm selection problem AS defined by Rice [133] falls within the field of meta-learning. It comprises four main components. The problem space  $P$  that represents the data used in the study. The feature space,  $F$ , that stands for the range of measures that characterize the problem space  $P$ . The algorithm space,  $M$ , that consists in the list of candidate algorithms which can be used to find solutions to the problems in  $P$ . The performance metric,  $\mathcal{L}$ , is a measure of algorithm performance such as the Mean Squared Error (MSE). The goal is to select, for a given  $x \in P$ , the algorithm  $M^i \in M$  that maximizes (minimizes) the performance metric. This problem can be cast a classification, ranking or regression task.

Following the same problem formulation, Talagala, Hyndman, Athanasopoulos, et al. [150] have introduced the meta-learning based model selection for the time series forecasting task as described below.

#### Problem formulation

For a given instance  $x \in P$ , with features in  $F$ , find the selection mapping  $S$  into the algorithm space  $M$ , such that the selected algorithm  $M^j \in M$  minimizes forecast error metric  $\mathcal{L}(M^j)$ .

- $P$ : the problem space stands for each instance in the stream
- $F$ : the feature space that forms the meta-knowledge for selection and comprises data characteristics or meta-features that provide valuable information to differentiate the performance of a set of given learning algorithms [20, p. 6]. The idea is to gather descriptors about the data that correlate well with the performance of learning models.
- Algorithm space  $M$  is the pool of base models trained to forecast future values in the temporal stream.
- Loss function  $\mathcal{L}$  is the evaluation metric to differentiate between the predictive performance of base models in order to select a subset from the pool of models

Brazdil et al. [20] argue that ensemble methods and model selection and combination may be regarded as a form of meta-learning as it uses results at the base level to construct a classifier at the meta level. Consider the problem of DES where the goal is to select a subset of predictive models from a pool of models for a given test instance. The problem can be cast as a meta-learning problem and the aim is to identify the set of learning algorithms with

best expected performance for the test instance  $x_t$ . Besides, meta-Learning can be used to cope with stream specific challenges such as re-occurring concepts. Thus, one should involve remembering patterns from past data to be reused in future situations. If a recurring concept is detected, reusing a model previously trained on that concept can perform more efficient in terms of predictive performance and time [5, 59, 163].

Rijn et al. [134] have investigated the use of meta-learning for algorithm selection on data streams where meta-knowledge can improve the predictive performance of data stream algorithms. They consider an ensemble of algorithms and for each window of size  $w$ , an abstract meta-algorithm determines which algorithm will be used to predict the next window of instances. The decision is based on data characteristics computed in the previous window and the meta-knowledge. The Algorithm selection problem was treated as a classification task using meta-learning. Similarly, `MetaStream` [137] is a meta-learning based dynamic models selection and combination framework for regression task in time-evolving stream environment. `MetaStream` works by periodically choosing between single learning algorithms or their combination to be used on a window of incoming observations. It maps the characteristics extracted from the past and incoming data to the expected performance of regression models in the ensemble.

The Arbitrated Dynamic Ensemble (ADE) [38] is based on arbitration [117] to achieve DES for time-series one-step-ahead point forecasting. ADE comprises a pool of heterogeneous base forecasters trained off-line and a set of meta-learners where each meta-learner is trained to predict for each test instance the expected error of its base counterpart. The  $\alpha\%$  best base-model (with the lowest predicted errors) are then selected in a committee and their predictions weighted and combined using the softmax. Talagala, Hyndman, Athanasopoulos, et al. [150] introduce a general framework for forecast-model selection using meta-learning designed for traditional learning. They train a random forest to identify the best forecasting method based on time-series features such as seasonality, trend and auto-correlation.

Candela et al. [33] use meta-learning to automatically perform model selection. Based on the assumption that forecasting performance decays in time, they introduced a framework that constantly monitors and compares the performance of deployed forecasting models to guarantee accurate forecasts of travel products' prices. A meta-model is in charge of predicting the forecasting error (`sMAPE`) of each forecasting model in the pool and then ranked, where the model with the lowest error is ranked #1. The estimated forecasting error is also used to detect accuracy deterioration over time and send alert signals. In fact, if the estimated performance is too poor, this means that the model has become obsolete and consequently be replaced.

**FFORMA** (Feature-based **FOR**ecast **MO**del **A**veraging) [112] is an automated method for obtaining weighted forecast combinations using time-series features. First, a collection of time series is used to extract features that are then used to train a meta-model to assign weights to various possible forecasting methods while minimizing the average forecasting loss obtained from a weighted forecast combination. The meta-model predicts the weight of each forecast for new and unseen series. **FFORMA** outperforms a simple forecast combination, and all of the most popular individual methods in the time-series forecasting literature.

## 2.4 SUMMARY

Predicting future values of data streams is a challenging task due to the evolving nature of data subject to concept drift. Besides, data streams often exhibit a temporal dependency where past data impacts both current and future behavior. Data Stream (DS) mining methods rely on the vast research work achieved in the field of Time-Series (TS) analysis to perform forecasting. On the other hand, TS analysis takes advantage of DS related methods such as concept drift detection and adaptation in order to improve predictive performance. We revised several dynamic forecast combination and ensemble methods that have demonstrated highly competitive results in the streaming setting. More particularly, we focus on Dynamic Ensemble Selection (DES) that aims at selecting the most competent models only according to the data at hand in order to combine their predictions. We overviewed both windowing and meta-learning based DES methods that can meet the requirements of the streaming setting. In the next chapter, we will explore these topics in more detail and propose several novel methods for temporal data streams forecasting using DES approaches to tackle the main limitations of the current state-of-the-art approaches.



## Streaming Dynamic Ensembles Selection

In this chapter, we will overview several approaches to estimate models performance such as windowing and meta-learning that were applied in the context of evolving data streams. Besides, as mentioned in the previous chapter, diversity among experts is a key aspect in ensemble methods. However, the diversity-accuracy dilemma is a challenging issue, particularly in the streaming setting since all the models are trained and updated on the same data. Finally, predictions fusion approach describes how individual predictions of selected experts are aggregated into the single ensemble output. In the context of temporal stream forecasting, we will pay special attention to changes and drifts that may occur in the characteristics of the data and may alter overall predictive performance.

The remainder of the Chapter is organized as follows, Section 3.1 addresses the first task of DES methods that is the estimation of base-models performance. We discuss several windowing and meta-learning based methods tailored to the streaming setting. Section 3.2 details different selection methods that aim at enhancing both accuracy and diversity among selected experts. Section 3.3 illustrates an extensive experimental study to compare the proposed DES methods against state-of-the-art methods in terms of predictive performance as well as computational cost (time and memory). Finally, Section 3.4 concludes the chapter by discussing the results achieved by the proposed DES methods and presents potential research directions for future work.

Every Dynamic Ensemble Selection (DES) approach is based on three main steps: models' performance evaluation, committee selection and finally predictions aggregation. Let  $M = \{M^1, M^2, \dots, M^m\}$  be the ensemble of fixed size  $m$  and  $x_{t+1}$  be the test instance provided at time  $t$  and serves as a feature vector to predict  $\hat{y}_{t+1}$ . First, we need to estimate the predictive performance of each individual model  $M^i \in M$  according to a function  $P(\cdot)$ . Second, we select a subset of the most competent base-models according to  $x_{t+1}$  in a committee  $M^c$  following some selection approach  $S(\cdot)$ . Finally, the predictions of the selected experts are combined using some predefined fusion function  $g(\cdot)$  which is often a weighted average of experts' individual predictions. Figure 3.1 details a general framework for DES on data streams. For each incoming test instance, the DES framework, estimates base-models' predictive performance, selects experts and combines their respective outputs. When the true value of the target is released, all base-models are updated and performance evaluation refreshed.

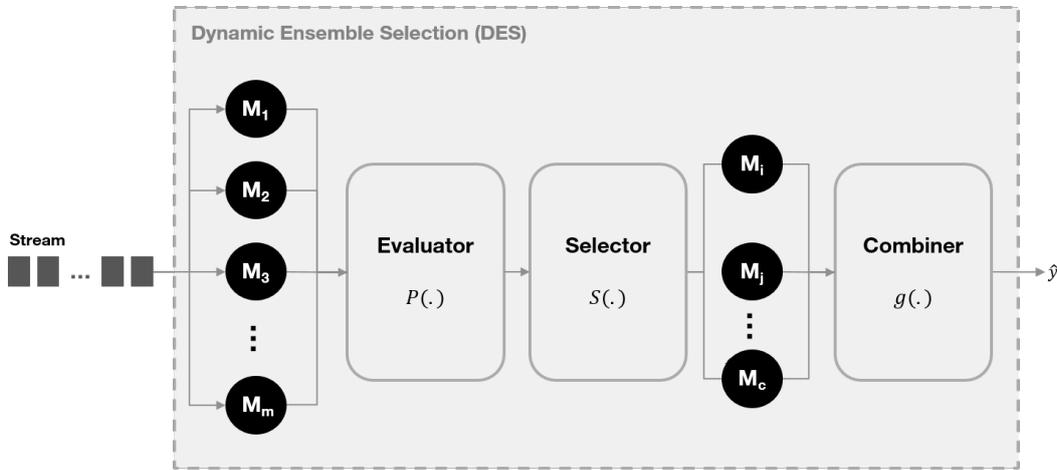


Figure 3.1: General Dynamic Ensemble Selection pipeline for data streams

### 3.1 INDIVIDUAL MODELS' COMPETENCE ESTIMATION FOR DES

Dynamic environments such as in temporal streaming data are subject to changes and concept drift. A common strategy to deal with these changes is to continuously monitor the predictive performance of the learning models and select, on the fly, the most suitable model or subset of models for the current data at hand. In a heterogeneous ensemble, where the pool comprises different learning algorithms, it is expected that these models exhibit different levels of predictive performance along the temporal stream. The goal of Dynamic Ensemble Selection (DES) is to dynamically select the most effective individual models only and combine their predictions to compute the final ensemble prediction.

Hence, estimating the performance of base-models is of crucial importance in any DES approach. In fact, model competence defines the extent of the reliability of the model itself to perform well on a given unseen test instance. We discussed in Section 2.3.2 two main approaches to dynamically estimate base-models' performance, namely *windowing* and *meta-learning*. Nevertheless, estimating the competence of each base model  $M^i$  for a future test sample  $x_{t+1}$  is a challenging task in the field of DES. Besides, data streams are very likely

to evolve over time, thus it is very important to capture the underlying dynamics and cope with concept drift by discarding old and outdated data. We discuss here several competence evaluation methods tailored to streaming data following both windowing and meta-learning based approaches.

### 3.1.1 WINDOWING-BASED DES

Windowing DES methods are based on the assumption that the region of competence of base-models is solely temporal. The performance of experts is determined on a window of recent data or by using a forgetting mechanism that accentuates the importance of the most recent instances. The rationale is that recent observations are more likely to be similar to the ones we intend to predict in the near future. Thus, good-performing experts on recent observations are more likely to be better on the upcoming instances. This is even more valid in the streaming setting where the stream is supposedly infinite and evolving due to concept drift. Let  $\hat{e}_t^i$  be the estimated error that reflects the predictive performance of base model  $M^i$  at time  $t$  using some loss function  $l$ , the lower the error the better the base-model. In our context,  $l$  can be any loss measure described in Section 2.1.7.2.

#### Fixed-size sliding window

One way to estimate the predictive performance of each base model is using a sliding windows ( $SW$ ) such that only the last  $w$  samples are taken into account while computing the loss function  $l$ . As time goes on, the sliding window moves over the stream while keeping the same size  $w$ . Let  $y_w$  and  $\hat{y}_w^i$  be the true values and the predicted values of model  $M^i$  respectively in the recent window  $w$ , then the error  $\hat{e}_t^i$  of base-model  $M^i$  at time  $t$  is:

$$\hat{e}_t^i(w) = l(y_w, \hat{y}_w^i) \quad (3.1)$$

Older samples are drastically excluded as new samples are released in the stream. Consequently, the computed error measurements can considerably vary between two consecutive windows, which translates in faster transitions over the temporal stream and thus faster adaptation yet very sensitive to outliers. However, choosing the right size of the window is not trivial. In practice, varying the value of the window size  $w$  follows the stability-plasticity dilemma [35]. In fact, small values of  $w$  lead to faster reactivity, but makes the model sensitive to outliers and noise. On the other hand, higher values (larger windows) lead to greater stability at the cost of some responsiveness and possibly including outdated data.

#### Fading Factor

An alternative to sliding windows are fading factors ( $FF$ ) [60], which are faster and memory-less. This method decreases the influence of older samples on the performance measurement as the stream progresses. Computing the loss  $\hat{e}_t^i$  for  $t \geq 1$  using a fading factor is estimated by:

$$\begin{aligned} \hat{e}_t^i &= s_t^i / B_t \\ S_t^i &= l(y_t, \hat{y}_t^i) + \lambda \times S_{t-1}^i \\ B_t^i &= 1 + \lambda \times B_{t-1} \end{aligned}$$

Where  $\lambda \in \mathbb{R} : 0 \ll \lambda \leq 1$  is the forgetting factor that controls the speed at which older data are discarded.  $S_t^i$  stands for the faded sum of loss incurred by model  $M^i$  along the stream whereas  $B_t$  represents the faded total number of instances seen so far. The initial values of  $S_0^i$  and  $B_0$  are set to zero at time  $t = 0$ .

In the fading factor variant, each data object is assigned a different weight according to its arrival time so that new observations receive higher weights than old ones. It allows to smoothly reduce the effect (importance) of old and possibly outdated instances on performance computation. A decreasing exponential function is usually used in the fading model such as in AEC [141] to dynamically weight forecasters and aggregate their predictions.

### 3.1.1.1 Feature and temporal based performance region

We present SLOPE for Sliding Local Performance that adapts the concept of local performance to the streaming setting. The idea of "Local Accuracy" was used for classification on batch learning tasks where the goal is to estimate each individual classifier's accuracy in local regions of the feature space surrounding a test instance [166]. The decision of the most locally accurate classifier is then used as the ensemble's prediction. It was later extended to DES called KNORA (K-Nearest-ORAcles) [90] where the most suitable subset of classifiers is selected instead of a single one. For each incoming test instance, KNORA finds its  $k$  nearest neighbors in a predefined validation set, and selects the classifiers that correctly classify those neighbors to aggregate their predictions.

The proposed SLOPE approach is based on a dual-locality assumption for temporal data streams that considers the recency of the data using a sliding window and the feature space using the  $k$  nearest neighbors. Similarly to fixed size sliding window, SLOPE considers the  $w$  latest observed instances in the stream up to time  $t$ . However, instead of evaluating base models on all the instances comprised in the sliding window, we evaluate the base models on a smaller "local region" surrounding the given test instance  $x_{t+1}$  called the region of competence defined by the  $k$  nearest neighbors. In fact, the proposed approach is twofold:

- The sliding window emphasizes on the recency of the data and discards outdated instances;
- The feature-based local performance region defines a small region in the feature space using the  $k$  nearest neighbors surrounding a given test instance  $x_{t+1}$  among all instances comprised in the sliding window.

The idea is to measure the performance on the most recent and the most similar instances only. Besides, each sample belonging to the region of competence can be weighted by its distance to the test instance. Thus, instances that are closer to the test sample have a higher influence when computing the performance [145]. SLOPE is particularly well-suited as we are using the prequential evaluation approach for streaming data. In fact individual models' performance are continuously updated using the newly released true values and assume no delay. However, the forgetting method is drastic since the oldest examples in the window are completely excluded as soon as a new instances are released. Naturally, the performance of SLOPE is closely related to the choice of the parameters values  $w$  as described in fixed-size sliding window (paragraph 3.1.1) as well as the number of nearest neighbors  $k$ .

### 3.1.2 META-LEARNING BASED DES

Meta-learning is one of the most promising approaches employed for dynamic ensemble selection [20]. It focuses on understanding the behavior of learning algorithms by exploiting the knowledge acquired from previous experience on similar data in order to predict the behavior of the algorithms on future unseen instances. As discussed in Section 2.3.2.2, the DES problem can be cast as a meta-learning task where the goal is to predict the performance of each base model in the pool and accordingly select and weight their predictions. Every meta-learning approach is based on two main components: *meta-features* and the *meta-model*. Meta-features represent a set of characteristics extracted from the data and used to train the meta model [20]. The meta-model investigates the relation between data characteristics and the performance of learning algorithms. Figure 3.2 illustrates a general pipeline to achieve meta-learning based DES on data streams. We can distinguish two separate layers of learning. The first one involves the pool of base models  $M$  that learn on the data released in the stream to predict future observations. The second layer, referred to as meta-layer, extracts meta-features from the data stream and trains a meta-model  $Z$  to predict future behavior of the base-models.

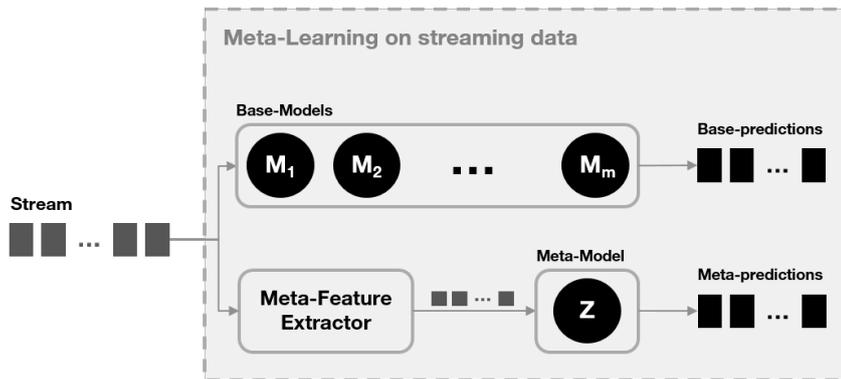


Figure 3.2: Meta-learning pipeline for streaming data

The performance of meta-learning approaches is closely related to the quality and usefulness of the meta-features. In fact, for the ensemble selection task, important characteristics are the ones that represent properties of the data that influence model predictive performance.

In the next section, we discuss a set of meta-features that are deemed to be the best descriptors of the data characteristics that relate to models predictive performance and are tailored to the streaming setting.

#### 3.1.2.1 Meta-features for data stream characterization

The first challenge to be addressed in any meta-learning based DES approaches, is the definition of a set of *meta-features*. In traditional batch learning, the goal of meta-learning is to determine a set of meta-features that can be common to several data sets that have different morphologies (different attributes, number of instances . . .). The extracted characteristics are then used to train a meta-model to link to the performance of different learning algorithms from one data set to another. However, the goal of meta-learning in DES for data streams is to designate the best model or subset of models for each test instance. The stream is described by the same set of attributes (same morphology) while the process generating the

data may evolve over time. We discuss here a set of meta-features used in the experiments to implement meta-learning based DES approaches.

### Meta-features for temporal data stream

Data stream characterization approaches can be divided into three categories: general (simple), landmarks, and model-based meta-features. Model-based meta-features exploit properties of some induced model to build a set of meta-features. The properties are related to the morphology of the model itself such as the number of leaf nodes or the maximum depth in a decision tree [9, 122]. In our work, we will focus on general and landmarks meta-features only assumed to be the most appropriate ones for temporal data stream. Besides, we describe a set of stream-specific information that can be integrated in the set of meta-features.

- **General (Simple):** Simple meta-features include statistical and information-theoretic characteristics estimated from the data stream. Meta-feature related to the morphology of the data (number of instances, features, ...) are not addressed since we assume that the stream at hand has the same morphology despite the evolving process generating the instances. In fact, in data streams, the set of attributes typically does not change over time while the number of instances is supposedly infinite. In this case, meta-learning approaches are usually used for ensemble selection at different time points of the same stream, rather than many different data sets. Therefore, the characteristics extracted from the data stream (each attribute or its relations) at a time point can be used directly as meta-features. Besides, due to the phenomenon of concept drift, meta-features extraction must be a regular process [137].

In temporal data streams, the observations are not i.i.d and, hence, temporal dependence between successive examples is very important. Therefore, measures that consider this temporal dependence can be useful for meta-learning based DES methods. Particularly, measures such as serial correlation that are often investigated for feature extraction of time-series data. We recall that, the original feature vector  $x_{t+1}$  includes the previous  $p$  values observed up to time  $t$  (embedding vector). We compute the following statistics for each embedding vector in order to characterize the recent dynamics of the stream: mean, standard deviation, skewness, kurtosis, recent trend and serial correlation. Following the definition in [41], recent trend is defined as the ratio between the standard deviation of the embedding vector and the standard deviation of the differenced embedding vector. The serial correlation is estimated using the Ljung-Box test statistic.

The confidence that the extracted characteristics correctly represent the data is related to the number of examples used such that the larger the number of examples, the higher the reliability.

- **Landmarkers:** Landmarking is a strategy to describe tasks through the performance of simple and efficient learners. Landmarking based meta-features exploit information obtained from the performance of one or several simple and fast learners that exhibit significant differences in their learning mechanism [123]. We use the predictive performance of two landmarks to characterize the data stream namely, KNN and

Hoeffding Tree (RHT) regression models and compute their respective loss for each instance along the stream.

- **Stream-specific:** Rijn et al.[134] investigated the algorithm selection problem on data streams using meta-learning. They introduced a set of stream-specific meta-features, based on a change detector. They run both Hoeffding Trees and Naive Bayes algorithms with both the ADWIN [12] and DDM [61] change detectors over a set of data streams. They recorded the number of warnings and changes triggered and include this information to train a meta-model. Following this approach, we propose to use a warning and a drift detector based on ADWIN to trigger changes in the performance of the two landmarks discussed above. In fact, we use two ADWIN detectors where the first one is more indulgent and detects warnings whereas the second one is more rigorous and detects drifts. Warning and drift flags for each landmark are then included in the set of meta-features along with statistical information on the data and landmarks performance.

Moreover, in temporally dependent data streams, past values hold valuable information on current and future values as described in the auto-regressive model (Section 2.1.6.1). The behavior, and consequently the errors incurred by base-models, are very likely to exhibit temporal dependency as well. We propose to include a set of lagged errors for each base-model in order to enhance the meta-features set. We summarize in Table 3.1 all the meta-features and their respective description. These meta-features are used along with the original set of features  $x_t$ , namely the auto-regressive vector, and referred to with the tag ORI.

Table 3.1: Meta-Features grouped by category and their respective description

Category	Description
General (STA)	Mean, standard deviation, skewness, kurtosis, recent trend and serial correlation.
Landmark (LAN)	sMAPE loss of KNN and RHT as landmarks
Stream specific	Warning (resp. Drift) detection feature is set to True if a warning (resp. drift) was detected in the performance of the KNN and RHT landmarks. These features are used along with the landmarking features (LAN) described above
Lagged errors (LAG)	A number $q$ of the past base-errors incurred by each base-model is included in the set of meta-features

Now that we have determined the set of meta-features that best describe the data stream at hand with different characteristics, we determine the best approach for the meta-model to link between the predictive performance of individual models in the pool and the set of meta-features that are continuously extracted from the data stream.

### 3.1.2.2 Meta learner

The role of the meta-layer is to determine at time  $t$  which models are good given a test instance  $x_{t+1}$  by predicting their respective predictive performance (loss) based on a set of meta-features. The goal is to learn the relationship between a stream's meta-features and candidate models' performance. For example, Guerra, Prudêncio, and Ludermir [74] used an SVM meta-regressor per classification algorithm to predict its accuracy. In our case, we use the meta-model to predict and quantify the loss (using any of the evaluation metrics detailed in Section 2.1.7.2) that each individual model  $M^i \in M$  is likely to incur with regards to the test instance at hand  $x_{t+1}$ . Once the performance is estimated, we can then proceed to the selection stage where only the most competent ones are included in the committee of experts  $M^c \subset M$  whose predictions are aggregated to output the one-step-ahead forecast  $\hat{y}_{t+1}$ .

#### Arbitration

Ortega, Koppel, and Argamon[117] introduced the arbitrated architecture that assigns to each base model  $M^i \in M$ , a meta-model  $Z^i$ , also called referee or arbiter, that is in charge of learning the behavior and predictive performance of its base-counterpart. We propose to use **STreaming Arbitrated Dynamic Ensemble (STADE)**, that is a streaming arbitration meta-learning based DES approach. The work follows the ADE [38] framework that uses arbitration for time series forecasting using dynamic ensemble selection. The system uses a two-layered learning schema where each layer trains its own models and receives its own data [58]. The base-learner  $M^i$  learns to predict the future values of the target  $y_{t+1}$  of the stream, whereas the meta-model  $Z^i$  learns the behavior of its base counterpart  $M^i$  and predicts its future errors  $\hat{e}_{t+1}^i$  using a set of meta-features. The rationale is that each meta-model holds meta-information on the area of expertise of its base-counterpart, and thus can infer its predictive performance given a test instance  $x_{t+1}$ . The meta-model (arbiter), is often a decision tree but not constrained to and can be any regression model. The features used in training the meta-models consist of the original attributes that define the base-level dataset, namely the vector of features  $x_{t+1}$ , augmented by other computed meta-features discussed in Section 3.1.2.1.

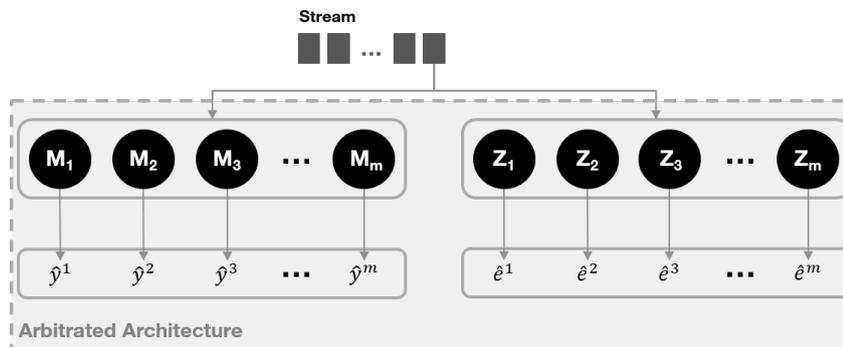


Figure 3.3: Streaming arbitrated meta learning architecture

Figure 3.3 highlights the general approach for arbitration on data streams. Conversely to windowing approaches that track the error on past instances, the meta-learning arbitrated approach is more proactive as it is based on predicting future loss of base-models. This

can lead to faster adaptation and better anticipation to changes in the case of evolving data streams where concept drift is very likely to happen.

"Arbitration" is often related to Stacking [164] that can also be considered as a meta-learning approach. Stacking runs the pool of base-models on the same data as well and builds a new meta-dataset, using the predictions of all base-models. The meta-dataset is then used to train a meta-model to link the predictions of base-models to the true value of the target. On the other hand, arbitration separately learns the individual expertise of each forecasting base-model and accordingly selects the ones that are expected to be the best. META-STREAM [137] is another example of meta-learning based DES where a set of meta-features are periodically extracted to train a meta classifier to predict the base-model that is expected to perform best, otherwise combines all the available base-models.

In this section, we discussed windowing and meta-learning based approaches to estimate the predictive performance of base models in the pool. The goal is to quantify the extent at which each base model will perform well on a given test instance. The estimated performance serves as a basis for the expert's committee selection that will be discussed next.

### 3.2 EXPERTS SELECTION: SELECT OR NOT SELECT, THAT IS THE QUESTION

Once the performance of each individual model in the pool has been estimated, it is important to proceed to the selection stage in order to prune the models that are likely to fail in predicting the upcoming instance (i.e incur a high loss). We recall that up to now, we have the vector of estimated loss of each base-model  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$  based on the information seen up to time  $t$  and the set of individual predictions  $\vec{y}_{t+1} = \langle \hat{y}_{t+1}^1, \hat{y}_{t+1}^2, \dots, \hat{y}_{t+1}^m \rangle$ . The goal is to select a committee  $M^c \subset M$  of base-models that are expected to perform best in order to combine their predictions.

In this section, we investigate several selection methods in order to pick the best committee of experts  $M^c \subset M$ . The number of base-models included in  $M^c$  can either be fixed or variable from one instance to another according to each approach.

#### 3.2.1 TRIMMING (TRIM)

A plethora of DES methods are based on trimming, where a fixed ratio  $\alpha$  of the best performing base-models are included in the committee  $M^c$ . Trimmed means [86] suggest that moderate trimming of 10 to 30% of the forecasts can provide improved combined predictions. ADE [38] uses a meta-learning approach that selects the 50% best base models. Authors have conducted an extensive study of the impact of varying selection ratio values on the overall predictive performance and have concluded that the best performing values are the ones in the middle of the searched distribution. In principle, the parameter depends to a great extent on the number of experts and their predictive performance. Finally BLAST [135] can be considered as a trimming-based approach as it selects the single best base-model over past instances.

#### 3.2.2 ABSTAINING (ABS)

Trimming-based DES approaches select a fixed-size committee of experts containing the  $\alpha\%$  best base-models regardless of the actual value of the performance score. However, following

this "blind" approach, the committee may include base models that are poorly performing despite their "good" rank. In fact, this may happen if all base models are expected to perform poorly. Hence, there is no clear evidence that better ranked base-models are going to perform better. Besides, considering the scenario where all base-models are expected to achieve very good predictive performance, selecting the  $\alpha\%$  best ranked ones only may exclude valuable information from other non-selected experts despite their high predictive performance.

We present an abstaining-based DES that allows base-models to abstain from contributing to the committee of experts  $M^c$  if their respective predictive performance is low. In fact, the overall loss achieved by the ensemble can be significantly improved by allowing base models to abstain in case of low confidence. The abstaining strategy is based on the actual values of the performance that reflects the confidence of the base-models. If a base-model's confidence does not meet a certain requirement, it is allowed to abstain. Loeffel et al. [102] have considered a similar problem in the regression task on evolving data streams that are subject to concept drifts. They addressed the issue on the precision of each prediction made from a single model using a predefined set of constraints. They assume that if the costs associated with a "good" and "bad" prediction are known, the overall prediction cost can be improved by allowing the regressor to abstain. The abstaining approach uses a single or a set of several Reliability Estimator Score (*RES*) and a competence threshold  $\theta$  to estimate the extent at which the model meets the required competence constraints.

The abstaining approach requires a continuous monitoring of the set of *RES* for each  $M^i \in M$ . We explore several reliability estimators used to assess the confidence of base-models for a given test instance.

### 3.2.2.1 Reliability estimation score

Abstaining relies on a set of reliability estimators to determine whether the constraints for the required precision can be met by a given model  $M^i \in M$  for a given test instance  $x_{t+1}$ . When the reliability requirements are not met, the forecaster is allowed to abstain in order to avoid misleading the committee of experts with a highly erroneous predictions. A reliability estimator is used to refer to the prediction error that will be incurred on instance  $x_{t+1}$ . We can attach a different threshold value  $\theta$  for each reliability estimator in *RES*. At time  $t$ , when test instance  $x_{t+1}$  is released in the stream, a set of  $r$  reliability estimators  $RES_t^i = \{R_t^1, R_t^2, \dots, R_t^r\} \in ([0, 1])^r$  is associated with each base model  $M^i \in M$  and computed based on the information observed up to time  $t$ . Small values of  $R_t^i$  indicate that the reliability estimator is confident that the forecaster  $M^i$  will output an accurate prediction  $\hat{y}_{t+1}^i$  according to the criterion. Inversely, a large value of  $R_t^i$  indicates a lack of confidence on  $M^i$ 's predictive performance, consequently, it is allowed to abstain (excluded from  $M^c$ ). A threshold  $\theta^i$  is set for each reliability estimator, and if  $R_t^i \leq \theta^i$ , the base-model is assumed to be reliable. The final decision about  $M^i$  abstaining or no, is computed by aggregating the decision of each  $R_t^j \in RES_t^i, j \in \{1, \dots, r\}$ . The aggregation strategy is the majority vote among all *RES*<sup>*i*</sup>. This means that if more than half of the reliability scores assume that the model  $M^i$  should abstain, then it is excluded from the committee  $M^c$ .

We detail here the set of reliability scores investigated for abstaining and are not bound to any forecasting algorithm. Nonetheless, reliability estimators must fulfill a set of computational requirements to cope with the constraints imposed in the streaming setting [102].

The requirements are basically the ones mentioned for incremental and adaptive learning algorithms discussed in Section 2.1.3. Therefore, each *RES* must i) operate on-line as new instances are released, ii) use limited memory, iii) process in low time, iv) cope with non-stationary distribution. Previous studies demonstrated that the best reliability estimators depend on both the model and the dataset [25]. Hence, it is commonly assumed that using an ensemble of different estimators to assess the reliability of individual models has a considerable advantage. In the same way as for ensembles of predictive models, relying on several reliability estimators reduces the risk of making the wrong decision about abstaining. This is particularly interesting in the data stream setting where both the data characteristics and the models' performance can evolve over time due to concept drift.

Toplak et al. [153] provided a taxonomy to categorize reliability estimators based on the information used for computation and can be classified into three main categories:

- Feature range-based: use feature values to place higher confidence in predictions of examples whose features values fall within the range of values encountered during the training phase.
- Nearest neighbor-based: use the distance to the most similar examples in the training set to infer reliability scores from the real error incurred in the neighbors.
- Sensitivity-based: use sensitivity analysis that samples or perturbs the composition of the training set to estimate a distribution of predictions error.

In our work, we focus on the nearest-neighbors based reliability estimators only as they are best suited for the streaming setting. In fact, sensitivity-based *RES* often require different copies of the model at time  $t$  and make several rounds to perturb the data in order to estimate the confidence. This is particularly fastidious for ensemble methods comprising several base-models as the complexity increases.

The estimated error  $\hat{e}_{t+1}^i$  of each base-models  $M^i$  computed using windowing or meta-learning approaches (Section 3.1) can be used as the first intuitive reliability estimator. The lower the error the higher the confidence placed in  $M^i$  to make an accurate one-step-ahead forecast  $\hat{y}_{t+1}^i$ . We discuss below other reliability estimators to be included in the abstaining decision.

### Nearest neighbors confidence

Briesemeister, Rahnenführer, and Kohlbacher [25] introduced the `CONFINE` (CONFidence estimation based on the Neighbors' Errors) measure for traditional batch learning. It measures the error in the local environment of the test instance  $x_{t+1}$  using the training dataset. `CONFINE` simply analyzes how good the model is in the neighborhood of the test instance. If the observed error of the nearest neighbors is high, the model is not expected to be very good on novel instances falling in this neighborhood either. If  $x_{t+1}$  lies within a densely populated subspace, `CONFINE` is able to interpolate the error based on very similar instances. On the other hand, if  $x_{t+1}$  lies within a sparsely populated subspace, the errors of instances within this subspace are likely to be high.

We propose `S-CONFINE`, for **Streaming-CONFINE**, that is a reliability estimator tailored for the streaming setting and emphasizes on the recency of the data in addition to

the test instance neighborhood as described for SLOPE in Section 3.1.1.1. Given a sliding window  $W_t$ , let  $N = \{(x, y)_1, \dots, (x, y)_k\}$  be the set of the  $k$  nearest neighbors for test instance  $x_{t+1}$  computed over  $W_t$ . The S-CONFINE of model  $M^i$  is defined in Equation 3.2 below as:

$$\text{S-CONFINE}_w(x_{t+1}, i) = \frac{1}{k} \sum_{j=1}^k e_j^i \quad (3.2)$$

where  $k$  is the number of neighbors and  $e_j^i$  is the actual error incurred by model  $M^i$  on neighbor instance at position  $j$ .

### CNeighbors-K

Similarly, Bosnić and Kononenko [17] proposed CNK as another approach to local estimation of prediction reliability score using the nearest neighbors' true target values. The value of the reliability estimator  $CNK^i$  for model  $M^i$  is the difference between the average target values of the  $k$  nearest neighbors and the prediction of the model  $M^i$  on example  $x_{t+1}$ .

$$\text{CNK}(x_{t+1}, i) = \frac{1}{k} \sum_{j=1}^k y_j - \hat{y}_{t+1}^i \quad (3.3)$$

where  $k$  stands for the number of neighbors,  $y_j$  denotes the  $j$ -th neighbors' true target value and  $\hat{y}_{t+1}^i$  denotes the example's prediction of model  $M^i$ . Similarly to S-CONFINE, we adapt CNK's computation to the streaming setting by using a sliding window of the  $w$  most recent labeled instances to query the  $k$  nearest neighbors. The assumption is that the prediction of the model should be as close as possible to the average value of the  $k$  nearest neighbors since instances laying in a close neighborhood are expected to share similar target values.

### 3.2.2.2 Abstaining with evolving data streams

Due to the evolving nature of data streams, one can experience a considerable loss in the predictive performance of forecasting models. The deterioration in the overall performance is related to the fact that poorly performing base models were included in the committee of experts  $M^c$ . This happens when the required competence threshold  $\theta$  was too wide and allowed base models with low confidence to be included. Conversely, if the threshold value  $\theta$  is too restrictive, highly performing base-models might be excluded from the committee and consequently diminish the overall predictive performance.

Abstaining was used in the context of online classifier ensembles where an adaptive abstaining strategy was proposed to deal with drifting and noisy data streams [93]. To cope with changes that are likely to occur in the stream and alter the confidence of base-models, authors have introduced a flexible and self-adaptive threshold value  $\theta_t$ . In fact, concept drift can alter the reliability of models' predictions over time and consequently affect overall predictive performance. Moreover, the forecasting models are expected to experience considerable loss and relatively different levels of confidence due to the evolving nature of data streams and concept drift. Therefore, using a static threshold will lead to poor predictive performance on drifting data. In this section, we propose and investigate two different update strategies to achieve adaptive abstaining based on competence threshold  $\theta_t$ .

**Adjustment update**

Following [93], the value of the threshold  $\theta_t$  at time  $t$  is modified based on the feedback on the error incurred by the ensemble prediction. If the committee of selected base-models was able to make an accurate prediction, this means that we have selected competent base-models and we can afford easing the reliability constraints to look for similarly performing base-models and include them in the committee for upcoming instances. On the other hand, a highly incorrect prediction can translate into a concept drift. The threshold  $\theta_t$  must be tightened in order to exclude poorly performing base-models and keep the most competent ones only. Besides, base-models have different abilities and speed for change adaptation, hence, it is important to keep the most competent ones only in case of concept drift.

