Universidade Estadual de Campinas
Faculdade de Tecnologia

Jorge Andrés Bueno Barajas

Dynamic Ensemble Mechanisms to improve Particulate Matter Forecasting

Mecanismos para Ensemble Dinâmicos aplicados para a Previsão de Material Particulado

Limeira
2018

Dynamic Ensemble Mechanisms to improve Particulate Matter Forecasting

Mecanismos para Ensemble Dinâmicos aplicados para a Previsão de Material Particulado

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Tecnologia na área de Sistemas de Informação e Comunicação.

Presented to the School of Technology (FT) of the University of Campinas (Unicamp), in partial fulfillment of the requirements for the Degree of Master of Technology.

Orientador: Prof. Dr. Guilherme Palermo Coelho

.

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO JORGE ANDRÉS BUENO BARAJAS, E ORIENTADA PELO PROF. DR GUILHERME PALERMO COELHO.

Limeira

2018

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

Bueno Barajas, Jorge Andrés, 1991-
B862d Dynamic ensemble mechanisms to improve particulate matter forecasting / Jorge Andrés Bueno Barajas. – Limeira, SP : [s.n.], 2018.

Orientador: Guilherme Palermo Coelho.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Mineração de dados (Computação). 2. Aprendizado de máquina. I. Coelho, Guilherme Palermo, 1980-. II. Universidade Estadual de Campinas. Faculdade de Tecnologia. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Mecanismos para Ensemble Dinâmicos aplicados para a Previsão de Material Particulado
Palavras-chave em inglês:
Data mining
Machine Learning
Área de concentração: Sistemas de Informação e Comunicação
Titulação: Mestre em Tecnologia
Banca examinadora:
Guilherme Palermo Coelho [Orientador]
Leandro Nunes de Castro Silva
Ana Estela Antunes da Silva
Data de defesa: 10-05-2018
Programa de Pós-Graduação: Tecnologia

FOLHA DE APROVAÇÃO

Abaixo se apresentam os membros da comissão julgadora da sessão pública de defesa de dissertação para o Título de Mestre em Tecnologia na área de concentração de Sistema de Informação e Comunicação, a que submeteu a (o) aluna (o) Jorge Andrés Bueno Barajas, em 10 de maio de 2018 na Faculdade de Tecnologia- FT/ UNICAMP, em Limeira/SP.

Prof. (a). Dr (a)

Guilherme Palermo Coelho

Prof. Dr.

Leandro Nunes de Castro Silva

Prof. Dr.

Ana Estela Antunes da Silva

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmica da aluna na Universidade.

Abstract

Microscopically small solid particles and liquid droplets suspended in the air, known as particulate matter (PM), may significantly affect not only human health, but also urban, natural and agricultural systems. Therefore, it is imperative to keep the concentration levels of these pollutants below harmful thresholds. To do so, forecasting mechanisms are particularly relevant, as they may help public offices and environmental agencies define strategies to control PM concentration in the atmosphere. Forecasting tools based on Machine Learning have been used to estimate the concentration of PM and other pollutants in the atmosphere, as they are capable of learning from examples and identifying hidden insights in the data without being explicitly programmed. Nevertheless, most of these techniques were developed to learn from data with stationary probability distributions and, considering that PM data are uninterruptedly collected, thus producing a stream of data whose distribution may evolve over time, which is known as *concept drift*, such traditional machine learning techniques may offer limited accuracy. The overall goal of this work is to evaluate whether online sequential learning, combined with mechanisms and techniques to handle concept drift such as ensemble learning and sliding windows, can improve the estimation accuracy of PM forecasting. Online and offline algorithms based on Extreme Learning Machines (ELM) were compared, in order to evaluate their performance when applied to forecast daily concentrations of PM, specifically particles with aerodynamic diameter smaller than 10 μm (known as PM_{10}). The experiments were performed using real world datasets of PM_{10} concentration from different cities of the State of São Paulo, Brazil. The obtained results indicate that PM data distributions slowly evolve over time, so new mechanisms were proposed to keep information of past concepts into ensembles, so they can adapt to new concepts. These new mechanisms have shown good performance in dynamic ensembles.

Keywords: Particulate Matter, Machine Learning, Online Learning, Extreme Learning Machines, Ensembles.

Resumo

Partículas sólidas e gotículas microscópicas suspensas no ar, conhecidas como material particulado (MP), podem afetar significativamente não só a saúde humana, mas também os sistemas urbanos, naturais e agrícolas. Portanto, é imperativo manter os níveis de concentração destes poluentes abaixo de limiares nocivos. Para isso, os mecanismos de previsão são particularmente relevantes, pois podem ajudar os órgãos públicos e agências ambientais a definir estratégias para controlar a concentração de MP na atmosfera. As ferramentas de previsão baseadas em Aprendizagem de Máquinas têm sido usadas para estimar a concentração de MP e outros poluentes na atmosfera, devido à sua capacidade de aprender com exemplos e identificar relações nos dados, sem serem explicitamente programadas para isto. No entanto, a maioria destas técnicas foi desenvolvida para aprender à partir de dados com distribuições de probabilidade estacionárias e, como é provável que as distribuições dos dados de concentração de MP mudem ao longo do tempo, o que é conhecido como *concept drift*, tais técnicas podem oferecer acurácia limitada. O objetivo geral deste trabalho é avaliar se algoritmos *online* de aprendizado de máquina, combinados a técnicas de detecção de *concept drift* tais como ensembles e janelas deslizantes, podem melhorar a acurácia da estimativa de valores futuros de MP. Neste trabalho, foram comparados algoritmos *online* e *offline* baseados em *Extreme Learning Machines* (ELM), a fim de avaliar seu desempenho quando são aplicados para prever as concentrações diárias de MP, especificamente partículas com diâmetro aerodinâmico inferior a 10 μm (conhecidas como MP_{10}). Experimentos foram realizados utilizando conjuntos de dados reais de concentração de MP_{10} de diferentes cidades do Estado de São Paulo, Brasil. Os resultados obtidos indicaram que os dados de concentração de MP evoluem lentamente com o passar do tempo, o que levou à proposição de novos mecanismos que permitem manter a informação de conceitos anteriores nos ensembles. Tais mecanismos têm mostrado bom desempenho em ensembles dinâmicos.

Palavras-chave: Material Particulado, Aprendizagem de Máquina, Concept Drift, Máquinas de Aprendizado Extremo, Ensembles.

List of figures

List of Tables

List of Symbols

PM: Particulate Matter

PM_{10}: PM with aerodynamic diameter ø ≤ 10 μm

$PM_{2.5}$: PM with aerodynamic diameter ø ≤ 2.5 μm

CO: Carbon Monoxide

CO_2: Carbon Dioxide

SW: Sliding Window

ELM: Extreme Learning Machine

OS-ELM: Online Sequential Extreme Learning Machine

OS-ELMsw: The Online Sequential Extreme Learning Machine with Sliding Window

N_1: Initial training data set

S: An idealized data stream

L: Number of hidden neurons

T: Number of samples of the data stream

ANN: Artificial Neural Network

SVM: Support Vector Machine

MLP: Multilayer Perceptron

MLR: Multiple Linear Regression Model

MLPNN: Multi-layer Perceptron Neural Networks

DOER: The Dynamic and Online Ensemble Regression

EOS: Ensemble of Online Learners with Substitution of Models

EOS-rank: The Ensemble of Online Learners with Substitution of Models with ranking of components

DOER-rank: The Dynamic and on-line ensemble regression with ranking of components

EOS-D: The Dynamic Ensemble of Online Learners with Substitution of Models using weighted average

Σ: Ensemble

D_t: Batch of samples

f: An online supervised learner

m: Window's size

R: maximum number of models in the ensemble

λ: inclusion/replacement frequency of EOS-base approaches

α: Factor of inclusion of new models of DOER-base approaches

MSE: Mean Squared Error

$life$: The age of a component of the ensemble

c: Subset of the most accurate components of the ensemble

l: The size of the subset of components

SRU1: Sulfur Recovery Unit Output 1

SRU2: Sulfur Recovery Unit Output 2

PM_{max}: Maximum value of particulate matter concentration reported by CETESB for each city

SGBP: Stochastic Gradient Descent Back-Propagation

RAN: Resource Allocation Network

RAEKF: Resource Allocation Network via Extended Kalman Filte

MRAM: Minimal Resource Allocation Network

GAP-RBF: Growing and Pruning RBF Network

Summary

1 Introduction

Advances in processing power, affordable data storage and the multiplication of data sensors have led to a significant growth in the overall volume of data produced. As a result, the demand for complex and automated data analysis tools, capable of dealing with bigger and more complex data and of delivering faster and more accurate results, have also increased. In order to satisfy this demand, machine learning and data mining techniques have evolved, becoming more reliable and accurate, allowing the creation of decision-making models in a wide variety of fields (Han, Kamber, & Pei, 2012).