Algorithm 1 details the adaptive abstaining ensemble based on adjustment factor. The adjustment factor is a user defined parameter  $s \in [0, 1]$  to update the threshold  $\theta_t$ . We explore two update strategies: **iterative** where  $\theta \leftarrow \pm s$  and **multiplicative** where  $\theta \leftarrow \theta(1 \pm s)$ . The  $MajorityVote(RES_t^i, \theta_t)$  outputs the majority vote on abstaining (or not) regarding base-

**Algorithm 1:** Adaptive abstaining using adjustment

---

**Input:**  $S = \{y_1, y_2, \dots, y_t \dots\}$ ,  $M = \{M^1, M^2, \dots, M^m\}$ ,  
 $RES = \{RES^1, RES^2, \dots, RES^m\}$ ,  $\vec{y}_{t+1} = \langle \hat{y}_{t+1}^1, \hat{y}_{t+1}^2, \dots, \hat{y}_{t+1}^m \rangle$ ,  
 $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$ , Initial competence threshold:  $\theta$ , Adjustment factor  $s$

**Output:**  $\hat{y}_{t+1}$  one-step-ahead forecast

```

1  $\theta_0 \leftarrow \theta$ 
2 while  $EndOfStream = False$  do
3   Get new test instance  $x_{t+1}$ 
4    $M^c \leftarrow \emptyset$  // Committee of experts
5   foreach  $M^i \in M$  do
6      $abstain \leftarrow MajorityVote(RES_t^i, \theta_t)$ 
7     if  $abstain = False$  then
8        $M^c \leftarrow M^c \cup \{M^i\}$ 
9     end
10  end
11   $\hat{y}_{t+1} \leftarrow Aggregate(M^c)$  // Aggregate expert's predictions
12  Get true label  $y_{t+1}$ 
13   $e_{t+1} \leftarrow error(y_{t+1}, \hat{y}_{t+1})$ 
14  if  $e_{t+1} \leq \theta$  then
15     $Increase(\theta_t, s)$  // Less restrictive for next step
16  end
17  else
18     $Decrease(\theta_t, s)$  // More restrictive for next step
19  end
20   $Update(M, x_{t+1}, y_{t+1})$ 
21   $Update(RES_t, x_{t+1}, y_{t+1})$ 
22 end

```

---

model  $M^i$  according to the value of the threshold  $\theta_t$ . We use the same threshold value  $\theta \in ]0, 1]$  for all the reliability estimators in  $RES$ . A base-model  $M^i$  is considered as an expert to predict  $y_{t+1}$ , and therefore be included in the committee  $M^c$ , if and only if the majority voting among all its associated reliability estimates in  $RES_t^i$  agree according to the current value of the threshold  $\theta_t$ . When the training instance is released, if the committee's incurred error  $e_{t+1}$  meets the initial competence threshold  $\theta$ , we assume that the committee has made a "good" prediction, if not, the prediction is inaccurate. The *Increase(.)* (resp. *Decrease(.)*) function are used to update the current value of the competence threshold  $\theta_t$  that will be used for next step. The *error(.)* function computes the  $sMAPE$  of the aggregated expert's predictions. Finally  $(x_{t+1}, y_{t+1})$  are used to update all the base models in  $M$  as well as their respective reliability scores  $RES$ .

#### Drift-based update

The second approach for self-adaptive competence threshold  $\theta_t$  is based on warning and drift detection methods to trigger drops in the overall predictive performance. If a warning is triggered, this translates to a deterioration in the overall predictive performance of the selected committee. Consequently, we must adjust  $\theta_t$  to be more restrictive and downward its value in order to force the worst performing base-models to abstain. On the other hand, if no warning was triggered, we can afford some room and increase the value of  $\theta_t$  in order to encompass base-models that abstained on the previous instances but whose reliability estimators lay in a very close proximity to the threshold value  $\theta_t$ . If a drift is detected, this means that there was a significant loss in the overall predictive performance and that the current value of  $\theta_t$  has completely deviated from the original tenet of selecting expert base models' only. Consequently, the original setting is reestablished. The drift-based adaptive abstaining approach is depicted in Algorithm 2.

### 3.2.3 RANDOMIZED SELECTION

The performance of both trimming and abstaining based DES methods are closely related to the choice of their respective parameters, namely the selection ratio  $\alpha$  and the competence threshold  $\theta$ . It is not trivial to set the value of these parameters without any prior human expert knowledge. Many dynamic ensemble methods are based on some random process in order to promote diversity among base-models such as bagging. Following this idea, we investigate randomization of the expert selection process without losing sight of the original objective that is to select the most accurate base models. The rationale is that the higher the error, the less likely is the base model to be selected among experts. However, concept drift may happen anywhere in the stream [173]. Thus, it might be interesting, from time to time, to select other base-models with higher predicted errors and inversely, prune the ones with low predicted errors. Besides, randomizing the selection process leaves more room to inject diversity among committee members. We investigate different scenarios where the parameters of the random selection are deduced from the predicted error of each base-model.

#### Binary randomized

We model the selection of each  $M^i \in M$  as a separate *Bernoulli* trial with a variable parameter  $p_{t+1}^i = 1 - \hat{e}_{t+1}^i$ . This means that a base-model is selected with probability  $p_{t+1}^i$  and not selected with a probability  $q_{t+1}^i = 1 - p_{t+1}^i$ . The smaller the expected error is, the greater is

**Algorithm 2:** Drift-based adaptive abstaining

---

**Input:**  $S = \{y_1, y_2, \dots, y_t \dots\}$ ,  $M = \{M^1, M^2, \dots, M^m\}$ ,  
 $RES = \{RES^1, RES^2, \dots, RES^m\}$ , Initial competence threshold:  $\theta$ ,  
Adjustment factor  $s$ ,  $\delta_w$ : warning threshold,  $\delta_d$  drift threshold

**Output:**  $\hat{y}_{t+1}$  one-step-ahead forecast

```

1  $\theta_0 \leftarrow \theta$ 
2 while EndOfStream = False do
3   Get new test instance  $x_{t+1}$ 
4    $M^c \leftarrow \emptyset$  // Committee of experts
5   foreach  $M^i \in M$  do
6      $abstain \leftarrow MajorityVote(RES_t^i, \theta_t)$ 
7     if  $abstain = False$  then
8        $M^c \leftarrow M^c \cup \{M^i\}$ 
9     end
10  end
11   $\hat{y}_{t+1} \leftarrow Aggregate(M^c)$  // Aggregate expert's predictions
12  Get true label  $y_{t+1}$ 
13   $e_{t+1} \leftarrow error(y_{t+1}, \hat{y}_{t+1})$ 
14  if  $C(\delta_w, e_{t+1})$  then
15     $Decrease(\theta_t, s)$  // Warning detected, decrease  $\theta_t$ 
16  end
17  else
18     $Increase(\theta_t, s)$  // Increase  $\theta_t$ 
19  end
20  if  $C(\delta_d, e_{t+1})$  then
21     $Reset(\theta_t)$  // Reset  $\theta_t$  to  $\theta$ 
22  end
23   $Update(M, x_{t+1}, y_{t+1})$ 
24   $Update(RES_t, x_{t+1}, y_{t+1})$ 
25 end

```

---

the probability of  $M^i$  to be selected among the committee of experts.

**Beta randomized**

The beta distribution was used in the *Thompson* sampling method which is an algorithm for online decision problems where actions are taken sequentially in a manner that must balance between exploiting what is known to maximize immediate performance and investing to accumulate new information (exploring) that may improve future performance [139].

The beta distribution requires two parameters  $Beta(a, b)$  and we associate to each individual model  $M^i \in M$  a pair of parameters  $(a^i, b^i)$ . A beta distribution with parameters  $(a^i, b^i)$  has mean  $a^i / (a^i + b^i)$ , and the distribution becomes more concentrated as  $a^i + b^i$  grows. For each test instance  $x_{t+1}$ , the algorithm generates an estimate  $\theta^i$  for each base model in the pool, that represents the current expectation of the success probability. The  $\alpha\%$  models with

the largest estimate  $\theta^i$  are then selected and their predictions aggregated. Initial values of  $a^i$  and  $b^i$  are set to 1.

---

**Algorithm 3:** Beta-Randomized selection
 

---

**Input:**  $S = \{y_1, y_2, \dots, y_t, \dots\}$ ,  $M = \{M^1, M^2, \dots, M^m\}$ , Selection ratio  $\alpha$

**Output:**  $M^c \subset M$

```

1 while EndOfStream = False do
2    $M^c \leftarrow \emptyset$  // Committee of experts
3   Get new test instance  $x_{t+1}$ 
4    $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$ 
5   foreach  $M^i \in M$  do
6      $r^i \leftarrow 1 - \hat{e}_t^i$ 
7      $(a^i, b^i) \leftarrow (a^i + r^i, b^i + 1 - r^i)$ 
8     Sample  $\hat{\theta}^i \sim \text{Beta}(a^i, b^i)$ 
9   end
10   $M^c \leftarrow \text{Top}_\alpha(\hat{\theta}^i)$  // Select  $\alpha\%$  with highest success probability
11   $\hat{y}_{t+1} \leftarrow \text{Aggregate}(M^c)$  // Aggregate expert's predictions
12
13 end
  
```

---

### 3.2.4 PERFORMANCE-DIVERSITY TRADE-OFF

The accuracy-diversity dilemma is a crucial aspect when selecting a committee of experts. The goal is to select the most accurate base-models with the highest diversity among their predictions. In fact, if base-models make the same predictions, there would be no benefit from combining their outputs. This issue was previously addressed in the Information Retrieval (IR) field where the Maximal Marginal Relevance (MMR) [34] was introduced as a document re-ranking method to promote novelty. The purpose is to return a set of documents deemed to be relevant for a given query  $Q$ . MMR selection is based on a combined criterion of query relevance and novelty of information. MMR strives to reduce redundancy while maintaining query relevance in re-ranking retrieved documents.

The MMR criterion is defined as:

$$\text{MMR} \stackrel{\text{def}}{=} \arg \max_{D_i \in R \setminus S} \left[ \lambda \times (\text{Sim}_1(D_i, Q) - (1 - \lambda) \times \max_{D_j \in S} \text{Sim}_2(D_i, D_j)) \right] \quad (3.4)$$

- $Q$  denotes the query,
- $R$  denotes the ranked list of documents according to query relevance
- $S$  denotes the subset of documents in  $R$  that are already selected
- $\text{Sim}_1$  stands for the measure of relevancy
- $\text{Sim}_2$  stands for the measure of redundancy between documents
- $\lambda$  is the trade-off factor of the combined MMR criterion between relevance and novelty.

A document  $D_i$  is said to have a high MMR score if it is relevant to the query  $Q$  and contains minimal similarity, i.e maximal novelty, with the documents that were previously selected.

### MMR for streaming DES

We discuss the use of MMR combined criterion selection for DES in the case uni-variate temporal data streams forecasting. We draw an analogy with the previous definition in the IR field. First, the query  $Q$  at time  $t$  stands for the test instance  $x_{t+1}$  to be used to predict  $y_{t+1}$ . The documents  $D_i$  is equivalent to base-model  $M^i \in M$ . The goal of MMR in our case is to select a subset of highly accurate models with high novelty (low redundancy).

- The notion of relevancy to the query is related to the expected performance of model  $M^i$  on instance  $x_{t+1}$ , the lower the error, the higher the relevancy. Therefore,  $Sim_1(M^i, x_{t+1})$  is inversely related to  $\hat{e}_{t+1}^i$  and defined as  $Sim_1(M^i, x_{t+1}) = 1 - \hat{e}_{t+1}^i$ .
- The redundancy measure  $Sim_2$  can be assimilated to a diversity measure in the context of ensemble methods. We choose the redundancy measure to be the correlation between models' predictions collected over a sliding window. Pearson's correlation factor  $\rho \in [-1, 1]$  is used to measure the strength of the linear relationship between two model's predictions. In fact, a correlation value greater (resp. lower) than 0 means a positive (resp. negative) relationship while a value zero indicates no relationship between the two variables being compared. We set  $Sim_2 = \frac{1+corr_w(M^i, M^j)}{2} \in [0, 1]$ . A value of 0 (resp. 1) stand for inversely (resp. positively) correlated base models. Low correlation values stand for higher "novelty" among base models predictions.

Concretely, for DES approach using MMR, we start  $M^c$  with the best ranked model according to  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$ , the lower the error, the better the rank. Before adding any model to  $M^c$ , we compute its MMR score using its correlation to all the previously selected models in  $M^c$ . Similar to trimming selection, the trade-off approaches requires a selection ratio parameter  $r$  to determine the size of the subset committee  $M^c$  and a trade-off factor  $\lambda$ . If  $\lambda = 1$ , this leads to a standard performance based ranking. If  $\lambda = 0$ , it computes a maximal diversity ranking among base models. Finally, for intermediate values  $\lambda \in ]0, 1[$  it leverages a linear combination of both performance and diversity criteria. The MMR-based DES approach is detailed in Algorithm 4 where  $MMR$  function computes the MMR score of model  $M^i$  relative to the previously selected models in  $M^c$ .

## 3.3 EMPIRICAL EXPERIMENTS

In this section, we present the empirical experiments carried out to compare the performance of the proposed streaming DES methods relative to state-of-the-art approaches for dynamically selecting and combining expert models within an ensemble. More particularly, we focus on the proposed SLOPE (windowing) and STADE (meta-learning) to estimate the predictive performance of individual models in the pool. Besides we study different approaches to select individual models that are as accurate as possible and as diverse as possible following the discussion in Section 3.2.

**Algorithm 4:** Trade-off MMR selection

---

**Input:**  $M = \{M^1, M^2, \dots, M^m\}$ ,  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$ , Selection ratio  $r \in [0, 1]$ , Trade-off factor  $\lambda \in [0, 1]$

**Output:**  $M^c$  a set of expert base models

- 1  $M^r \leftarrow \text{Rank}(M)$  // Rank models by ascending error
- 2  $M^c \leftarrow \text{Best}(M^r)$  // Select best model
- 3  $M^r \leftarrow M^r \setminus M^c$
- 4  $size_c \leftarrow r \times len(M)$
- 5 **while**  $|M^c| < size_c$  **do**
- 6 **foreach**  $M^i \in M^r$  **do**
- 7  $mmr\_score_i^i \leftarrow \text{MMR}(\lambda, \hat{e}_t^i, M^i, M^c)$
- 8 **end**
- 9  $M^k \leftarrow \arg \max mmr\_score$  // Include model with highest MMR score
- 10  $M^c \leftarrow M^c \cup \{M^k\}$
- 11  $M^r \leftarrow M^r \setminus \{M^k\}$  // Remove  $M^k$  from remaining models  $M^r$
- 12 **end**

**Return:**  $M^c \subset M$

---

**3.3.1 EXPERIMENTAL DESIGN**

The thesis falls within the one-step-ahead uni-variate temporal data streams point forecasting task as described in Section 2.1.2. We intend to forecast at time  $t$  future values  $\hat{y}_{t+1}$  of the stream  $S$  based on the historical information  $\mathcal{I}_t$  available up to time  $t$  where  $y_t \in \mathbb{R}, \forall y \in S$ . Our work focuses on dynamic ensemble selection (DES) using a heterogeneous pool of models as described in Section 2.2.3.2. This means that the ensemble comprises different incremental algorithms tailored to the forecasting task.

Experiments were conducted on 30 temporal streams including both real and synthetic data that are detailed in Appendix A (TableA.1). Each data stream comprises at least 10000 temporally dependent instances with varying sampling frequencies. The data streams are derived from different application domains such as IoT sensors monitoring air quality and traffic data. The heterogeneous ensemble of size  $m = 30$  comprises different base models using incremental regression models such as k-nearest neighbors (KNN) and adaptive regression trees (RHT). Besides, we use simple forecasting methods discussed in Section 2.1.5.1. The algorithms were designed using different parameters settings to promote diversity. For a detailed summary on the set of base forecasters that were used to generate the ensemble, we refer to Section A.2. The methods were evaluated and ranked using the Root Mean Square Error (RMSE) as described in Section 2.1.7.2 in a prequential setting detailed in Section 2.1.7.1. This means that each instance in the stream is first used to test the model *before* it is used to train it. The prequential evaluation promotes the idea that more recent examples are more important. We assume that instances are released one by one in the stream without any delay. In addition to the rank, we analyze the percentual difference (relative change) in terms of predictive performance according to Equation 3.5 where pivot

often stands for the best performing method on average.

$$Impr. = 100 \times \frac{Pivot_{perf} - Model_{perf}}{Model_{perf}} \quad (3.5)$$

A negative (resp. positive) value  $r$  means that the pivot model scores an improvement (resp. deterioration) of  $|r|\%$  compared to the other model.

Likewise, we discuss the computational cost where a ratio  $\rho$  as described in Equation 3.6 means that the pivot method requires as much as  $\rho$  times the resources required by the other model. We address the computational cost related to memory, that is the space required to store the learning model measured in kilobytes (kB) as well as the time (measured in seconds) required to predict and update each model on all instances of the stream.

$$\rho = \frac{Pivot_{cost}}{Model_{cost}} \quad (3.6)$$

Both windowing and meta-learning based DES methods use the sMAPE loss function to assess the predictive performance of base-models according to each instance  $x_t$ . However, the factor 2 is removed in order to bound the values of sMAPE to be  $[0, 1]$  and meet the requirements for the abstaining policy using a threshold. We recall that, experiments are based on auto-regressive model (AR) with  $p = 10$ . The optimal embedding dimension ( $p$ ) can be optimized using the method of False Nearest Neighbours [89] that looks at the behavior of the nearest neighbours as the value of  $p$  increases. However, this is beyond the scope of the thesis as the goal is to compare different DES methods on the same data in a streaming setting. Besides, data streams are subject to concept drift where the generating process changes and impacts the optimal embedding dimension.

In order to avoid performance issues related to the cold start, a first batch of labeled instances is dedicated to enforce a "warm" start using the blocked prequential procedure described in A.3. The initial batch of 1000 instances is divided into  $\beta$  equally sized and sequential blocks of contiguous observations. Blocks containing 10 observations are sequentially used to train then test to retrieve base-models predictions on the next batch and compute their respective predictive performance. The first part of the experiments addresses windowing and meta-learning DES methods using the trimming selection approach. The  $\alpha\%$  best base-model with the lowest predicted errors are selected in a committee  $M^c$  and their outputs weighted using the softmax function given in Equation 3.7.

$$w_{t+1}^i = \frac{\exp(-\hat{e}_{t+1}^i)}{\sum_{i \in M^c} \exp(-\hat{e}_{t+1}^i)} \quad (3.7)$$

where  $\hat{e}_{t+1}^i$  is the predicted error that model  $M^i$  will incur in  $y_{t+1}^i$  and  $w_{t+1}^i$  its weight where  $\exp$  denotes the exponential function. In the function *softmax*, the weight of a given model decays exponentially as its loss increases and aims at increasing the influence of the best performing models. The final prediction is a weighted average of experts committee  $M^c$  members' predictions as described in Equation 3.8.

$$y_{t+1} = \sum_{j \in M^c} w_{t+1}^j \times \hat{y}_{t+1}^j \quad (3.8)$$

Subsequently, we address different selection and abstaining approaches discussed in Section 3.3.5 and compare their predictive performance relative to trimming.

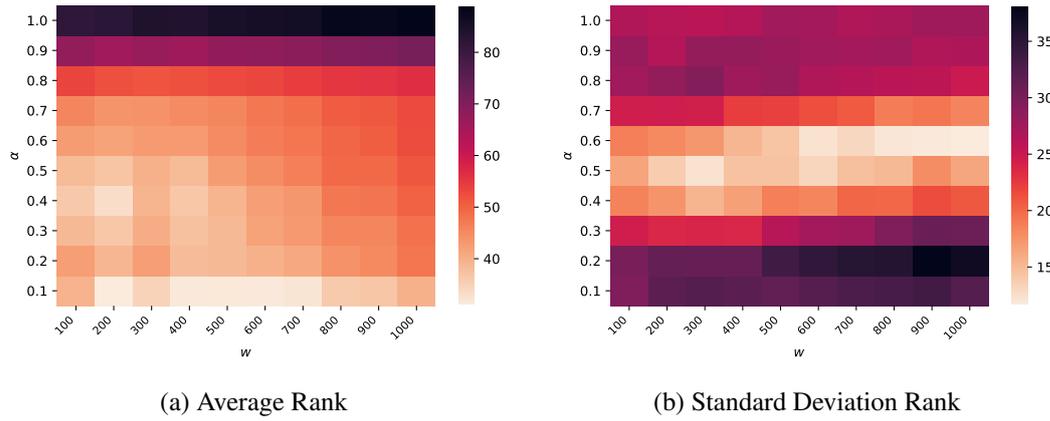


Figure 3.4: Heatmaps illustrating the average rank (3.4a) and respective standard deviation (3.4b) of SW-DES for varying  $w$  and  $\alpha$  parameter values. Darker tiles mean higher values

### 3.3.2 COMPARING WINDOWING METHODS

In this section, we analyze windowing-based DES methods discussed in Section 3.1.1 using trimming (TRIM) that selects a fixed ratio  $\alpha$  of the best performing base-models.

#### 3.3.2.1 Sliding-Window (SW) DES analysis

First, we address the fixed-size sliding window (SW) method that estimates the performance of individual models in the ensemble  $M$  based on their respective predictions on instances comprised in a recent window of data. SW based trimming DES performance is related to  $\alpha$  values, which denotes the ratio of selected experts in the committee and  $w$ , which represents the maximum size of the sliding window comprising the most recent instances. We analyze how the performance of SW-DES varies as the values of the two parameters  $\alpha$  and  $w$  change. We considered  $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$  and  $w \in \{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000\}$ . This renders a total of 100 variants. This analysis was carried out using the 30 temporal data streams and models were ranked according to their respective RMSE in each problem.

The results are shown in Figure 3.4 that illustrates two heatmaps for the average rank (Subfigure 3.4a) and respective standard deviation (Subfigure 3.4b) of each  $(w, \alpha)$  combination. Higher average rank (i.e. worse performance) and higher rank standard deviation are denoted by darker tiles. Varying the value of  $w$  follows the stability-plasticity dilemma. Small values of  $w$  lead to better reactivity, but also make the model more susceptible to outliers. Conversely, higher values of  $w$  lead to better stability while losing some responsiveness and possibly including outdated information. This is particularly fastidious for evolving data streams where concept drifts are likely to occur and alter the overall predictive performance.

Subfigure 3.4a shows that the best performing methods (lowest average rank) lay in the lower-left corner that translates to smaller  $w$  and  $\alpha$  values whereas the worst performing ones (higher average rank) lay in the upper-right corner (high  $w$  and high  $\alpha$ ). We can notice a smooth diagonal increase of the average rank values. On the other hand, Subfigure 3.4b suggests that the  $(w, \alpha)$  combinations with the lowest rank standard deviation are in the middle of the searched distribution. In fact, despite relatively good average rank for smaller values of  $w$  and  $\alpha$ , their combination results in very high standard deviation which indicates

greater reactivity, but higher susceptibility to noise and outliers.

### 3.3.2.2 Sliding local performance (SLOPE) analysis

We validate the predictive performance of the proposed SLOPE method that combines both a temporal criteria to emphasize on the most recent instances and a feature-based similarity to select the most similar training instances (using KNN) comprised in the sliding windowing to estimate the respective performance of  $M^i \in M$  (Section 3.1.1.1). Similarly to SW, we conduct an empirical study on TRIM-based SLOPE-DES method and its sensitivity to the variation of its parameters values. Performance are related to the size of the sliding window  $w$  and the selection ratio  $\alpha$  along with  $k$  that stands for the number of nearest neighbors We recall that we are using the time delay embedding that comprises the past  $p$  observed values in the stream and the euclidean distance to achieve the KNN search.

Figure 3.5 illustrates the heatmaps of the average rank and respective standard deviation for varying values of  $w$ ,  $\alpha$  and  $k$  of the SLOPE method. The values of  $w$  and  $\alpha$  are similar to the experiments conducted for SW. Each row stands for a different value of  $k \in \{5, 10, 15, 20\}$ . Results are illustrated for each value of  $k$  for the sake of better readability, however the ranking was carried using all the variants over the 30 temporal data streams. Fixing the value of  $k$  shows the sensitivity of SLOPE to the combination of parameters  $(w, \alpha)$  as discussed for SW methods. On the other hand, fixing  $w$  and  $\alpha$  allows analyzing the effect of varying nearest neighbors parameter  $k$  on the predictive performance. Results show that, for fixed  $(w, \alpha)$  (reading by column), lower values of  $k$  lead to worse (higher) average rank where the darkest zones are recorded for  $k = 5$  whereas, higher values  $k \in \{15, 20\}$  lead to better average rank overall. In fact, the brightest zone for average rank recorded overall Figure 3.5 tends to the right side lower corner for  $k \in \{15, 20\}$ . However, analyzing the standard deviation for  $k = 20$ , higher values are recorded despite lower average rank compared to SLOPE when  $k = 15$ . Selecting a number  $k$  of nearest neighbors that is too low or too high alters the estimation of the predictive performance of base-models.

### 3.3.2.3 Comparing all windowing DES methods

We compare here the proposed SLOPE based DES methods to their SW counterparts. We select the best performing SW and SLOPE methods, namely the methods with the lowest average rank and respective standard deviation. We investigate the predictive performance as well as the computational cost in terms of time and memory. Figure 3.6 highlight the the distribution of the rank and respective standard deviation of different SW and SLOPE based DES methods with varying window size  $w \in \{200, 300, 400, 500, 800, 900\}$ , selection ratio  $\alpha \in \{0.3, 0.4, 0.5\}$  and nearest-neighbors  $k = 15$  values. Results show that the proposed SLOPE variants achieve promising results and outperform their SW counterparts for similar sliding window size and selection ratio. Additionally, higher  $w$  in SW methods lead to lower performance whereas the inverse tendency is noted for SLOPE where larger sliding window in addition to KNN based selection lead to better average rank. In fact, computing individual models' loss on more similar instances according to the test instance  $x_t$  selected from more distant past gives valuable information on the behavior of base-models in the near future. SW using  $w = 200$  and  $\alpha = 0.3$  is the best ranked fixed size sliding window DES method and the only one that could outperform its SLOPE counterpart. Besides, using  $\alpha \in \{0.3, 0.5\}$  lead to higher average rank compared to  $\alpha = 0.4$  which indicates lower

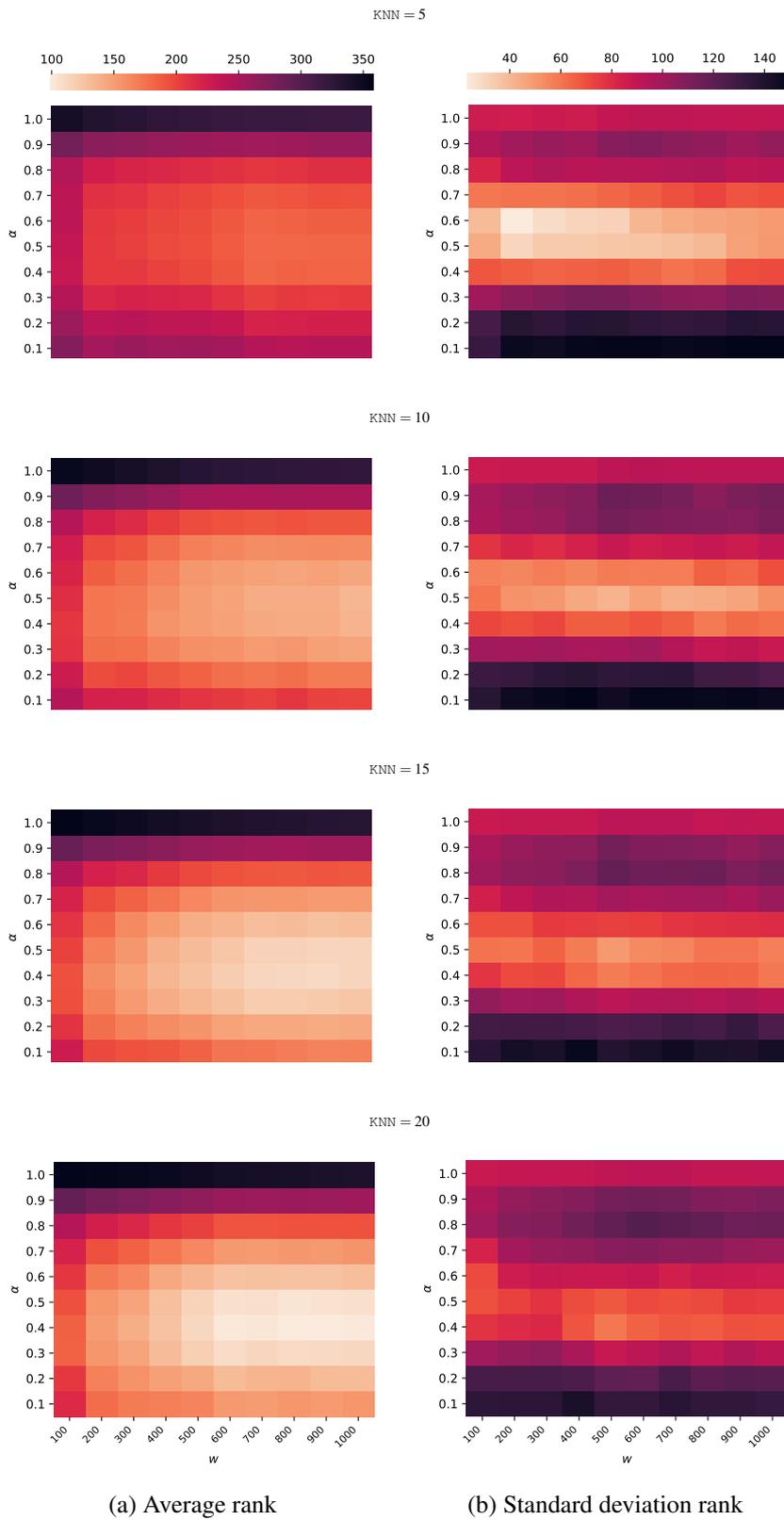


Figure 3.5: Heatmaps illustrating the average rank (left) and respective standard deviation (right) of SLOPE for varying  $w$ ,  $k$  and  $\alpha$  parameters. Darker tiles mean higher values

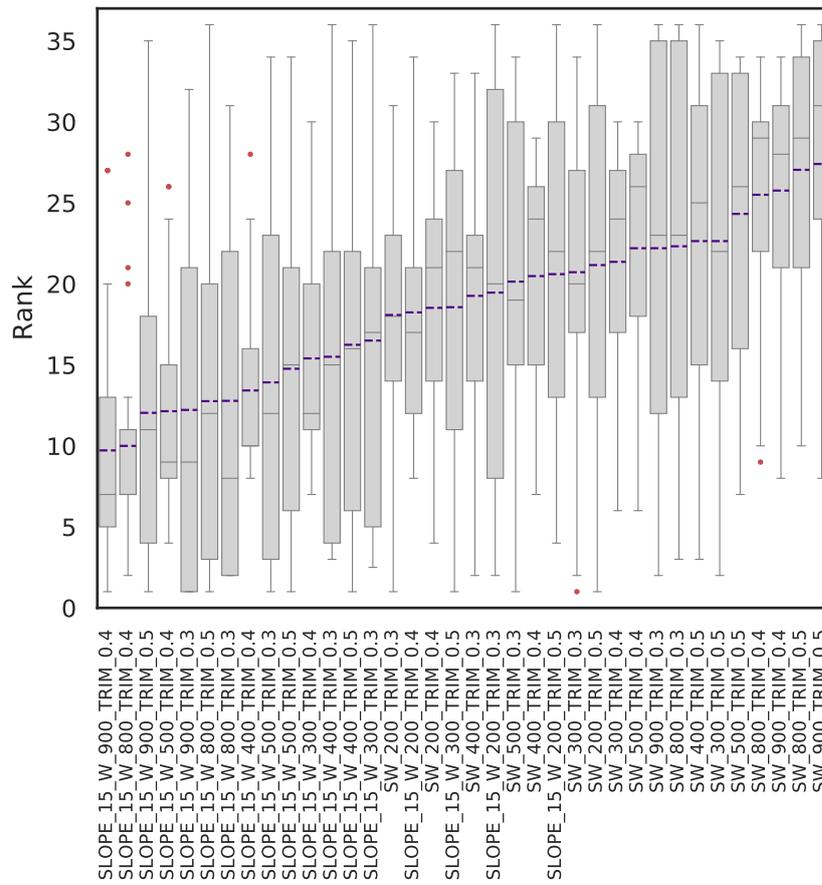


Figure 3.6: Boxplot illustrating the distribution of the rank and respective standard deviation of SW and SLOPE trimming based DES methods in terms of RMSE

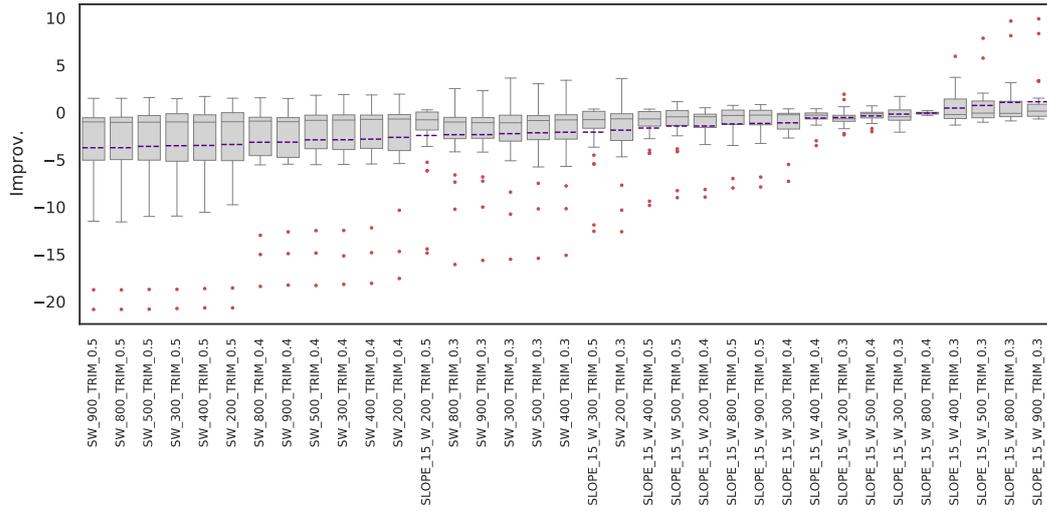


Figure 3.7: Boxplot illustrating the relative difference expressed in percentage achieved by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) against other windowing DES methods. Values below (resp. above) the zero line represent improvement (resp. loss) in the predictive performance.

certainty on their superiority. Next, we measure the pairwise relative percentual difference in terms of predictive performance compared to the best performing approach that is SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ). This analysis is presented in Figure 3.7 that illustrates the distribution of the percentual difference and respective standard deviation to visualize the magnitude of the difference in the predictive performance. Values below (resp. above) the zero line represent improvement (resp. loss) in the predictive performance. Results show that for all SW methods, the best SLOPE variant leads to a better predictive performance in average. Overall, the distribution of the percentual difference shows that SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) performs relatively better with varying average improvement compared to other windowing DES methods. Moderate improvement (less than 5%) is scored against SW methods whereas SLOPE variants have very close performance. In order to analyze the significance of the previous study, we carried a Bayesian analysis of the results to test for consistent differences between pairs of methods following the approach presented in [41]. Particularly, we employed the Bayes sign test to compare all the methods across the 30 temporal data streams against the best performing one on average rank. We define the Region of Practical Equivalence (ROPE) to be the interval  $[-0.01, 0.01]$ . This means that two methods show indistinguishable performance if the relative difference in performance between them falls within this interval. Otherwise, the respective pair of methods are considered practically equivalent. In order to perform the Bayesian analysis of the results, we normalize the results of the RMSE loss metric relative to the RMSE of Simple method that aggregates the all predictions with equal weights. This aims at avoiding issues related to the scale since RMSE loss function varies according to the scale of the temporal data stream.

Figure 3.8 illustrates the application of the Bayes sign test to analyze the significance of the superiority of SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) compared to the other windowing based DES methods. Each bar stands for a method and describes the probability of each outcome as follows: SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) winning, drawing (within the ROPE),

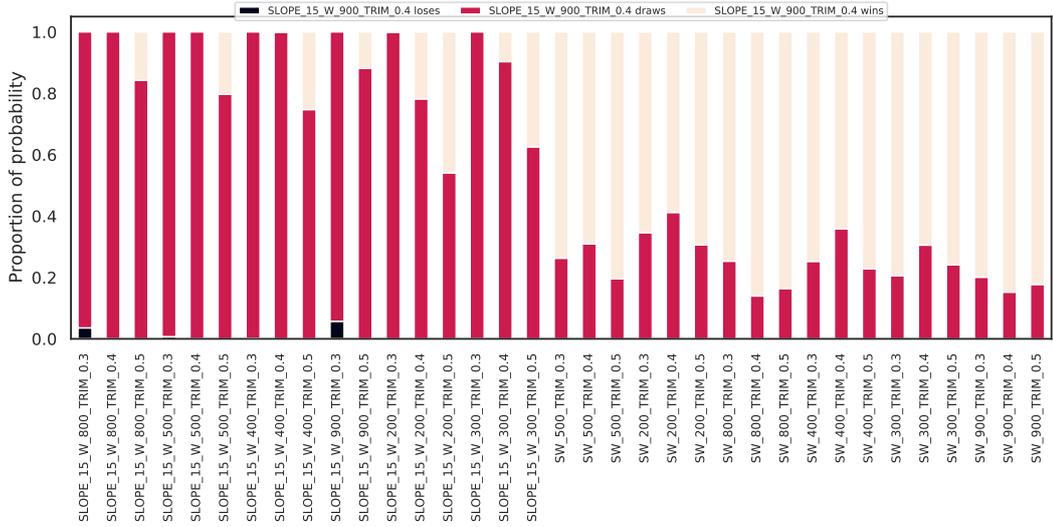


Figure 3.8: Proportion of probability of SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) winning/drawing/losing against other windowing DES methods according to the Bayes sign test

Table 3.2: Average ratio and respective standard deviation of the memory and time required by SLOPE compared to SW counterparts for similar  $w$  and  $\alpha$  and varying values of  $k$ .

$k$	Memory	Time
5	$1.000584 \pm 0.018463$	$1.023422 \pm 0.076779$
10	$1.000654 \pm 0.018498$	$1.019095 \pm 0.075788$
15	$1.000732 \pm 0.018537$	$1.021346 \pm 0.076499$
20	$1.000796 \pm 0.018562$	$1.019072 \pm 0.075785$

or losing. In general, SLOPE variants have high probability to be practically equivalent (probability of the event draw is higher) except for  $\alpha = 0.3$  that scores small probability of winning. More particularly, SLOPE has higher probability to win against all other SW variants. The rank analysis combined with the percentual difference and the Bayesian test results corroborate the advantage of the proposed SLOPE as a windowing DES method and the benefit of leveraging both temporal and feature based criteria to estimate the predictive performance of each individual model  $M^i \in M$  according to the test instance at hand  $x_t$ .

### Computational cost

In this section, we analyze the computational cost in terms of memory (model size) and time resources of SLOPE relative to SW counterparts. In fact, SW and SLOPE DES methods are both based on a sliding window  $w$  however, SLOPE introduces an additional stage to search for the  $k$  nearest neighbors to the test instance  $x_t$  comprised in the sliding window  $w$ . Table 3.2 details the average computational cost ratio and respective standard deviation (model size and total running time) of SLOPE-DES methods compared to SW counterparts (similar sliding window size  $w$  and selection ratio  $\alpha$ ).

Results show that SLOPE has slightly higher computational cost in average compared to SW (ratio above 1). In fact, SLOPE methods are marginally larger due to the data structure

KDTree [10] maintained online to partition the  $p$ -dimensional data space and perform the nearest-neighbors search<sup>1</sup>. Besides, SLOPE takes lightly more time in average due to KNN search in the feature space despite lower number of instances involved in the loss computation. The experimental results have demonstrated very promising results in terms of predictive performance as well as computational cost of the proposed SLOPE approach relative to SW DES methods. In fact, considering the feature-based similarity to the test instance  $x_t$  using the  $k$ -nearest neighbors along with the recency of the data using a sliding window leads to promising improvements in terms of predictive performance. Besides, SLOPE implies moderate and lightweight additional computational cost in terms of time and memory resource requirements.