In some application fields, data is often generated as a timely ordered sequence of numerical data points, so it can be seen as a time series. This allows the application of machine learning-based techniques to build forecasting models capable of automatically identifying important insights and of predicting future behavior of the time series. Online learning algorithms are suitable for this task as, in practice, data becomes available sequentially (as a *data stream*) and online algorithms allow the update of the forecasting model whenever new data becomes available. This approach offers good performance even when the underlying data distribution changes over time, where traditional batch-based models become less accurate (Cavalcante & Oliveira, 2015).

Given the features mentioned above, many applications of online learning algorithms have been reported in the literature. Such algorithms have been applied, for example, to environmental monitoring. In these scenarios, the forecasting task is difficult due to the uncertainties involved in the behavior of the natural phenomena. Hence, it is imperative to update the forecasting model with the information introduced by new incoming data, as the accuracy of the model is critical for planning and implementing counter-measures to protect human lives. The capability of fast updating the forecasting model (without a significant increase in the computational times) makes online learning suitable for short term forecasting, as required in such scenarios (Yaday et al., 2016).

Monitoring and forecasting systems have also been applied to predict the concentration of pollutants in the air. The aim of those systems is to support policies for the control of the concentration of various air pollutants that affect human health, such as ground-level

ozone (O_3), nitrogen dioxide (NO_2) and sulfur dioxide (SO_2). Results indicate the importance of considering data changes over time for real-time forecasting or air pollutants (Bashir et al., 2016), which can be achieved by online learning algorithms. In addition to the pollutants mentioned above, forecasting can be also applied to many other air pollutants, such as particulate matter (PM), which has drawn the attention of scientists and academics, given the high concentration levels of this pollutant in the air that have been observed in the last years.

Fast growing population in urban regions has increased human-related activities such as agriculture, industry and transportation. These activities may lead to the increase of the concentration of different pollutants in the air (Calderón-Garcidueñas et al., 2015; Pozza, 2009), including extremely small particles and liquid droplets known as particulate matter (PM).

Breathable fractions of PM with aerodynamic diameter ø ≤ 2.5 µm and ø ≤ 10 µm (known as $PM_{2.5}$ and PM_{10}, respectively) have a greater impact on human health (Oprea et al., 2015), as these particles easily enter through the airways up to the lungs, increasing the likelihood of respiratory diseases and even death (Souza et al., 2015). Besides affecting human health, these particles can cause environmental and crops damage. Therefore, real-time monitoring, forecasting and alert systems are helpful to environmental agencies and other authorities to manage air quality, in order to avoid that PM concentrations reach harmful levels.

Given the nature of PM concentration data, they can also be seen as time series, since they correspond to a sequence of measures collected over time (Bell, Samet, & Dominici, 2004). Therefore, it is possible to apply Machine Learning techniques to forecast PM concentration (Oprea et al., 2015; Raimondo et al., 2007; Souza et al., 2015). Nevertheless, most of these approaches assume that the underlying distribution of data, from which the model learns, do not change over time (stationary environments).

Considering that dynamic behavior is inherent to data streams, it is possible to say that the values to be predicted may depend on some hidden context that evolves over time (Gonçalves et al., 2014; Han et al., 2012). Therefore, it is likely that the data distribution

will change over time, thus compromising the accuracy of the forecasting system. This phenomenon is known as *concept drift*, and it implies an important challenge for online learning (Gonçalves et al., 2014).

In conclusion, given the PM effects over human health and urban, natural and agricultural systems, real-time monitoring, forecasting and alert systems are needed to support control strategies and policies to keep the concentration of PM below the harmful thresholds for humans and environmental systems. Short-term exposures to high concentrations of PM_{10} demands immediate mitigation exposure actions to protect especially children and elders exposed to these episodes. Air quality monitoring systems can support these strategies to predict and anticipate several air pollution episodes. Nevertheless, this kind of data (data streams) may present dynamic behaviors (concept drifts) that can make this task difficult, so it is important to consider this issue to properly mine PM data.

Online sequential learning capability of learning new data patterns over time enables models to handle concept drifts in an implicit way by updating the models with new arriving samples every time they are available. Additionally, Ensembles of Models has shown important results in dynamic scenarios, especially when the components of the ensemble are trained with different batches of data, thus, learning different concepts of the data stream. Such model's outputs may be combined according to their accuracy in recent past predictions, ensuring that the most accurate models contribute to the ensemble output. Those approaches can be combined with Sliding Windows, which selects the most recent samples in a predefined window to update or re-train the models. This window acts like a limited memory, which retain information of the current drift in the most recent samples and forget old information.

The aim of this work is to evaluate whether the use of techniques to handle concept drift together with online sequential learning forecasting models and online dynamic ensembles, improves the accuracy of estimations. Here, a first step to achieve this goal was made with a thorough analysis of a state-of-the-art online learner: Extreme Learning Machine (ELM)-based forecasting models trained to predict daily PM_{10} concentration were

evaluated and compared. Besides, such ELMs were also combined into ensembles, and two mechanisms to enhance the accuracy of dynamic ensembles were proposed: a ranking scheme, which was applied to the Ensemble of Online Sequential Learners (EOS) and to the Dynamic and Online Ensemble for Regression (DOER) algorithms (namely EOS-rank and DOER-rank, respectively), and a procedure of dynamic adaptation of models using weighted average, which was applied to EOS (named here EOS-D). Experiments were performed using real-world datasets of PM concentration from different cities of the State of São Paulo, Brazil.

This document is organized into 6 chapters. Chapter 2 provides a theoretical background of the main concepts related to concept drift detection techniques and online learning algorithms. Chapter 3 resumes the literature review of pollutant concentration forecasting and techniques to handle concept drift. Chapter 4 presents detailed information and description of the algorithms evaluated in this work. The configuration of the algorithms, the datasets used in the experiments and the methodology adopted here, together with the obtained results are discussed in Chapter 5. Finally, Chapter 6 presents the conclusions and future steps of this research.

2 Theoretical Background

The aim of this chapter is to address the main concepts related to this work and provide key definitions that will be required for the next chapters. The first section of the chapter introduces the particulate matter phenomena, explains the ways it can be characterized and its impact on human and environmental systems, which motivated this study. Particulate matter data can be seen as time series, a special kind of time series, known as data streams and characterized by high volumes of incoming data, is also briefly introduced (Han et al., 2012). Finally, the last section explores the inherently dynamic behavior of data streams, provides formal definitions of Concept Drift and presents approaches to handle this phenomenon.

2.1 Particulate Matter

Particulate Matter (PM) is the general term used to describe microscopic solid particles and liquid droplets found in the air. Some of these particles are so small they can only be detected using an electron microscope. The origins of PM can vary, as some particles may be emitted directly, as in industrial emissions, fires and construction sites, while others can be formed from photochemical reactions of chemical compounds, such as sulfur dioxide and nitrogen oxides, that occur in the atmosphere (Bell et al., 2004).

The source of these particles can be classified as follows: (*i*) *natural*, such as suspension of soil and saline particles from the ocean; and (*ii*) *anthropogenic*, originated from human activities, such as industries and farms (stationary sources), and fuel combustion (mobile self-propelled sources) (Pozza, 2009).

The particle size is frequently expressed in terms of its aerodynamic diameter, which characterizes an irregularly shaped particle in terms of the diameter of an idealized particle. Particles suspended in the atmosphere with an aerodynamic diameter smaller than 100 µm, also known as respirable or inhalable particles, are characterized by the fact that some fraction of them remains in the upper respiratory tract when inhaled. Regarding human health, PM with aerodynamic diameter ø ≤ 2.5 µm and ø ≤ 10 µm ($PM_{2.5}$ and PM_{10} respectively) are the most important particles regarding to their impact on human health (Taylor, 2008). The relative size of PM particles can be appreciated in Figure 1.

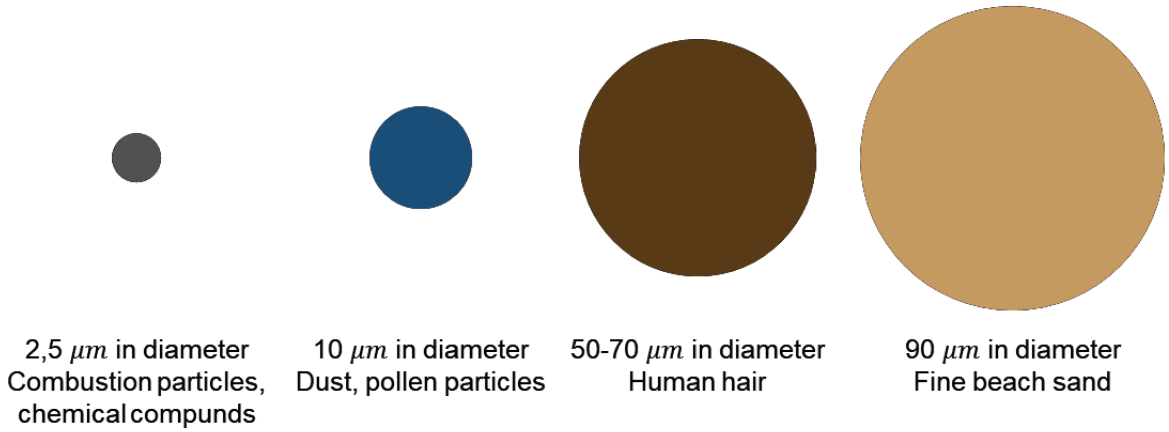

Figure 1. Relative Particle Sizes

In most urban environments, both PM_{10} and $PM_{2.5}$ are presented. However, the proportion of particles of the two sizes varies in cities around the world according to local geography, meteorology and specific PM sources. Usually, unfavorable meteorological conditions have great influence on high concentrations of PM_{10} episodes (Bianco et al., 2017; Mao, Shen, & Feng, 2017; Peng et al., 2017).

2.1.1 Particulate matter effects

Particulate matter composed of Sulfur Oxides (SO_2), Nitrogen Oxides (NO_x) and other volatile compounds can significantly affect urban, natural and agricultural systems. Acid rain and reduced visibility are some of the environmental effects of high concentrations of PM (Bhattacharjee et al., 1999). Furthermore, PM has a greater impact on human health. Studies in the past 30 years found a strong exposure-response relationship between elevated air pollution levels of PM and several health outcomes, including premature mortality (Taylor, 2008). Effects are stronger on children and elders, since these fine and ultrafine particles can easily enter through the airways deep up to the lungs, increasing the likelihood of causing respiratory and cardiovascular diseases, infections such as pneumonia, asthma, lung cancer and even death (Pozza, 2009; Taylor, 2008).

Correlations between several episodes of PM_{10} and the increase of hospital admissions for respiratory and heart diseases were also identified (Bianco et al., 2017). Calderón-Garcidueñas et al., (2015) warn that sustained exposures to high concentration of air pollutants have short and long-term neurodegenerative consequences. Although air

pollution effects on human health have been mainly linked to heart and respiratory diseases, and brain effects are not as broadly recognized in children, particulate matter pollution is a risk factor for the development of neuro-inflammation and neuro-degeneration. Results showed that inflammation of the upper and lower respiratory tracts produce a natural inflammatory response based on inflammatory mediators. These mediators can ultra-pass the placental barrier and reach the brain of the embryo and fetus, affecting its development (Calderón-Garcidueñas et al., 2015).

2.1.2 Air quality standards

Given the negative effects mentioned above and considering that the concentration levels of PM have increased in most cities worldwide, PM has become a big concern for public health authorities and environmental agencies. It is necessary to create control strategies for PM concentrations in urban areas and to strengthen policies to keep the concentration of these particles below harmful thresholds, so damages to human health, human welfare and the environment can be mitigated.

Usually, air quality standards are set by each country in order to protect its population from health diseases and are considered an important component of the national risk management and environmental policies. Those standards not only vary according to economic, political and social factors but also according to the level of development and air quality management capabilities.

The World Health Organization air quality guidelines (AQGs) is one of the most widely accepted standards and guidelines for air quality management. The AQGs are designed to offer guidance in order to reduce the health impacts of air pollution. These guidelines are continuously updated based on expert evaluation of scientific evidence, incorporating new studies of effects of air pollution that have been published in the literature. The objective of these guidelines is to inform policy-makers, support actions to achieve air quality that protects public health and define appropriate targets in order to create environmental policies for air quality management around the world (WHO, 2005).

It is important to highlight that there is not enough evidence to suggest a PM threshold below which no negative effects would be expected. Additionally, there are individual

factors that influence the response to a given exposure, making difficult to create a global guideline that leads to a complete protection for every individual against air pollution adverse health effects (WHO, 2005).

The WHO AQGs defines short-term (24 hour) and long-term (annual mean) indicators of PM pollution. Long-term and short-term guideline values for PM_{10} and $PM_{2.5}$ concentration are presented in Table 1.

Table 1. World Health Organization Guideline values for PM_{10} and $PM_{2.5}$

Particulate matter size	Term	Concentration
$PM_{2.5}$	Annual mean	10 $\mu g/m^3$
	Hour mean	25 $\mu g/m^3$
PM_{10}	Annual mean	20 $\mu g/m^3$
	Hour mean	50 $\mu g/m^3$

For $PM_{2.5}$ and PM_{10}, annual average concentrations of 10 $\mu g/m^3$ and 20 $\mu g/m^3$ were defined. Those values represent the lower limits over which significant effects on human health were observed in an American Cancer Society's study (Pope III et al., 2002). The WHO AQGs warns that adverse health effects cannot be entirely ruled out below the levels defined above. These levels represent values of PM concentration that not only have been shown to be achievable by different cities in developed countries around the world, but also that allows significant reduction of adverse effects in human health.

Besides the guideline values presented in Table 1, three interim targets (IT) are defined for $PM_{2.5}$ and PM_{10} (Table 2). The interim targets present values that have been shown achievable through different air quality management policies in order to reduce population exposure and can be helpful to support progress evaluation of current policies over time.

The guideline values of 24-hour mean, presented in Table 2, aim to protect populations against peaks of pollution that lead to a substantial increase in mortality. Published risk coefficients from multi-center studies and meta-analyses were considered to determine each interim target.