### 3.3.3 COMPARING META-LEARNING BASED METHODS

In this section, we address the STreaming-ADE (STADE) meta-learning based DES method and compare the predictive performance using several meta-features discussed in Section 3.1.2.1 to characterize the temporal data stream and the behavior of base models. Arbiters were implemented using incremental regression trees (FIMT-DD) [84] that can cope with the evolving nature of temporal data streams and meet the computational requirements. Similarly to the empirical study conducted in Section 3.3.2, we first compare STADE using the trimming selection approach against other meta-learning based methods such as STACKING and META-STREAM.

First, we analyze several variants of STADE using the set of meta-features discussed in Table 3.1 for varying selection ratio values  $\alpha$ . Figure 3.9 illustrates the heatmaps of the average rank (Sub-Figure 3.9a) and respective standard deviation (Sub-Figure 3.9b) for varying meta-features and selection ratio values  $\alpha$ . We recall that STADE uses arbitration and assigns to each base-model  $M^i \in M$  a dedicated meta-model  $Z^i$  that learns to predict future errors according to the test instance at hand  $x_t$ . The ORI label stands for the original feature vector of the test instances using time delay embedding to train the base-models. Similarly to the sensitivity analysis conducted for SW and SLOPE, the best selection ratio values  $\alpha$  on average lie in the middle when fixing the set of meta-features  $\alpha \in \{0.3, 0.4, 0.5\}$  whereas values that are too small (resp. too high) on the far left (resp. right) achieve higher average rank. Besides, Sub-Figure 3.9b supports the relevant findings above where the lightest zone that stands for smaller standard deviation values lies in the middle of the research distribution as well. Results also show the superiority of using the proposed meta-features related to the performance of landmarks including warning and drift flags (LAN) as well as the lagged base errors (LAG) for each meta-model in addition to the statistical information extracted from the data referred to with STA. Figure 3.10 is here to visualize the magnitude in the improvement (percentual difference) in the predictive performance when using the set of meta-features that led to the best average rank according to the results illustrated in Figures 3.9a and 3.9b namely ORI\_STA\_LAG\_LAN, against other variants of STADE (using other meta-features) for similar selection ratio values  $\alpha$ . Results show very similar performance compared to ORI\_STA\_LAG where most of the values are clustered around the value 0. On

<sup>1</sup>We have used the *Scikit-Multiflow* Python package implementation available at [https://github.com/scikit-multiflow/scikit-multiflow/blob/master/src/skmultiflow/lazy/base\\_neighbors.py](https://github.com/scikit-multiflow/scikit-multiflow/blob/master/src/skmultiflow/lazy/base_neighbors.py)

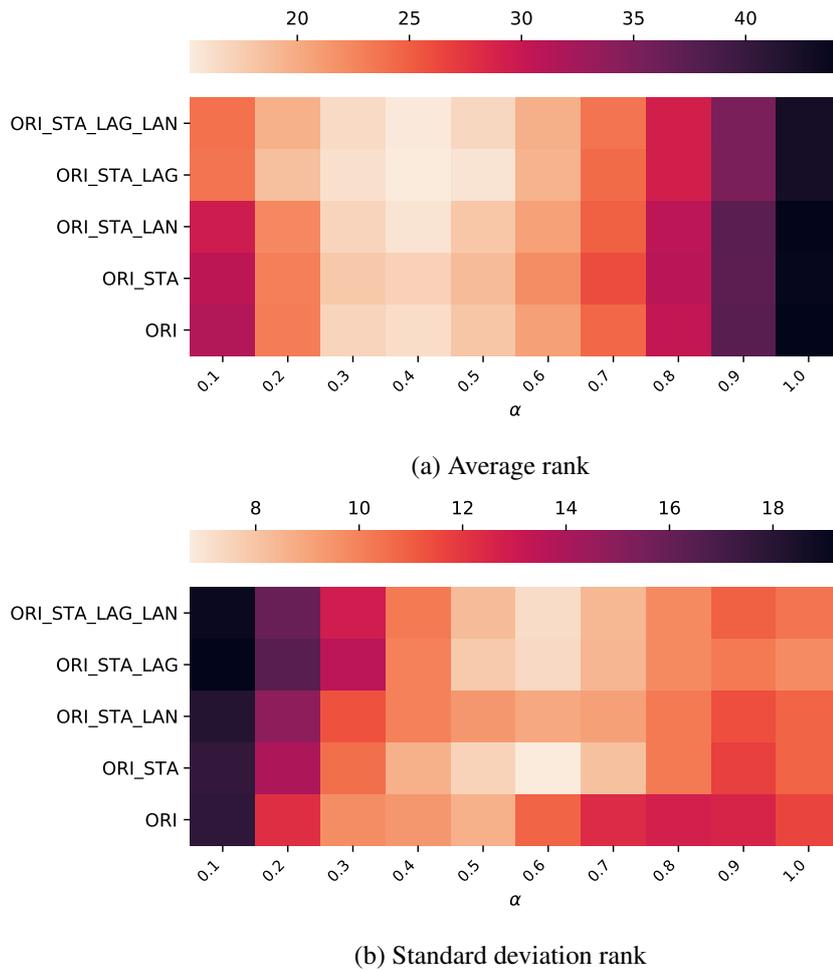


Figure 3.9: Heatmaps illustrating the average rank (3.9a) and respective standard deviation (3.9b) of STADE for varying meta-features set and  $\alpha$  parameters. Darker tiles mean higher values.

the other hand, higher percentual difference values (in absolute value) are scored for the other variants which highlights the positive impact of using landmarking and stream-specific meta-features along with the original feature set and statistical information extracted from the stream.

Figure 3.11 depicts the distribution of the rank and respective standard deviation of STADE variants using the best performing selection ratio values  $\alpha \in \{0.3, 0.4, 0.5\}$  according to the results highlighted in Figures 3.9 and 3.10. Results show that including additional meta-features discussed in Table 3.1 improves overall predictive performance and leads to lower average rank. Besides, using  $\alpha = 0.4$  achieves lower standard deviation which reflects higher certainty. We analyze in Figure 3.12 the percentual difference in the predictive performance achieved by the best performing STADE that is the variant using ORI\_STA\_LAG meta-features along with  $\alpha = 0.3$ .

Overall, STADE variant using ORI\_STA\_LAG performs better on average (relative average difference below 0) compared to other variants for similar selection ratio  $\alpha = 0.3$  despite moderate percentual difference values recorded. On the other hand, better

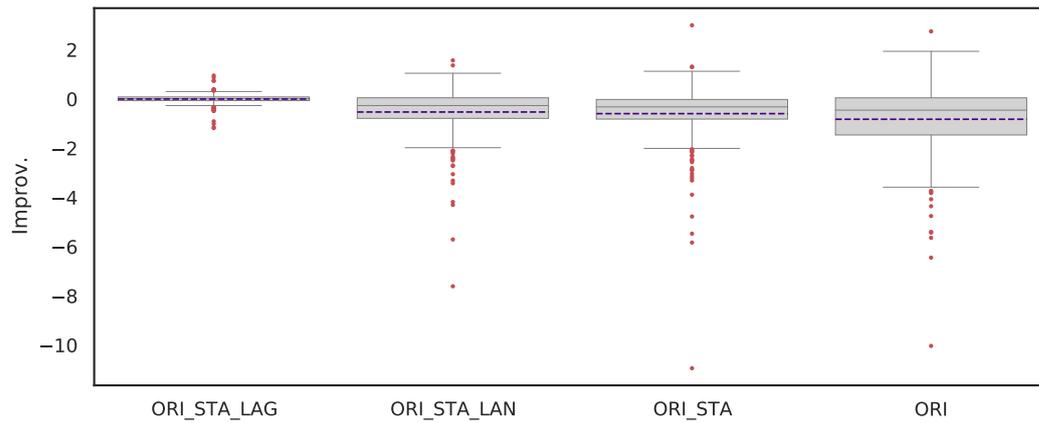


Figure 3.10: Boxplot illustrating the distribution of the percentual difference in terms of RMSE achieved by ORI\_STA\_LAG\_LAN against other STADE variants using different meta-features for corresponding selection ratio  $\alpha$ .

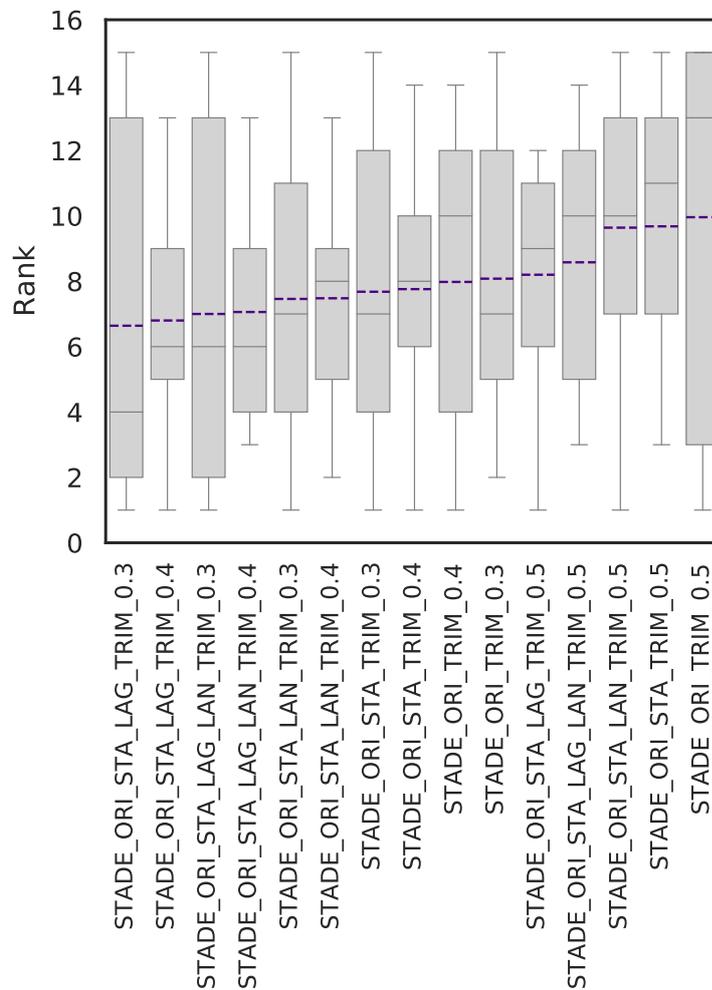


Figure 3.11: Boxplots illustrating the distribution of the rank and respective standard deviation of the best performing STADE trimming DES methods

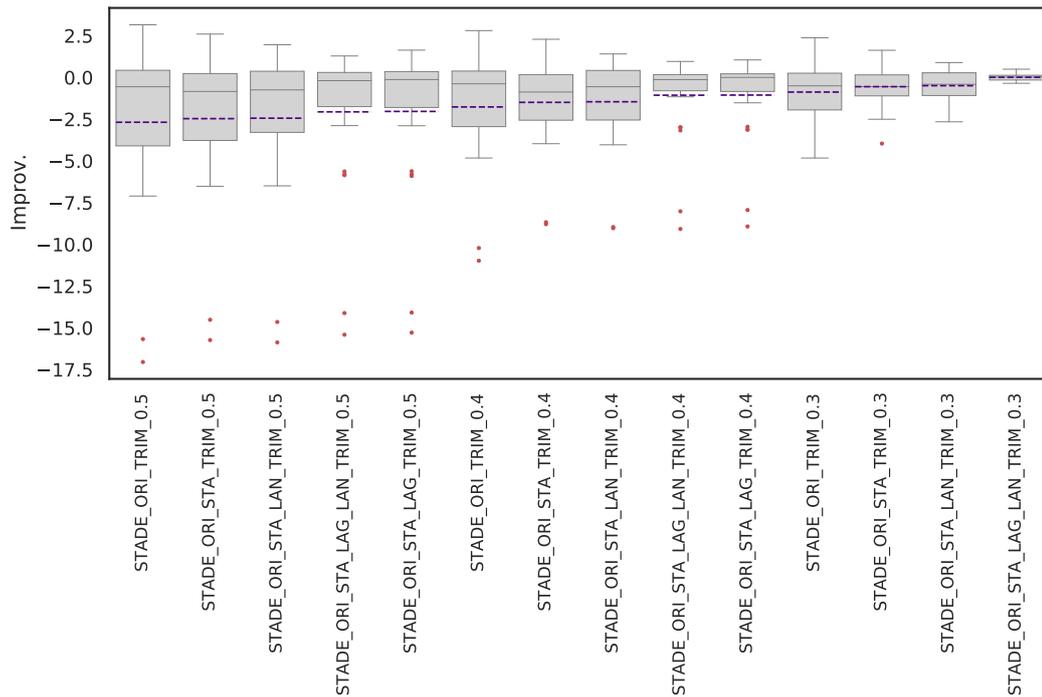


Figure 3.12: Boxplot illustrating the distribution of the percentual difference in terms of RMSE achieved by the best STADE.

improvement is scored compared to other STADE variants particularly for higher selection ratio  $\alpha$ . In the same way as discussed in Section 3.3.2 for windowing DES methods, we analyze the significance of the results stemming from the rank and relative difference in terms of predictive performance using the Bayes sign test. Figure 3.13 illustrates the proportion of probability that the best performing STADE variant loses, draws or wins against each method represented by a separate bar. In general, the probability that STADE using statistical and lagged errors with  $\alpha = 0.3$  wins against other STADE variants that use the original feature space and statistical information only with  $\alpha \in \{0.4, 0.5\}$  is higher. However, the probability to achieve a draw is higher for other variants despite better average rank (Figure 3.11 and negative average values of percentual difference (Figure 3.12).

Finally, we compare STADE-DES methods in terms of computational cost including model size and total time required to test and train on all the instances of the stream. We analyze the distribution of the ratio between the resources (memory and time) required by the best STADE variant (ORI\_STA\_LAG) relative to the resources required by other methods. Results are depicted in Figure 3.14 where methods were compared for similar selection ratio values and grouped by their respective set of meta-features. As expected, including landmarking, lagged and statistical meta-features leads to an increase in the memory required by the meta-layer relative to the use of the original feature vector only. In fact, STADE with ORI\_STA\_LAG is almost 4 times more memory consuming and twice more time consuming compared to methods using the original set of features only (STADE-ORI). Using the  $q = 5$  previous errors (LAG) of each individual in the set of meta-features of each arbiter prompts moderate additional complexity compared to other variants. Lastly, including landmarking (LAN) meta-features that are computed by monitoring the performance of landmark models

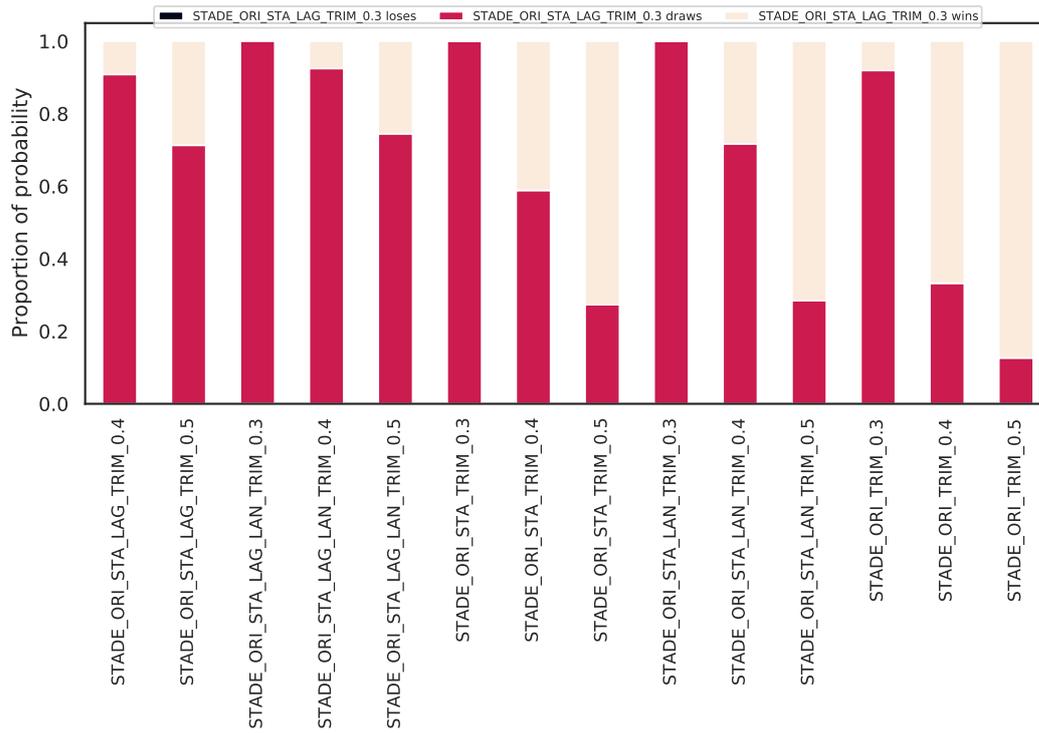


Figure 3.13: Proportion of probability of best STADE variant winning/drawing/losing according to the Bayes sign test

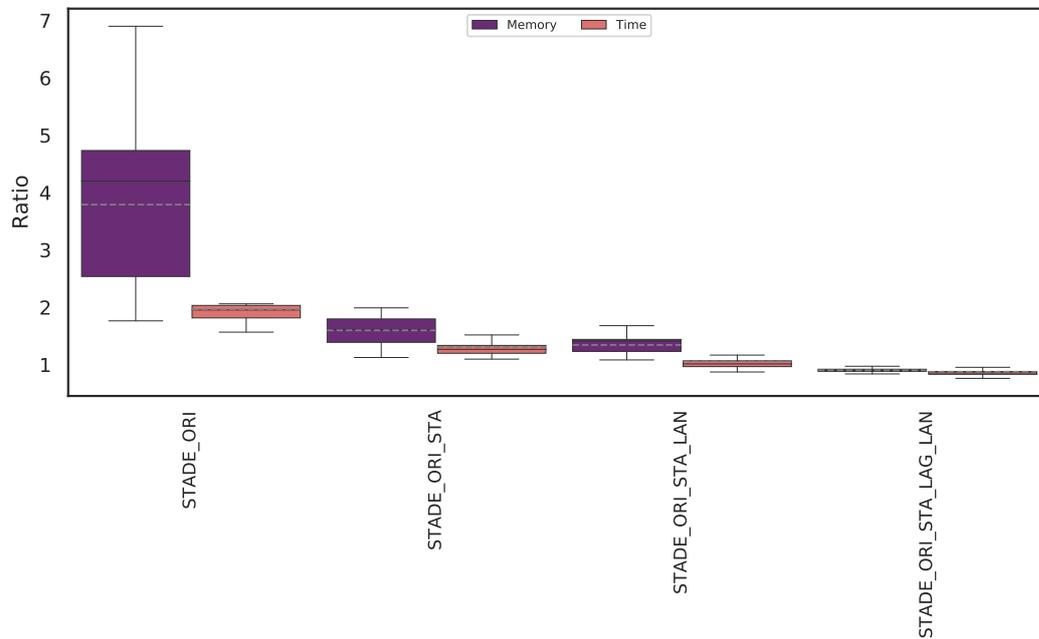


Figure 3.14: Boxplot illustrating the distribution of the ratio between the resource spent by best STADE in terms of memory and time relative to other variants. Values below (resp. above) the value 1 stand for lower (resp. higher) computational resource requirement.

(KNN and RHT) has mild impact on the overall computation cost.

The proposed Streaming Arbitrated Dynamic Ensemble (STADE) using different meta-features that describe the characteristics of the temporal data stream using statistical information in addition to landmarking models and stream specific information achieves very promising results compared to the use of the original feature vector only. On the other hand, this leads to reasonably higher computational cost in terms of model size and running time.

### 3.3.4 COMPARING SW, SLOPE AND META-LEARNING BASED TRIMMING DES

Finally, we conclude the section by putting all the experiments together to compare the best performing SW, SLOPE and STADE based DES methods against other state-of-the-art dynamic ensembles. Similarly, we compare the methods in terms of average rank according to RMSE and compute the relative difference to the best performing approach. Besides, we perform the Bayes sign test to assess the significance of the results by comparing the proportion of probability of winning, drawing and losing. Figure 3.15 illustrates the distribution of the rank and respective standard deviation of the compared methods.

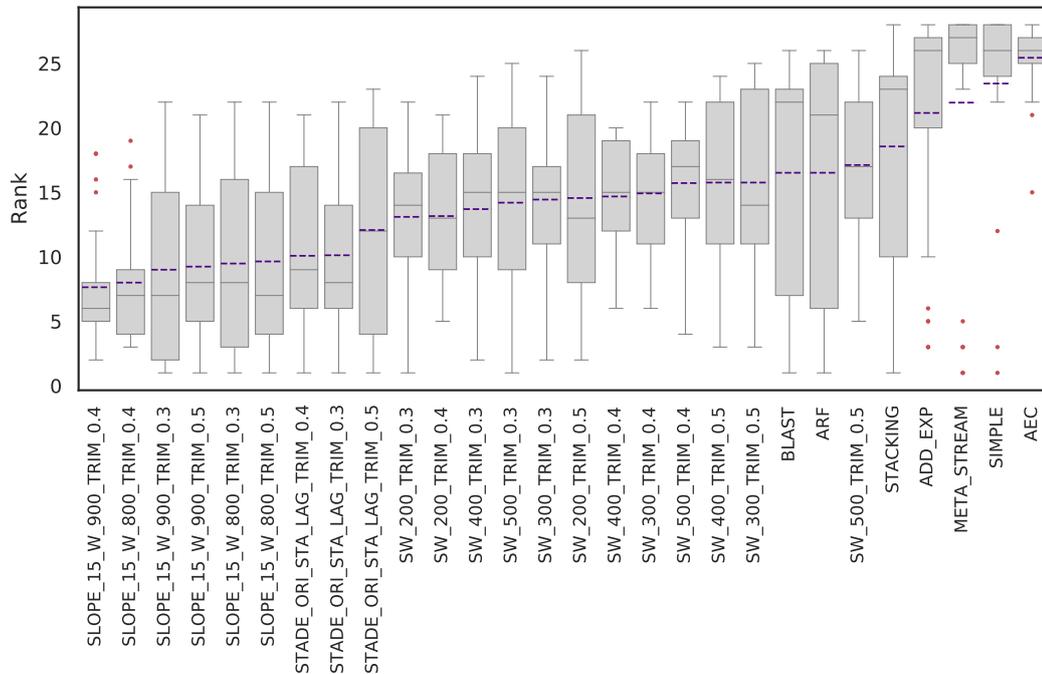


Figure 3.15: Boxplot illustrating the distribution of the rank and respective standard deviation of trimming based windowing and meta-learning based DES methods in terms of RMSE.

The results show that the proposed SLOPE-DES methods outperform in average rank the best performing SW and STADE trimming variants as well as state-of-the-art methods. In fact, SLOPE variants exhibit competitive results compared to STADE as they occupy the first positions in terms of average ranking. On the other hand, STADE meta-learning methods surpass the SW trimming DES methods and other state-of-the-art dynamic ensemble methods such as BLAST that selects the single best individual model according to past performance. Other state-of-the-art dynamic ensemble methods including ARF, STACKING, AddExp, META-STREAM, BLAST and AEC perform relatively worse compared to SLOPE. In the

following experiments, we do not include the SW and SLOPE variants other than the best performing ones (SLOPE ( $w = 900, \alpha = 0.4, k = 15$ )) as this has already been discussed in Section 3.3.2. Figure 3.16 depicts the distribution of the relative difference expressed in percentage achieved relative to STADE and state-of-the-art dynamic ensembles.

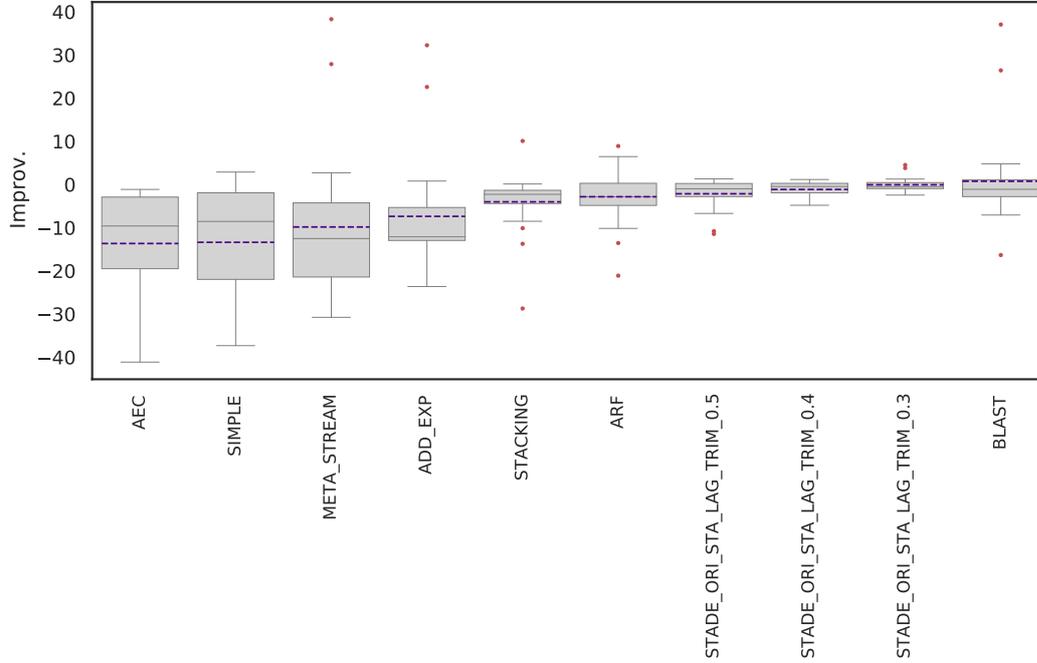


Figure 3.16: Boxplot illustrating the percentual/relative difference expressed in percentage achieved by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) compared to other DES methods.

SLOPE achieves very promising improvement compared to state-of-the-art dynamic ensemble methods, particularly for AEC and META-STREAM. Mild improvement is reached in average relative to STADE variants while higher values are scored for other ensemble methods particularly for STACKING and ARF. Finally, we conduct the Bayes sign test to compare the proportion of probability of the events that SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) winning, drawing or losing. Figure 3.17 illustrates the proportion of the probabilities of each event. BLAST and ARF have non-negligibly small probability of winning against SLOPE whereas other state-of-the-art methods always lose across all the 30 temporal data streams. Besides, SLOPE has very high probability to achieve a draw relative to STADE variants with  $\alpha \in \{0.3, 0.4\}$  whereas it scores higher probability to win when  $\alpha = 0.5$ .

Finally, we compare the methods enumerated above in terms of computational cost (memory and time). Again, the comparison does not include the SW methods as this was previously discussed in Table 3.2. We discuss the distribution of the ratio of the resource required by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) relative to STADE and other state-of-the-art dynamic ensemble methods. The results are presented in Figure 3.18 over all the 30 temporal data streams and show that the proposed SLOPE-DES methods require substantially less memory and time compared to STADE. Arbitration in general is more resource demanding compared to other meta-learning based methods such as STACKING and META-STREAM that use a single model at the meta layer that is an adaptive random forest and a tree classifier

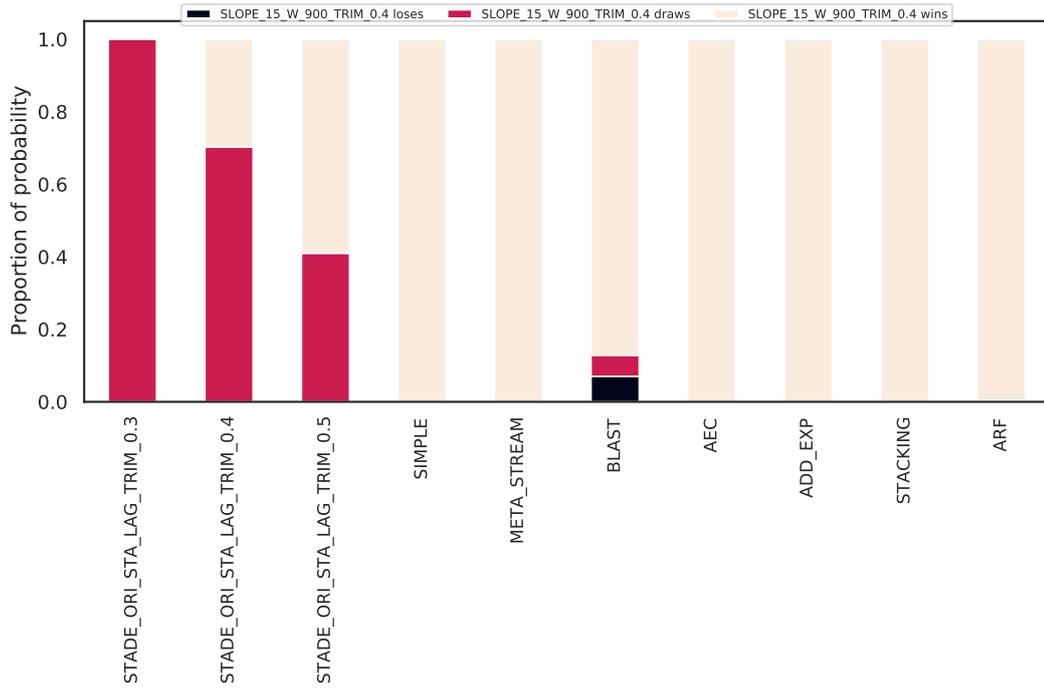


Figure 3.17: Proportion of probability of best trimming DES method winning/drawing/losing according to the Bayes sign test.

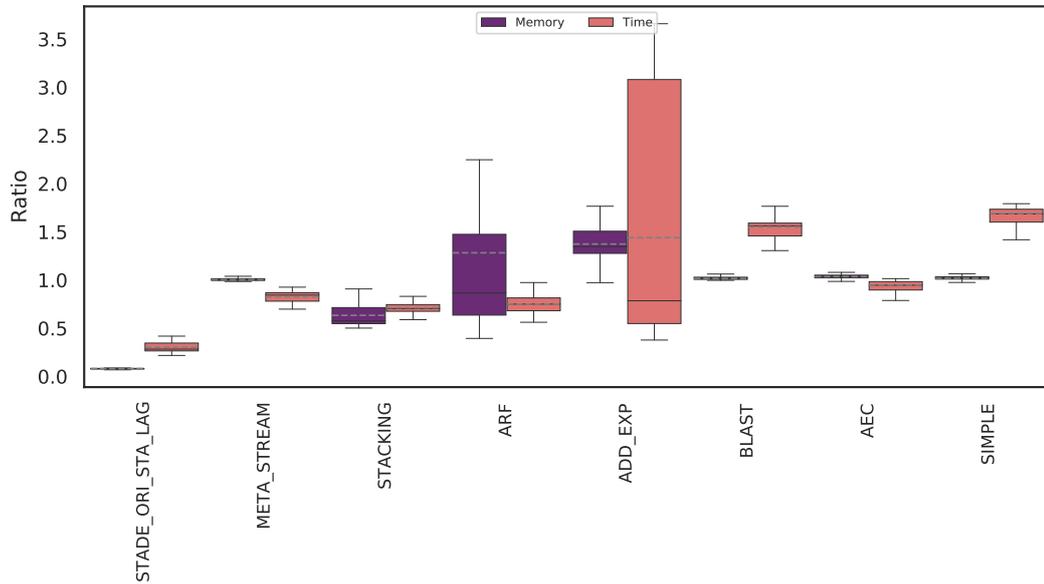


Figure 3.18: Boxplot illustrating the distribution of the ratio of the memory (left) and time (right) required by SLOPE ( $w = 900, \alpha = 0.4, k = 15$ ) relative to STADE and state-of-the-art dynamic ensembles.

respectively. In fact, STADE requires more memory to maintain the models online as well as more time to process all the instances released in the stream including additional training and testing time of meta-models (arbiters). Particularly, the highest average memory ratio

is recorded for ARF and AddExp. Principally, ARF is based on an informed adaptation strategy that keeps replacing low performing trees and replaces them with background trees trained on more recent instances which leads to lower memory requirements. On the other hand, AddExp maintains a variable size ensemble using the additive strategy to include new models and discard the worst performing one. Besides, SLOPE requires less time in average compared to ARF that combines all the available base trees and monitors their respective predictive performance. SLOPE scales better in average (time and memory) compared to other meta-learning based ensembles namely META-STREAM and STACKING that include an additional single meta-model to predict the best single or forecast combination. The meta-model meets the streaming setting computation requirements and leads to a lightweight increase in resources requirements. Likewise, BLAST and AEC, that are windowing DES approaches, exhibit the same pattern and have similar memory cost.

Overall the proposed SLOPE based DES methods that leverage both temporal criterion as well as feature based similarity to the test instance to compute the local predictive performance of each individual model in the pool reveal very promising results in terms of predictive performance as well as computational cost relative to other windowing and meta-learning based DES methods along with other state-of-the-art dynamic ensembles. SLOPE performance can be further improved by using additional meta-features as described in Section 3.1.2.1 and other distance metrics that are more appropriate to temporally dependent data such as DTW to determine the local region of competence.

### 3.3.5 EVALUATING EXPERTS SELECTION

The second part of the experimental study addresses different experts' selection approaches described in Section 3.2 and compares to the best TRIM-based DES methods that select a fixed ratio  $\alpha$  of the best performing base models within the pool  $M$ . Similarly, we conduct an empirical experimental study and compare methods in terms of average rank according to the RMSE loss for each of the SW, SLOPE and meta-learning based DES. The goal is to determine if any of the proposed selection approaches is able to outperform the simple trimming approach.

#### 3.3.5.1 Abstaining Vs. Trimming

In this section, we compare TRIM based DES methods (using SW, SLOPE and STADE) against the proposed abstaining policy (ABS) that forces non-confident base-models to abstain from contributing to the aggregation of the final prediction. We study the effect of different Reliability Estimation Scores (RES) discussed in Section 3.2.2.1 namely S-CONFINE and CNK as well as the adaptive abstaining strategies described in Section 3.2.2.2. The SUM and PRO labels stand for the additive and multiplicative adjustment factor based abstaining policy respectively whereas AUT stands for the drift based approach described in Section 3.2.2.2. We set the performance threshold  $\theta = 0.2$  and an update step  $s = 0.01$  to be used in the adaptive abstaining approaches (Algorithm 1). The drift-based abstaining uses the ADWIN detector with  $\delta_w = 0.01$  (resp.  $\delta_d = 0.001$ ) to trigger warnings (resp. drifts) in the overall predictive performance.

Results depicted in Figures 3.21a, 3.21b, and 3.21c show the distribution of the rank and respective standard deviation of the SW, SLOPE and STADE trimming and abstaining

variants respectively. Overall, the outcome shows that abstaining-based (ABS) DES methods perform worse in average relative to their trimming counterparts for SW, SLOPE and STADE methods and other state-of-the-art dynamic ensembles such as BLAST. Besides, using the CONFINE and CNK (Section 3.2.2.1) along with the predicted error as reliability estimators did not lead to the desired results that is enhancing overall predictive performance except for SW with  $w = 200$ . Besides, the proposed adaptive abstaining did not help improving the predictive performance. The same pattern repeats itself for SW, SLOPE and STADE approaches. Some abstaining-based DES methods could surpass in average existing dynamic ensemble methods such as META-STREAM, AddExp and AEC. The performance of the abstaining policy is closely related to the value of the threshold parameter  $\theta$  that highly depends on the data and task at hand and often requires human expertise to set the adequate value.

### 3.3.5.2 Randomized Vs. Trimming

We compare the predictive performance of randomized DES discussed in Section 3.2.3 relative to trimming approaches. Similarly, we compare the different methods in terms of average rank according to the loss incurred for each of the 30 temporal data streams. Experiments include the Bernoulli (BERN) as well as the BETA based DES using varying selection ratios. Similarly, randomized selection methods perform worse relative to their trimming counterparts for SW, SLOPE and STADE approaches. Overall, the BETA-based randomized selection shows promising performance compared to BERN particularly for selection ratio values that are in the middle of the search distribution. Promoting diversity through randomized selection did not achieve the expected results on the predictive performance.

Future work can take advantage of the theoretical background and guarantees of the Thompson sampling approach and the multi armed bandit in non stationary environments in order to improve the BETA randomized selection. Thompson sampling is a very well known algorithm for online decision problems to study the exploration/exploitation trade-off in sequential decision making. The goal is to find a balance between exploiting what is known to maximize immediate performance and investing to accumulate new information that may improve future performance [139]. In fact, model or ensemble selection for streaming data can be cast as a regret-minimization problem in non-stationary environments using fading factor or windowing to update the parameters of prior distribution to systematically reduce the effect of outdated observations.

### 3.3.5.3 Trade-off MMR Vs. Trimming

We presented in Algorithm 4 a trade-off selection approach that leverages both predictive performance and pairwise diversity using correlation in order to select a subset of experts that is as accurate and as diverse as possible. The goal is to increase diversity among selected models while maintaining a reasonable individual error. The method is inspired from the Maximal Marginal Relevance (MMR) in the Information Retrieval field for documents re-ranking. Figure 3.21 depicts the distribution of the rank and respective standard deviation of MMR-DES relative to their trimming counterparts using SW, SLOPE and STADE based DES methods and state-of-the-art methods. different values of the trade-off parameter  $\in \{0.9, 0.8, 0.5\}$  and selection ratio values  $\{0.2, 0.5, 0.8\}$ .