Table 2. WHO air quality guidelines and interim targets for particulate matter: 24-hour concentrations. Adapted from (WHO, 2005)

Interim target	PM_{10} $\mu g/m^3$	$PM_{2.5}$ $\mu g/m^3$	Basis for the selected level
Interim target -1 (IT-1)	150	75	About 5% increase of short-term mortality over the AQG value
Interim target -2 (IT-2)	100	50	About 2.5% increase of short-term mortality over the AQG value
Interim target -3 (IT-3)	75	37.5	About 1.2% increase of short-term mortality over the AQG value
Air quality guideline (AQG)	50	25	Based on relationship between 24-hour and annual PM levels

These studies suggest that risk for short-term exposure to PM_{10} are the same in developed and developing countries. It is important to consider that values presented in Table 2 tend to vary between countries due to the strong influence of specific characteristics of pollutant sources and their locations. Results also suggest that every 10 $\mu g/m^3$ in daily concentration produce an increase of 0.5% in mortality, this is to say, for a PM_{10} concentration of 150 $\mu g/m^3$, it is expected an increase of 5% in daily mortality. This is an important concern for public health and requires immediate mitigation actions. Therefore, real-time monitoring, forecasting and alert systems are helpful to environmental agencies and other authorities to manage air quality, in order to avoid that PM concentration reach harmful levels.

2.2 Concept Drift

Air quality monitoring stations support the air quality management in cities through the continuous measure of air pollutant concentrations. Some of these stations can process hourly average concentration levels of PM from samples collected within intervals of seconds. Thus, PM concentration data can be seen as an uninterrupted flow that is constantly sampled by air quality monitoring stations over time, which is also known as data streams.

A data stream is a flow of data that arrives in high volumes. This ordered sequence of samples can change dynamically, is possibly infinite and composed of multidimensional features. Sequences of numerical data points recorded sequentially over time are known

as time series and can be considered as an particular case of data stream (Cavalcante & Oliveira, 2015). Since in many applications the data streams cannot be stored, due to the high volume of data and the speed that the samples arrive, the effective mining of data streams is not a trivial task. Therefore, the development of efficient methods for mining data streams has grown in different areas of data mining, including classification, clustering and online detection of rare events in data streams (Han et al., 2012).

As the nature of data streams is dynamic, data patterns may evolve over time, which is a challenge to conventional batch learning algorithms. As the underlying data distribution may change over time, the accuracy of the forecasting models may also degrade. This problem is referred to as concept drift (Cavalcante & Oliveira, 2015).

Most of the work in the machine learning literature assumes that the underlying distributions of training samples are stationary (Gama et al., 2004). However, the probabilistic distribution of the data can change over time, so the model learned may become less accurate after a period of time. This problem is known as *concept drift*, and it implies a big challenge to conventional batch learning algorithms (Cavalcante & Oliveira, 2015). A concept drift is generally described as a modification in the relationship between input and output data over time (Elwell & Polikar, 2011; Gama et al., 2004).

In order to explain concept drift, this document considers the following definitions. Forecasting can be seen as the task of making long or short-term predictions of future values of a time series, based on a mathematical model adjusted to approximately represent the historical patterns of the series (Han & Kamber, 2006). Environments where the underlying data distributions change over time are known as *non-stationary* environments. The objective variable for classification and forecasting models are known as *classes* and *target values*, respectively.

According to the above terminology, concept drift can be formally described from the Bayesian posterior probability (Gama et al., 2014). According to Bayes' theorem,

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)} \qquad (1)$$

where a target variable $y \in \Re^1$ must be predicted according to a set of inputs $x \in \Re^P$, $P(x)$ corresponds to the feature-based probability of the data, $P(y)$ defines the objective variable prior probability and $P(x|y)$ describes the likelihood of x within a particular set of possible outcomes. In this context, a concept drift can be defined as any scenario where the posterior probability changes, i.e., $P_{t+1}(y|x) \neq P_t(y|x)$.

2.2.1 Types of drift

While a shift in $P(x)$ could indicate that the predictive decision can be shifting as well, the observation of a shift on $P(x)$ is not enough to indicate a concept drift, due to its independence from the objective variable. However, if the data distribution $P(y|x)$ changes, the decision boundary is affected. Changes that affect the decision boundary are a concern both from forecasting and classification perspectives.

In this context, it is possible to distinguish two types of drifts:

1. *Real concept drift*, which refers to changes in $P(y|x)$, thus representing a change in the decision boundary. Such change can occur either with or without a modification in the probabilities of the input data $P(x)$ (Elwell & Polikar, 2011).
2. *Virtual drift*, which corresponds to changes in the distribution of the input data $P(x)$ that do not affect $P(y|x)$ (Tysmbal, 2004; Gama, 2014).

Some authors have characterized concept drift differently, according to the way the concept drift occurs. Such classification is based on the drift's speed, randomness and cyclical nature. Drift speed is defined as the displacement rate between $P_t(y|x)$ and $P_{t+1}(y|x)$, from one time step to the next. Figure 2 shows the main patterns of concept drift, which can be:

- *Sudden drift,* also known as *abrupt drift,* corresponds to a larger displacement within a time step and may occur by switching from one concept to another. This usually results in high prediction error.
- *Incremental drift* presents smaller displacements, therefore results in lower prediction errors and is more difficult to detect.

- *Gradual drift*, which is associated with an even smaller displacement. The data switches between two concepts and finally stabilizes in the new concept.
- *Cyclic or recurrent drifts* can be described as the recurrence of concepts over time. They can be observed in many real-world applications such as climate or electricity demand. The recurrence of the drift could be periodic or random.

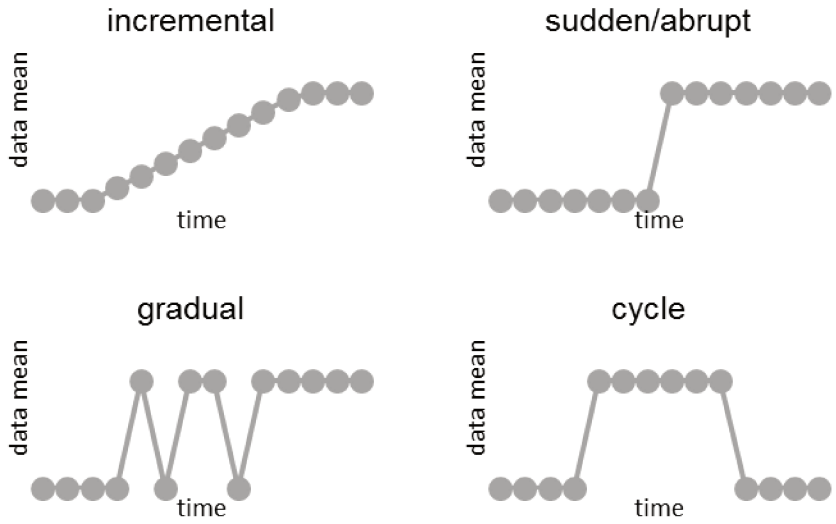

Figure 2. Main patterns of concept drift

2.2.2 Approaches to handle concept drift

Considering machine learning-based forecasting, the literature categorizes concept drift detection approaches according to Table 3. *Active* (or explicit) approaches employ mechanisms to detect the concept drift and, when it occurs, the model is trained using recent samples. Early Drift Detection Method (EDDM) is an example of an active approach (Baena-García et al., 2006).

Table 3. Main concept drift detection approaches based on machine learning

Active (explicit)	Passive (implicit)
• Mechanisms monitor the forecasting system to detect a concept drift. • The forecasting model is re-trained when a *concept drift* is identified.	• Assumes a possible ongoing concept drift and the model is continuously updated from new data. • Main techniques: Time window, Instance Weighting, Ensemble learning

Passive (or implicit) approaches, on the other hand, do not employ techniques to detect the beginning of a concept drift: a possible ongoing drift is constantly assumed, and the model is continuously updated with the most recent samples. Sample-based online learners are considered passive approaches, and these algorithms assure faster adaptation to changing environments, offering good performance in cases where the rate of incoming data is not very fast (Gomes Soares & Araújo, 2015).

Within the passive approach, three methods for concept drift handling are widely used: sliding windows or instance selection, instance weighting and ensemble learning (Cavalcante & Oliveira, 2015; Fdez-Riverola et al., 2007; Liao, Member, & Carin, 2009).

Sliding Window (or *instance selection*) approaches train or updated models from the most recent data in a predefined window, allowing the model to represent and predict the current concept. The window acts like a limited memory that forgets the older samples that are left out of the window. An important issue is to find the ideal window size, which should capture the rate of the concept drift**.** Small windows provide faster adaptation and large windows provide more stability, but also slower adaptation to drifts. The main disadvantage of this approach is the high computational cost of continuously training a new model whenever a new sample is available (Soares & Araújo, 2015). This approach is used to handle concept drifts since it allows models to represent and predict the current concept (Soares & Araújo, 2015).

Instance Weighting assigns weights to data or part of the data, according to its age or utility. These weights reflect the importance of such samples for the classification/forecasting task (Tsymbal, 2004). Weights are useful in concept drift scenarios when only the new samples represent the current concept. Thus, weights can be determined according to the age of each sample: one approach is the exponential decrease of each weight according to the age of the sample.

Finally, *ensemble learning* employs a set of models, usually trained from different sets of data, to forecast target variables. Predictions of each model could be combined using voting, weighted voting or selecting the most relevant model. Mechanisms to include and remove models in the ensemble are important factors to improve the prediction performance of the ensemble in concept drift scenarios. In this approach, new models

trained with the current concept data, can be added to the ensemble in order to adapt the ensemble to the current concept. On the other hand, the dynamic removal of inaccurate models, which are not able to predict the current concept, avoid degrading the ensemble's performance (Soares & Araújo, 2015).

2.2.3 Online Sequential Learning

In machine learning can be identified two main learning approaches: *offline learning* and *online learning*. In offline learning, the whole training data set must be available for the training phase. Only when the model is completely trained, it is available for predicting. In contrast, online learning process data sequentially. In this approach, a model is produced and put into operation without having the complete training data set at the beginning. This model is continuously updated with the new incoming samples. The online learning algorithms are suited for scenarios where it is impossible to have the whole training data set at the beginning of the operation (Gama et al., 2010).

While offline learning algorithms use past and new data in a complete retraining of the model, which can be computationally expensive, online learning only uses new data to update the model. In this context, linear regression models are more suitable for online learning, since they are generally easy to update even when updating with batches of data, linear models are not expensive computationally. This capability is not feasible for nonlinear methods, where frequent updating via batch/sample learning is too expensive to implement, as those models tend to have more parameters to train, making the process more slowly when compared with linear models (Peng et al., 2017).

Online learning can be formally defined as follows. A forecasting model for regression $F(x)$ that maps a set of inputs x into an output $y = F(x)$ da for classification and regression tasks. The online learning procedure is the following:

1. An unlabeled sample x_t is received by the algorithm
2. A prediction of $\hat{y}_t$ is made using x_t
3. Receive the true label y_t
4. Update the model using (x_t, y_t)

In step 2 can be added a loss estimation module $f(y_t, \hat{y}_t)$ that tracks the performance of the online learning algorithms and sends information about the changes in the environment. This is information is used to identify where the environment change and is necessary to train or retrain completely the model.

The ability of continuously learning with the incoming data make online learning able to handle concept drifts in an implicit way. Online learning offers the capability of learning new pattenrs on data that were not present in the training data set, this is an advantage in scenarios with concept drift, compare with offline learners, which can only predict based in the concepts presented in the training data set.

2.3 Studied Approaches

Since online sequential learning has shown good performance in dynamic scenarios (where underlying distributions evolve over time), given its capability of constantly learning from new arriving instances, thus enabling faster adaptations to changes (Cavalcante & Oliveira, 2015; Soares & Araújo, 2015, 2016), this work considered Extreme Learning Machines (ELM), specifically the online version of ELM, the Online Sequential Extreme Learning Machine (OS-ELM), as the base data stream forecaster.

2.3.1 Extreme Learning Machines

In this section, ELMs are introduced so that OS-ELM, the online version of the ELM, can be described. The discussion presented here considers ELMs and OS-ELMs for regression with a single output and assumes that, to train/update the forecasting models, the arriving samples of the data stream can be organized as $S = \{(x_t, y_t) | x_t \in \mathbb{R}^{r\mathbf{x1}}, y_t \in \mathbb{R}, t = 1, \dots, T\}$ where x_t is a vector of r input variables, and y_t is the target variable.

The ELM algorithm (Huang, Zhu, & Siew, 2006) is derived from single hidden layer feedforward neural networks (SLFNs). In contrast to SLFNs, ELMs randomly assign the input weights and bias without any iterative tuning. Figure 3 shows the general structure of an ELM.

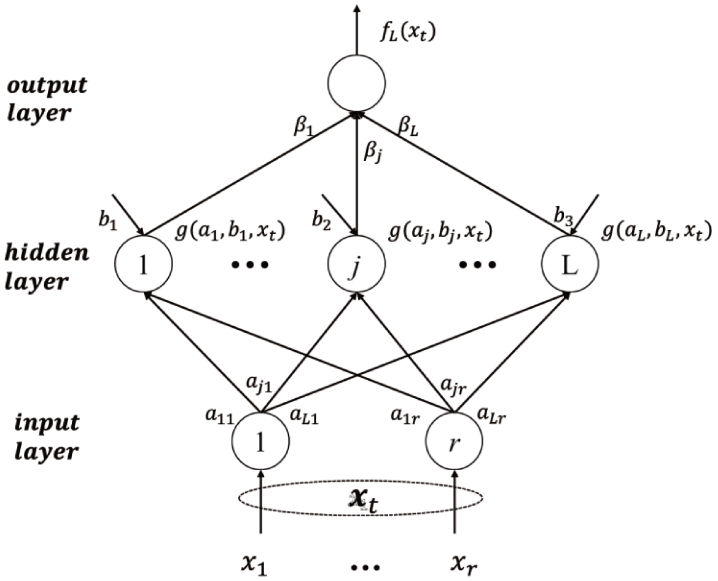

Figure 3. Structure of an ELM

Considering a dataset $D_1 = \{(x_t, y_t)|t = 1, \dots, N_1\} \subset S$, with N_1 distinct samples from the data stream S, the number of hidden nodes $L \leq N_1$ and $g(\cdot)$ an activation function, the output of an ELM is mathematically given in Eq. 2.

$$f_L(x_t) = \sum_{j=1}^{L} \beta_j g(a_j, b_j, x_t) = y_t \quad \text{for } t = 1, \dots., N_1, \tag{2}$$

where $a_j = [a_{j1}, a_{j2}, \dots a_{jr}]^{\mathrm{T}}$ is the input weight vector connecting the r input nodes and the j-th hidden node ($j = 1, \dots, L$); b_j is the bias of the j-th hidden node; β_j is the weight associated with the connection between the j-th hidden node and the output node; and y_t is the predicted output. Therefore, the ELM can be mathematically represented as:

$$\mathbf{H}\beta = \mathrm{y} \tag{3}$$

$$\mathbf{H} = \begin{bmatrix} g(a_1, b_1, x_1) & \cdots & g(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ g(a_1, b_1, x_T) & \cdots & g(a_L, b_L, x_T) \end{bmatrix} \tag{4}$$

$$\beta = [\beta_1, \dots, \beta_L]^{\mathrm{T}} \text{ and } y = [y_1, \dots, y_L]^{\mathrm{T}}, \tag{5}$$

where β is the output weight vector and y is the output vector. $\mathbf{H}$ is the hidden layer output matrix, where the j-th column of $\mathbf{H}$ represents the output vector of the j-th hidden node, with respect to all the inputs; and the j-th row of $\mathbf{H}$ is the output vector of the hidden layer with respect to x_t.