MMR approach using  $\lambda = 0.5$  that is a balanced trade-off between diversity and per-

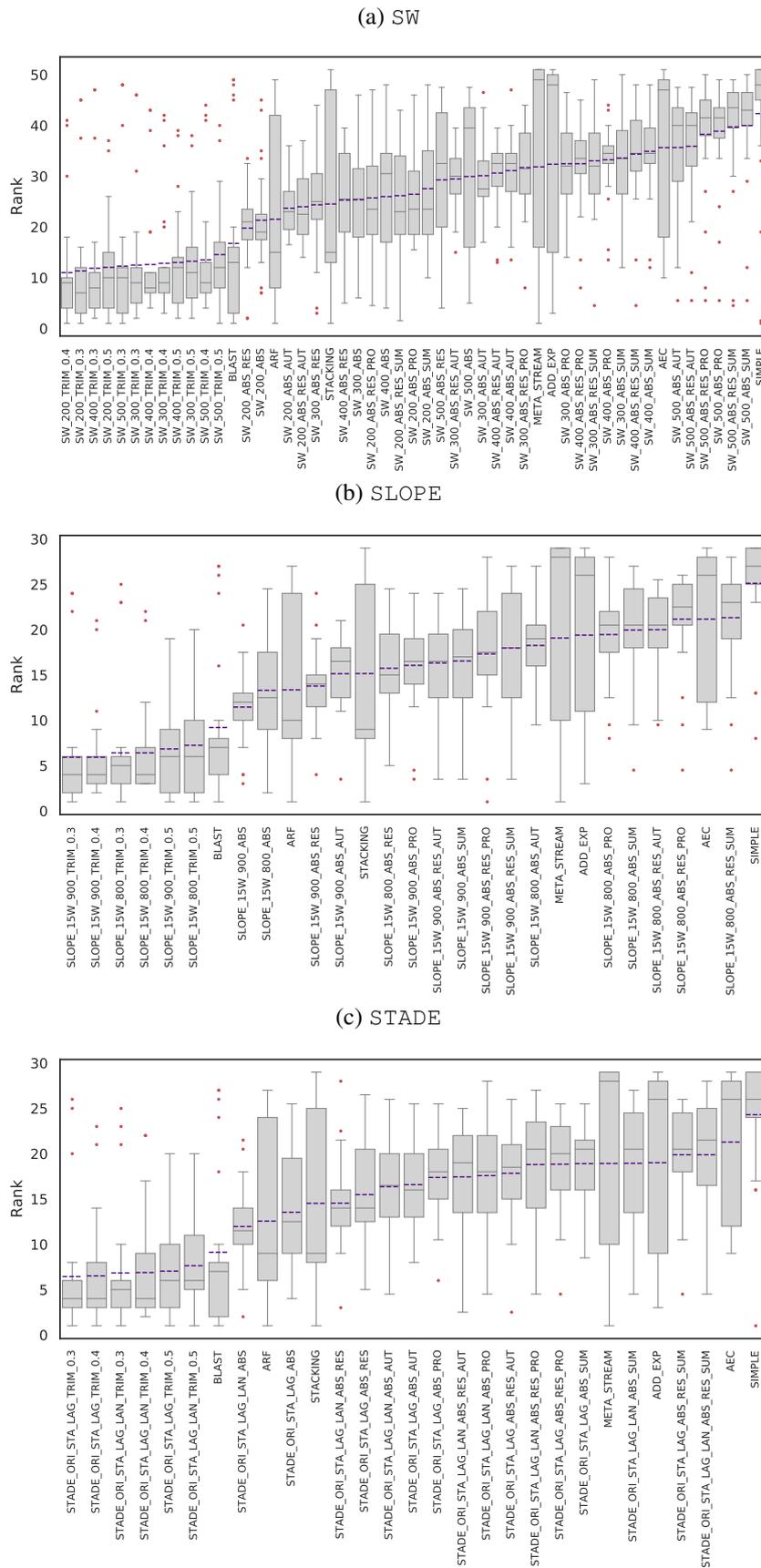


Figure 3.19: Boxplots illustrating the distribution of the rank and respective standard deviation of trimming and abstaining (ABS) based DES methods for SW, SLOPE and STADE approaches in terms of RMSE.

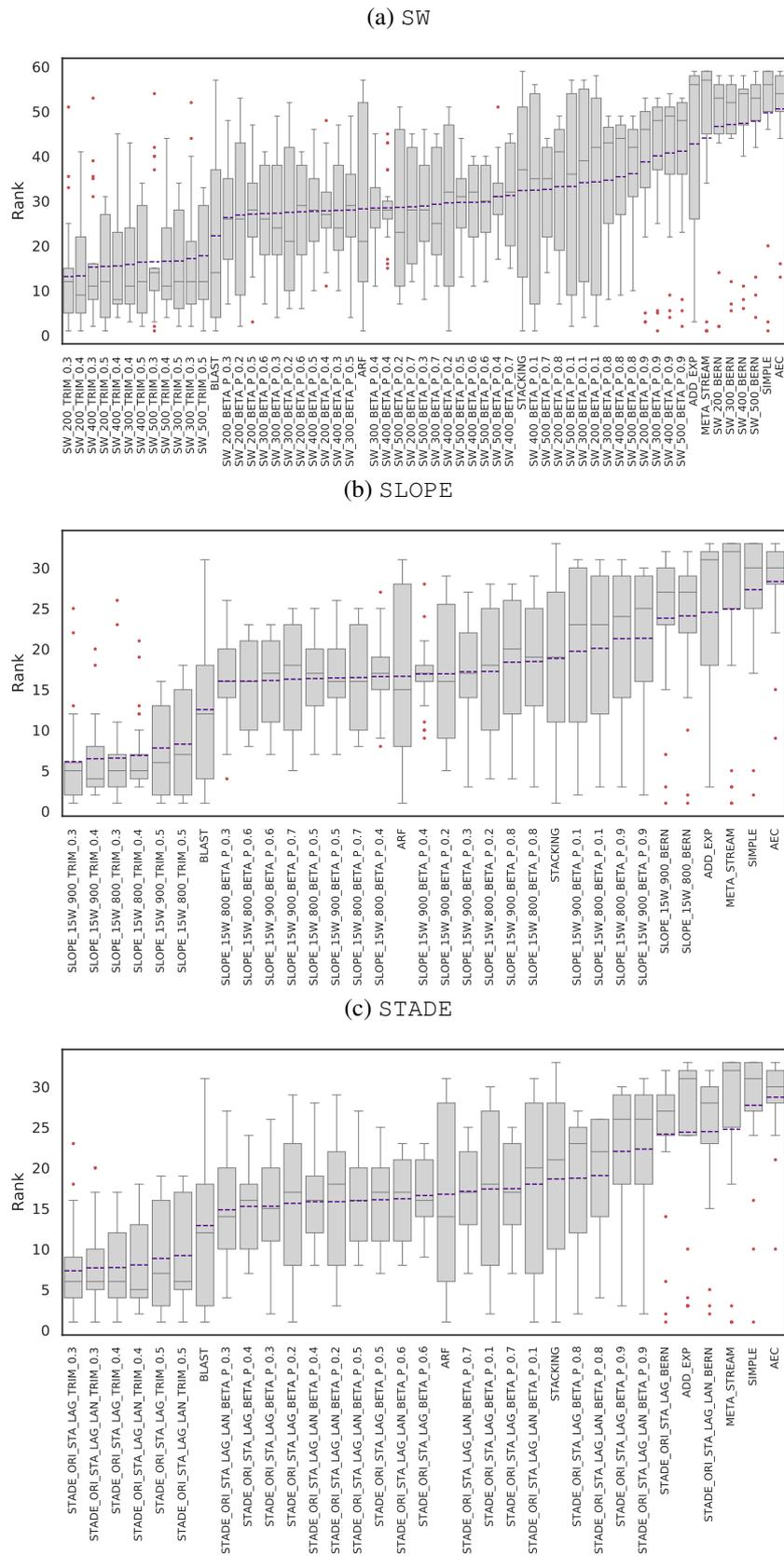


Figure 3.20: Boxplots illustrating the distribution of the rank and respective standard deviation of trimming and randomized based DES methods for SW, SLOPE and STADE approaches in terms of RMSE.

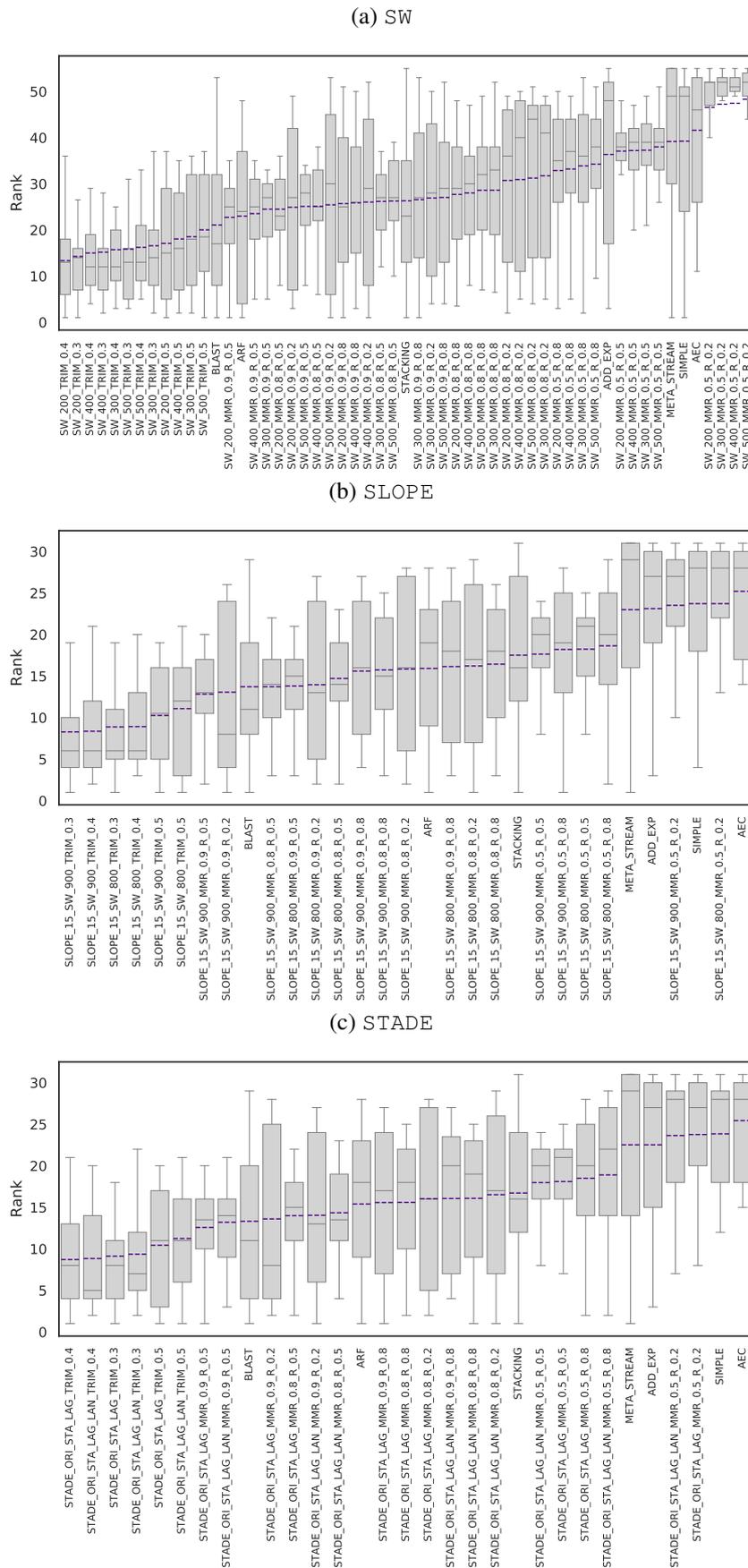


Figure 3.21: Boxplots illustrating the distribution of the rank and respective standard deviation of trimming and MMR based DES methods for SW, SLOPE and STADE approaches in terms of RMSE.

formance performs worse relative to higher  $\lambda$  values  $\{0.8, 0.9\}$  that tend to grant more importance to the predictive performance. Controlling the performance-diversity trade-off using the  $\lambda$  factor did not achieve the expected results, that is surpassing simpler trimming methods (TRIM). On the other hand, some MMR methods could outperform in average rank existing methods such as BLAST and ARF. Despite the evidence of the benefits of diversity within ensemble methods, MMR-based selection using correlation did not render a positive effect on overall predictive performance. In fact, incremental models that are trained on the same instances of the stream and for the same task are highly correlated and using Pearson's correlation as a measure of similarity can be limited in this case. Future work includes information theoretic based diversity measures to quantify the mutual information between two base-models or more (multivariate analysis) [110].

### 3.4 SUMMARY

DES methods are based on the assumption that heterogeneous forecasting models have varying expertise along a temporal data stream. Their goal is to accurately select on the fly for each incoming test instance a committee of experts deemed to be the best, and aggregate their predictions provided some weighting mechanism to emphasize on the most competent ones. Dynamically selecting and combining forecasters lead to better overall predictive performance, particularly in the streaming setting where the characteristics of the data are very likely to evolve due to concept drift.

In this chapter we introduced several DES methods for streaming data. We presented SLOPE a novel performance estimation approach based on local predictive performance using a combined criterion. In fact SLOPE brings together the temporal criteria to emphasize on the most recent instances using a sliding window and a feature-based similarity using the  $k$  nearest neighbors to the test instance. SLOPE has proved to be more effective compared to traditional SW based methods using trimming selection approach at the cost of lightweight additional computational cost in terms of memory and time. Subsequently, we introduced STADE, a streaming meta-learning based DES approach following the arbitration framework [38]. We focused on a set of meta-features including statistical, landmarking and stream-specific information to train the meta-models to link between the characteristics of the data and the predictive performance of individual base-models. SLOPE method demonstrated encouraging results in terms of predictive performance compared to other meta-learning based DES methods as well as state-of-the-art dynamic ensembles. Besides, SLOPE scales better in terms of computational cost (memory and time) compared to STADE meta-learning DES.

The second part of the chapter focused on novel methods to select a committee of experts and compare to trimming that selects a fixed ratio  $\alpha$ . We introduced an adaptive abstaining policy that forces less confident forecasters to decline from contributing to the final output. Subsequently, we presented randomized selection approach where base-models with lower expected error are more likely to be selected in the committee. Selecting the best models leads to the best predictive performance overall. However, concept drift may happen anywhere in the stream. Thus, it might be interesting, from time to time, to select other base-models with higher errors and inversely, prune the ones with low predicted errors. Finally,

we presented an MMR-based selection using a combined criterion as a trade-off between individual models' performance and ensemble diversity. The empirical results demonstrated that simple selection methods such as trimming often perform better on average relative to more sophisticated selection methods.

## Dynamic Ensemble using Multi-Target Regression

Meta-learning based dynamic ensemble selection and combination methods have demonstrated highly competitive predictive performance for temporal data streams forecasting. Meta-Learning based DES methods work by learning the behavior of individual models in the pool in order to predict their future performance and accordingly select and combine the best performing ones. As mentioned in the previous chapters of the thesis, DES methods can be divided into two categories: *individual-based* and *group-based* [26]. The first category estimates the competence and selects each model in the pool separately whereas the second one explicitly captures dependencies among individual models. This chapter presents our second contribution that is a group-based DES approach using meta-learning that exploits the dependency information among base-models when learning and predicting their respective behavior. We formulate the DES task as a Multi-Target Regression (MTR) problem where the goal is to simultaneously predict individual models expected performance according to the test data.

The remainder of the chapter is organized as follows, Section 4.1 introduces the DES problem formulation using multi-output learning and details the MTR based DES methods for temporal data streams forecasting. Section 4.2 describes different MTR approaches to tackle the DES task for evolving temporal data streams. Section 4.3 highlights the experimental setup used to compare the proposed MTR-based DES against other state-of-the-art DES methods. Lastly, Section 4.4 concludes the chapter with a summary of our contributions and discuss encountered challenges and future directions.

## 4.1 DES USING MULTI-OUTPUT LEARNING

Meta-learning based Dynamic Ensemble Selection (DES) methods have demonstrated highly competitive results in terms of predictive performance. The meta-layer is in charge of learning the behavior of each base-model in the pool according to the data at hand and subsequently predict future behavior. The goal being to select the most competent ones for each incoming test instance and combine their predictions in order to boost overall predictive performance. DES methods can be divided into two categories: *individual-based* and *group-based* [26]. In *individual-based* methods, the selection of models is achieved by estimating the competence of each model in the pool *individually* without considering its dependencies with other models. On the other hand, *group-based* methods explicitly consider models' dependencies while learning and predicting their behavior.

Arbitrated Dynamic Ensemble (ADE) is a meta-learning based DES that addresses time-series forecasting and falls within the individual-based DES methods. ADE is based on arbitration where the behavior of each base-model is learned separately by its meta-counterpart. Thus, the correlations among the base-models behavior are ignored at the risk of losing valuable information and competitive advantage. ADE was later enhanced with an additional stage where experts weights are recomputed using their pairwise correlation [41]. The idea of meta-learning approach using Multi-Output Learning [155] has emerged in order to overcome the aforementioned drawbacks and to learn and predict base-models' behavior all at once.

### 4.1.1 DES VIA MULTI-LABEL CLASSIFICATION

Markatopoulou, Tsoumakas, and Vlahavas [108] first suggested that the meta-learning based DES can be cast as a Multi-Label Classification (MLC) task. It addresses classification task, where the goal is to learn to predict the subset of classifiers, among the pool of base classifiers, that are likely to correctly classify a given test instance. The proposed framework comprises a meta-layer where the feature space training set is the same as the original feature space, whereas the label space contains one label for each classifier. The multi-label training examples are computed based on the ability of each classifier to correctly classify each of the original examples. The meta-model assigns a positive label (1) if the given classifier is expected to predict the right class for the given instance otherwise the label is negative (0). Conversely to arbitration that learns the behavior of each base model separately, the MLC based DES takes advantage of the essence of multi-output-learning to explicitly take into account the correlation among individual models. In fact, the problem of predicting whether a base-model will correctly predict a given unlabeled instance can be addressed collectively instead of separately. All base-models are trained on the same data and predict on the same test instances, thus, the behavior of one model is likely to provide valuable information on the behavior of other models. Following this approach, Narassiguin, Elghazel, and Aussem [114] showed that the dependencies of the errors made by each model in the ensemble have to be exploited. They used Probabilistic Classifier Chains to cast the DES classification task for batch learning.

However, MLC approaches remain very challenging when it comes to ensembles of forecasters. In fact, it is not clear how to determine if a forecaster has made a correct or wrong predictions. Nevertheless, as an alternative, we can quantify the loss incurred using

one of the evaluation metrics discussed in Section 2.1.7.2. In this chapter we propose a novel meta-learning based DES approach using Multi-Target Regression (MTR). MTR is a multi-output-learning framework that deals with real valued targets. The goal is to simultaneously predict the error of each ensemble component and accordingly select and combine the best performing ones and combine their predictions. The MTR-DES approach has the advantage of explicitly capturing models' dependency information at an early stage in order to learn and predict their respective behavior. In fact, ADE [41] and DEMSC [140] addressed the dependency issue among models' predictions by introducing an additional stage that aims at involving model's pairwise correlation in their performance estimation. This additional step increases the computational complexity and scales with the number of models in the pool.

#### 4.1.2 DES VIA MULTI-TARGET REGRESSION

Let's reconsider the described temporal one-step-ahead forecasting problem using dynamic ensembles.  $S$  is a stream of instances  $S = \{y_1, y_2, \dots, y_t \dots\}$  where  $y_t$  is the observed value at time  $t$  and  $x_t$  it's  $p$  dimensional associated feature vector. We recall that our approach is based on auto-regressive modelling approach. This means that the  $x_t$  comprises the  $p$  previous observed values  $x_t = \langle y_{t-1}, y_{t-2}, \dots, y_{t-p} \rangle$  used as a feature vector to model the temporal dependency that characterizes temporal data streams. We use a meta-learning based DES with a pool  $M = \{M^1, M^2, \dots, M^m\}$  of heterogeneous base-models and the goal is to learn their respective behavior and predict future error values. In fact, at time  $t$ , the meta-layer is in charge of predicting  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$  where  $\hat{e}_{t+1}^i$  is the loss that the base-model  $M^i$  is expected to incur when predicting next value to be observed in the stream  $\hat{y}_{t+1}^i$  compared to the true value  $y_{t+1}$ . The most competent base-models are then selected in a committee  $M^c$  and their predictions combined to act as the ensemble's prediction  $\hat{y}_{t+1}$ . Base models' error in DES are very likely to exhibit compound dependencies and must be considered at the meta-level.

That said, the meta-layer error prediction task can be cast as a multi-target regression (MTR) problem. The learning task is to simultaneously predict multiple continuous targets that are assumed to be related. The assumption holds in the context ensemble methods for streaming data since base-models learn on the same data to predict the same one-step-ahead forecast. More formally, Borchani et al. [16] define the task as learning a multi-target regression model consisting of finding a function  $h$  that assigns to each instance, given by the vector  $x_t$  and comprising  $p$  features, a vector  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$  of  $m$  continuous valued targets.

$$\begin{aligned} h : \Omega_{X^1} \times \dots \times \Omega_{X^p} &\rightarrow \Omega_{E^1} \times \dots \times \Omega_{E^m} \\ x = (x^1, \dots, x^p) &\rightarrow e = (e^1, \dots, e^m), \end{aligned} \quad (4.1)$$

where  $\Omega_{X^i}$  and  $\Omega_{E^j}$  stand for the sample spaces of each predictive variable for  $i \in \{1, 2, \dots, p\}$  and each target variable  $j \in \{1, 2, \dots, m\}$ , respectively. The learned model will be used to simultaneously predict the value  $e_{t+1}$  of the new incoming unlabeled instance  $x_{t+1}$ .

Models' dependencies have been considered in previous works [41, 140] by introducing an additional stage prior to aggregation where selected models' weights are recomputed to

involve their respective correlations. Conversely, we consider models' dependencies at an early stage while predicting their performance for a given test instance  $x_{t+1}$  without any additional stage. In fact, multi-target regression methods have proven to yield a better predictive performance, in general, when compared against the single-output methods. MTR methods provide as well the means to effectively capture base-models relationship by considering not only the underlying relationships between the features and the corresponding targets but also the relationships between the targets, guaranteeing thereby a better representation and interpretability. A further advantage of the multi-target approaches is that they often produce simpler models and computationally efficient.

We provide throughout next section a survey on state-of-the-art MTR learning methods that meet the streaming requirements discussed in Section 2.1.3.

## 4.2 STREAMING MULTI-TARGET REGRESSION

MTR methods can be categorized into two main approaches: *problem transformation* and *algorithm adaptation* methods [16].

### 4.2.1 PROBLEM TRANSFORMATION METHODS

These methods are based on transforming the MTR problem into several single-target problems, and building a separate model for each target. Finally all the individual predictions are concatenated.

#### 4.2.1.1 Binary Relevance

Binary Relevance (BR) or Single-Target [170], is a simple approach that decomposes the MTR problem into  $m$  independent single-target models. Each model is trained on a transformed data stream:  $\mathcal{S}_i = \{(x_1^1, e_1^i), \dots, (x_t^1, e_t^i)\}$ ,  $i \in \{1, \dots, m\}$  to predict at time  $t$  the value of a single target  $\hat{e}_{t+1}^i$ . All the individual predictions are then concatenated to produce a final  $m$  dimensional prediction vector  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$ . Clearly, ADE [38] that uses arbitration at the meta-level falls within this category of MTR methods. The main drawback of these methods is that the relationships among the targets are ignored, and the targets are predicted independently, which may affect the overall quality of the predictions.

#### 4.2.1.2 MTR-Stacking

Multi-Target Regressor Stacking (MTRS) takes advantage of the *stacked generalization* [164] approach involving a two-layered learning process. In addition to the first layer of single-target models (BR), a second set of  $m$  meta-models are learned over the first layer models predictions, one for each target  $e^i, i \in \{1, \dots, m\}$ . MTRS is based on the idea that a second-stage model is able to correct the predictions of a first-stage model by using information about the predictions of other first-stage models. Each meta-model is learned on a transformed stream where  $S^{*i} = \{(x_1^{*i}, e_1^i), \dots, (x_t^{*i}, e_t^i)\}$ , where  $x_t^{*i} = (x_t^1, \dots, x_t^p, \hat{e}_t^1, \dots, \hat{e}_t^m)$  is an input instance consisting of the original input instance  $x$  augmented with the predictions of their target variables yielded by the first-stage single-target models.

#### 4.2.1.3 Regressor Chains

Regressor Chains (RC) is inspired by the Multi-Label Classifiers Chain [127] based on the idea of chaining single-target models. The training of RC consists of selecting a random

chain (order) of  $m$  regression models, one for each target. The features of each link in the chain are augmented with the predictions of all previous links. Unlike BR method, the RC method models targets dependency, however, it might be sensitive to the selected order. This issue can be mitigated using an ensemble with different random chain ordering and aggregating their outputs, this is referred to as Ensemble Regressor Chains (ERC).

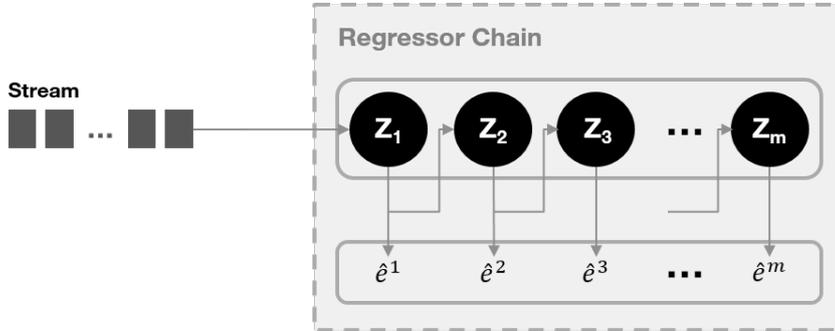


Figure 4.1: Regressor Chain

Problem transformation methods are very simple to implement and their adaptation for the streaming setting is straightforward. In fact, one can use any incremental and adaptive regression model such as FIMT-DD [84] as a base estimator.

#### 4.2.2 ALGORITHM ADAPTATION

Algorithm adaptation methods are based on the idea of simultaneously predicting all the targets using a single model that can capture the underlying dependencies and internal relationships between them. These methods have the advantage of being much smaller in size compared to the total size of the individual single-target models for all variables. Second, they better identify the dependencies between the different targets to achieve better predictive performance especially when the targets are correlated [24, 91, 144]. In this section we focus on multi-target regression trees given that they were extensively studied for the streaming setting. However, algorithm adaptation methods are not limited to trees, we point the reader to the survey of Borchani et al. [16] for further details on other methods that were proposed to tradition batch learning.

##### 4.2.2.1 Multi-Target Regression Trees

Multi-Target Regression Trees (MTRT) are regression trees that can simultaneously predict multiple real-valued target variables [23]. MTRT trees have the advantage of being much smaller than the total size of BR regression tree. Besides, they better identify the dependencies between the target variables. MTRT approaches were widely studied for the tradition batch setting, we refer to the detailed survey [16]. In our work we focus on MTRT for streaming data only.

FIMT-MT is an incremental [83] MTRT that addresses multi-target time-changing data streams. The algorithm extends the incremental single-target model tree proposed in [82] by adopting the principles of Predictive Clustering Tree (PCT) [15] in the split selection criterion and works under the assumption that grouping similar examples together, can improve the predictive performance of the model. FIMT-MT maintains incremental perceptrons at the

leaves of the tree, one for each target. The Hoeffding bound [79] is used to perform a split only when enough statistical evidence has been accumulated in favor of a particular splitting test.

### Incremental Structured Output Prediction Tree

The *i*SOUP [118], for incremental Structured Output Prediction Tree, is an extension of the *FIMT-MT* that uses an adaptive multi-target model at each leaf instead of a single linear perceptron. The *i*SOUP maintains in one hand, a multi-target perceptron and on the other hand the multi-target mean predictor that computes the prediction as the mean value of each of the targets observed at a given leaf. The fading Mean Absolute Errors  $fMAE_{Perceptron}^i$  and  $fMAE_{Mean}^i$  are monitored for each target. The decision about which local model to use is made for each target variable separately. The fading mean absolute error is computed as:

$$fMAE_{learner}^i(n) = \frac{\sum_{j=1}^n 0.95^{(n-j)} |\hat{e}_j^i - e_j^i|}{\sum_{j=1}^n 0.95^{(n-j)}} \quad (4.2)$$

where  $n$  is the number of instances observed in a leaf,  $\hat{e}_j^i$  and  $e_j^i$  are the predicted and the true value for the  $j$ -th example, respectively, and  $learner \in \{Perceptron, Mean\}$ . The fading error emphasizes on more recent examples to cope with concept drift. The model with the lowest error is used for each target. Therefore, the  $m$  dimensional output vector can contain predictions from both the *Perceptron* and the *Mean* predictors.

### Single Stacked Target Hoeffding Tree

The Single Stacked Target Hoeffding Tree (*SST-HT*) [109] takes advantage of all previously mentioned *MTRT* approaches but better exploits targets' inter-correlation by using an additional stacked model at the leaves. Previously studied *MTRT* methods used as many linear models as the number of targets and the predictions  $\hat{e}_t$  are computed separately, only considering the original features dependencies. *SST-HT*, in turn, adds a stacked perceptron to combine and enhance the predictions and adopts the same fading error approach to select the best performing local model for each target.

Table 4.1: Multi-Target Regression methods

Category	Method
Problem Transformation	BR Binary Relevance [170]
	MTRS Stacked Multi-Target
	RC Regressor Chain [127]
Algorithm adaptation	<i>i</i> SOUP [118]
	<i>SST-HT</i> [109]

## 4.3 EMPIRICAL EXPERIMENTS

### 4.3.1 EXPERIMENTAL DESIGN

In this chapter, we follow the experimental design described in 3.3.1, where the *DES* approach is based on heterogeneous ensembles and individual models' predictive performance is

assessed using the `sMAPE` loss metric. This means that the meta-layer is in charge of predicting the `sMAPE` loss that each model  $M^i \in M$  will incur for predicting  $\hat{y}_{t+1}^i$  based on the feature vector of the instance at hand  $x_{t+1}$ . Similarly, all methods were evaluated in the prequential setting and ranked according to their respective `RMSE` loss, the lower the loss the better the rank. In fact, we analyze the distribution of the rank and respective standard deviation as well as the distribution of the percentual difference relative to the best performing method in order to quantify the gain or loss in the overall predictive performance. We conduct the Bayesian analysis as described in Section 3.3.2.3 in order to assess the significance of the results above. Besides, we discuss the computational cost in terms of time and memory of all the `MTR` approaches.

According to the empirical study conducted in Chapter 3, present experiments include trimming (`TRIM`) selection only with  $\alpha \in \{0.3, 0.4\}$  due to the superior predictive performance relative to other selection methods. We focus on problem-transformation and algorithm adaptation `MTR` methods as discussed in Section 4.2. We note that, `STADE` method presented in Chapter 3 that uses arbitration is considered as a Binary-Relevance (`BR`) `MTR` approach.

### 4.3.2 COMPARING META-LAYER

First we analyze the predictive performance of the meta-layer as a multi-target regression problem using the Average Root Mean Squared Error (`aRMSE`) [16] as described in Equation 4.3. In fact the output of the meta-layer for each test instance  $x_t$  is a vector of size  $m$  and hence can be evaluated as a multi-target task. The goal is to evaluate how good is the meta-layer in learning and predicting the actual base-models' errors for each incoming instance in the stream using Equation 4.3

$$aRMSE = \frac{1}{m} \sum_{i=1}^m \sqrt{\frac{\sum_{t=1}^n (e_t^i - \hat{e}_t^i)^2}{n}} \quad (4.3)$$

where  $m$  is the size of the ensemble that translates to an  $m$  dimensional vector of errors, and  $e_t^i$  (resp.  $\hat{e}_t^i$ ) is the actual (resp. predicted) error of model  $M^i$  at time  $t$ .

Similarly to `STADE` that is considered as a Binary Relevance (`BR`) `MTR` task, all other problem transformation methods such as the (`MTRS` and `RC`) are implemented using Hoeffding regression trees as base estimators. Experiments do not include `ERC` due to the excessive computational cost that scales with the number of random chains included in the ensemble. Besides, we evaluate the use of additional meta-features (statistical and landmarking discussed in Section 3.1.2.1) along with the original feature vector  $x_t$  with `MTR` meta-learning methods.

Figure 4.2 illustrates the distribution of the rank and respective standard deviation of the `MTR` meta-layers methods according to their `aRMSE` on the 30 temporal data streams. A method ranked #1 stands for the method with the lowest `aRMSE`. Results show that `MTR` methods that explicitly consider individual models' dependencies do not outperform binary relevance approaches that learn the behavior of each model individually. In fact, `STADE` that uses arbitration (separate meta-model for each base-model) along with the proposed meta-features (statistical, lagged errors and landmarking) surpass all the other methods in

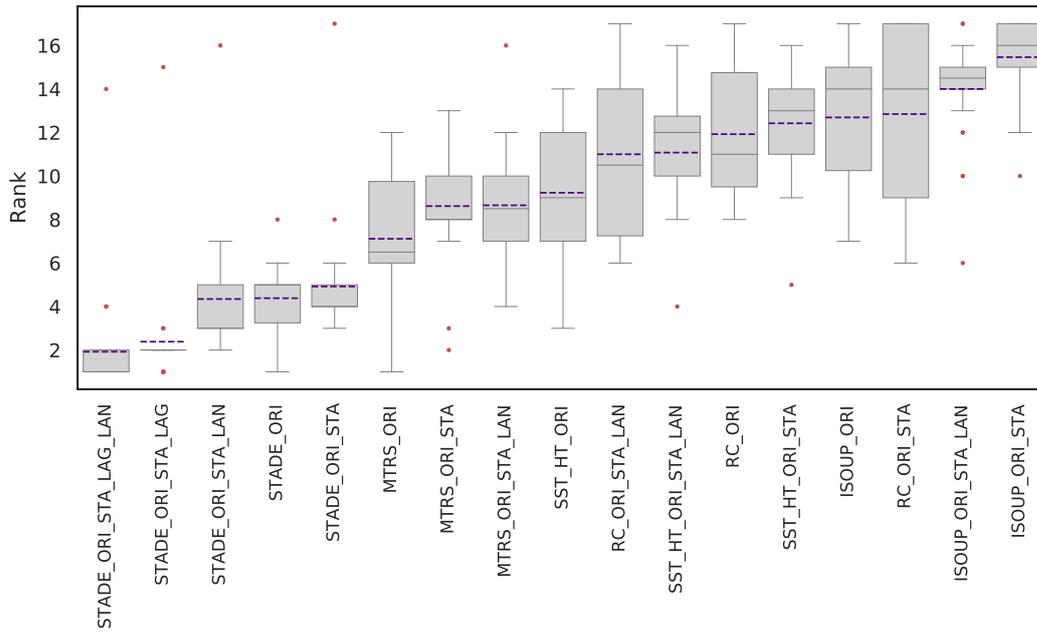


Figure 4.2: Boxplot illustrating the distribution of the rank and respective of MTR based meta-layer in terms of aRMSE.

estimating the actual loss of individual models. Besides, the problem transformation methods MTRS approach performs better in average compared to other methods (except STADE). Also, including statistical and landmarking meta-features along with problem transformation (MTRS and RC) and algorithm adaptation (iSOUP and SST-HT) MTR methods often deteriorate overall meta-layer predictive performance conversely to STADE.

We measure the magnitude of change in the predictive performance that translates to the improvement or deterioration achieved by the best performing method in average in terms of aRMSE relative to other methods. Figure 4.3 depicts the distribution of the percentual difference where a value  $p$  below (resp. above) 0 stands for an improvement (resp. deterioration) of  $|p|\%$ . In general, STADE\_STA\_LAG\_LAN tends to improve in average all the other methods in terms of aRMSE that reflects the ability of the meta-layer in predicting the actual error of individual base-models. Significant improvement is achieved against the MTR methods that explicitly consider base-models dependencies and more particularly iSOUP that is an algorithm adaptation method using tree models. On the other hand, moderate improvement is highlighted against other STADE variants. Experiments demonstrate that STADE meta-learning using additional meta-features described in Table 3.1 achieve competitive results and are able to learn and predict the behavior of base models in the pool  $M$ .

### 4.3.3 COMPARING MTR-BASED DES METHODS

In this section we compare the MTR-based DES methods to other windowing and meta-learning based DES methods as well as state-of-the-art dynamic ensembles. According to the results depicted in Chapter 3, experiments will cover trimming methods only where a fixed ratio of the best performing base-model are included in the committee. In fact,

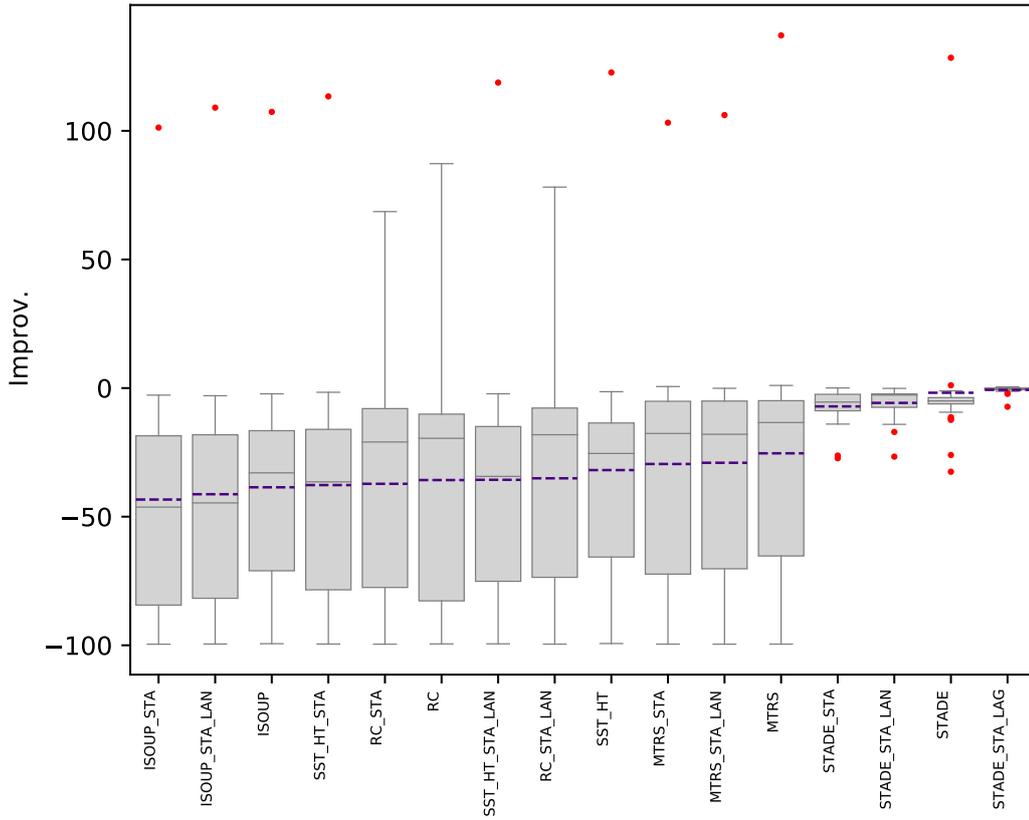


Figure 4.3: Boxplot illustrating the distribution of the percentual difference and respective of of the best performing MTR based meta-layer in average (STADE\_STA\_LAG\_LAN) in terms of aRMSE.

simply pruning a percentage of the worst performing base-models leads to the best results on average in terms of predictive performance overall. Similarly, we compare methods on the 30 temporal data streams using the RMSE loss function in a prequential evaluation setting. Figure 4.4 details the distribution of the rank and respective standard deviation of MTR-based DES methods against other state of the art dynamic ensembles.

Results attest to their superiority of STADE when selection is applied relative to other MTR approaches. The best performing MTR methods, except the BR approaches, is MTRS that is also based on a two-layers learning framework where the second layer leverages the predictions of the first one and the original feature vector in order to output the final  $m$ -dimensional vector of errors. Besides, algorithm adaptation methods (iSOUP and SST-HT) perform worse in average relative to problem transformation ones. Figure 4.5 depicts the distribution of the percentual difference of the different MTR-DES methods using trimming (TRIM) relative to the best performing one on average that is STADE\_STA\_LAG with  $\alpha = 0.3$  according to the results of Figure 4.4. This comparison does not include other STADE variants as this was discussed in Chapter 3.