Since the weights and biases of the hidden layer are randomly assigned, the learning process in ELMs is based on finding the output weights β. This can be accomplished by:

$$\hat{\beta} = \mathbf{H}^{\dagger} y, \tag{6}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse (or pseudoinverse) of matrix $\mathbf{H}$ (Liang et al., 2006)**.** Which can be calculated as Eq. 7 if the inverse of $\mathbf{H}^T\mathbf{H}$ exists.

$$\mathbf{H}^{\dagger} = (\mathbf{H}^T\mathbf{H})^{-\mathbf{1}}\mathbf{H}^T. \tag{7}$$

Substituting Eq. 7 into Eq. 6, β becomes

$$\beta = (\mathbf{H}^T\mathbf{H})^{-\mathbf{1}}\mathbf{H}^T y \tag{8}$$

Therefore, the ELM algorithm can be summarized as in Algorithm 1.

Algorithm 1. Extreme Learning Machine (ELM) algorithm

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; the size of training data N_1; a number of hidden nodes L;

1. Assign input weights a_j and bias b_j randomly, $j = 1, \ldots, L$;
2. Calculate $\mathbf{H}$ with $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$; and Eq. (4);
3. Calculate the output weight β through Eq. (8);

end

In many industrial applications, it is impossible to have all the training data available before the learning process, as the observations arrive sequentially to the learning algorithm, i.e., they arrive one-by-one or chunk-by-chunk (in batches). In these cases, traditional ELMs are not suitable. Hence, the OS-ELM was proposed to deal with online learning (Liu et al., 2015).

2.3.2 Online Sequential Extreme Learning Machine

Two phases compose the learning process in OS-ELMs. The *initialization phase* and the *sequential learning phase.* In the initial phase, a training dataset of size $N_1 < T$ is used to build the initial ELM. In the sequential learning phase only the new one-by-one (or chunk-by-chunk) arriving samples are used to update the ELM. Once the step is completed, those samples are discarded. For the initialization and update phases, the $(k+1)$-th batch of new observations can be expressed as:

$$D_{k+1} = \{(x_t, y_t)\}_{t=\left(\sum_{l=0}^{k} N_l\right)+1}^{t=\sum_{l=0}^{k+1} N_l} \tag{9}$$

where $k \geq 0$, D_{k+1} represents the $(k+1)$-th batch of observations, N_{k+1} is the number of samples in the $(k+1)$-th batch and $N_0 = 0$.

In the initialization phase of the OS-ELM, a training dataset $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$ is used to build the initial ELM. Then, the output weights β_0 are determined as in Eq. 10.

$$\beta_0 = \left(\mathbf{H_0^T H_0}\right)^{-1} \mathbf{H_0^T} y_{0,} \tag{10}$$

where $y_0 = [y_1, \ldots, y_T]^{\mathrm{T}}$ is the output vector from D_1 and $\mathbf{H_0}$ is the initial hidden layer matrix obtained with D_1 (Eq. 11).

$$\mathbf{H_0} = \begin{bmatrix} g(a_1, b_1, x_1) & \cdots & g(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ g\left(a_1, b_1, x_{N_1}\right) & \cdots & g\left(a_L, b_L, x_{N_1}\right) \end{bmatrix} \tag{11}$$

Equation 10 can be rewritten as $\beta_0 = P_0 \mathbf{H}_0^T y_0$, where P_0 is the initial covariance matrix $P_0 = (\mathbf{H}_0^T \mathbf{H}_0)^{-1}$.

When a new batch of samples arrives, the new output weight vector β_{k+1} is computed using concepts of the Recursive Least Squared algorithm (RLS) (Liang et al., 2006), as follows:

$$\beta_{k+1} = \beta_k + P_{k+1} \mathbf{H}_{k+1}^{\mathrm{T}} (y_{k+1} - \mathbf{H}_{k+1} \beta_k), \tag{12}$$

$$P_{k+1} = P_k - \mathbf{H}_{k+1}^{\mathrm{T}}\left(I + \mathbf{H}_{k+1}P_k\mathbf{H}_{k+1}^{\mathrm{T}}\right)^{-1}\mathbf{H}_{k+1}P_k, \tag{13}$$

$$\mathbf{H}_{k+1} = \begin{bmatrix} g\left(a_1, b_1, x_{\left(\sum_{l=0}^{k} N_l\right)+1}\right) & \cdots & g\left(a_1, b_L, x_{\left(\sum_{l=0}^{k} N_l\right)+1}\right) \\ \vdots & \ddots & \vdots \\ g\left(a_1, b_1, x_{\sum_{l=0}^{k+1} N_l}\right) & \cdots & g\left(a_1, b_L, x_{\sum_{l=0}^{k+1} N_l}\right) \end{bmatrix}, \tag{14}$$

$$y_{k+1} = \left[y_{\left(\sum_{l=0}^{k} N_l\right)+1}, \cdots, y_{\sum_{l=0}^{k+1} N_l}\right]^{\mathrm{T}} \tag{15}$$

Liang et al., (2006) provide a detailed derivation of Eqs. 12 and 13. Therefore, the OS-ELM algorithm in a sample-based scenario, where each sample from S is provided sequentially, is depicted in Algorithm 2.

Algorithm 2. Learning algorithm for the sample-based OS-ELM

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; the number of samples for initial training phase, N_1; a number of hidden nodes L;

1. **Initialization**: assign input weights a_j and bias b_j randomly, $j = 1, \ldots, L$;
2. Calculate the hidden layer matrix H_0 with $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$ and Eq. 11;
3. Calculate the output weight β_0 through Eq. 10, where $y_0 = [y_1, \ldots, y_{N_1}]^{\mathrm{T}}$ and set $k = 1$, $t = N_1$;
4. **while** $t \leq T$ do:
    a. Present the $(k+1)$ −th batch $D_{k+1} \subset S$ defined in Eq. 9;
    b. Obtain the matrix H_{k+1} using D_{k+1} and Eq. 14;
    c. Set y_{k+1} using Eq. 15;
    d. Obtain P_{k+1} and β_{k+1} using Eqs. 13 and 12;
    e. Set $k \leftarrow k+1$, $t \leftarrow t + N_{k+1}$
5. **end while**

end

2.3.3 Ensemble of models

Ensemble methods can be used to increase the overall prediction or classification accuracy. In this work, ensembles are studied in order to enhance the predictions of single OS-ELMs. An ensemble for prediction is a composite model made up of a combination of

individual base models. An ensemble combines a series of *k* learned models (or base predictors) $M_1, M_2, \dots, M_k$, with the aim of creating an improved composite prediction model. A given dataset *D* may be used to create *k* training subsets $D_1, D_2, \dots, D_k$, where D_i is used to train the classifier M_i. Given the new incoming samples, the individual models predict the future values. The ensemble returns a prediction based on the combination of individual predictors of the ensemble. Figure 4 presents the general scheme of the ensembles. (Han et al., 2012)