The STADE method achieves mild improvement compared to MTRS while values are higher for iSOUP and SST-HT variants yet under 5% in average. In order to analyze the significance of the results stemming from the rank analysis, we use the Bayes sign test to

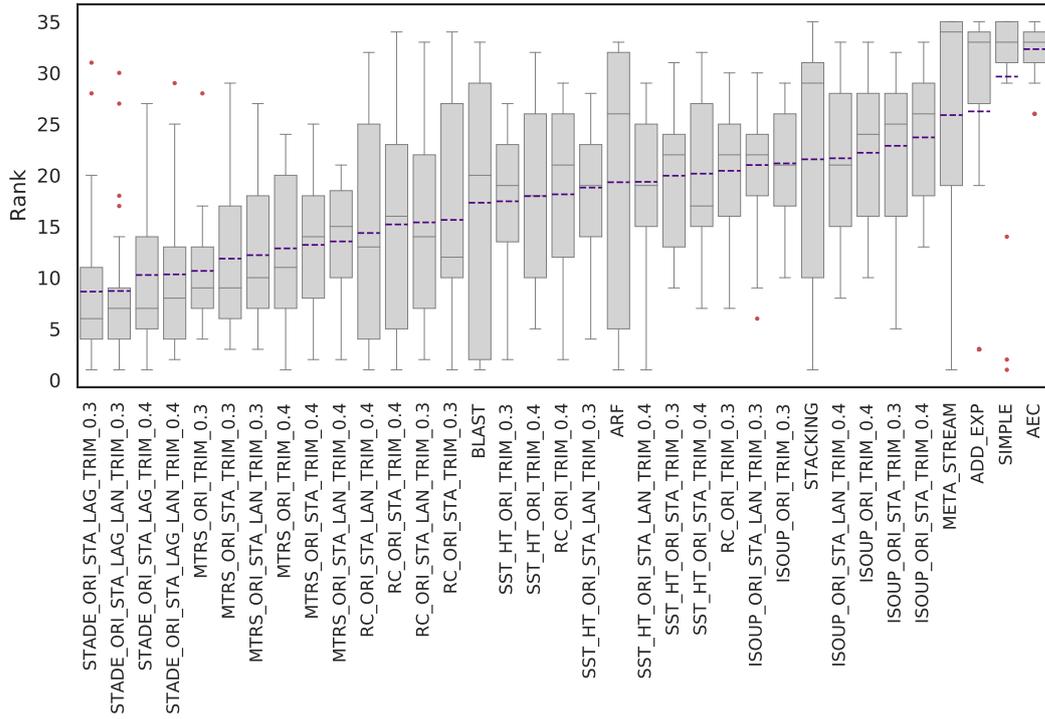


Figure 4.4: Boxplot illustrating the distribution of the rank and respective standard deviation of MTR-based DES methods and state of the art methods in terms of RMSE

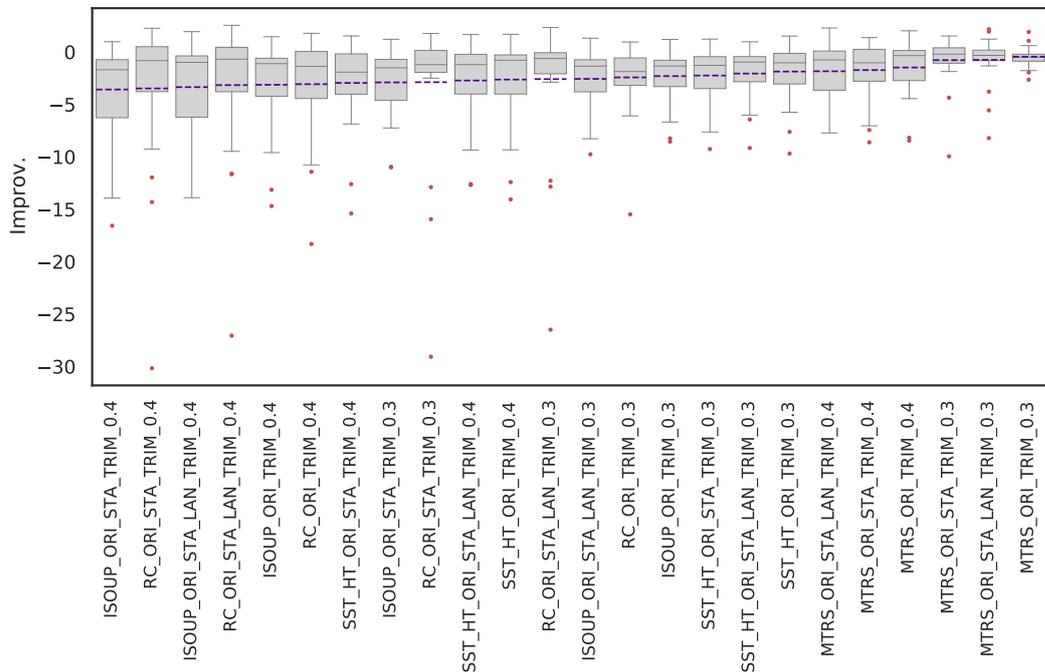


Figure 4.5: Boxplot illustrating the distribution of the percentage difference and respective of the best performing MTR based (STADE\_ORI\_STA\_LAG in terms of RMSE

compare all the methods across the 30 temporal data streams against the best performing one on average rank that is STADE\_ORI\_STA\_LAG along with a Region of Practical

Equivalence (ROPE) set to  $[-0.01, 0.01]$ . We refer the reader to Chapter 3 for more details on the Bayesian analysis. Figure 4.6 illustrates the proportion of probability that the best performing STADE variant loses, draws or wins against each method represented by a separate bar. MTRS variants with similar selection ratio  $\alpha = 0.3$  have high probability to

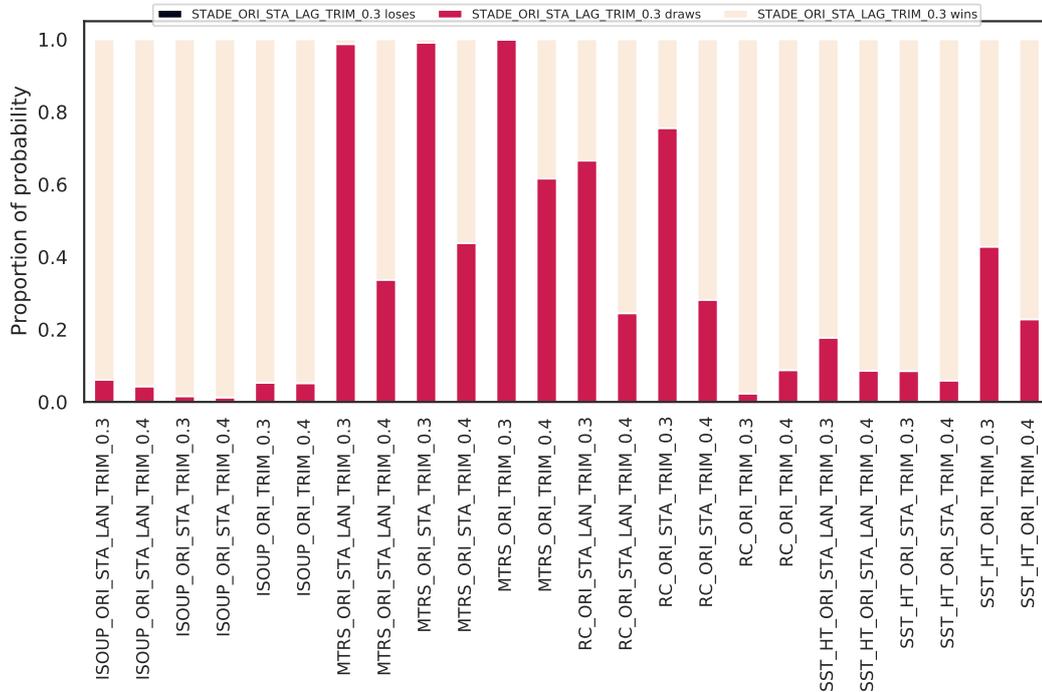


Figure 4.6: Proportion of probability of STADE\_ORI\_STA\_LAG\_LAN winning/drawing/losing against other MTR based DES and state of the art methods according to the Bayes sign test

draw with the best performing STADE approach whereas algorithm adaptation methods have very little probability to achieve a draw. Using additional statistical and landmarking meta-features along with RC enhances predictive performance.

#### 4.3.4 COMPUTATIONAL COST

Finally, we compare the MTR meta-learning approaches in terms of computational cost including memory that stands for the model size and total time required to test and train on all the instances of the stream. We analyze the distribution of the ratio between the resources required by the STADE binary relevance method to the resources required by other problem transformation and algorithm adaptation MTR methods. Figure 4.7 depicts the distribution of the ratio of the memory (left) and time (right) required by STADE using ORI\_STA\_LAG relative to other MTR methods and respective standard deviation where values below (rep. above) 1 translate to lower (rep. higher) computational cost. STADE variants are grouped by their respective set of meta-features regardless of the selection ratio as this step has negligible impact on the computational cost.

MTRS variants have similar computational requirements relative to STADE while RC variants are lightweight less demanding. On the other hand, algorithm adaptation methods namely, ISOUP and SST-HT are significantly smaller and faster relative to STADE. Indeed,

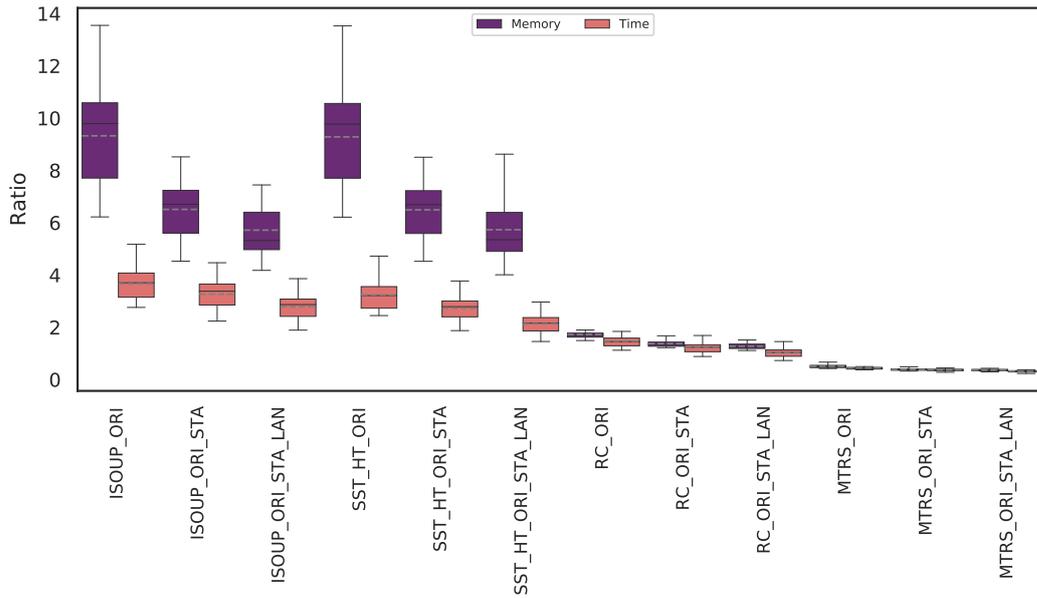


Figure 4.7: Boxplot illustrating the distribution of the ratio of the memory (left) and time (right) required STADE using ORI\_STA\_LAG relative to other MTR methods.

iSOUP and SST-HT using the original set of features only, are approximately 10 times less memory consuming in average and 4 times faster whereas the same models using additional meta-features (STA and LAN) lead to a reasonable additional computational cost.

In fact, despite their superiority in terms of predictive performance, problem transformation methods (STADE, MTRS and RS) are considerably more memory and time demanding relative to algorithm adaptation methods that is iSOUP and SST-HT. In fact, computational resource usage is a crucial aspect in the streaming setting and various application domains where both memory and time are limited. Thus, one may wish to favor lower computational cost at the expense of predictive performance in some application domains according to the sensitivity of the data and the impact of decisions.

### 4.3.5 DISCUSSION

This chapter served as an introduction to formulate the streaming meta-learning DES problem as a multi-target regression task. The idea has emerged due to the encouraging results of Multi-Label Classification (MLC) problem that was successfully used to address DES problems on the classification task. The results raise interesting discussion on several aspects that can considerably improve the proposed approach.

#### On multi-variate concept drift

The predictive performance of the meta-layer can be highly altered by concept drift in evolving data streams. There is a large collection of change detection methods for single target variables. However, the application of these methods on streaming multi-dimensional data is not straightforward. One can use a separate change detector for each target and trigger drifts separately following the problem transformation approach for MTR. Nonetheless, this approach is unable to detect changes in the pairwise behavior, particularly when targets are

correlated. In fact, incrementally training heterogeneous ensembles on the same data and for the same task renders highly correlated models. Nonetheless, when a drift occurs, the collective behavior is very likely to be altered.

Multi-variate change detectors were studied and can be used to enhance the predictive performance of MTR methods. Kuncheva [97] presented SPLM (Semi-Parametric Log-Likelihood) as its name indicates is a log-likelihood based change detection method that is able to detect changes in the variance and covariance between the variables if the means are the same. Using multi-variate change detectors for streaming data can help improving meta-layer performance at modelling individual model's behavior in non-stationary environments.

#### **On the dimensionality problem**

For a large number of base-models, problem transformation methods such as MTRS and RC face the challenging task of solving a large number of single-target problems and their complexity scales with the number of base-models. In this case, it is computationally more interesting to consider the algorithm adaptation approaches such as Multi-Target Regression Trees (MTRT). Besides, algorithm adaptation methods, are assumed to provide better interpretability according to the number of targets. In fact, it becomes bulky to analyze each model and retrieve information about its relationships to other models if the number of targets is too large.

## **4.4 SUMMARY**

In this Chapter, we reformulated the dynamic ensemble selection (DES) problem for the forecasting task in temporal data streams task as a multi-target regression (MTR) problem. Conversely to the STADE method presented in Chapter 3 that uses arbitration where a separate meta-model is in charge of each base model, we have studied several MTR approaches for streaming data that explicitly consider base-models dependencies while learning their respective behavior. We conducted an extensive empirical study on 30 temporal data streams to compare both problem transformation and algorithm adaptation methods at the meta-level. More particularly, we have compared the ability of the meta-layer to learn and predict the actual errors of each base-model in the pool using MTR-specific loss measure such as  $\alpha\text{RMSE}$ .

Results were unconvincing to what was expected from explicitly considering dependencies of the errors made by each model in the ensemble. In fact, STADE, that is considered as a binary relevance (BR) MTR algorithm outperforms all the other methods notably when additional meta-feature reflecting statistical information extracted from the data stream as well as information on the performance of other simple and fast models are used. Besides, including a number of the  $k$  previous real-error values of each base-model highly contributes to enhancing overall predictive performance of STADE.



## Model Compression

Ensemble methods that dynamically combine several models have shown superior predictive performance in data streams forecasting compared to individual models. However, ensembles are renowned for their complexity and computational costs which makes them unsuitable in cases where both resources and time are limited such as IoT applications. We investigate model compression in the streaming setting in order to overcome the aforementioned drawbacks. We train a single model to mimic the behavior of a highly performing complex ensemble while drastically reducing its complexity in terms of time and memory consumption.

The rest of the chapter is organized as follows: Section 5.1 introduces Model Compression (MC) for Dynamic Ensemble Methods (DES) to address the forecasting task in the streaming setting. Section 5.2 details our contribution for adaptive model compression on DES methods to cope with evolving temporal data streams. Experiments on both real and synthetic data streams using relevant state-of-the-art DES methods are presented in Section 5.3 to measure the impact of compression in terms of predictive performance as well as computational cost. Finally, Section 5.4 concludes with a summary of our contributions and raises issues and directions for future work.

## 5.1 COMPRESSING DYNAMIC ENSEMBLES FOR STREAMING DATA

Dynamic ensemble methods have demonstrated highly competitive results in terms of predictive performance for temporal data streams forecasting. We have over-viewed different state-of-the-art and novel dynamic ensemble methods that are able to cope with the evolving nature of temporally dependent data. Nonetheless, despite their superior predictive performance, DES methods lead to high computational costs in terms of time and memory compared to individual models. The overhead issue is particularly restraining in stream setting applications where predictive models are deployed on devices with very limited resources such as IoT and embedded systems. Besides, instances are released at high frequency in the stream which requires much faster predictive models for the sake of real-time reactivity. In fact, dynamic ensembles are time consuming as they have to frequently update models' weights. Last but not least, DES methods behave like "black boxes" and lack of interpretability which leads to serious challenges in understanding model's decision [99]. In fact, understanding and explaining predictive models decisions has become a major task in the research field of AI and ML over the last years. This is a particularly important aspect in sensitive application domains including ML driven decision-making which require accurate yet transparent models.

The computational complexity as well as the lack of interpretability have motivated the emergence of new research fields to solve the aforementioned drawbacks. In fact, Model Compression (MC) is a promising approach to create smaller and faster models while maintaining highly competitive predictive performance. We investigate the use of model compression for the forecasting task in the streaming setting using dynamic ensemble methods to cope with the evolving nature of temporal data. We show that compressing a highly performing dynamic ensembles into an individual model leads to better predictive performance when compared to an individual learner while significantly reducing computational costs. We conduct an extensive experimental study on both real and synthetic temporal data streams to measure the impact of compression on predictive performance as well as the computational cost.

### 5.1.1 MODEL COMPRESSION FOR BATCH LEARNING

Model Compression (MC) has emerged as a very promising solution to overcome the computational shortcomings of dynamic ensemble methods. The main idea is to use a fast and compact model to approximate the function learned by a slower, larger, but highly performing model [31]. MC consists in training a single predictive model (Student) to mimic the behavior of a highly performing yet complex model (Teacher). The teacher model is often an ensemble combining several individual models. We will refer to this approach as the Student-Teacher (ST) framework. The goal of a predictive model on streaming data is to generalize well to unseen observations. MC promotes this by training a learning algorithm to mimic the function of a dynamic ensemble that proved to generalize well [76]. The Student-Teacher (ST) approach consists of a two-stage learning process described in the following steps:

- Train the teacher  $T$  using a training set  $D_{tr}(x, y)$ ;
- Retrieve the predictions of the teacher  $\hat{y}^T$  on unseen observations  $D_{ts}(x, y)$ ;

- Train the Student  $s$  using a newly constructed set of observations  $D_{ts}(x, \hat{y}^T)$ , where the explanatory variables are the original ones, but the target variables are replaced with the predictions of the teacher;
- Deploy the student and use it to predict on all the upcoming test instances.

Most of the existing MC methods are tailored for batch learning and focus on the problem of classification [31, 76, 87, 103, 161, 169]. The ST approach was often applied in order to reduce the complexity for neural networks. Hinton, Vinyals, and Dean [76] used ST and obtained a more compact single neural network that has comparable predictive performance relative to the complex ensemble of neural networks. Handful studies tackle MC for regression tasks [45, 142, 148] and more particularly for the forecasting problem using dynamic ensemble methods. Cerqueira et al. [39] applied model compression to address uni-variate time series forecasting using DES methods such as ADE and AEC, to name only a few. Figure 5.1 describes the proposed ST-based model compression approach for DES as described in [39].

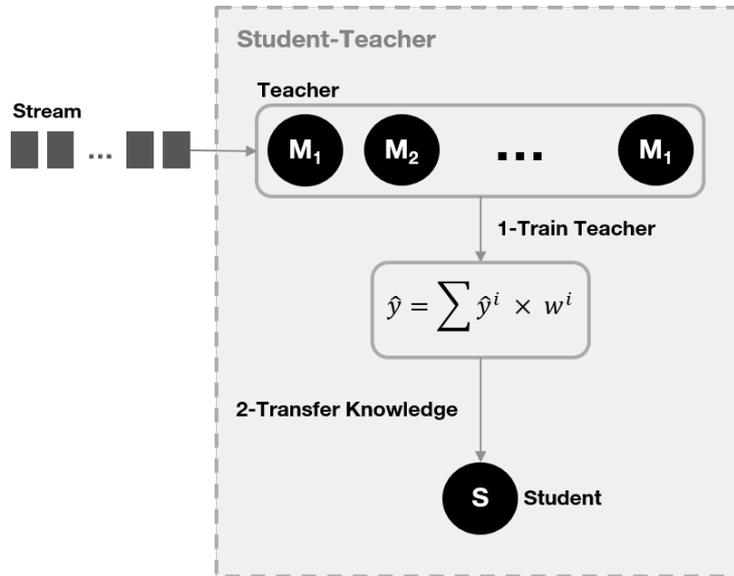


Figure 5.1: Student-Teacher general framework

### 5.1.2 COMPRESSION FOR STREAMING DATA

Model Compression has proved to be particularly effective when applied to dynamic ensembles of forecasters in traditional batch learning [39]. Nonetheless, the two-stage scenario described so far is likely to be limited when dealing with evolving data streams. In fact, if a concept drift occurs, the predictive performance of the deployed student is expected to be affected and decrease when novel concepts emerge and consequently lose the advantage of compressing a highly performing teacher [40].

In this chapter, we propose an adaptive Student-Teacher (ST) based model compression approach for temporal data streams where both the teacher and the student keep learning as new data are released. The teacher learns from new data instances while the student learns from new predictions retrieved from the teacher. The complex teacher model can

be deployed on a remote machine with high computational capacity whereas the student is deployed on a low capacity device such as IoT applications. The proposed method uses an adaptive ST approach for uni-variate temporal stream forecasting to cope with changes in the characteristics of the data. The proposed approach allows taking advantage of the predictive performance of dynamic ensembles along the stream while drastically reducing their computational costs.

## 5.2 ADAPTIVE MODEL COMPRESSION FOR DATA STREAMS FORECASTING

Following the uni-variate temporal data streams forecasting task described in Section 2.1.2, dynamic ensemble methods have demonstrated highly competitive predictive performance for temporal data streams forecasting compared to single models. In the streaming setting, it is important for learning models to meet tough time and memory requirements. However, ensemble methods are too complex whereas simple individual models are fast and compact to the detriment of prediction quality. We propose a novel approach for model compression tailored to evolving data streams to maintain a fast and compact but yet highly performing predictive model.

Similarly, we describe the proposed approach in three main steps: i) create the dynamic ensemble to serve as the Teacher ( $T$ ); ii) train the student model(s) using teacher's predictions; iii) handle concept drifts when they occur in the stream. We have over-viewed in Section 2.2 several dynamic ensemble methods tailored to the streaming data forecasting task that can serve as a teacher. We will discuss in more details steps ii) and iii) in the next sections.

### 5.2.1 TEACHING DATA

Model compression methods for classification often rely on the predictions of the teacher only, without considering the error made regarding the true value of the target. However, the real-valued predictions are unbounded in uni-variate numeric temporal stream forecasting task. Thus, the teacher can give highly erroneous guidance to the student, hence the importance of emphasizing on the real value of the target. On the other hand, ensemble methods are known for their generalization ability and robustness to noise and concept drift in the data stream mining field. We leverage both teacher's predictions and the true values of the target in training the student model. Therefore, instead of using the teacher's prediction directly as a target, we exploit the true value in order to lessen the impact of highly inaccurate predictions provided by the teacher that are likely to alter the predictive performance of the student. Besides, instead of training the student on the true values of the stream directly, smoothing the true values of the target using teachers' predictions aims at taking advantage of the robustness of ensemble methods against noise and concept drift that can occur in the stream.

We achieve this by controlling the contribution of the teacher according to its error relative to the ground truth, the higher the error, the lower the weight granted to the teacher. This will allow us to take advantage of the generalization ability and the predictive power of the teacher while mitigating the risk related to highly inaccurate predictions. Equation 5.1 describes the regularization function using a smoothing factor  $\alpha_t$  applied to harness teacher's predictions and the ground truth. We refer to teacher's prediction by  $\hat{y}_t^T$ , the true values by  $y_t$

and  $y'_t$  is the input value used to train the student.

$$\hat{y}'_t = \alpha_t \times \hat{y}_t^T + (1 - \alpha_t) \times y_t \quad (5.1)$$

The weight  $\alpha_t$  is dynamic and defined as  $\alpha_t = 1 - e_t^T$  where  $e_t^T$  is the sMAPE error incurred by the teacher while predicting  $y_t$ .

### 5.2.2 STUDENT MODEL TEACHING APPROACH

The proposed ST-based adaptive MC leverages both teacher's predictions as well as the true target values to train the student model. In this section, we describe different teaching scenarios based on the way the student model "invokes" the teacher to retrieve its predictions. As a matter of fact, the teacher model is often deployed on remote high-powered/resources machines scaled to maintain highly complex dynamic ensemble models whereas the student model is deployed on low-energy and capacity devices such as IoT and embedded systems. Hence, teacher's predictions are costly to obtain and the student model must be judicious/cautious to determine when to invoke the teacher in order to meet with the resource-wise environment.

The first compression scenario is strongly supervised where we assume that the student has the benefit to afford teacher's predictions for each instance in the stream. The two remaining scenarios are weakly supervised and invoke the teacher at specific times or instances based on the predictive performance of the student. The first weakly supervised strategy invokes the teacher for a given instance  $x_t$  if it was apparently hard to predict. The second weakly supervised compression approach uses a warning detector and invokes the teacher when a deviation is triggered in the predictive performance of the student model. The goal of the weakly supervised methods is to minimize resource consumption while maintaining high performance. We note that the term supervised is used for teacher predictions availability only to perform compression and not for the true values of the target that are assumed to be available for each instance for both the teacher and the student with no delay.

#### 5.2.2.1 Strongly supervised compression

We present SIMON for StreamIng MOdel compressionN which is strongly supervised using teacher's predictions. The predictions of the teacher are available for each instance in the stream in order to update the student model w.r.t. the true value of the target. This implies permanent communication between the teacher and the student which is often unfeasible particularly in IoT devices with limited resources. Algorithm 5 describes the fully supervised ST-based streaming model compression. The function *predict(.)* is used to retrieve model's prediction given the test instance  $x_t$ . The function *update(.)* is used to update the predictive model with the newly released labeled instances in the stream. The *Regularization(.)* function computes a smoothed target value that leverages both the teacher's prediction and the true value according to one of the methods detailed in Section 5.2.1.

#### 5.2.2.2 Semi-supervised compression

The SIMON-SMILE, for SIMON-SeMI-supervised LEarning, is a weakly supervised compression approach that assumes limited access to the predictions of the teacher. The student

**Algorithm 5:** SIMON

---

**Input** :  $S$ : stream,  $P()$ : performance estimation function,  $T$ : Teacher,  $s$ : Student  
**Output** :  $\hat{y}_t$  for each instance in the stream

```

1 while  $HasNext(S)$  do
2    $(x_t, y_t) \leftarrow next(s)$  // Get next instance
3    $\hat{y}_t \leftarrow predict(S, x_t)$  // Return Student's prediction
4    $\hat{y}_t^T \leftarrow predict(T, x_t)$  // Get Teacher's prediction
5    $\hat{y}' \leftarrow Regularization(y_t, \hat{y}_t^T)$  // Regularized target
6    $update(T, x_t, \hat{y}_t)$ 
7    $update(s, x_t, \hat{y}'_t)$ 
8 end

```

---

model "invokes" the teacher if it considers necessary due to instance prediction hardness (difficulty). The semi-supervised approach is a desirable property in the case of model compression due to the cost associated with the teacher's expertise query.

A test instance  $x_t$  is said "*hard*" to predict if the incurred error by the student is greater than the average error computed on very similar and past instances. In fact, since the model is deployed in a prequential setting, the predictive performance of the student model is expected to get better as it is updated using newly released training instances. However, the predictive performance might be altered by noise or change (concept drift). Thus, the student can request teacher's advice since the teacher is assumed to be better than individual models to handle noise and adapt to concept drift. Similarly to SLOPE defined in Section 3.1.1.1, we compute the average loss incurred in a local region defined using the  $k$  nearest neighbors queried over a sliding window  $w$ . Algorithm 6 describes in more details the steps of the SIMON-SMILE approach.

### 5.2.2.3 Performance based compression

SIMON-PELE (SIMON-**P**erformance based **L**earning) is a weakly supervised ST based MC where the student model invokes the teacher when a warning is triggered in its predictive performance. Technically, the student keeps learning from the real values of the target released in the stream, however, when a change occurs in the characteristics of the data, the predictive performance of the student is very likely to be altered or damaged. When a warning is flagged, we refer to the teacher and invoke its predictions in order to update the student model. In fact, complex dynamic ensemble methods are deemed to be better for their robustness to noise and recovery from concept drift. Algorithm 7 depicts the drift based adaptive model compression where any state-of-the-art change detection method can be used to monitor student's loss and trigger warnings.

### 5.2.3 CONCEPT DRIFT ADAPTATION FOR STREAMING MC

Temporal data streams are expected to evolve over time and therefore be subject to concept drift that translates into changes in the characteristics of the data generation process [156]. Concept drift may alter the predictive performance or completely invalidate the current learned model. One of the best assets of dynamic ensembles methods is their ability to

**Algorithm 6:** SIMON-SMILE

---

**Input** :  $S$ : stream,  $T$ : Teacher,  $s$ : Student,  $P(\cdot)$ : performance estimation function,  
 $k$ : number of neighbors,  $w$ : sliding window size

**Output** :  $\hat{y}_t$  for each instance in the stream

```

1 while HasNext( $S$ ) do
2    $(x_t, y_t) \leftarrow \text{next}(s)$  // Get next instance
3    $\hat{y}_t \leftarrow \text{predict}(S, x_t)$  // Return Student's prediction
4    $e^T \leftarrow P(\hat{y}_t, y_t)$  // Student's error on current instance
5    $e_{k,w}^T \leftarrow P_{k,w,S}$  // Student's error using SLOPE
6   if  $e_{k,w}^T \leq e^T$  then
7     // Student performs worse, invoke teacher
8      $\hat{y}_t^T \leftarrow \text{predict}(T, x_t)$  // Get Teacher's prediction
9      $\hat{y}' \leftarrow \text{Regularization}(y_t, \hat{y}_t^T)$  // Regularized target
10  end
11 else
12  // Student performs better, no need for teacher's prediction
13   $\hat{y}' \leftarrow y_t$ 
14 end
15 update( $T, x_t, \hat{y}_t$ )
16 update( $s, x_t, \hat{y}'$ )
17 update( $w, x_t, y_t, \hat{y}_t$ )
18 end

```

---

**Algorithm 7:** SIMON-PELE

---

**Input** :  $S$ : stream,  $T$ : Teacher,  $s$ : Student,  $P(\cdot)$ : performance estimation function,  
 $\delta_w$ : warning threshold,  $C(\cdot)$ : change detection method

**Output** :  $\hat{y}_t$  for each instance in the stream

```

1 while HasNext( $S$ ) do
2    $(x_t, y_t) \leftarrow \text{next}(s)$  // Get next instance
3    $\hat{y}_t \leftarrow \text{predict}(S, x_t)$  // Return Student's prediction
4    $\text{err}^T = P(\hat{y}_t, y_t)$ 
5   if  $C(\delta_w, \text{err}^T)$  then
6     // Warning Detected, call teacher for help
7      $\hat{y}_t^T \leftarrow \text{predict}(T, x_t)$  // Get Teacher's prediction
8      $\hat{y}' \leftarrow \text{Regularization}(y_t, \hat{y}_t^T)$  // Regularized target
9   end
10  update( $T, x_t, y_t$ )
11  update( $s, x_t, \hat{y}'$ )
12 end

```

---

handle concept drift. However, single models do not share the same strength and speed of adaptation and recovery when drifts occur compared to dynamic ensemble methods. In this context, STUDD [37] (Student-Teacher approach for Unsupervised Drift Detection) is a

novel unsupervised drift detection approach based on an ST learning paradigm. A teacher is used for predicting new instances and monitoring the mimicking loss of the student for concept drift detection. The student’s imitation loss that stands for the discrepancy between the teacher’s prediction and student’s prediction on the same instance is monitored using any state-of-the-art drift detector. When concept drift occurs, the collective behavior between the teacher and student models is likely to be altered.

That said, it is crucial that the student keeps learning from the teacher as new data arrive in order to carry on the advantage of the teacher’s adaption, robustness and predictive capacities. To cope with evolving data streams, the informed adaptation approach explicitly uses a concept drift detection mechanism to trigger any decline in the overall predictive performance. If a drift is flagged, the default behavior is to replace the current model with a new one in order to consolidate the new concept. However, this may decrease overall performance since the new model has not been trained on any instance. We have followed the twofold adaptation strategy presented in [66] to implement Adaptive Random Forest (ARF). Instead of resetting the model as soon as a drift is detected, we use two different drift detection levels. The first one is more indulgent and detects warnings whereas the second one is more rigorous and detects drifts. After a warning has been detected, another background student ( $s_b$ ) starts learning in parallel without impacting the predictions. The background student replaces the old student ( $s$ ), if and only if the warning flag amplifies to a drift. Most of the concept drift detection methods focus on detecting changes by tracking significant deviation in the loss of the model, we have used the sMAPE error measure to monitor the predictive performance of the student model. If a warning or drift is reported, this means that the student model is under-performing in predicting the actual values of the stream and hence should be replaced. The proposed streaming model compression method is not bounded to a specific drift detector method. We have used the ADWIN change detector [11] with confidence level  $\delta_w$  and  $\delta_d$  for warning and drift detector respectively.

The Adaptive Model Compression is depicted in Algorithm 8. To cope with stationary data streams any incremental forecasting algorithm can be used as the student model and be updated as new instances are released in the stream. We monitor the predictive performance of the student model  $p_t$  along the stream. When a warning is detected (Line 13), a background student  $s_b$  starts learning on new data and replaces the current/old student  $s$  if the warning escalates to a drift (Line 16).

### 5.3 EXPERIMENTS

In this section, we investigate and analyze each compression scenario discussed above (SIMON, SIMON-SMILE, and SIMON-PELE) to build a compressed student and leverage both teacher predictions and true values of the target. We provide an extensive empirical study to measure the impact of the proposed adaptive model compression for temporal data streams forecasting. We will answer the following questions throughout the set of experiments.

- Q1** How does an individual model trained to mimic the behavior of a dynamic ensemble performs relative to the dynamic ensemble itself?

**Algorithm 8:** Adaptive Model compression

---

**Input** :  $s$ : stream,  $P(\cdot)$ : performance estimation function,  $T$ : Teacher,  $s$ : Student,  $s_b$ : Background Student,  $\delta_w$ : warning threshold,  $\delta_d$ : drift threshold,  $C(\cdot)$ : change detection method

**Output** :  $\hat{y}_t$  for each instance in the stream

```

1  $s_b \leftarrow \text{NULL}$ ;
2 while  $\text{HasNext}(S)$  do
3    $(x_t, y_t) \leftarrow \text{next}(s)$  // Get next instance
4    $\hat{y}_t^T \leftarrow \text{predict}(T, x_t)$  // Get Teacher's prediction
5    $\hat{y}_t \leftarrow \text{predict}(S, x_t)$  // Return Student's prediction
6    $p_t \leftarrow P(\hat{y}_t, y_t)$  // Student's error
7    $\hat{y}_t' \leftarrow \text{Regularization}(y_t, \hat{y}_t^T)$ 
8    $\text{update}(T, x_t, \hat{y}_t)$ 
9    $\text{update}(s, x_t, \hat{y}_t')$ 
10  if  $s_b$  then
11    // Background student exists
12     $\text{update}(s_b, x_t, \hat{y}_t')$ 
13  end
14  if  $C(\delta_w, p_t)$  then
15    // Warning Detected
16     $s_b \leftarrow \text{CreateStudent}()$ 
17  end
18  if  $C(\delta_d, p_t)$  then
19    // Change Detected
20     $S \leftarrow s_b$ 
21     $s_b \leftarrow \text{NULL}$  // Reset background student
22  end
23 end

```

---

- Q2** How does a compressed model perform relative to the same model trained directly on the real values of the target ?
- Q3** What is the advantage of leveraging both teacher's predictions and true values of the target when training the student model?
- Q4** What is the relative gain in terms of predictive performance of compressed students relative to the same model trained directly on the data ?
- Q5** What is the loss in terms of predictive performance of the compressed models relative to their teachers ?
- Q6** What is the relative computational gain by compressing a dynamic ensemble into a single individual model?
- Q7** What is the advantage of using an explicit adaptation approach to maintain the student model ?

First we analyze each compression scenario (*SIMON*, *SIMON-SMILE*, *SIMON-PELE*) using different teacher ensembles and compare compressed models (students) against their respective teachers as well as the same individual model trained directly on the data stream (*RHT*). We examine three different methods to leverage both teacher’s predictions and true values of the target as described in Section 5.2.1. We conduct a comparison in terms of predictive performance and computational cost.

Table 5.1 describes the abbreviations used throughout the experimental study. The first three rows address the regularization/smoothing to leverage both teacher’s predictions and true values (ground truth) described in Equation 5.1 where *M\_PERF* (resp. *M\_0.5*) stands for dynamic (resp. equal) weighting. Finally, *ADAPT* points to the adaptive model compression described in Section 5.2 that monitors the loss incurred by the student and replaces it with a newer one where a drift is triggered.

Table 5.1: Terminology (tags) and respective description for compression approaches

Tag	Description
None	The student learns from the predictions of the teacher only without any smoothing/regularization that translates to $\alpha = 1$ .
<i>M_PERF</i>	Adaptive target regularization using $\alpha_t$ based on the performance of the teacher 5.1.
<i>M_0.5</i>	Static target regularization using $\alpha = 0.5$ that grants the same weight to both teacher prediction and true value of the target
<i>ADAPT</i>	Adaptive model compression described in Algorithm 8

### 5.3.1 EXPERIMENTAL SETUP

For dynamic forecast combination methods, we focus on the following approaches: *ARF*, *BLAST*, and *STACKING* described in Chapter 2. Our study includes real-world and synthetic uni-variate temporal data streams described in Chapter A. In summary, our analysis is based on 30 temporal data streams using an embedding size  $p = 10$ . We use the root mean squared error (RMSE) as evaluation metric in a prequential setting, where each instance is first used to test the model that to update it. Similarly to Chapter 3, a large batch is used to ensure a warm start for the dynamic ensemble methods using the blocked prequential approach described in Section A.3. Notwithstanding, in most of our analysis we will compare different forecasting approaches according to their average rank in the 30 temporal data streams. A rank #1 means that the method was the best performing one (lowest RMSE) on a given data stream.

For the weakly supervised *SIMON-SMILE*, we used the *SLOPE* (Section 3.1.1.1) evaluation method based on a sliding window of  $w = 200$  and  $k = 5$  nearest-neighbors to compute the confidence of the compressed model by computing the loss in the local region surrounding the test instance. In addition to the rank, we analyze the percentual difference (relative change) in terms of predictive performance between the compressed student and both teachers and *RHT*. This allows us to quantify the improvement or deterioration in

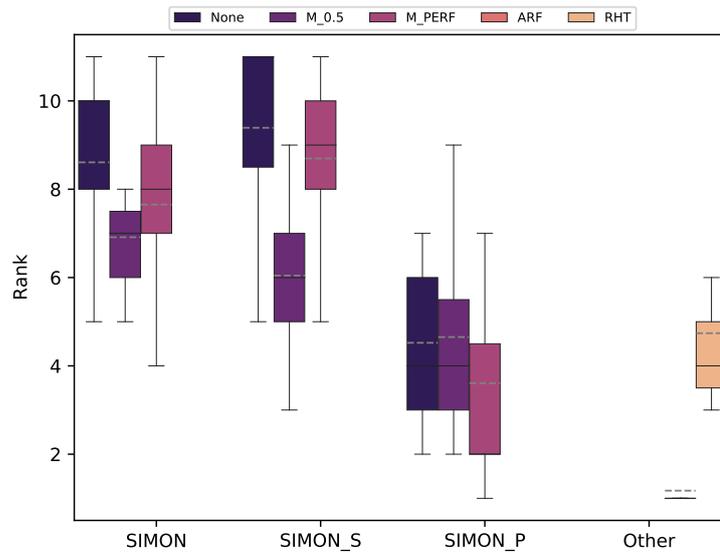


Figure 5.2: Boxplots illustrating the distribution of the rank and respective standard deviation of compressed student relative to ARF and RHT in terms of RMSE grouped by their respective compression scenario (SIMON, SIMON-SMILE, and SIMON-PELE)

the predictive performance achieved using compression. The gain (resp.loss) in predictive performance is computed as described in Equation 3.5 and expressed in percentage.

### Ranking Student vs Teacher vs RHT

Figures 5.2, 5.3, and 5.4 show the results that address the research questions **Q1** and **Q2** and analyze how the predictive performance of a compressed student compares with ensemble methods (teachers) and individual model trained directly on the data RHT. We detail the results for each teacher enumerated above (ARF, BLAST and STACKING). First, Figure 5.2 addresses model compression using ARF as a teacher and illustrates the distribution of the rank and respective standard deviation of student models relative to their teacher and RHT grouped by compression scenario (SIMON, SIMON-SMILE, SIMON-PELE). Models were ranked according to the RMSE scored for each temporal data stream. Besides, we display for each compression scenario the three approaches to regularize the target value used to train the student model and weight teacher's predictions and trues values of the target observed in the stream. The category *Other* groups both the dynamic ensemble method that served as teacher as well as RHT trained directly on the data.

Results, show, as expected, that ARF outperforms all its compressed counterparts variants using different compression scenarios (SIMON, SIMON-SMILE and SIMON-PELE) as well as the individual RHT. Compressed students in a strongly supervised scenario (SIMON) and weakly supervised one using student's confidence (SIMON-SMILE) perform worse in average compared to RHT (rightmost box). In more details, training the student model using teacher predictions only noted with *None* scores the highest average rank compared the two other variants that leverage both teacher predictions and trues values of the target. Compression using equal weights ( $M_{0.5}$ ) accomplish the best average rank in the SIMON and SIMON-SMILE and outperform the use of the dynamic smoothing factor  $\alpha_t$  based on

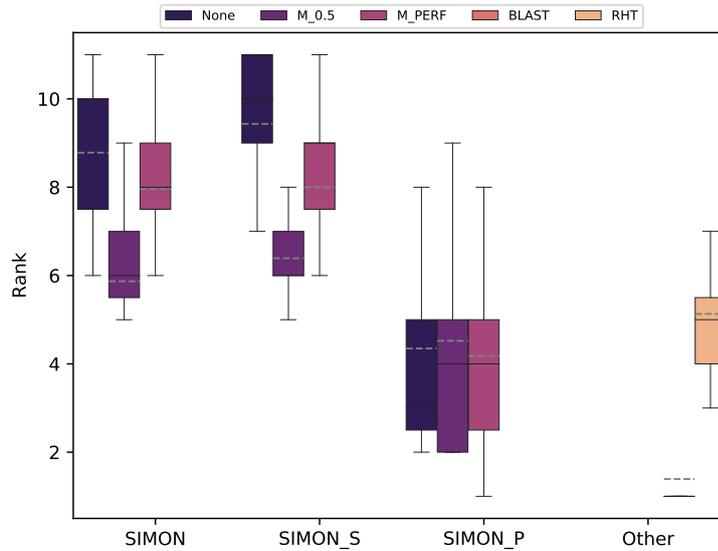


Figure 5.3: Boxplots illustrating the distribution of the rank and respective standard deviation of compressed student relative to BLAST and RHT in terms of RMSE grouped by their respective compression scenario (SIMON, SIMON-SMILE, and SIMON-PELE)

the performance of the teacher (M\_PERF).

The proposed weakly supervised SIMON-PELE scenario performs better on average compared to SIMON and SIMON-SMILE. We can notice that the M\_PERF variant shows the best average rank and outperforms RHT. SIMON-PELE is based on monitoring the loss of the student and "invoking" the teacher when a warning is triggered. Notwithstanding, SIMON-PELE-based compression of ARF ensemble using the three different teaching approaches, despite their competitive average rank, shows high standard deviation which reflects the uncertainty about its predictive performance compared to RHT.

Similarly, we conduct in Figure 5.3 and 5.4, the same analysis addressing BLAST and STACKING as teachers respectively. Overall, we can notice the same behavior where compressing using SIMON and SIMON-SMILE do not manage to compete the average rank of RHT. Besides, compressing BLAST and STACKING in SIMON-PELE scenario demonstrates the best average rank among all other compression scenarios and achieves competitive results compared to RHT despite higher standard deviation as well.

The SIMON-PELE demonstrates promising results for the application of model compression on evolving temporal data streams forecasting. In fact, experiments using ARF, BLAST and STACKING show that compressed students combining teacher's predictions and true values of the target perform slightly better on average compared to RHT. We bring together all the results for the best performing scenario SIMON-PELE and analyze how the predictive performance of compressed student compares to all the teachers and RHT. Figure 5.5 depicts the distribution of the rank of all compressed variants in SIMON-PELE grouped by the target smoothing value approach (None, M\_0.5, and M\_PERF) as well as the teachers and RHT. Table 5.2 supports the results and details the average rank and standard deviation of the compared methods. The rank column presents the average rank achieved by each variant of compression and its respective standard deviation. A rank of #1 means the

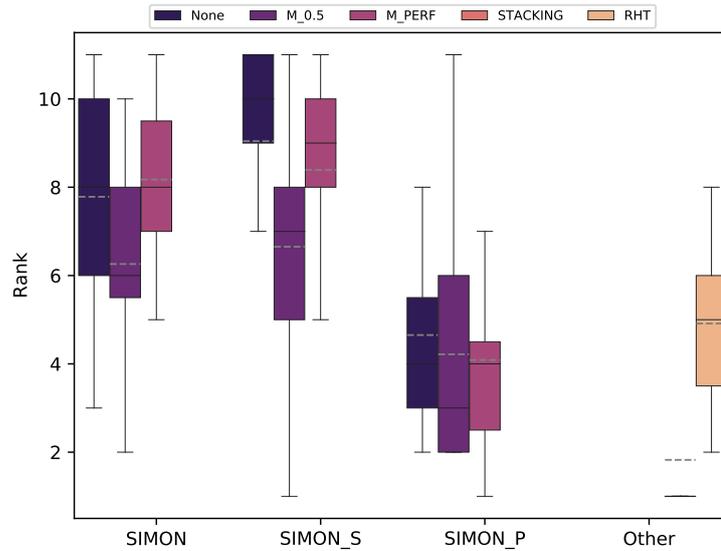


Figure 5.4: Boxplots illustrating the distribution of the rank and respective standard deviation of compressed student relative to STACKING and RHT in terms of RMSE grouped by their respective compression scenario (SIMON, SIMON-SMILE, and SIMON-PELE)

method was the best performing (lowest RMSE). Compressed students show better average rank compared to RHT, particularly ST\_ARF\_M\_PERF and ST\_BLAST\_None. However, high standard deviation is scored, that shows that the results are widely spread and less reliable.

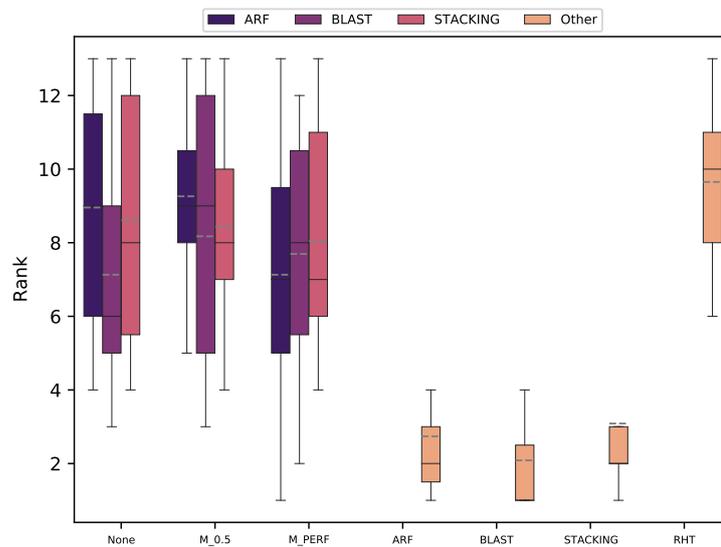


Figure 5.5: Boxplots illustrating the distribution of the rank and respective standard deviation of all compressed student relative to all teachers and RHT with the best performing compression scenario (SIMON-PELE) in terms of RMSE

Table 5.2: Average rank and respective standard deviation of compression variants with different teachers (ARF, BLAST and STACKING) in terms of RMSE for 30 temporal streams using SIMON-PELE compression scenario.

	Method	Avg. Rank
ARF	None	$8.957 \pm 3.126$
	M_0.5	$9.261 \pm 2.115$
	M_PERF	$7.130 \pm 3.123$
BLAST	None	$7.130 \pm 2.959$
	M_0.5	$8.174 \pm 3.499$
	M_PERF	$7.696 \pm 3.066$
STACKING	None	$8.609 \pm 3.354$
	M_0.5	$8.435 \pm 2.873$
	M_PERF	$8.043 \pm 2.836$
Single	RHT	$9.652 \pm 2.080$
Teacher	STACKING	$3.087 \pm 3.190$
	ARF	$2.739 \pm 2.072$
	BLAST	$2.087 \pm 2.151$

#### Quantifying the gain (loss) in the predictive performance to RHT and teacher:

We address both **Q5** and **Q4** to quantify the gain or loss in terms of predictive performance of compressed models relative to their respective teachers and RHT per scenario. Figure 5.6 shows the distribution of the percentual difference in terms of RMSE scored by the compressed students using a smoothing factor  $\alpha$  (M\_0.5 and M\_PERF) or training the student model on the prediction of the teacher only (None). Students are grouped by compression scenario and each row stands for a different teacher (ARF, BLAST and STACKING). Figures on the left (resp. right) side depict the distribution of the relative predictive performance loss/gain in RMSE of the student compared to teacher (resp. RHT). We refer the reader to Equation 3.5 for detailed explanation on the significance of the values.

Not surprisingly, compressed students score high loss in RMSE compared to their respective teachers particularly for BLAST and ARF where the average loss lays around/close to 100% in average. In fact, compressing the best performing teachers causes high loss in the predictive performance compared to STACKING where the average loss value gravitates around 60%. Besides, student learning from the predictions of the teacher only show lightweight higher average loss compared to M\_0.5 and M\_PERF that combine teacher predictions and true values of the target. This pattern runs through all the compression scenarios and teachers. Results also show a small advantage for smoothing that leverages both real value of the target and the predictions of the teacher with equal weights. On the other hand, compression does not achieve high improvements compared to RHT as one would have expected/hoped. In fact, all variants of compressed student in the SIMON and SIMON-SMILE scenarios for all teachers score positive average relative change which means that the students do not improve RHT predictive performance. The best performing

compression scenario `SIMON-PELE` achieves little performance gain with lower standard deviation and have comparable behavior across all teachers..

#### Q6 Computational costs:

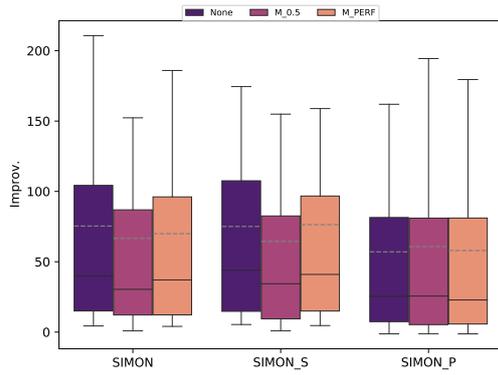
We discuss in this section the relative difference in the computational costs of the compressed student relative to both dynamic ensemble teachers and single model `RHT`. We analyze the distribution of the computational cost ratio between the teacher (resp. `RHT`) and all variants of compressed students across all scenarios. We recall that a ratio  $\rho$  means that teacher (resp. `RHT`) requires as much as  $\rho$  times the resources required by the compressed student. First, we address in Figure 5.7 the computational cost related to memory, that is the space required to store the learning model.

Figure 5.7a shows the distribution of the model size ratio between all variants of compressed students grouped by compression scenario and their respective teachers across 30 temporal data streams. On average, compressed students are almost 20 times smaller which means that the student model requires less than 5% the memory required by the ensemble to be stored. `SIMON-PELE` and `SIMON-SMILE` are slightly larger (smaller average ratio) compared to `SIMON` due to the additional complexity related to the proposed enhancement. In fact, `SIMON-SMILE` confidence evaluation is based on `SLOPE` that stores a sliding window of size  $w = 200$ . The `SIMON-PELE` on its side, uses two `ADWIN` drift detectors that are designed to meet the streaming settings requirements in terms of memory usage and do not cause substantial additional cost.

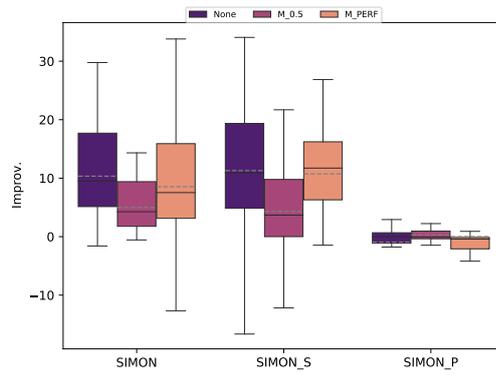
Similarly, Figure 5.7b details the distribution of the model size ratio between all variants of compressed students grouped by their respective compression scenario and the single model `RHT` that is trained directly on the data stream. As the results suggest, students have comparable model size to `RHT` with an average ratio gravitating around  $\rho = 1$  with small variation. `SIMON` and `SIMON-SMILE` compressed variants using `None`, `M_0.5` and `M_PERF` exhibit higher standard deviation compared to `SIMON-PELE` due to the difference in the tree induction related to the modified values of the target value used to train the student. In fact, `SIMON` and `SIMON-SMILE` make more use of the teacher predictions and thus the tree is build differently compared to `RHT` that uses the true values of the target only. On the other hand, `SIMON-PELE` induces a tree that is close to the one induced by `RHT` since it invokes the teacher less frequently (only in the case of warnings).

Next, we address in Figure 5.8 the computational cost in terms of time (measured in seconds) required to predict and update each model on all instances of the stream. In fact, our experimental setup is based on prequential evaluation which means that each instance in the stream is first used to test the model (predict) and then to update it as the true value of the target is released. We emphasize that the total running time does not include communication cost with the teacher and assume perfect communication with no delays. Figure 5.8a reports the distribution of the time ratio between teacher and all variants of compressed student counterparts grouped by their respective compression scenario. All the teacher methods are relatively more time consuming compared to compressed students with a factor mean around  $\rho = 10$ . The variants `None`, `M_0.5`, and `M_PERF` from the same compression scenario have comparable running time. In fact the only difference lies in the computation of the teaching data that include basic operations for `M_0.5` and `M_PERF`. Finally, `SIMON-SMILE` is

## ARF

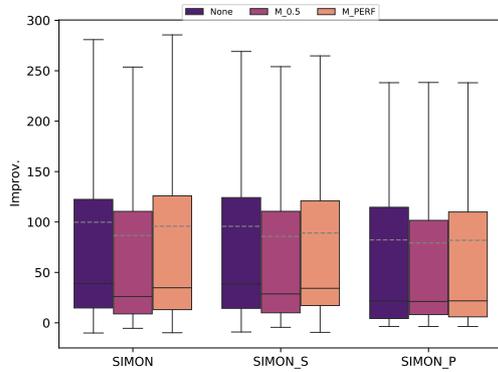


(a) ARF Student teacher

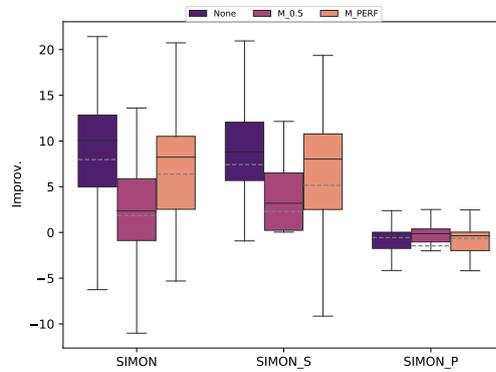


(b) ARF Student to RHT

## BLAST

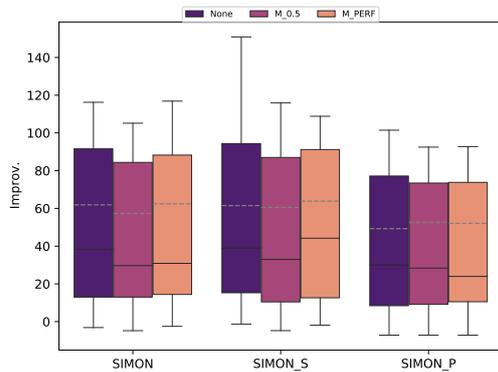


(c) BLAST Student teacher

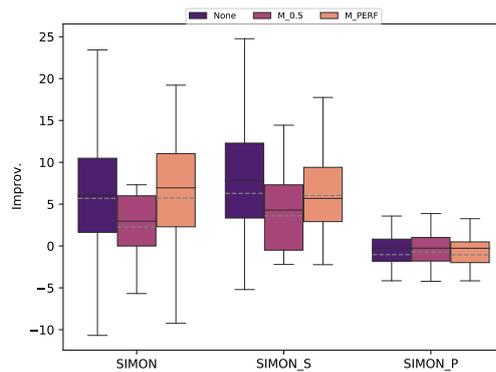


(d) BLAST Student to RHT

## STACKING



(e) STACKING Student teacher



(f) STACKING Student to RHT

Figure 5.6: Boxplots illustrating the distribution of the average percentual improvement (expressed in percentage) and respective standard deviation of compressed student relative to teacher (left) and RHT (right) grouped by their respective compression scenario (SIMON, SIMON-SMILE, SIMON-PELE) in terms of RMSE.

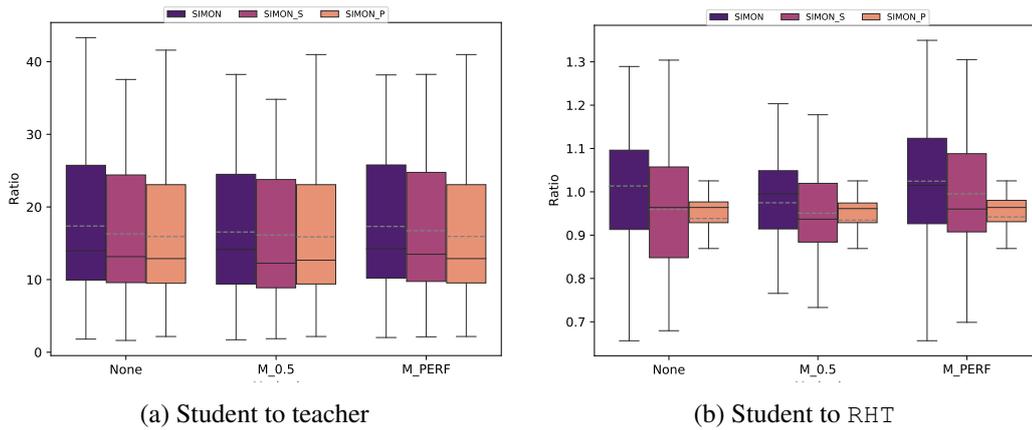


Figure 5.7: Boxplots illustrating the distribution of the ratio and respective standard deviation of compressed student relative to RHT (left) and respective teacher (right) grouped by compression scenario (*SIMON*, *SIMON-SMILE*, *SIMON-PELE*) in terms of model size.

slower (smaller ratio) on average compared to *SIMON* and *SIMON-PELE* due to the *SLOPE* component that computes for each incoming instance the local predictive performance of the student on the  $k$  nearest-neighbors over a sliding window. Likewise, Figure 5.8b depicts the distribution of the time ratio of compressed students relative to RHT. On average, RHT is faster and the same comments arise regarding *SIMON-PELE*. Modifying the target value used to train the student model compared to RHT impacts tree induction. In fact, RHT uses the Hoeffding bound to control its split decisions and relies on calculating the reduction of variance in the target space to decide among the split candidates. The smallest the variance at its leaf nodes, the more homogeneous the partitions are.

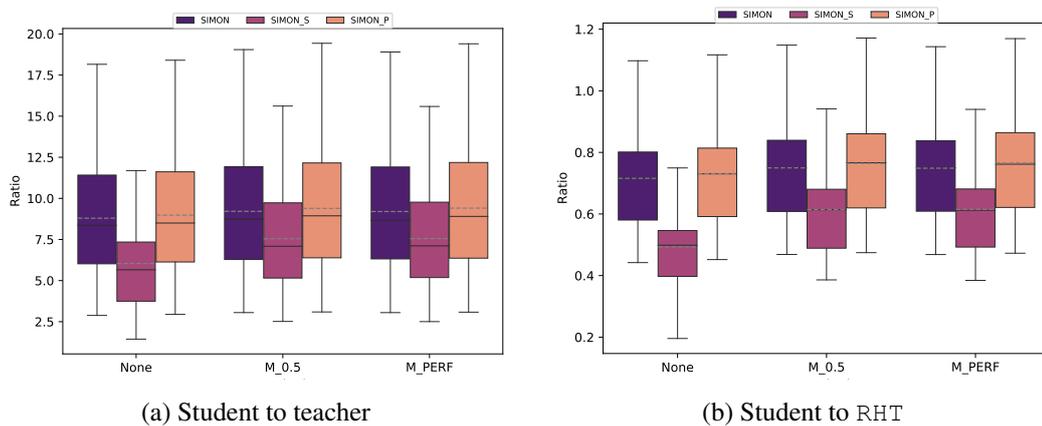


Figure 5.8: Boxplots illustrating the distribution of the ratio and respective standard deviation in terms of total running time of compressed student relative to respective teacher (left) and RHT (right) grouped by compression scenario (*SIMON*, *SIMON-SMILE*, *SIMON-PELE*).

### 5.3.2 ADAPTIVE COMPRESSION

In this section we address **Q7** and analyze the proposed adaptive model compression that uses an informed adaptation strategy. We have proposed to monitor the *sMAPE* loss in

the performance of the compressed student and trigger warnings and drifts. We have used the ADWIN drift detector with  $\delta_w = 0.01$  (resp.  $\delta = 0.01$ ) being the confidence parameter of the warning (resp. drift) detector. As described in Algorithm 8, when a warning is triggered a background student starts learning from new upcoming instances and replaces the old one if the warning escalates to a drift. We analyze the predictive performance of adaptive compressed students compared to their non-adaptive counterparts, the teachers and RHT. We investigate adaptation for all the proposed scenarios (SIMON, SIMON-SMILE and SIMON-PELE) with or without using a smoothing factor (None, M\_0.5, and M\_PERF).

Figure 5.9 shows the average rank and respective standard deviation of adaptive compressed students and their non-adaptive counterparts in addition to teachers and RHT. Each sub-figure illustrates a different compression scenario (SIMON, SIMON-SMILE, SIMON-PELE) where student models are grouped by their respective teachers (ARF, BLAST, STACKING). Likewise, the group `Other` includes both the dynamic ensembles and RHT. Adaptive students are referred with the tag `ADAPT` whereas non-adaptive ones use the same terminology as described in Table 5.1. Results depicted in sub-figures 5.9a and 5.9b for SIMON and SIMON-SMILE scenarios respectively using adaptive compression are inconclusive. In fact, adaptive compressed students do not manage to outperform RHT on average and show high uncertainty (standard deviation is high). Besides, adaptive students perform worse on average compared to their non-adaptive counterparts for all the teachers and cause significant loss in terms of predictive performance. This means that using an informed adaptation strategy using ADWIN could not help improving the predictive performance of student models in the strongly supervised scenario (SIMON) and weakly supervised one SIMON-SMILE. Indeed, as illustrated before, the two scenarios with or without using the smoothing factor  $\alpha$  did not succeed compared to RHT. Finally, adaptive compression results on the SIMON-PELE scenario illustrated in sub-figure 5.9c show the same pattern where adaptation worsens the average rank compared to non-adaptive counterparts with high disruptive distribution of the ranks scored. However, some of the adaptive compression methods have comparable average rank compared to RHT even slightly better such as `ADAPT_M_PERF` using ARF as teacher. By proposing a more proactive adaptive compression that relies on explicitly triggering warnings and drifts in the predictive performance of the student, one would have expected better results overall when dealing with evolving temporal data streams. However, results are closely related to the drift detector as well as the loss criteria. Besides, data could include noise and outliers that can mislead the active adaptation strategy.

## 5.4 SUMMARY

In this chapter we have investigated Model Compression (MC) for evolving temporal data stream forecasting. Model compression using the Student-Teacher (ST) framework was widely studied for classification and has demonstrated promising results for traditional learning. Compressing dynamic ensembles for time series in the batch-learning [39] has enabled the transfer of the capacity of large ensembles into compact predictive models. In this chapter we investigated the applicability of model compression for dynamic forecast combination approaches in the streaming setting. In fact, temporal data streams are expected

to experience concept drift that has negative impact on the predictive performance. We aim at building an individual model to mimic the behavior of an ensemble to retain a competitive predictive performance while being faster and more compact.

Conversely to existing methods that rely on the prediction of the teacher only to train the student model, we proposed to leverage both teacher's predictions and true value of the target using a smoothing function that allows taking advantage from both. Besides, we introduced three different compression scenarios based on the communication schema between the teacher and the student. The first scenario *SIMON*, is strongly supervised where the student model requires permanent communication with the teacher. However, communication is costly in resource wise environments such as embedded systems related to the internet-of-things where devices must save energy. We proposed two other compression scenarios where the student has limited access to teacher's predictions based on its predictive performance. The *SIMON-SMILE* scenario is weakly supervised where the student is trained on the true values of the target and requests teacher's predictions in case of low confidence. The confidence is computed on a local region defined by the nearest and most recent instances observed in the stream. The last scenario *SIMON-PELE* is based on monitoring the loss incurred by the student and updating it with the predictions of the teacher in case of warning.

Finally, we presented an adaptive model compression approach to cope with evolving data streams. In fact, when a concept drift occurs, the collective behavior between the teacher and the student model is expected to be altered. Hence, it is crucial to have an informed adaptation strategy to discard the old student and train a new student model with recent instances to consolidate the new concept.

The Experimental results presented in this chapter have demonstrated limited success of the proposed compression methods across the different scenarios in terms of predictive performance. Nevertheless, this work will establish a basis for model compression on temporal data streams and discuss several leads to improve the results. In particular, it offers some promising developments for the analysis of various compression and adaptation methods to cope with concept drift.

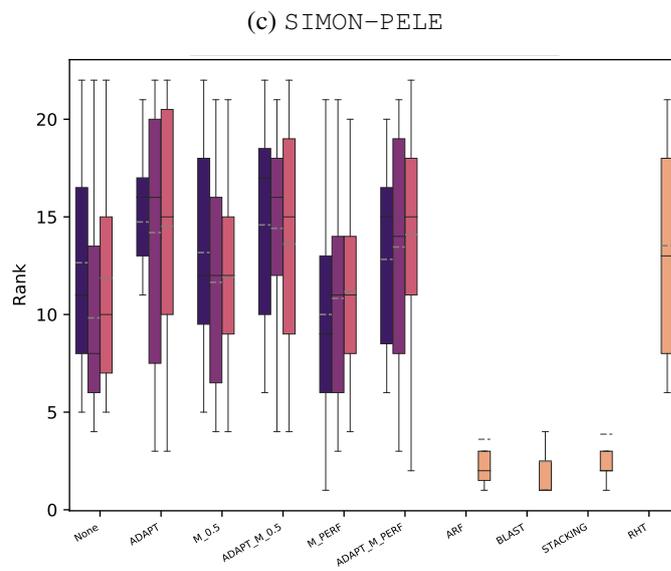
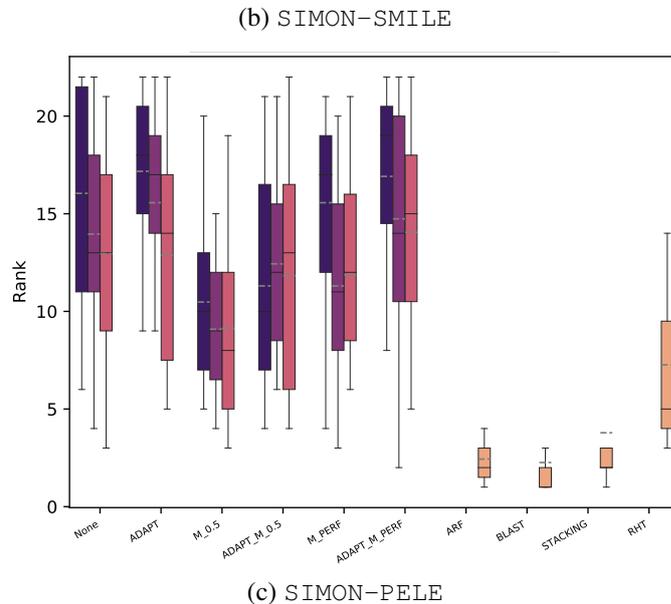
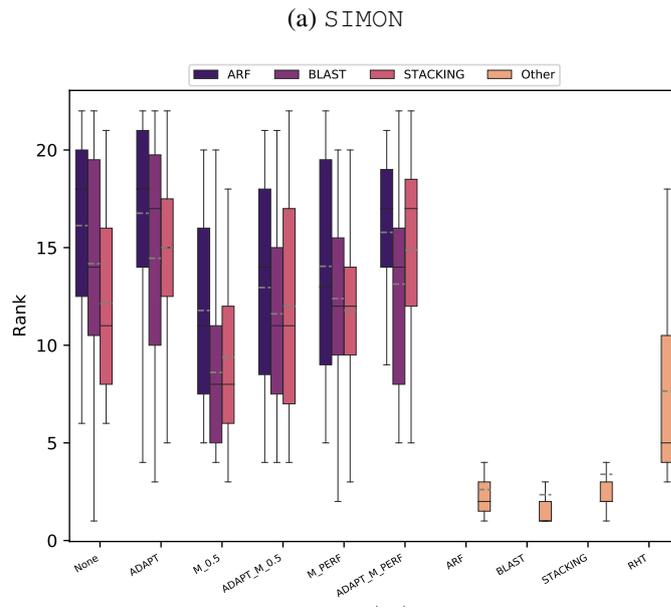


Figure 5.9: Boxplots of the distribution of the rank and respective standard deviation in terms of RMSE of the compressed students relative to teachers and RHT by scenario.

## Real-time Machine Learning Competitions with SCALAR

In this chapter, we present the competition "Real-time Machine Learning Competition on Data Streams", a BigDataCup Challenge of the IEEE Big Data 2019 conference. This track in the IEEE Big Data 2019 Big Data Cup represents Machine Learning competition on data streams. There already exists a great community organizing various competitions on machine learning tasks for batch learners. We introduce a similar approach to engage the whole community in solving essential problems in the data stream mining field involving fast incremental learners. The main idea is to release competition data in the form of stream of small batches to build incremental learning models and participants continuously submit their predictions in the form of stream as well. We run a data science competition based on a real-time prediction setting, using a novel platform that meets the streaming setting requirements. Besides, participants are urged to submit the predictions rapidly in the form of stream as well to cope with newly released data. To the best of our knowledge, this is the first data science competition conducted in real-time.

The chapter is organized as follows: Section 6.1 details the competition and describes the winning solutions, Section 6.2 describes the SCALAR platform designed to meet the requirement of online and streaming machine learning competition and contributes to the Free and Open Source Software (FOSS) in the research community. Finally, Section 6.3 concludes this chapters and addresses potential future improvement directions.

## 6.1 REAL-TIME MACHINE LEARNING COMPETITION ON DATA STREAMS

The real-time machine learning competition was organized as a track in the IEEE Big Data 2019 Big Data Cup<sup>1</sup> [18]. We have conducted a challenge on real-world case study with data streams. The competition involved several participants from all over the world to adhere to this novel challenge. The competition scenario falls within the network activity monitoring application. Following the multi-step ahead point *forecasting* task described in Chapter 2, the goal is to predict at time  $t$  the  $h$  upcoming values to be observed in the stream based on historical data.

### 6.1.1 COMPETITION PROTOCOL AND TASK

The competition is run continuously for a specified duration where the test and training data are issued in the stream with regards to the batch size as well as the time intervals defined in the competition settings. The flow is made available to all the participants at the same time and can be consumed independently. The competition was held in two phases. In the first phase, which is not be taken into account for the final ranking, participants could build and train their models and setup their environment based on one part of the data stream. They will have enough time to tune their models and evaluate them against the baselines. In the second phase is online where participants continuously receive batches coming from the rest of the data stream to test their models and submit their predictions. Predictions are submitted online as well and evaluation is performed in a prequential setting. Since the data are released in a streaming fashion including small batches, participants have to submit their predictions for the current batch before the next batch is released. Otherwise, they are penalized with a worst-case value (a value is defined for each evaluation metric).

Figure 6.1 illustrates the general workflow of the real-time competition.

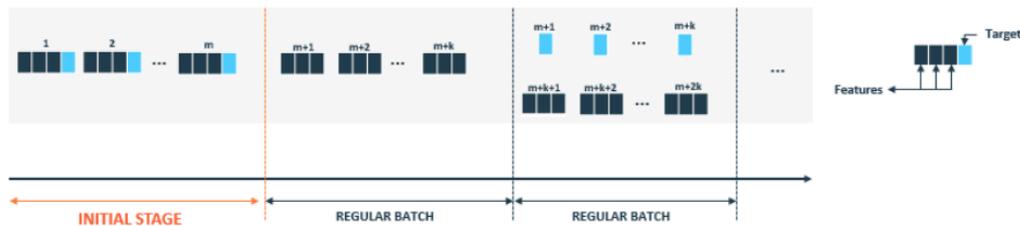


Figure 6.1: Competition streaming workflow

- **Registration/Subscription:** all participants are asked to register to the competition platform. Thereafter, each user can subscribe to the competition and get a secret key to be used subsequently to secure the bi-directional stream communication. Data is released within a specified schedule and following a predefined format detailed in a *Protobuf* file. All participants must respect the predefined format when submitting predictions.

<sup>1</sup><https://bigmine.github.io/real-time-ML-competition/index.html>

- **Submission:** users must meet the technical requirements in order to create an appropriate machine-learning environment for supporting the streaming setting. The **gRPC** and **Protobuf** are required and support several programming languages such as *Python* and *Java* to provide more freedom to participants in setting up their environment. Test and learning instances are released in batches in regular time interval. When a test batch is released, participants must submit their predictions before a specified due date, otherwise, they will be penalized with a worst-case value. Participants are allowed to use any additional libraries and resources for this competition, that includes: hardware setup, software setup, programming language, and additional data.
- **Evaluation and Results:** submitted predictions are evaluated online and results are updated live and displayed in the web application. Evaluation metrics comprise common loss functions used for forecasting as discussed in Section 2.1.7.2.

Participants must build models able to predict the future values of the series with high accuracy while taking into account the evolutionary nature of the data stream. These models must be able to learn incrementally and detect changes in order to adapt to them as soon as they occur following the requirements enumerated in Section 2.1.3. When the competition starts, data are released as a continuous stream of training and test batches as defined in the competition settings.

### 6.1.2 COMPETITION DATA

Telecommunication companies and network providers like Orange are very concerned about the behavior of their network to ensure high-quality customer services. They collect data in real-time about network traffic and need to extract valuable knowledge to learn and predict future behavior for capacity planning or anomaly detection in case of malicious attacks or malfunctioning devices. It is of great importance to process the data online and make fast decisions when required. The temporal data stream used in the competition consists of many recorded parameters related to quality of the signal, loss in packet transmission or traffic intensity. Our competition proposal addresses the network activity analysis application that can be used for the following scenarios:

- **Capacity planning and activity prediction:** Predicting metrics about the network such as the number of devices and number of messages that passed through the network will allow companies to predict future behavior and deploy necessary resources when needed such as network expansion with new devices.
- **Anomaly detection:** Monitoring certain metrics may reflect malfunctioning devices among the network such as signal strength, noise ratio, and packet loss. The goal is to predict the future value of these metrics and spot abnormal values and trigger alarms.

The data collected from the network activity is recorded at regular time intervals (every second for example) summarizing the density of communications in terms of the number of messages conveyed. The dataset used in the competition is synthetic, yet realistic. In fact, the characteristic of the data such as serial correlation and seasonality were simulated from real-world time-series and several transformations were applied to inject noise, anomalies and

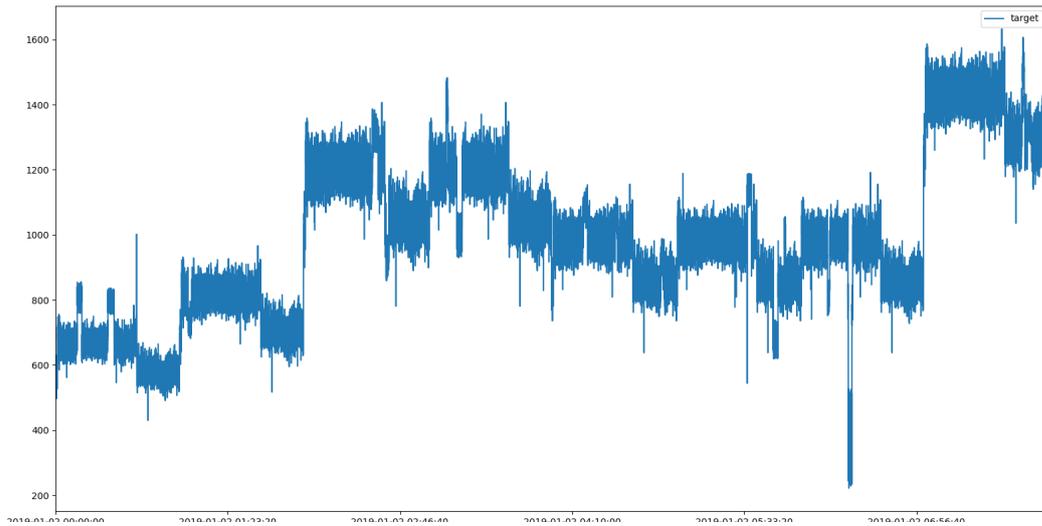


Figure 6.2: Competition temporal data stream

concept drift. In fact, the temporal stream was deliberately designed to meet the challenges of stream mining, including:

- **Concept drift:** stands for a change in the statistical properties of the target variable that we are trying to predict. The change can be *Abrupt* or *Gradual* or *Incremental* as detailed in Section 2.1.4.
- **Anomalies:** stand for patterns in data that do not conform to a well defined notion of normal behavior [43]. We address two types of anomalies: *Point* and *Group* anomalies. If an individual data instance can be considered as anomalous to the rest of the data, then the instance is referred to as a point anomaly. On the other hand, group anomaly is a continuous set of data points that are collectively anomalous even though the individual points may or may not be point anomalies.