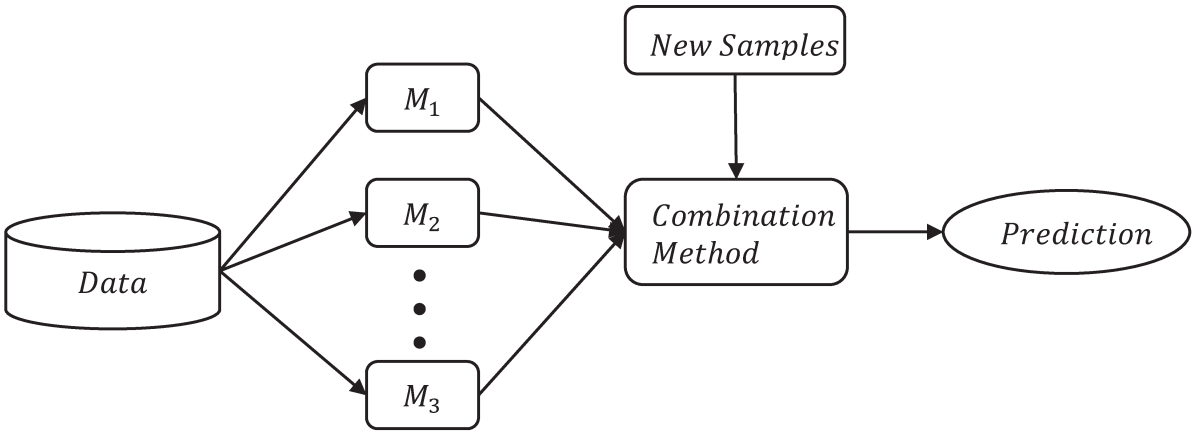

Figure 4. General structure of an Ensemble of models

Ensemble approaches tend to be more accurate than its base predictors. For example, when implementing the simple average combination strategy, where the ensemble's output is calculated with the arithmetic average of the individual models' outputs, some predictors may make mistakes predicting the sample X, but the ensemble will misclassify X only if over half of the base classifiers are wrong. Additionally, ensembles tend to perform better when there is significant diversity among the models (Han et al., 2012).

Considering the effects of PM mentioned above, real-time monitoring, forecasting and alert systems are needed to protect the population of harmful episodes of PM. However, the dynamic behaviors (concept drifts) presented by this kind of data streams can make this task more difficult. Therefore, traditional mining of these data may not be efficient, making necessary the incorporation of mechanisms to deal with these dynamic behaviors. Instance weighting, sliding windows and ensemble approaches have been used in previous works to deal with concept drifts, showing promising results. Furthermore, online

learning has proven to be efficient in changing environments due to the capability to adapt the prediction model to new concepts through the sequential learning. Thus, the aim of this work is to combine those approaches together with the online sequential version of the ELM to forecast hourly concentrations of PM.

3 Literature Review

The literature review was focused on the two main topics of this work: air pollutant concentration forecasting, particularly PM, and techniques to forecast time series (or data streams) with concept drift. Therefore, the goal of this chapter is to present the current state-of-the-art on these two topics.

3.1 Particulate Matter Forecasting

Two main approaches have been used to study and forecast air pollutants' concentration: deterministic and traditional machine learning approaches. Novel deterministic approaches incorporate online meteorological-chemistry models based on elaborated 3D chemistry transport models. These models allow the study of physicochemical and meteorological processes, directly linked to the composition of atmospheric aerosols (Mao et al., 2017). Nevertheless, these models depend on detailed knowledge about the emissions and sources of different pollutants, complex chemical processes that occur in the atmosphere and meteorological conditions. Without this knowledge and information, air pollution concentration forecasting can be limited (Mao et al., 2017). Besides the advances achieved by current forecasting technologies, air quality is still a challenge because of the complexity of the processes involved and the influence of many parameters, which affect the forecasting models' performance (Bianco et al., 2017; Peng et al., 2017).

On the other hand, machine learning approaches are easier to be implemented, compared with deterministic models. Usually, traditional machine learning models like neural networks require large amounts of data associated with different pollutants and atmospheric variables to be trained. Nevertheless, the widespread concern about air quality by public authorities around the world has led authorities to monitor different air pollutants and meteorological variables, thus making available large amounts of data to train this kind of models. Additionally, online learning algorithms which can operate without the need of large training datasets are suitable for this task (Bueno, Coelho, & Bertini, 2017; Peng et al., 2017).

Statistical and machine learning approaches are based on identifying patterns of data that allow predicting future pollutant concentration. Usually, these approaches require less computational resources and offer results with considerable accuracy, compared with deterministic models. Among the most recent and widely used approaches for air pollution forecasting, multiple nonlinear regression, neural networks, neuro-fuzzy and hidden Markov models can be found (Peng et al., 2017).

Monitoring and forecasting systems have been developed and applied to forecast PM concentration. In this context, information systems, especially those based on machine learning techniques, have shown promising results in PM forecasting. For example, Oprea et al., (2015) developed an intelligent system capable of performing 24-hour ahead forecasting of $PM_{2.5}$ concentration levels and of sending early warnings to protect children with health problems. Raimondo et al., (2007) used Artificial Neural Networks (ANN) and Support Vector Machines (SVM) to forecast PM_{10} concentration, and Souza et al., (2015) proposed an ensemble of ANNs to forecast daily concentrations of PM_{10}. In Souza et al. (2015), the ensemble approach presented better performance compared to individual ANNs. Shaban et al., (2016) indicate the importance of considering data changes over time for real-time forecasting of air pollutants, which can be achieved by online sequential learning algorithms. Such algorithms constantly update their forecasting models with newly received data, either sample-by-sample or batch-by-batch (blocks of data). In contrast with off-line approaches, online learners do not require a full retraining of the forecasting model whenever new data is available, which speeds up the whole process.

Mao et al., (2017) joined deterministic and machine learning approaches to forecast hourly $PM_{2.5}$ concentrations. This approach used meteorological data as the input of a Multilayer Perceptron (MLP) together with data from a satellite remote sensing technique to monitor air quality: the Satellite-derived Aerosol Optical Depth (AOD). The resulting configuration presented good performance, predicting hourly $PM_{2.5}$ concentrations in the south of China with a number of steps ahead. The incorporation of transport of “dirty” and “clean” air information, measured by the AOD, improved the accuracy of $PM_{2.5}$ predictions.

Biancofiore et al., (2017) evaluated the PM forecasting performance of three models: a Multiple Linear Regression Model (MLR), an ANN with recursive architecture and an ANN

without recurrent architecture. Meteorological parameters and PM_{10} concentration served as inputs of the models, which were developed to predict daily average PM_{10} concentrations three days ahead. Data collected from 2011 to 2013 in the urban area of Pescara, Italy, was used in this work. Measurements included temperature, relative humidity, wind speed/direction, pressure, and concentration of PM_{10}, CO, ozone and nitrogen oxide, among others. An analysis over PM_{10} data allowed the identification of an annual cycle pattern with higher concentrations during winters and lower concentrations during summers. This can be seen as an evidence of the strong influence of meteorological parameters, in this case determined by the seasons of the year, on the concentration of PM (Mao et al., 2017; Peng et al., 2017). Furthermore, a strong correlation between the concentration of PM_{10} and carbon monoxide (CO) was identified, suggesting common sources for PM_{10} and CO. Results showed that the recursive neural network performed better in all the evaluated scenarios than MLR and the ANN without recurrent architecture. The inclusion of CO as an additional parameter improved the performance of all models. This may suggest that, in scenarios with a well-identified CO source (like emissions due to fossil fuel combustion in the case of Pescara), the concentration of this pollutant can be used as an additional input of models that forecast PM.

Evaluation of online sequential learning approaches was conducted by Peng et al., (2017), in order to study the impact of online updating capabilities to air pollution forecasting models. Peng et al., (2017) evaluated MLRs, Multi-layer Perceptron Neural Networks (MLPNN) and ELMs. Online learning versions of MLR and ELM (OSMLR and OS-ELM respectively), updated with daily collected data, were compared with MLPNN updated seasonally and with the climatology model GEM-MACH15. Data from 2009 to 2014 of meteorological variables and air quality of six monitoring stations of Canadian cities were used in the experiments. The OS-ELM outperformed the other methods over the six stations, including the climatology model. The MLPNN, updated every 3 months due to its high computational cost, presented poorer performance than the daily updated approaches. This is an important issue, considering that all models were initially trained with the same data corresponding to the first 2 of the 5 years available. Results regarding the MLPNN may indicate that the initial training data did not provide enough statistical

information for the MLPNN to learn all the patterns of the dataset, and also that new patterns may have appeared over time. Therefore, since MLPNN cannot adapt to changes of the underlying distributions of the data, online sequential learning approaches like OS-ELM tend to perform better. Such approach required about 10 times less computing resources, even with daily updates than the MLPNN (Huang et al., 2006; Liang et al., 2016; Peng et al., 2017).