Figure 6.2 illustrates a part of the data stream used to run the competition including concept drift and anomalous data.

### Competition stream setting

In order to set-up the competition, we provide the following settings that describe the way data are released in the stream:

- **Initial batch:** a first batch was provided comprising data collected over 2 days with a record every second summarizing the number of messages conveyed in the network. A period of time was granted to competitors to tune their models at first, before the stream of test instances is released to ensure a warm start.
- **Regular batch:** a batch of fixed size  $\beta = 5$  is released in the stream at regular time intervals (5 seconds). The latter contains  $\beta$  new test instances to be predicted before a predefined submission deadline. This translates to a multi-step ahead forecasting task with a horizon  $h = 5$ . The deadline stands for the expected time of arrival of

the next batch. Each batch contains the true target values corresponding to the test instances released in the batch before. If predictions are submitted after the due date, the competitor is penalized. There is no delay in the arrival of the true target values for updating.

### 6.1.3 WINNING SOLUTIONS

In this section, we present the solutions which achieved the best scores. Three solutions that yielded the best results are:

#### 6.1.3.1 Adaptive moving average on $n$ previous equally distant windows method (AMA) - proposed by Ghislain Fievet:

This model takes advantage of the periodicity of the released data stream. It takes into account the current window and previous  $k$  windows where the distance between all the windows is the period of the stream computed using Partial Auto-correlation Function (PACF) using the initial batch. The size of the window is updated during the stream and the average value of

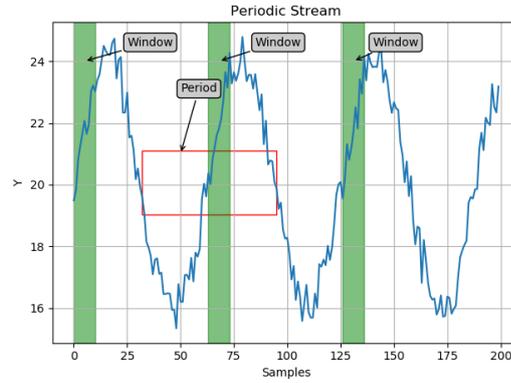


Figure 6.3: Windows with distance equal to stream period

the target is computed for each window.

$$window\_mean_k = \frac{1}{win\_size} \sum_{i=t-k*period}^{t-k*period+win\_size} y_i \quad (6.1)$$

The model keeps only  $n$  windows whose mean is the closest to the mean value of the current window. This avoids considering the mean of the windows during anomalies periods. The prediction for next  $h$  values is made by computing the distance between the mean of each window and every one of next  $m$  points following the window.

$$prediction\_mean_l = \frac{1}{n} \sum_{i=1}^n |Y_l - window\_mean_i| \quad (6.2)$$

where  $l$  in  $[1, h]$ . The prediction is then computed as the sum of the mean value of current window and average distance for every of  $m$  points.

$$prediction_l = (current\_window\_mean + prediction\_mean_l) \quad (6.3)$$

Additionally, the model adapts its prediction methods to deal with anomalies. Anomalies are assumed to be the values whose distance to the window mean value is greater than the threshold  $\theta$ . If the prediction is to be made inside, what is detected as, anomaly period, the prediction is made by simple moving average algorithm. The method requires 9 parameters such as the windows size, the threshold to detect anomalies, the number of windows to consider. During the competition a custom genetic algorithm makes these parameters evolve to fit as best as possible the past 30 minutes of released data.

### 6.1.3.2 Moving average method with Robust Random Cut Forest for anomaly detection (MA-RRCF) - proposed by Sohn Jimin:

The second solution uses moving average method and Robust Random Cut Forest (RRCF)[75] to predict the next values. The prediction model, is based on the assumption that the temporal data stream has increasing, monotonous pattern and thus using moving average method seems adequate. The training phase consists of keeping the last  $m$  values, where  $m$  is the size of the batch in the stream, and computing the average value:

$$prediction_t = \frac{1}{m} \sum_{i=t-m}^{t-1} y_i \quad (6.4)$$

The average value of last  $m$  points is then used for the whole next batch and then updated in the same way. To deal with anomalies the solution uses RRCF. RRCF provides anomaly detection on streaming data. It uses the ensemble of independent Robust Random Cut Trees as a sketch of the input stream and constructs a tree of 10 – 1000 vertices from a random sampling of the pool and creates more trees of the same size 1-1000 times which creates the forest. It decides whether the new data point is an anomaly or not by injecting it into the trees and analyzing the extent of changes experienced in the complexity of the forest(*e.g.* depth). Previous values in the stream are used to build the trees and new values are inserted and the average displacement degree is computed to decide if the point is an anomalous or no. If the degree is larger than a given threshold, that point is assumed to be an anomaly and replaced by the average computed on the whole stream of data released so far.

### 6.1.3.3 Mondrian Forest with Robust Random Cut Forest for anomaly detection (MF-RRCF) - proposed by Yeonwoo Nam:

The third solution uses regression Mondrian Forest[100] for prediction and RRCF for anomaly detection. Mondrian Forests are fast incremental random forests that use Mondrian Processes[138] to build the ensembles of random decision trees. For this specific use case, since the data stream represents the univariate time series, to use Mondrian Forest to make predictions the data had to be transformed. They used auto-regressive representation to built feature vector including the  $p$  previous values as described in Section 2.1.6.1. The second part of the solution addresses anomaly detection on data streams. Similarly to the previous, RRCF is used to detect anomalous data points. However, instead of replacing the target value of the anomalous point with the average observed on the whole stream. they propose to average value the last received batch. This is based on a crucial assumption in temporal data streams that recent instances are more important than old ones.

### 6.1.4 COMPETITION RESULTS

In this part, we present the results achieved by the mentioned solutions. We present the results achieved during the competition and following all the rules of the competition which include penalties for late predictions or not sent predictions at all. After we present the results that have been achieved during independent testing of provided solutions, on a wider test set and without any disturbance where prediction has been sent for all records in the stream on time. We evaluate models using Mean Absolute Percentage Error(MAPE), described in Section 2.1.7.2. Also, we note the time latency, to measure how fast were models in sending predictions.

We compare proposed methods against the baseline model *MEAN*, which predicts at time  $t$  the next values as the average value on whole data stream released up to time  $t$ .

Table 6.1 shows the competition results in terms of predictive performance measure in Mean Absolute Percentage Error *MAPE* and latency that stands for the time required to the participant to submit the predictions for next batch. Baseline model *MEAN* is deployed on the competition platform and does not suffer from delays related to network communication which explain the large difference compared to other competitors latency. The results show that Adaptive Moving Average *AMA* proposed by *Ghislain Fievet* performs best by scoring the lowest *MAPE* error and the minimum latency among all other participants.

Team	Method	MAPE	Latency[s]
Ghislain Fievet	<i>AMA</i>	12.71261	0.68632
Sohn Jimin	<i>MA – RRCF</i>	13.10381	0.91592
Yeonwoo Nam	<i>MF – RRCF</i>	13.73831	1.04440
Baseline	<i>MEAN</i>	16.50072	0.0055

Table 6.1: Real-time Competition top results

## 6.2 SCALAR PLATFORM

There exists a wide range of platforms to promote and organize challenges related to data science and artificial intelligence and machine learning. Platforms such as *Kaggle*<sup>2</sup> and *Ali baba Tianchi*<sup>3</sup> to name only a few. These platforms provide public datasets and code to perform different data analysis tasks such as classification or regression. Besides, one can organize data science related competitions designed to provide challenges for competitors of varying expertise in the field of machine learning. Featured competitions are full-scale machine learning challenges which pose difficult, generally commercially-purposed prediction problems. They attract the best experts, and offer prize pools.

Similarly, *OpenMl*<sup>4</sup> for Open Machine Learning, is an inclusive movement to build an open, organized, online ecosystem for machine learning. The platform is an open source tools to discover and share interesting open datasets from different application

<sup>2</sup><https://www.kaggle.com/>

<sup>3</sup><https://tianchi.aliyun.com/>

<sup>4</sup><https://www.openml.org/>

domains, and quickly build models in collaboration with thousands of other data scientists and practitioners. Besides, one can create learning problems (e.g., classification) by creating tasks to describe the goals. OpenMl allows collaboration, benchmarking or different methods, and comparison to the state-of-the-art approaches. OpenMl tasks are machine-readable, and support to automatic data extraction to train and evaluate models. The platform easily integrates into the most popular machine learning tools and provides a wide range of APIs to download and upload datasets, tasks, run algorithms, and share the results.

Machine learning competitions are widely used in a batch setting to compare prediction models. Being able to compare incremental prediction models would be of great importance for increasing accuracy and efficiency while analyzing data in real-time. However, existing platforms such as Kaggle and OpenMl meet nearly the same model for challenges where participants download the whole training and test data sets to build models and submit their predictions before a predefined deadline. The platform then proceeds to the evaluation based on all the submissions and select the best approach based on a set of measures.

To be able to organize the real-time competitions, there was a need to have a novel platform dedicated to handle online machine learning contests on data streams. To the best of our knowledge, there is no such platform for organizing competitions that meet these criteria. Out of all platforms already available, none of them meets the streaming setting requirements. We present SCALAR [126], a new platform for running real-time machine learning competitions on data streams. Following the intent of Kaggle, which serves as a platform for organizing machine learning competitions adapted for batch learning, SCALAR is a novel platform explicitly designed for stream learning in real-time and support both classification and regression problems. It was developed in Python, using state of the art open-source solutions such as Apache Kafka, Apache Spark, gRPC, Protobuf, and Docker.

SCALAR allows creating competition by providing specific data in the form of CSV file that will be used to recreate a continuous stream using specific settings such as the time interval between two releases and the number of instances to be included in the batch. It supports a wide range of evaluation metrics depending on the problem under study (classification, regression ...). Submitted predictions are evaluated online, and results are updated continuously and displayed on the dedicated competition web page. SCALAR platform allows participants full independence and freedom in choosing appropriate hardware and software to take part in the competition and enhance predictive performance as well as time latency. Figure 6.4<sup>5</sup> details the architecture of the platform that includes a web application and a streaming engine following the fundamentals of Information Flow Processing (IFP) [48].

The SCALAR platform supports multiple bi-directional streams for several users at a time. A user first creates an account and signs in to the platform before subscribing to a specific competition. Every subscriber gets a unique secret credential to connect to a secure channel dedicated to the competition and ensure safe and continuous data communication. Every competitor can use his environment to perform machine learning models using either Python, Java or R languages. Competition data is first read from the respective CSV file and then pushed to the competition stream. Training and test instances are issued in the right order and at the right moment according to the competition settings. A bi-directional streaming flow

---

<sup>5</sup>All icons are open and come from <https://www.flaticon.com/>

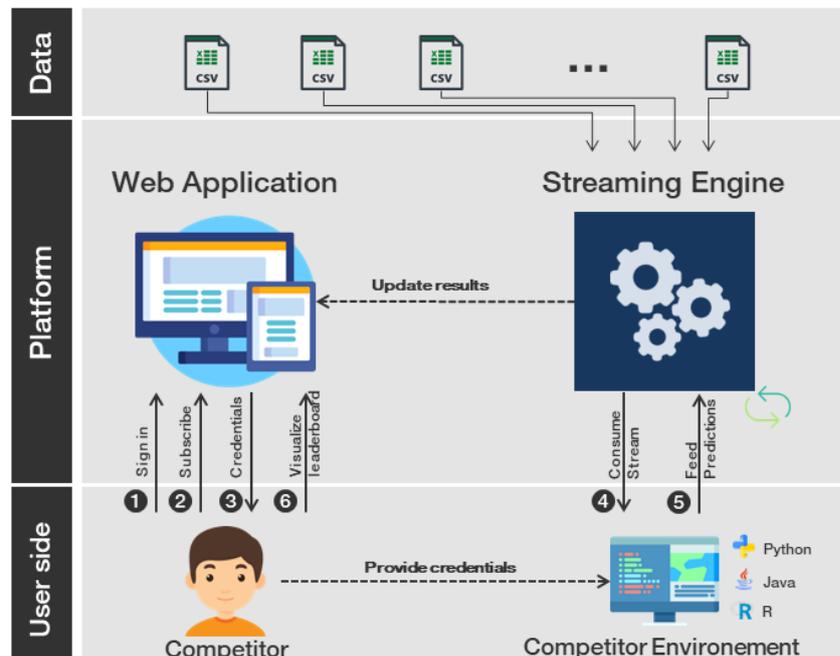


Figure 6.4: Platform architectures and services

is dedicated to each user to consume training and test instances and submit the respective predictions according to the competition indications. Predictions are then processed and evaluated online to allow real-time updates on the competition results and rankings. The web application is here to allow users to browse through competition description and live results and rankings.

The platform was designed to provide users as much freedom as possible in choosing their own resources for building models. They can use any setup they desire as long as they are able to connect to the secure channel and send the predictions on time and in the right format. Users are also allowed to use external data sources if they think that can help improve their models.

SCALAR architecture comprises two principal components: streaming engine and web application. Participants will be prompted to interact with the both during the competition. The streaming engine is responsible for ensuring bi-directional streams to provide test and training data and receive predictions from every participant. This component will mainly inter-operate with the developing environment (Python, Java or R). The web application offers a plethora of options allowing users to browse through competitions' information and real-time results and rankings. The two components communicate with each other in order to update users' performance based on the predictions received.

### 6.2.1 STREAM SERVER AND COMMUNICATION

The data stream server is the module that is responsible for pulling data from different sources and recreating a stream for every competition based on the settings in terms of batch size and time interval. The streamer releases data to separate streams that can be read by multiple users at the same time. A user may subscribe to many competitions at a time. Figure 6.5 illustrates the multiple bi-directional streaming setting.

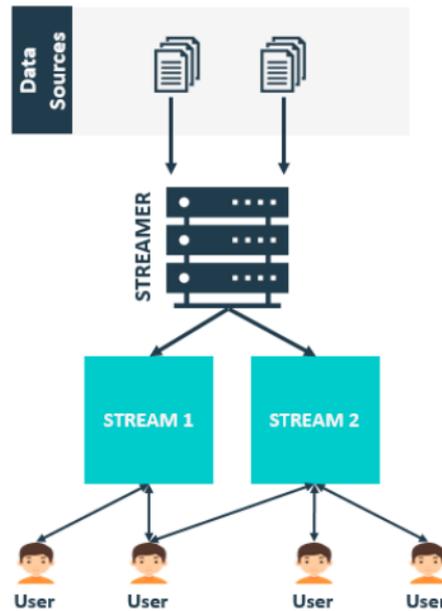


Figure 6.5: SCALAR data stream server workflow

The Streaming engine and client communication are based on **gRPC**<sup>6</sup> framework and **Protobuf**<sup>7</sup>. They provide secure communication, full-duplex bidirectional streaming and an easy way to describe services. Besides, the framework is language- and platform-neutral and supports several programming languages (Java, Python, Go, C#, Ruby). Users must install **gRPC** package and compiler for **.proto** file in order to be able to communicate with the streaming engine. **gRPC** is an open Remote Procedure Call(RPC) framework that can run on different platforms( more than 10 supported languages). It enables client and server applications to communication and and exchange data. It is used to build highly scalable, low latency and distributed systems that connect services, mobile applications, real time communication and IoT while at the same time ensuring efficiency in CPU use and bandwidth.

Second requirement for successful communication is **Protobuf** that stands for **Protocol buffers** and is a language-neutral, platform-neutral, extensible way of serializing structured data for use in communications protocols or data storage. **Protobuf** is thought in the same way as XML but smaller, faster and simpler. SCALAR uses **Protobuf** file for each competition to describe the structure of messages conveyed in the bi-directional stream including test and training batches as well as expected predictions. The user needs to download the **.proto** file from competition page and compile it to generate new files containing data structures and classes to be used for future communication. An example of the **.proto** file is shown in figure 6.6.

The **.proto** file the following parts:

1. We define the syntax and some parameters that are related to the languages that we will be used on the user side environment.

<sup>6</sup><https://grpc.io/>

<sup>7</sup><https://developers.google.com/protocol-buffers/docs/overview>

```

syntax = "proto3";
option java_package = "ex.grpc";
package file;

```

← 1 - Syntax

```

// The data service definition.
service DataStreamer {
// Sends multiple greetings
rpc sendData (stream Prediction) returns (stream Message) {}
}

```

← 2 - Service definition

```

message Message{
int32 rowID =1;
string Day = 2;
string Period = 3;
int32 Target = 4;
string Deadline = 5;
string Released = 6;
string tag=7;
}

message Prediction{
int32 rowID = 1;
int32 Target= 2;
}

```

← 3 - Data format definition

Figure 6.6: *.proto* file example

- We define the service to be called, which in this case is `DataStreamer`. It is defined as bi-directional streaming service where the user sends stream of `Predictions` and receives stream of `Messages`.
- We define the data formats used for communication `Message` and `Prediction`. The structure depend on the competition and dataset used and define the format of message the client receives from the server (vector of features and target) and the expected format of the predictions. The messages that do not correspond to one of the formats will not be conveyed.

### 6.2.2 WEB APPLICATION

SCALAR provides a user-friendly web application where users can register and connect in order to create or participate to real-time competitions. There are two type of user profiles, organizers and competitors. Organizers can create competitions and provide data to run a challenge in real-time according to the provided settings. Creating a stream requires a CSV file in order to be added to the platform as described in Figure 6.7. Figure 6.8 shows the interface to add a competition to the system and requires an existing data stream as well other settings to determine the flow of the challenge. Once the competition is added, it can be accessed in the global list of competitions and users can participate.

A competitor must first subscribe to a competition in order to connect to the stream server, read data stream and submit predictions. A secret key is allocated to each user as described in Figure 6.9 and users must provide this secret key as a part of the credentials needed to authenticate to the stream server.

Once the user has registered, subscribed and prepared the environment, he should be ready to participate in the competition. Start the competition on time is important, so that the competitor can benefit from all the training time to prepare the model. When the competition

**Add Datastream** ✕

Name  
 ☰ RTC Rregression

Description  
 T Daily Energy Consumption 24/255

📎 Choose File previ\_conso18-04-2017.csv

CANCEL    ADD

Figure 6.7: SCALAR web interface to create a stream

**Add Competition** ✕

Name  
 ⚙ Regression RTC

Datastream  
 ⚙ RTC Consumption ▼

🔒 Has Target

🔒 Valeurs ☰ -

Target  
 🔒 Valeurs

Figure 6.8: SCALAR web interface to create a competition

**Regression RTC** **UNSUBSCRIBE**

🕒 Started at : Mon, 18 Sep 2017 11:00:00 GMT 📅 Ends at : Tue, 19 Sep 2017 00:59:00 GMT 📄 code: J2

🔒 Secret Key  
 eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjb21wZXRpZGl9pZCI6ImJAsInVzZXJfaWQiOiJkaWhpYS5ib3'

Figure 6.9: SCALAR web interface with subscription information

starts, users connect and receive the first training batch containing several instances to ensure a warm start and tweak the predictive model. Following this step, each participant to the

competition receives regular batches and submits their respective predictions. Online results can be tracked on the competition page including

- Leaderboard - graphical representation of contestants scores (Figure 6.10),
- Ranking - ranking list of the contestants on selected evaluation metrics (Figure 6.11).

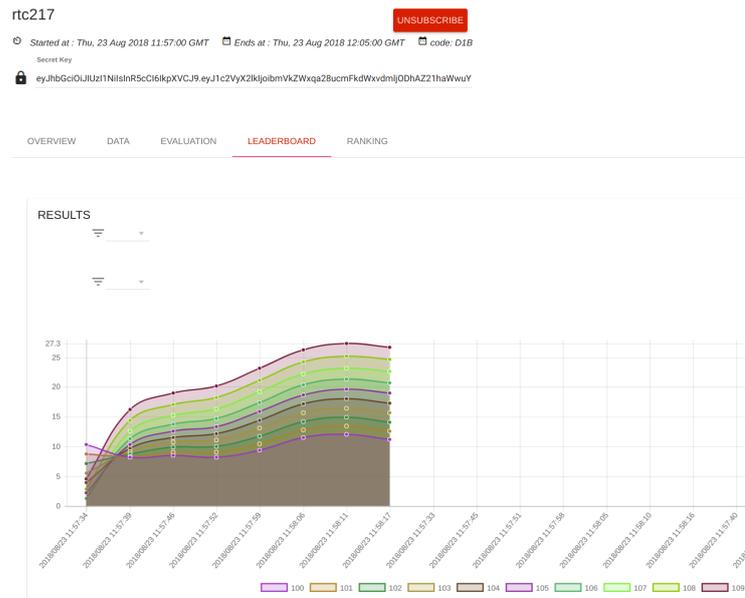


Figure 6.10: Live results on SCALAR web application

The screenshot shows the 'Ranking' tab of the SCALAR web application. At the top, it displays the competition ID 'rtc217', a red 'UNSUBSCRIBE' button, and the start/end dates: 'Started at: Thu, 23 Aug 2018 11:57:00 GMT' and 'Ends at: Thu, 23 Aug 2018 12:05:00 GMT'. Below this is a 'Secret Key' and a long alphanumeric string. The navigation menu includes 'OVERVIEW', 'DATA', 'EVALUATION', 'LEADERBOARD', and 'RANKING' (highlighted). The main content area features a search bar with a magnifying glass icon and the text 'search'. Below the search bar is a table with the following columns: '#', 'First Name', 'Last Name', 'Mail', and 'Valeurs'. The table contains 10 rows of data, all with 'Test' as the first name and 'Baseline' as the last name. The 'Valeurs' column shows scores ranging from 13.09621907 to 30.7443759.

#	First Name	Last Name	Mail	Valeurs
1	Test	Baseline		13.09621907
2	Test	Baseline		14.85506211
3	Test	Baseline		16.68074304
4	Test	Baseline		18.53761056
5	Test	Baseline		20.41250596
6	Test	Baseline		22.33632411
7	Test	Baseline		24.31987351
8	Test	Baseline		26.38848188
9	Test	Baseline		28.54979011
10	Test	Baseline		30.74437591

Figure 6.11: Participants ranking on SCALAR web application

### 6.3 SUMMARY

We have organized the first-ever real-time machine learning competition on data streams. We have conducted an online machine learning challenge on a real-life use case dealing with temporal data streams forecasting. Our contribution in this chapter was twofold: **(i)** We have conducted a real competition involving several competitors to solve a pending industry challenge on network activity monitoring. **(ii)** We have developed a novel platform for data science challenges on data streams.

The goal of the competition was to predict the future behavior of the temporal stream that falls within the task of *forecasting*. We have presented the three best solutions proposed by the competitors that were evaluated using the MAPE measure to assess the quality of submitted predictions while penalizing late ones. The time latency is of crucial importance for online learners. In order to run the competition, we have developed a dedicated platform SCALAR, designed to meet the real-time requirements of the streaming competition. The SCALAR platform handles secure and independent bidirectional streams to serve test and training data to all competitors and receive their predictions in a streaming fashion. It implements real-time evaluation and provides a user-friendly web application to browse through the competitions and the leaderboard results. We strongly believe that having such dedicated platform machine learning competitions on data streams could provide a strong contribution to the field of data stream mining.

## Conclusions and Future Work

Temporal data streams are essential to represent real-world phenomena from different application domains including infrastructure monitoring and planning. Forecasting future behavior in the context of streaming data is a very challenging task due to the non-stationary and evolving nature of the data as well as the computational constraints. Stream mining paradigm aims at extracting valuable knowledge from data flows to support decision-making in different applications fields such as health, infrastructure monitoring, to name only a few.

In the present work, we investigated the problem of evolving temporal data stream forecasting using dynamic ensemble methods while addressing the aforementioned challenges. In particular, we focused on dynamic ensemble selection methods as a promising approach to improve predictive performance. Besides, we addressed the computational limitations using model compression in order to train a smaller model with comparable performance while reducing their complexity. This chapter concludes the thesis with a discussion on the work achieved and future research directions.

### 7.1 CONCLUSIONS

The first part of the thesis deals with the one-step-ahead point forecasting task in the context of evolving data streams using ensemble methods. We addressed the following question:

*How can we dynamically select and combine a set of forecasting models and cope with the changing dynamics of temporal data streams?*

We focused on Dynamic Ensemble Selection (DES) methods that aim at selecting, on the fly, the most accurate models only from a pool of diverse models in order to combine their predictions. First, we proposed different methods to assess the predictive performance of each individual model in the pool in order to select the most accurate ones according to the data at hand. Evaluating models' performance in the context of temporal data streams can be

divided into two main categories: windowing and meta-learning. Windowing approaches are based on the assumption that future values are very likely to behave like instances from the recent past. We proposed SLOPE that estimates the performance of models using the notion of "locality" or "vicinity". A local region is computed using a dual criterion: temporal locality using a sliding window as well as a "feature" based proximity using the nearest neighbors. On the other hand, meta-learning based methods are more proactive and comprise two key components, namely meta-features and meta-model. Meta-features stand for the set of characteristics extracted from the data that best describe the underlying generating process. Meta-features are then used to train a meta-model in order to predict the performance of individual models on unseen instances. We investigated different meta-features including statistical, landmark and stream specific characteristics that are deemed to be the best characteristics to relate to the predictive performance to the data.

Subsequently, we have considered several methods to select expert base models in a committee based on their estimated expertise stemming from the previous step. Experts' predictions are then combined through a weighted average where more weight is granted to the best performing models. Existing selection methods are mainly based on trimming, i.e a fixed percentage of the best performing forecasters are selected in the committee regardless of the relative value. We have proposed a self-adaptive abstaining method where the less confident models are allowed to abstain based on a set of reliability estimators and required competence threshold. Besides, we have presented a randomized selection method that aims at enhancing diversity with less human intervention knowledge. The higher the predictive performance, the more likely is the forecaster to be selected. Finally, we have introduced a selection approach based on a combined criterion as a trade-off between predictive performance and diversity. In fact, diversity is an important aspect in the success of ensemble methods.

We have proposed in Chapter 4 to cast the meta-learning based DES method as a Multi-Target Regression (MTR) task where the goal is to simultaneously predict all predictive performance of the forecasters in the pool in order to explicitly capture models' dependency and improve overall predictive performance.

The second part of the thesis tackles the complexity drawbacks of dynamic ensemble methods. In fact, despite their high predictive performance, DES methods are greedy resource consuming in terms of time and space. This makes the use of ensemble methods unpractical in applications where resources are limited such as IoT and embedded systems. In this second part, we addressed the following question:

*How can we train a single model that has comparable performance compared to a complex dynamic ensemble while considerably reducing its computational cost ?*

We have investigated Model Compression (MC) for dynamic ensembles of forecasters in the streaming setting using the Student-Teacher framework that consists in training a single predictive model (Student) to mimic the behaviour of an ensemble (Teacher) [31]. Different teaching scenarios to induce the student model to leverage both teacher's predictions as well as the true value of the target were proposed. The first scenario is strongly supervised where the student model has access to all teacher's predictions. However, this scenario is impractical for streaming data applications as it requires permanent communication between the Student

and the Teacher whilst resources are limited and communications costly. The second and third scenarios are weakly supervised and more suitable in the streaming context. In fact, the student invokes the teacher at specific times only based on the student performance in order to learn from teacher's predictions. Besides, we have proposed an adaptive model compression approach to better cope with concept drift and changes that may damage the predictive performance of student models. We used an informed adaptation strategy that trains a background student when a warning is triggered and replaces the student model when a warning escalates to a drift.

We presented in Chapter 6 the "Real-time Machine Learning Competition on Data Streams", organized in the scope of the BigDataCup Challenge of the IEEE Big Data 2019 conference. Competition data are released in the form of stream of small batches to build incremental learning models and participants continuously submit their predictions in the form of stream as well. A dedicated platform SCALAR was used to host the competition and released as an open source framework.

## 7.2 FUTURE DIRECTIONS

We highlight some promising research directions and provide recommendations on potentially future studies.

### 7.2.1 HANDLING DELAYED LABELS IN TEMPORALLY EVOLVING DATA STREAMS

In the thesis, we formalized *Forecasting* task in streaming setting assuming that all labels are available immediately (no delay). However, the assumption is unrealistic. Indeed, in real-world applications labels for previously observed feature vectors may only arrive after considerable lag meaning that the labels are delayed. Delayed labels are an important aspect of streaming analysis and particularly for the forecasting task as the uncertainty increases. Appropriately handling delayed labels in the context of a temporally evolving data streams can improve the predictive performance

### 7.2.2 BEHAVIORAL DATA AND EXCEPTIONAL (DIS)AGREEMENT

The behavioral data consist of a set of individuals (models) who express outcomes (predictions) on entities (instances). The main objective of behavioral data analysis is to "breakdown individuals (users) into groups to gain a more focused understanding of their behavior" [116]. This offers a great opportunity to study the behavior of individual models and the interactions between them. In particular, such data can be leveraged to investigate and search for exceptionally consensual/controversial data. In fact, identifying contexts where individual models exhibit an exceptional (dis)agreement pattern compared to what is usually observed can translate to a drift in the data. One possibility is to study how groups of individual models sharing the same data and the same task behave with regards to specific data over time. This can also contribute to the transparency and interpretability of DES methods that often behave like black boxes. In fact, developing forecasting methods which are able to provide some sort of explanation for the predictions could improve the decision-making process.

### 7.2.3 CHANGE DETECTION FOR MULTI-VARIATE DATA

Concept drift detection is a key aspect in streaming data analysis and has been widely investigated for uni-variate data but much fewer works address the problem of detecting changes in multi-variate data streams. We presented in Chapter 4 the Multi-Target Regression (MTR) based meta-learning DES approach that explicitly considers model's dependencies while learning their respective behavior.

A straightforward extension to the multi-variate is to independently inspect each component of the data stream with a single uni-variate change detector. Nonetheless, this approach does not provide a true multi-variate solution, and is unable to detect changes affecting the correlation among different targets. One multi-variate change detection approach consists in computing the log-likelihood of the data stream and compare the distribution of the log-likelihood over different time windows as proposed by Kuncheva [97]. As a matter of fact, using multi-variate change detection with the MTR regression models to monitor the loss can help improving the predictive performance as described for the adaptive random forest ARF [66].

### 7.2.4 MODEL COMPRESSION FOR STREAMING DATA

Hinton, Vinyals, and Dean [76] introduced the concept of distilling knowledge from a larger teacher model onto compact and faster compressed student model by training the student on softened teacher output distribution in this case of classification task. Softened outputs reveal the dark knowledge in the ensemble and provide much richer information about the data compared to hard targets. This helps the student learn more generalized features compared to single labels. However, the dark knowledge is unavailable in the case of uni-variate point forecast. As discussed in Section 2.1.2, a prediction interval gives an interval within which we expect  $y_t$  to lie with a specified probability. This offer an unparalleled opportunity to extract dark knowledge on the forecasting task and improve the Student-Teacher model compression described in Chapter 5.

# Datasets and Forecasters

## A.1 TEMPORAL DATA STREAMS

In this section, we describe the temporal data streams used in the experiments conducted in Chapters 3, 4 and 5. The data comprises 30 temporal data streams from several domains of application such as IoT where each stream is a univariate sequence of continuous/numeric observations captured at regular intervals. Besides, each stream contains at least 10000 observations. Table A.1 summarizes the set of temporal data streams and details their respective characteristics and sources.

We describe in more details each temporal data stream as follows:

- 1–3 Streams with ID 1–3 falls within the field of smart cities and represent traffic data at different junctions. Instances were collected every hour from November, 1, 2015 to June, 3, 2017 and serve to understand traffic patterns of the city.
- 4–5 In this temporal data stream, instances are sampled every minute from April 1, 2018 to August 31, 2018 and represent water pump sensor data deployed in cities in order to detect and predict failures <sup>1</sup>. Detecting failures is crucial as it directly impacts several families daily life.
- 6–11 These temporal data streams are collected from different sensors deployed in several rooms of 4 different floors of the Sutardja Dai Hall(SDH) at UC Berkeley [80]. Each sensor captures several measurements and among them CO2 concentration over a period of one week from Friday, August 23, 2013 to Saturday, August 31, 2013 at a frequency of an observation every 5 seconds.
- 12 The data is provided by and industrial company Schneider-Electric<sup>2</sup> providing energy and automation digital solutions for efficiency and sustainability. Industry automatisms

---

<sup>1</sup>Available at <https://www.kaggle.com/nphantawee/pump-sensor-data/discussion/131429>

<sup>2</sup><https://www.se.com/ww/en/>

Table A.1: Data streams and respective summary

ID	Stream	Data source	Data characteristics	Size
1	Junction 1			
2	Junction 2	Analytics Vidhya	Hourly values from Nov. 1, 2015 to Jun. 3, 2017	14582
3	Junction 3			
4	Sensor 2	Pump sensor data	Observation every minute from Apr. 1, 2018 to Aug. 31, 2018	220320
5	Sensor 27			
6	Room 413			
7	Room 415			
8	Room 417	Smart building [80]	Observation every 5 seconds from Aug. 23, 2013 to Aug. 31, 2013	130912
9	Room 419			
10	Room 422			
11	Room 423			
12	Sensor fault	Sensor fault	Observation every minute from Mar. 1, 2017 to May 7, 2017	62629
13	Metro Traffic Volume	Metro traffic	Observation every minute from Oct. 2, 2012 to Sep. 30, 2018	48204
14	T1			
15	RH1	Energy Data	Observation every 10 minutes from Jan. 11, 2016 to May 27, 2016	19735
16	T9			
17	RH9			
18	Humidity			
19	Windspeed			
20	Total	Bike sharing	Hourly observations from Jan. 1, 2011 to Dec. 31, 2012	17372
21	Total registered			
22	Total casual			
23	Synth1			17280
24	Synth2	Synthetic data		21600
25	Synth3			28800
26	Synth4			21600
27	CO_AQI	Pakistan air quality		24384
28	DEWP			
29	O3	Beijing air quality [171]	Hourly observation from Mar. 1, 2013 to Feb 28, 2017	35064
30	Pressure			

systems are becoming more and more complex and working in evolutionary environment. It is vital to develop efficient fault detection methods in order to predict and locate malfunctioning operations in order to improve the performance and productivity and lessen the dramatic consequences of failures <sup>3</sup>.

- 13 The data contains Hourly sampled instances of the westbound traffic volume Minneapolis interstate traffic <sup>4</sup>.
- 14–17 The data falls within the appliances energy use in a low energy buildings field [32] where observations are recorded every 10 minutes from January, 11, 2014 to May 27, 2014 (about four months and a half). Measures about the temperature and humidity conditions were monitored in different rooms using wireless sensors <sup>5</sup>.
- 18–22 Data are collected from a bike-sharing and rental system and contains instances recorded on an hourly basis between January, 1, 2011 and December, 31 2012 in Capital bikeshare system in Washington, DC with the corresponding weather and seasonal information such as average levels of humidity and average windspeed along with the total number of rentals [54].

<sup>3</sup>Available at <https://www.kaggle.com/arashnic/sensor-fault-detection-data>

<sup>4</sup>Available at <https://www.kaggle.com/mikedev/metro-traffic-volume>

<sup>5</sup>Available at <https://www.kaggle.com/loveall/appliances-energy-prediction>

- 23–26 Synthetic temporally dependent data were generated using TSimulus<sup>6</sup> designed to generate random, yet realistic, time series data. TSimulus provides tools for specifying the shape of a time series including general patterns, cycles, noise, etc<sup>7</sup>.
- 27 The dataset contains monoxide carbon (CO) measures where instances were collected every five seconds for a week from August,9 2018 to August,17 2018. All the values have been converted to the Air Quality Index (AQI) format provided by the World Health Organization(WHO) standards<sup>8</sup>.
- 28–30 The data stream includes hourly air pollutants data from the nationally-controlled air-quality monitoring site Aotizhongxin provided by the Beijing Municipal Environmental Monitoring Center [171]. Air quality measures include dew point temperature (DEWP), O3 concentration, and pressure level recorded on an hourly basis from March 1, 2013 to February 28, 2017.

## A.2 FORECASTERS

Experiments were conducted using a heterogeneous ensemble  $M$  of size 30 comprising different forecasting algorithms with different parameters values to promote diversity within individual models. The base-models are described as follows:

- Hoeffding Tree Regressor and :
  - Model at the leaves in  $\{mean, perceptron\}$
  - grace period: is the number of instances a leaf should observe between split attempts and values are in  $\{100, 300, 500, 1000\}$
- Hoeffding Adaptive Tree Regressor: in addition to previous parameters, the adaptive tree uses ADWIN detector to detect drift and monitor sub-trees.
- Streaming K-Nearest Neighbors:
  - the number  $k$  of the nearest neighbors to search for in  $\{5, 20, 30, 50\}$ ;
  - the max window size in to store the last observed samples in  $\{200, 500, 2000\}$ ;
  - the distance metric to use for the KDTree (euclidean).
  - the aggregation method using the *median* or *mean* of the  $k$  nearest neighbors true target values.
- Simple forecasting methods:
  - Simple exponential smoothing with fading  $\{0.3, 0.35, 0.4, 0.6, 0.7, 0.8\}$
  - Naive drift:  $\{0.3, 0.35, 0.45, 0.5, 0.65, 0.8\}$
  - Mean regressor  $\{0.3, 0.35, 0.4, 0.45, 0.5, 0.9\}$

<sup>6</sup><https://github.com/cetic/TSimulus>

<sup>7</sup>Available at [https://github.com/dihiaboulegane/tsimulus\\_data](https://github.com/dihiaboulegane/tsimulus_data)

<sup>8</sup>Available at <https://www.kaggle.com/mahmedphdcs17seecs/air-quality-monitoring-dataset-pakistan>

We have used different parameters combination for each individual model in the pool in order to enhance diversity.

### A.3 BLOCKED PREQUENTIAL TRAINING

To avoid performance issues related to the cold start, a first batch of labeled instances is dedicated to retrieve base-models predictions to enforce a "warm" start to the evaluation method (windowing or meta-learning). In the original arbitrating strategy, the meta-learning layer only starts at run-time, using only information from test observations [117] which may lead to poor predictive performance. A blocked Prequential procedure was introduced in [38] to retrieve Out-of-Bag base-models' predictions in order to increase the amount of data available to train the meta-layer. Base-models' predictions are then used to estimate their respective loss/error. The blocked prequential procedure [49] splits the first available batch into  $\beta$  equally sized and sequential blocks of contiguous observations. In the first iteration, the first block is used to train all the base learners  $M^i \in M$  and the second/next block is used to test them in order to retrieve their predictions. Then, the second block is merged with the first one for training and the third block is used for testing. This procedure continues until all blocks are used for testing.

When the online phase starts, each new instance  $x_{t+1}$  in the stream is first used to retrieve base-predictions  $\vec{y}_{t+1} = \langle \hat{y}_{t+1}^1, \hat{y}_{t+1}^2, \dots, \hat{y}_{t+1}^m \rangle$  involved in computing  $\vec{e}_{t+1} = \langle \hat{e}_{t+1}^1, \hat{e}_{t+1}^2, \dots, \hat{e}_{t+1}^m \rangle$  w.r.t. to the true value of the target  $y_{t+1}$ . The computed incurred base errors  $\vec{e}_{t+1} = \langle e_{t+1}^1, e_{t+1}^2, \dots, e_{t+1}^m \rangle$  are used to update the meta-layer whereas the true value of the target  $y_{t+1}$  is used to update all base models in the pool  $M$ .

Figure A.1 illustrates the blocked prequential procedure to retrieve base models' predictions to ensure warm start.

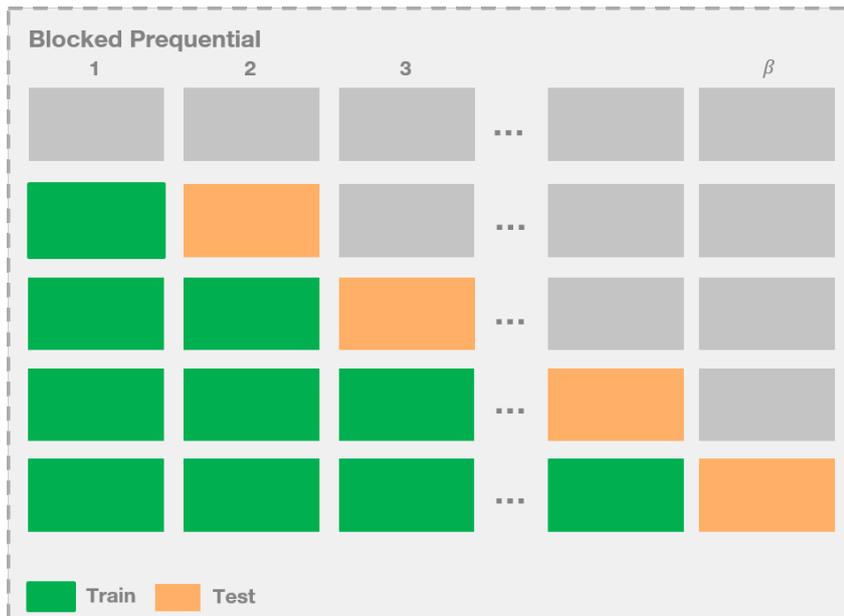


Figure A.1: Blocked prequential training workflow

#### A.4 ONLINE ENSEMBLES

- **BLAST** [135]: **Best LAST** was first proposed for classification tasks. It monitors the performance of base-models over a sliding window of recent data and selects the best one to be the only active model on the upcoming instances, when  $w = 1$ , BLAST translates to a DES.
- **AEC**: Adaptive Exponential Combination [141] is a method for adaptively combining a set of forecasters that uses an exponential weighting strategy to combine forecasters according to their past performance, including a forgetting factor to give more importance to recent values.
- **AddExp**: The Additive Expert Ensemble [92] is a general method for using any online learner for drifting concepts. It implements two pruning strategies, the 'oldest' and the 'weakest' that is often better performing one.
- **ARF**: Adaptive Random Forest for regression [69] is a streaming implementation of random forest that includes an effective resampling method and an adaptation strategy to cope with different types of concept drift.
- **STACKING**: Stacked generalization [164] is the process of learning an ensemble of heterogeneous models whose outputs will serve as features to a meta-model. Stacking model involves a two-levels learning architecture. The first level (base-models) learn on the data stream whereas the second level (meta-model) learns how to best combine the individual predictions.
- **SRP**: Streaming Random Patches [67] is based on patches that are combination of random sub-spaces of both data instances and features.
- **META-STREAM**: Meta-stream [137] is a meta-learning based DES methods that periodically builds a meta-model to predict the best (or combination of all the base-models) to be used as regressor on the upcoming instances based on data characteristics.
- **STADE**: Streaming arbitrated dynamic ensemble uses arbitration meta-learning following the ADE [38] for time series forecasting using dynamic ensembles in the batch learning.
- **SLOPE**: Sliding local performance is based the idea of "Local Performance" to estimate each individual model's performance in a local regions of the feature space surrounding a test instance. The proposed SLOPE approach is based on a dual-locality assumption for temporal data streams that considers the recency of the data using a sliding window and the feature space using the  $k$  nearest neighbors.



## References

- [1] Ratnadip Adhikari and Ramesh K Agrawal. “An introductory study on time series modeling and forecasting”. In: *arXiv preprint arXiv:1302.6613* () (cited on page 13).
- [2] Charu C Aggarwal. *A Survey of Stream Classification Algorithms*. 2014 (cited on page 10).
- [3] Marco Aiolfi, Carlos Capistran, and Allan Timmermann. “Forecast combinations”. In: *CREATES research paper 2010-21* (2010) (cited on page 26).
- [4] Marco Aiolfi and Allan Timmermann. “Persistence in forecasting performance and conditional combination strategies”. In: *Journal of Econometrics* 135.1-2 (2006), pages 31–53 (cited on pages 4, 26, 27).
- [5] Robert Anderson et al. “Recurring concept meta-learning for evolving data streams”. In: *Expert Systems with Applications* 138 (2019), page 112832 (cited on page 36).
- [6] J Scott Armstrong. “Combining forecasts: The end of the beginning or the beginning of the end?” In: *International Journal of Forecasting* 5.4 (1989), pages 585–588 (cited on page 26).
- [7] David K Arrowsmith, Colin M Place, CH Place, et al. *An introduction to dynamical systems*. Cambridge university press, 1990 (cited on page 13).
- [8] John M Bates and Clive WJ Granger. “The combination of forecasts”. In: *Journal of the Operational Research Society* 20.4 (1969), pages 451–468 (cited on pages 3, 26).
- [9] Hilan Bensusan, Christophe G Giraud-Carrier, and Claire Julia Kennedy. “A Higher-order Approach to Meta-learning.” In: *ILP Work-in-progress reports* 35 (2000) (cited on page 44).
- [10] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Communications of the ACM* 18.9 (1975), pages 509–517 (cited on page 64).
- [11] Albert Bifet. *Adaptive stream mining: Pattern learning and mining from evolving data streams*. Volume 207. Ios Press, 2010 (cited on page 100).
- [12] Albert Bifet and Ricard Gavaldà. “Learning from time-changing data with adaptive windowing”. In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM. 2007, pages 443–448 (cited on pages 15, 45).
- [13] Albert Bifet et al. “New ensemble methods for evolving data streams”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pages 139–148 (cited on page 29).
- [14] Albert Bifet et al. *Machine learning for data streams: with practical examples in MOA*. MIT press, 2018 (cited on pages 2, 4, 10, 13, 14, 20, 21).

- [15] Hendrik Blockeel, Luc De Raedt, and Jan Ramon. “Top-down induction of clustering trees”. In: *arXiv preprint cs/0011032* (2000) (cited on page 83).
- [16] Hanen Borchani et al. “A survey on multi-output regression”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.5 (2015), pages 216–233 (cited on pages 81–83, 85).
- [17] Zoran Bosnić and Igor Kononenko. “Comparison of approaches for estimating reliability of individual regression predictions”. In: *Data & Knowledge Engineering* 67.3 (2008), pages 504–516 (cited on page 50).
- [18] Dihia Boulegane et al. “Real-time machine learning competition on data streams at the IEEE big data 2019”. In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pages 3493–3497 (cited on page 114).
- [19] George EP Box et al. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015 (cited on pages 2, 10, 11, 16, 19).
- [20] Pavel Brazdil et al. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008 (cited on pages 35, 43).
- [21] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24.2 (1996), pages 123–140 (cited on page 29).
- [22] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pages 5–32 (cited on pages 25, 29, 31).
- [23] Leo Breiman. *Classification and regression trees*. Routledge, 2017 (cited on page 83).
- [24] Leo Breiman and Jerome H Friedman. “Predicting multivariate responses in multiple linear regression”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 59.1 (1997), pages 3–54 (cited on page 83).
- [25] Sebastian Briesemeister, Jörg Rahnenführer, and Oliver Kohlbacher. “No longer confidential: estimating the confidence of individual regression predictions”. In: *PLoS one* 7.11 (2012), e48723 (cited on page 49).
- [26] Alceu S Britto Jr, Robert Sabourin, and Luiz ES Oliveira. “Dynamic selection of classifiers—a comprehensive review”. In: *Pattern recognition* 47.11 (2014), pages 3665–3680 (cited on pages 4, 32, 79, 80).
- [27] Peter J Brockwell et al. *Introduction to time series and forecasting*. Springer, 2016 (cited on pages 2, 11, 16).
- [28] Gavin Brown et al. “Diversity creation methods: a survey and categorisation”. In: *Information fusion* 6.1 (2005), pages 5–20 (cited on pages 24, 25).
- [29] Gavin Brown et al. “Managing diversity in regression ensembles.” In: *Journal of machine learning research* 6.9 (2005) (cited on pages 26, 31).
- [30] Robert G Brown. “Exponential smoothing for predicting demand”. In: *Operations Research*. Volume 5. 1. INST OPERATIONS RESEARCH MANAGEMENT SCIENCES 901 ELKRIDGE LANDING RD, STE ... 1957, pages 145–145 (cited on page 17).

- [31] Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. “Model compression”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pages 535–541 (cited on pages 4, 94, 95, 128).
- [32] Luis M Candanedo, Véronique Feldheim, and Dominique Deramaix. “Data driven prediction models of energy use of appliances in a low-energy house”. In: *Energy and buildings* 140 (2017), pages 81–97 (cited on page 132).
- [33] Rosa Candela et al. “Model Monitoring and Dynamic Model Selection in Travel Time-series Forecasting”. In: *arXiv preprint arXiv:2003.07268* (2020) (cited on page 36).
- [34] Jaime Carbonell and Jade Goldstein. “The use of MMR, diversity-based reranking for reordering documents and producing summaries”. In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 1998, pages 335–336 (cited on page 54).
- [35] Gail A Carpenter and Stephen Grossberg. *Pattern recognition by self-organizing neural networks*. MIT Press, 1991 (cited on page 41).
- [36] Rich Caruana et al. “Ensemble selection from libraries of models”. In: *Proceedings of the twenty-first international conference on Machine learning*. 2004, page 18 (cited on page 30).
- [37] Vitor Cerqueira, Heitor Murilo Gomes, and Albert Bifet. “Unsupervised Concept Drift Detection Using a Student–Teacher Approach”. In: *International Conference on Discovery Science*. Springer. 2020, pages 190–204 (cited on page 99).
- [38] Vítor Cerqueira et al. “Arbitrated ensemble for time series forecasting”. In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer. 2017, pages 478–494 (cited on pages 30, 32, 36, 46, 47, 77, 82, 134, 135).
- [39] Vitor Cerqueira et al. “Model Compression for Dynamic Forecast Combination”. In: *arXiv preprint arXiv:2104.01830* (2021) (cited on pages 4, 95, 110).
- [40] Vitor Cerqueira et al. “STUDD: A Student-Teacher Method for Unsupervised Concept Drift Detection”. In: *arXiv preprint arXiv:2103.00903* (2021) (cited on page 95).
- [41] Vítor Manuel Araújo Cerqueira. “Ensembles for Time Series Forecasting”. In: (2019) (cited on pages 13, 44, 62, 80, 81).
- [42] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006 (cited on page 34).
- [43] Varun Chandola, Arindam Banerjee, and Vipin Kumar. “Anomaly detection: A survey”. In: *ACM computing surveys (CSUR)* 41.3 (2009), pages 1–58 (cited on page 116).
- [44] Chris Chatfield. *Time-series forecasting*. CRC press, 2000 (cited on pages 2, 11, 12, 16, 18).
- [45] Guobin Chen et al. “Learning efficient object detection models with knowledge distillation”. In: *Advances in Neural Information Processing Systems*. 2017, pages 742–751 (cited on page 95).

- [46] Robert T Clemen. “Combining forecasts: A review and annotated bibliography”. In: *International journal of forecasting* 5.4 (1989), pages 559–583 (cited on page 26).
- [47] Robert T Clemen and Robert L Winkler. “Combining economic forecasts”. In: *Journal of Business & Economic Statistics* 4.1 (1986), pages 39–46 (cited on pages 33, 34).
- [48] Gianpaolo Cugola and Alessandro Margara. “Processing flows of information: From data stream to complex event processing”. In: *ACM Computing Surveys (CSUR)* 44.3 (2012), pages 1–62 (cited on page 120).
- [49] A Philip Dawid. “Present position and potential developments: Some personal views statistical theory the prequential approach”. In: *Journal of the Royal Statistical Society: Series A (General)* 147.2 (1984), pages 278–290 (cited on pages 20, 134).
- [50] Francis X Diebold and Jose A Lopez. “8 Forecast evaluation and combination”. In: *Handbook of statistics* 14 (1996), pages 241–268 (cited on page 26).
- [51] Thomas G Dietterich. “Ensemble methods in machine learning”. In: *International workshop on multiple classifier systems*. Springer. 2000, pages 1–15 (cited on pages 22, 24).
- [52] Gregory Ditzler et al. “Learning in nonstationary environments: A survey”. In: *IEEE Computational Intelligence Magazine* 10.4 (2015), pages 12–25 (cited on page 14).
- [53] Joao Duarte and Joao Gama. “Ensembles of adaptive model rules from high-speed data streams”. In: *Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*. PMLR. 2014, pages 198–213 (cited on page 30).
- [54] Hadi Fanaee-T and Joao Gama. “Event labeling combining ensemble detectors and background knowledge”. In: *Progress in Artificial Intelligence* 2.2 (2014), pages 113–127 (cited on page 132).
- [55] Julian Faraway and Chris Chatfield. “Time series forecasting with neural networks: a comparative study using the air line data”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 47.2 (1998), pages 231–250 (cited on page 19).
- [56] Yoav Freund and Robert E Schapire. “A decision-theoretic generalization of on-line learning and an application to boosting”. In: *Journal of computer and system sciences* 55.1 (1997), pages 119–139 (cited on page 23).
- [57] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010 (cited on page 10).
- [58] João Gama and Petr Kosina. “Tracking recurring concepts with meta-learners”. In: *Portuguese Conference on Artificial Intelligence*. Springer. 2009, pages 423–434 (cited on page 46).
- [59] Joao Gama and Petr Kosina. “Recurrent concepts in data streams classification”. In: *Knowledge and Information Systems* 40.3 (2014), pages 489–507 (cited on page 36).
- [60] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. “On evaluating stream learning algorithms”. In: *Machine learning* 90.3 (2013), pages 317–346 (cited on page 41).

- [61] Joao Gama et al. “Learning with drift detection”. In: *Brazilian symposium on artificial intelligence*. Springer. 2004, pages 286–295 (cited on page 45).
- [62] João Gama et al. “A survey on concept drift adaptation”. In: *ACM computing surveys (CSUR)* 46.4 (2014), pages 1–37 (cited on pages 14, 15).
- [63] Stuart Geman, Elie Bienenstock, and René Doursat. “Neural networks and the bias/variance dilemma”. In: *Neural computation* 4.1 (1992), pages 1–58 (cited on page 25).
- [64] Véronique Genre et al. “Combining expert forecasts: Can anything beat the simple average?”. In: *International Journal of Forecasting* 29.1 (2013), pages 108–121 (cited on page 33).
- [65] Giorgio Giacinto and Fabio Roli. “Dynamic classifier selection”. In: *International Workshop on Multiple Classifier Systems*. Springer. 2000, pages 177–189 (cited on page 32).
- [66] Heitor M Gomes et al. “Adaptive random forests for evolving data stream classification”. In: *Machine Learning* 106.9 (2017), pages 1469–1495 (cited on pages 29, 100, 130).
- [67] 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)*. IEEE. 2019, pages 240–249 (cited on pages 29, 31, 135).
- [68] Heitor Murilo Gomes et al. “A survey on ensemble learning for data stream classification”. In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pages 1–36 (cited on pages 28, 30).
- [69] Heitor Murilo Gomes et al. “Adaptive random forests for data stream regression.” In: *ESANN*. 2018 (cited on pages 28, 31, 135).
- [70] Heitor Murilo Gomes et al. “Machine learning for streaming data: state of the art, challenges, and opportunities”. In: *ACM SIGKDD Explorations Newsletter* 21.2 (2019), pages 6–22 (cited on pages 10, 16).
- [71] Heitor Murilo Gomes et al. “On Ensemble Techniques for Data Stream Regression”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pages 1–8 (cited on pages 28, 31).
- [72] Clive William John Granger and Paul Newbold. *Forecasting economic time series*. Academic Press, 2014 (cited on pages 2, 11, 16).
- [73] CWJ Granger. “Combining Forecasts—Twenty Years Later”. In: *Essays in Econometrics: Collected Papers of Clive WJ Granger* 32 (2001), page 411 (cited on page 26).
- [74] Silvio B Guerra, Ricardo BC Prudêncio, and Teresa B Ludermir. “Predicting the performance of learning algorithms using support vector machines as meta-regressors”. In: *International Conference on Artificial Neural Networks*. Springer. 2008, pages 523–532 (cited on page 46).

- [75] Sudipto Guha et al. “Robust random cut forest based anomaly detection on streams”. In: *International conference on machine learning*. 2016, pages 2712–2721 (cited on page 118).
- [76] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015) (cited on pages 4, 94, 95, 130).
- [77] Tin Kam Ho. “The random subspace method for constructing decision forests”. In: *IEEE transactions on pattern analysis and machine intelligence* 20.8 (1998), pages 832–844 (cited on page 29).
- [78] Tin Kam Ho. “Multiple classifier combination: Lessons and next steps”. In: *Hybrid methods in pattern recognition*. World Scientific, 2002, pages 171–198 (cited on page 23).
- [79] Wassily Hoeffding. “Probability inequalities for sums of bounded random variables”. In: *The Collected Works of Wassily Hoeffding*. Springer, 1994, pages 409–426 (cited on page 84).
- [80] Dezhi Hong, Quanquan Gu, and Kamin Whitehouse. “High-dimensional time series clustering via cross-predictability”. In: *Artificial Intelligence and Statistics*. PMLR. 2017, pages 642–651 (cited on pages 131, 132).
- [81] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018 (cited on page 17).
- [82] Elena Ikonomovska and Joao Gama. “Learning model trees from data streams”. In: *International Conference on Discovery Science*. Springer. 2008, pages 52–63 (cited on page 83).
- [83] Elena Ikonomovska, João Gama, and Sašo Džeroski. “Incremental multi-target model trees for data streams”. In: *Proceedings of the 2011 ACM symposium on applied computing*. 2011, pages 988–993 (cited on page 83).
- [84] Elena Ikonomovska, Joao Gama, and Sašo Džeroski. “Learning model trees from evolving data streams”. In: *Data mining and knowledge discovery* 23.1 (2011), pages 128–168 (cited on pages 20, 31, 64, 83).
- [85] Elena Ikonomovska, João Gama, and Sašo Džeroski. “Online tree-based ensembles and option trees for regression on evolving data streams”. In: *Neurocomputing* 150 (2015), pages 458–470 (cited on page 31).
- [86] Victor Richmond R Jose and Robert L Winkler. “Simple robust averages of forecasts: Some empirical results”. In: *International journal of forecasting* 24.1 (2008), pages 163–169 (cited on pages 34, 47).
- [87] Jee-Weon Jung et al. “Knowledge Distillation in Acoustic Scene Classification”. In: *IEEE Access* 8 (2020), pages 166870–166879 (cited on page 95).
- [88] Holger Kantz and Thomas Schreiber. *Nonlinear time series analysis*. Volume 7. Cambridge university press, 2004 (cited on page 13).

- [89] Matthew B Kennel, Reggie Brown, and Henry DI Abarbanel. “Determining embedding dimension for phase-space reconstruction using a geometrical construction”. In: *Physical review A* 45.6 (1992), page 3403 (cited on page 57).
- [90] Albert HR Ko, Robert Sabourin, and Alceu Souza Britto Jr. “From dynamic classifier selection to dynamic ensemble selection”. In: *Pattern recognition* 41.5 (2008), pages 1718–1731 (cited on pages 31, 42).
- [91] Dragi Kocev et al. “Using single-and multi-target regression trees and ensembles to model a compound index of vegetation condition”. In: *Ecological Modelling* 220.8 (2009), pages 1159–1168 (cited on page 83).
- [92] Jeremy Z Kolter and Marcus A Maloof. “Using additive expert ensembles to cope with concept drift”. In: *Proceedings of the 22nd international conference on Machine learning*. 2005, pages 449–456 (cited on pages 30, 135).
- [93] Bartosz Krawczyk and Alberto Cano. “Online ensemble learning with abstaining classifiers for drifting and noisy data streams”. In: *Applied Soft Computing* 68 (2018), pages 677–692 (cited on pages 50, 51).
- [94] Bartosz Krawczyk et al. “Ensemble learning for data stream analysis: A survey”. In: *Information Fusion* 37 (2017), pages 132–156 (cited on pages 4, 22, 30).
- [95] Georg Kreml et al. “Open challenges for data stream mining research”. In: *ACM SIGKDD explorations newsletter* 16.1 (2014), pages 1–10 (cited on page 10).
- [96] Ludmila I Kuncheva. “Classifier ensembles for changing environments”. In: *International Workshop on Multiple Classifier Systems*. Springer. 2004, pages 1–15 (cited on page 28).
- [97] Ludmila I Kuncheva. “Change detection in streaming multivariate data using likelihood detectors”. In: *IEEE transactions on knowledge and data engineering* 25.5 (2011), pages 1175–1180 (cited on pages 91, 130).
- [98] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2014 (cited on pages 4, 24, 25, 27, 32).
- [99] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. “Interpretable decision sets: A joint framework for description and prediction”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pages 1675–1684 (cited on pages 4, 94).
- [100] Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. “Mondrian forests: Efficient online random forests”. In: *Advances in neural information processing systems*. 2014, pages 3140–3148 (cited on page 118).
- [101] William B Langdon, SJ Barrett, and Bernard F Buxton. “Combining decision trees and neural networks for drug discovery”. In: *European Conference on Genetic Programming*. Springer. 2002, pages 60–70 (cited on page 30).
- [102] Pierre-Xavier Loeffel et al. “Improving the Prediction Cost of Drift Handling Algorithms by Abstaining”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2016, pages 1213–1222 (cited on page 48).

- [103] David Lopez-Paz et al. “Unifying distillation and privileged information”. In: *arXiv preprint arXiv:1511.03643* (2015) (cited on page 95).
- [104] Gilles Louppe and Pierre Geurts. “Ensembles on random patches”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2012, pages 346–361 (cited on pages 29, 31).
- [105] Spyros Makridakis and Robert L Winkler. “Averages of forecasts: Some empirical results”. In: *Management science* 29.9 (1983), pages 987–996 (cited on page 26).
- [106] Spyros Makridakis et al. “The accuracy of extrapolation (time series) methods: Results of a forecasting competition”. In: *Journal of forecasting* 1.2 (1982), pages 111–153 (cited on page 26).
- [107] Massimiliano Marcellino. “Forecast pooling for European macroeconomic variables”. In: *Oxford Bulletin of Economics and Statistics* 66.1 (2004), pages 91–112 (cited on page 33).
- [108] Fotini Markatopoulou, Grigorios Tsoumakias, and Ioannis Vlahavas. “Instance-based ensemble pruning via multi-label classification”. In: *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*. Volume 1. IEEE. 2010, pages 401–408 (cited on page 80).
- [109] Saulo Martiello Mastelini et al. “Online Multi-target regression trees with stacked leaf models”. In: *arXiv preprint arXiv:1903.12483* (2019) (cited on page 84).
- [110] William McGill. “Multivariate information transmission”. In: *Transactions of the IRE Professional Group on Information Theory* 4.4 (1954), pages 93–111 (cited on page 77).
- [111] Joao Mendes-Moreira et al. “Ensemble approaches for regression: A survey”. In: *Acm computing surveys (csur)* 45.1 (2012), pages 1–40 (cited on page 27).
- [112] Pablo Montero-Manso et al. “FFORMA: Feature-based forecast model averaging”. In: *International Journal of Forecasting* 36.1 (2020), pages 86–92 (cited on page 36).
- [113] Luis Moreira-Matias et al. “Predicting taxi-passenger demand using streaming data”. In: *IEEE Transactions on Intelligent Transportation Systems* 14.3 (2013), pages 1393–1402 (cited on page 34).
- [114] Anil Narassiguin, Haytham Elghazel, and Alex Aussem. “Dynamic ensemble selection with probabilistic classifier chains”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2017, pages 169–186 (cited on page 80).
- [115] Paul Newbold and David I Harvey. “Forecast combination and encompassing”. In: *A companion to economic forecasting* 1 (2002), page 620 (cited on pages 4, 26).
- [116] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, and Ria Mae Borromeo. “User group analytics: hypothesis generation and exploratory analysis of user data”. In: *The VLDB Journal* 28.2 (2019), pages 243–266 (cited on page 129).
- [117] Julio Ortega, Moshe Koppel, and Shlomo Argamon. “Arbitrating among competing classifiers using learned referees”. In: *Knowledge and Information Systems* 3.4 (2001), pages 470–490 (cited on pages 36, 46, 134).

- [118] Aljaž Osojnik, Panče Panov, and Sašo Džeroski. “Tree-based methods for online multi-target regression”. In: *Journal of Intelligent Information Systems* 50.2 (2018), pages 315–339 (cited on page 84).
- [119] Nikunj C Oza. “Online bagging and boosting”. In: *2005 IEEE international conference on systems, man and cybernetics*. Volume 3. Ieee. 2005, pages 2340–2345 (cited on page 29).
- [120] Nikunj C Oza and Stuart Russell. “Experimental comparisons of online and batch versions of bagging and boosting”. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. 2001, pages 359–364 (cited on page 29).
- [121] Ewan S Page. “Continuous inspection schemes”. In: *Biometrika* 41.1/2 (1954), pages 100–115 (cited on pages 20, 30).
- [122] Yonghong Peng<sup>1</sup> Peter A Flach Pavel and Brazdil<sup>2</sup> Carlos Soares. “Decision tree-based data characterization for meta-learning”. In: *IDDM-2002* (2002), page 111 (cited on page 44).
- [123] Bernhard Pfahringer, Hilan Bensusan, and Christophe G Giraud-Carrier. “Meta-Learning by Landmarking Various Learning Algorithms.” In: *ICML*. 2000, pages 743–750 (cited on page 44).
- [124] Robi Polikar. “Ensemble based systems in decision making”. In: *IEEE Circuits and systems magazine* 6.3 (2006), pages 21–45 (cited on page 25).
- [125] Robi Polikar. “Ensemble learning”. In: *Ensemble machine learning*. Springer, 2012, pages 1–34 (cited on page 22).
- [126] Nedeljko Radulovic, Dihia Boulegane, and Albert Bifet. “SCALAR-A Platform for Real-time Machine Learning Competitions on Data Streams”. In: *Journal of Open Source Software* 5.56 (2020), page 2676 (cited on page 120).
- [127] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Machine learning* 85.3 (2011), page 333 (cited on pages 82, 84).
- [128] Jesse Read et al. “Data Streams Are Time Series: Challenging Assumptions”. In: *Brazilian Conference on Intelligent Systems*. Springer. 2020, pages 529–543 (cited on pages 3, 10, 11, 13, 16, 19).
- [129] David J Reid. “Combining three estimates of gross domestic product”. In: *Economica* 35.140 (1968), pages 431–444 (cited on page 26).
- [130] David J Reid. *A comparative study of time series prediction techniques on economic data*. University of Nottingham, Library Photographic Unit, 1969 (cited on page 26).
- [131] Gregory C Reinsel. *Elements of multivariate time series analysis*. Springer Science & Business Media, 2003 (cited on page 12).
- [132] Ye Ren, Le Zhang, and Ponnuthurai N Suganthan. “Ensemble classification and regression-recent developments, applications and future directions”. In: *IEEE Computational intelligence magazine* 11.1 (2016), pages 41–53 (cited on page 27).

- [133] John R Rice. “The algorithm selection problem”. In: *Advances in computers*. Volume 15. Elsevier, 1976, pages 65–118 (cited on page 35).
- [134] Jan N van Rijn et al. “Algorithm selection on data streams”. In: *International Conference on Discovery Science*. Springer. 2014, pages 325–336 (cited on pages 36, 45).
- [135] Jan N van Rijn et al. “Having a blast: Meta-learning and heterogeneous ensembles for data streams”. In: *2015 IEEE International Conference on Data Mining*. IEEE. 2015, pages 1003–1008 (cited on pages 30, 34, 47, 135).
- [136] Jan N van Rijn et al. “The online performance estimation framework: heterogeneous ensemble learning for data streams”. In: *Machine Learning* 107.1 (2018), pages 149–176 (cited on page 30).
- [137] André Luis Debiasio Rossi et al. “MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data”. In: *Neurocomputing* 127 (2014), pages 52–64 (cited on pages 36, 44, 47, 135).
- [138] Daniel M Roy, Yee Whye Teh, et al. “The Mondrian Process.” In: *NIPS*. 2008, pages 1377–1384 (cited on page 118).
- [139] Daniel Russo et al. “A tutorial on thompson sampling”. In: *arXiv preprint arXiv:1707.02038* (2017) (cited on pages 53, 73).
- [140] Amal Saadallah, Florian Priebe, and Katharina Morik. “A drift-based dynamic ensemble members selection using clustering for time series forecasting”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pages 678–694 (cited on pages 34, 81).
- [141] Ismael Sánchez. “Adaptive combination of forecasts with application to wind energy”. In: *International Journal of Forecasting* 24.4 (2008), pages 679–693 (cited on pages 34, 42, 135).
- [142] Muhamad Risqi U Saputra et al. “Distilling knowledge from a deep pose regressor network”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pages 263–272 (cited on page 95).
- [143] Jeffrey C Schlimmer and Richard H Granger. “Incremental learning from noisy data”. In: *Machine learning* 1.3 (1986), pages 317–354 (cited on page 13).
- [144] Timo Similä and Jarkko Tikka. “Input selection and shrinkage in multiresponse linear regression”. In: *Computational Statistics & Data Analysis* 52.1 (2007), pages 406–422 (cited on page 83).
- [145] Paul C Smits. “Multiple classifier systems for supervised remote sensing image classification based on dynamic classifier selection”. In: *IEEE Transactions on Geoscience and Remote Sensing* 40.4 (2002), pages 801–813 (cited on page 42).
- [146] James Surowiecki. *The wisdom of crowds*. Anchor, 2005 (cited on pages 3, 21).
- [147] Souhaib Ben Taieb et al. “A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition”. In: *Expert systems with applications* 39.8 (2012), pages 7067–7083 (cited on page 12).

- [148] Makoto Takamoto, Yusuke Morishita, and Hitoshi Imaoka. “An Efficient Method of Training Small Models for Regression Problems with Knowledge Distillation”. In: *arXiv preprint arXiv:2002.12597* (2020) (cited on page 95).
- [149] Floris Takens. “Detecting strange attractors in turbulence”. In: *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pages 366–381 (cited on page 19).
- [150] Thiyanga S Talagala, Rob J Hyndman, George Athanasopoulos, et al. “Meta-learning how to forecast time series”. In: *Monash Econometrics and Business Statistics Working Papers* 6 (2018), page 18 (cited on pages 35, 36).
- [151] George C Tiao and Ruey S Tsay. “Some advances in non-linear and adaptive modelling in time-series”. In: *Journal of forecasting* 13.2 (1994), pages 109–131 (cited on page 12).
- [152] Allan Timmermann. “Forecast combinations”. In: *Handbook of economic forecasting* 1 (2006), pages 135–196 (cited on page 27).
- [153] Marko Toplak et al. “Assessment of machine learning reliability methods for quantifying the applicability domain of QSAR regression models”. In: *Journal of chemical information and modeling* 54.2 (2014), pages 431–441 (cited on page 49).
- [154] Grigorios Tsoumakas, Lefteris Angelis, and Ioannis Vlahavas. “Selective fusion of heterogeneous classifiers”. In: *Intelligent Data Analysis* 9.6 (2005), pages 511–525 (cited on page 30).
- [155] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. “Mining multi-label data”. In: *Data mining and knowledge discovery handbook*. Springer, 2009, pages 667–685 (cited on page 80).
- [156] Alexey Tsymbal. “The problem of concept drift: definitions and related work”. In: *Computer Science Department, Trinity College Dublin* 106.2 (2004), page 58 (cited on page 98).
- [157] Naonori Ueda and Ryohei Nakano. “Generalization error of ensemble estimators”. In: *Proceedings of International Conference on Neural Networks (ICNN’96)*. Volume 1. IEEE. 1996, pages 90–95 (cited on page 25).
- [158] Joaquin Vanschoren. “Meta-learning: A survey”. In: *arXiv preprint arXiv:1810.03548* (2018) (cited on page 35).
- [159] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013 (cited on page 18).
- [160] Kenneth F Wallis. “Revisiting Francis Galton’s forecasting competition”. In: *Statistical Science* (2014), pages 420–424 (cited on page 22).
- [161] Hui Wang et al. “Progressive Blockwise Knowledge Distillation for Neural Network Acceleration.” In: *IJCAI*. 2018, pages 2769–2775 (cited on page 95).
- [162] Andreas S Weigend. *Time series prediction: forecasting the future and understanding the past*. Routledge, 2018 (cited on page 12).

- [163] Gerhard Widmer and Miroslav Kubat. “Learning in the presence of concept drift and hidden contexts”. In: *Machine learning* 23.1 (1996), pages 69–101 (cited on pages 13, 36).
- [164] David H Wolpert. “Stacked generalization”. In: *Neural networks* 5.2 (1992), pages 241–259 (cited on pages 30, 47, 82, 135).
- [165] David H Wolpert. “The supervised learning no-free-lunch theorems”. In: *Soft computing and industry* (2002), pages 25–42 (cited on page 23).
- [166] Kevin Woods, W. Philip Kegelmeyer, and Kevin Bowyer. “Combination of multiple classifiers using local accuracy estimates”. In: *IEEE transactions on pattern analysis and machine intelligence* 19.4 (1997), pages 405–410 (cited on page 42).
- [167] Michał Woźniak, Manuel Grana, and Emilio Corchado. “A survey of multiple classifier systems as hybrid systems”. In: *Information Fusion* 16 (2014), pages 3–17 (cited on page 27).
- [168] Yuhong Yang. “Combining forecasting procedures: some theoretical results”. In: *Econometric Theory* (2004), pages 176–222 (cited on page 26).
- [169] Junho Yim et al. “A gift from knowledge distillation: Fast optimization, network minimization and transfer learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pages 4133–4141 (cited on page 95).
- [170] Min-Ling Zhang and Zhi-Hua Zhou. “A review on multi-label learning algorithms”. In: *IEEE transactions on knowledge and data engineering* 26.8 (2013), pages 1819–1837 (cited on pages 82, 84).
- [171] Shuyi Zhang et al. “Cautionary tales on air-quality improvement in Beijing”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2205 (2017), page 20170457 (cited on pages 132, 133).
- [172] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012 (cited on pages 23–25, 27).
- [173] Indrė Žliobaitė et al. “Active learning with drifting streaming data”. In: *IEEE transactions on neural networks and learning systems* 25.1 (2014), pages 27–39 (cited on page 52).
- [174] Indrė Žliobaitė et al. “Evaluation methods and decision theory for classification of streaming data with temporal dependence”. In: *Machine Learning* 98.3 (2015), pages 455–482 (cited on pages 3, 10, 11).
- [175] Hui Zou and Yuhong Yang. “Combining time series models for forecasting”. In: *International journal of Forecasting* 20.1 (2004), pages 69–84 (cited on page 34).

**Titre :** Algorithmes d'apprentissage machine appliqués au contexte dynamique de l'internet des objets

**Mots clés :** apprentissage machine, flux de données, séries temporelles, prévision, selection dynamic d'ensemble, compression de modèle

**Résumé :**

La croissance rapide de l'Internet des Objets (IdO) ainsi que la prolifération des capteurs ont donné lieu à diverses sources de données qui génèrent continuellement de grandes quantités de données et à une grande vitesse sous la forme de flux. Ces flux sont essentiels dans le processus de prise de décision dans différents secteurs d'activité et ce grâce aux techniques d'intelligence artificielle et d'apprentissage automatique afin d'extraire des connaissances précieuses et les transformer en actions pertinentes. Par ailleurs, les données sont souvent associées à un indicateur temporel, appelé flux de données temporel qui est défini comme étant une séquence infinie d'observations capturées à intervalles réguliers, mais pas nécessairement.

La prévision est une tâche complexe dans le domaine de l'IA et vise à comprendre le processus générant les observations au fil du temps sur la base d'un historique de données afin de prédire le comportement futur. L'apprentissage incremental et adaptatif est le domaine de recherche émergeant dédié à l'analyse des flux de données.

La thèse se penche sur les méthodes d'ensemble qui fusionnent de manière dynamique plusieurs modèles prédictifs accomplissant ainsi des résultats compétitifs malgré leur coût élevé en termes de mémoire et de temps de calcul.

Nous étudions différentes approches pour estimer la performance de chaque modèle de prévision individuel compris dans l'ensemble en fonction des données en introduisant de nouvelles méthodes basées sur le fenêtrage et le méta-apprentissage. Nous proposons différentes méthodes de sélection qui visent à constituer un comité de modèles précis et divers. Les prédictions de ces modèles sont ensuite pondérées et agrégées. La deuxième partie de la thèse traite de la compression des méthodes d'ensemble qui vise à produire un modèle individuel afin d'imiter le comportement d'un ensemble complexe tout en réduisant son coût. Pour finir, nous présentons "Real-Time Machine Learning Compétition on Data Streams", dans le cadre de BigDataCup Challenge de la conférence IEEE Big Data 2019 ainsi que la plateforme dédiée SCALAR.

**Title :** Machine Learning Algorithms for dynamic Internet of Things

**Keywords :** machine learning, data streams, time series, forecasting, dynamic ensemble selection, model compression

**Abstract :** With the rapid growth of Internet-of-Things (IoT) devices and sensors, sources that are continuously releasing and curating vast amount of data at high pace in the form of stream. The ubiquitous data streams are essential for data driven decision-making in different business sectors using Artificial Intelligence (AI) and Machine Learning (ML) techniques in order to extract valuable knowledge and turn it to appropriate actions. Besides, the data being collected is often associated with a temporal indicator, referred to as temporal data stream that is a potentially infinite sequence of observations captured over time at regular intervals, but not necessarily.

Forecasting is a challenging tasks in the field of AI and aims at understanding the process generating the observations over time based on past data in order to accurately predict future behavior. Stream Learning is the emerging research field which focuses on learning from infinite and evolving data streams. The thesis ta-

ckles dynamic model combination that achieves competitive results despite their high computational costs in terms of memory and time.

We study several approaches to estimate the predictive performance of individual forecasting models according to the data and contribute by introducing novel windowing and meta-learning based methods to cope with evolving data streams. Subsequently, we propose different selection methods that aim at constituting a committee of accurate and diverse models. The predictions of these models are then weighted and aggregated. The second part addresses model compression that aims at building a single model to mimic the behavior of a highly performing and complex ensemble while reducing its complexity. Finally, we present the first streaming competition "Real-time Machine Learning Competition on Data Streams", at the IEEE Big Data 2019 conference, using the new SCALAR platform.