Bueno Coelho & Bertini (2017) compared online and offline algorithms based on Extreme Learning Machines (ELM) applied to forecast hourly concentrations of PM. In special, that work reported results organizing ELMs as a dynamic ensemble similar to the streaming ensemble algorithm (Street & Kim, 2001), the Ensemble of Online Learners with Substitution of Models EOS. The experimental results showed that online sequential approaches (OS-ELM) performed better in PM_{10} scenarios than offline approaches (ELM) and that the EOS ensemble improved the stability of the results.

The results reported by Peng et al., (2017) and Shaban et al., (2016), together with the theoretical background presented in the last chapter, reinforce the hypothesis that PM data may present concept drifts, therefore current approaches to deal with this phenomena are discussed in the next section.

3.2 Concept drift detection algorithms

Concept drift detection consists in determining whether the environment has changed sufficiently, so that the classification/forecasting models cannot predict accurately the current data.

The Drift Detection Mechanism (DDM) monitors the online error of the model. For each new instance (x_i, y_i), the model predicts the target variable $\hat{y}_i$, based on x_i. The classification/forecasting error is modeled as Bernoulli trials. If the new instances to be predicted have the same distribution, the error rate is expected to be constant or smaller. Otherwise, if the distribution of the incoming samples is different, the error increases, which means that a concept drift occurred and that the current model is not effective for the incoming instances (Gama et al., 2004). DDM offers good performance in detecting abrupt and fast gradual changes, but it faces difficulties when the rate of change is slow

(Gama et al., 2004). The early drift detection method (EDDM), proposed by Baena-García et al. (2006), offers an improvement over DDM, as its analysis is based on the distance between errors, instead of only considering the number of prediction errors. This method can be used with any learning algorithm.

3.2.1 Ensemble based approaches to handle Concept drift

The Incremental Local Learning Soft Sensing Algorithm (ILLSA) is an ensemble approach based on Recursive Partial Least Squares (RPLS). ILLSA divides historical data into partitions, which represent different states of the process. A model is built based on each dataset. The model's weights on the new sample are calculated using the posterior probability obtained by a Bayesian framework. Experiments showed that ILLSA leads to better accuracy, when compared to traditional RPLS (Kadlec & Gabrys, 2010).

Additive-Expert (AddExp) is an ensemble of predictive models (referred to as *experts*), each with an associated weight. The algorithm uses a weighted vote that considers the outputs of all experts. When a new sample arrives, the algorithm output is determined by the expert prediction with the greatest weight. The weights of the experts with low accuracy are decreased by a multiplicative constant β. If the overall prediction is incorrect, a new expert is added to the ensemble and all experts are re-trained with the sample. AddExp can be used with any online learner algorithm, such as least square regression learners and naïve Bayes for regression (Kolter & Maloof, 2005). Experimental results have shown better performance and faster adaptation of AddExp based on naïve Bayes, when compared to traditional naïve Bayes algorithm for regression. Similar results were found for AddExp based on the least squares regression algorithm.

Online Ensemble using Ordered Aggregation (OEOA) is an ensemble proposed by Soares & Araújo, (2016), which uses a quality metric to produce a decreasing order of the best models for a given data. This approach is capable of providing online prediction in non-stationary environments. A data window of fixed size is kept and a new model is trained with the new incoming data when the ensemble's performance is deteriorating. Artificial and real-world datasets were employed to evaluate the predictive performance of OEOA over state-of-the-art approaches. The results showed that OEOA delivers more

accurate estimations of output variables in industrial applications, when compared to other state-of-the-art ensembles in the literature such as AddExp and EOS-ELM.

Learn^{++}.NSE (Elwell & Polikar, 2011) is a batch-based ensemble learning algorithm that uses weighted majority voting. In this algorithm, the weights are updated based on the classification error on current and past environments. A drift detection mechanism is implemented and uses only current data for training. Learn^{++}.NSE can handle a wide variety of drifts such as abrupt, gradual and cyclical. It only discards a classifier temporarily, which is particularly useful in cyclical environments. In order to evaluate the Learn^{++}.NSE algorithm, several datasets that simulate different scenarios of non-stationary environments, such as abrupt, gradual and cyclical drift, were evaluated. Learn^{++}.NSE algorithm can be implemented using different base learners, such as Naïve Bayes, Support Vector Machines (SVM) and classification and regression trees (CART). The reported experiments allowed the comparison of Learn^{++}.NSE with other concept drift ensemble approaches, such as SEA, DWM and AdaBoost weighting. The results showed the versatility of Learn^{++}.NSE to adapt to a wide variety of drift scenarios and also its higher efficiency, since it uses existing knowledge by reactivating early classifiers when they are needed the most, and disabling them when they are not relevant.

Soares & Araújo (2015) proposed an Online Weighted Ensemble of forecasting models (OWE), which is able to incrementally learn, sample by sample, in the presence of several types of changes, and simultaneously retain old information in recurring scenarios. OWE employs several adaptive mechanisms to deal with different types of drifts (Gomes Soares & Araújo, 2015). OWE was compared with state-of-the-art approaches, using two artificial datasets and two real-world industrial datasets, and the results show the ability of OWE to handle several types of drifts, such as abrupt, gradual and cyclic.

The Dynamic and Online Ensemble Regression (DOER) offers fast adaptation capability for online prediction of variables measured at low sampling in non-stationary environments (Soares & Araújo, 2015). DOER is an online ensemble for regression with the following properties:

- Online inclusion and removal of models to keep only the most accurate models;
- Dynamic model weighting based on online predictions;

- Online adaptation of models' parameters.

Experiments were performed in scenarios that required faster adaptation capability and, when DOER was compared to four online strategies using the single model OS-ELM (Liang et al, 2016) algorithm and five online ensemble algorithms (EOS-ELM, AddExp, Online Bagging (OB), Learn^{++}.NSE and OAUE – Soares & Araújo, 2015), it showed higher accuracy.

3.2.2 Approaches that handle concept drift in an implicit way

Online Sequential Extreme Learning Machine (OS-ELM) is the online variant of Extreme Learning Machines (ELMs) that can learn from samples or batches of data. It combines the ELM advantages as speed and generalization performance with the sequential learning process (Liang et al, 2006). When a new sample or batch is available, it is used to update the learning model. Due to its online nature, OS-ELM is able to handle concept drift in an implicit way (as mentioned before). However, since OS-ELM updates the model every time a new instance is available**,** the computational cost is high compared with the original ELM, especially when the rate of incoming data is high (Cavalcante & Oliveira, 2015). Liang et al (2006) compared OS-ELM with other sequential learning algorithms (SGBP, RAN, RAEKF, MRAN, GAP-RBF– Liang et al., 2006) on real world datasets for regression, classification and time series forecast problems, and the results indicated that the OS-ELM achieves better generalization and requires lower training time.

Cavalcante & Oliveira (2015) proposed a learning method, which behaves like an online and offline learner, switching the operation to react to changes in the data in order to reduce the computational resources, when compared with single OS-ELM. Their work implemented OS-ELMs combined with DDM and also OS-ELMs combined with Exponentially Weighed Moving Average Concept Drift Detection, known as ECDD (Ross et al., 2010). Two metrics, accuracy and processing time, were evaluated in the experiments, which used artificial and real-world datasets. The results showed that the combination of OS-ELM with ECDD reduces the processing time when compared with single OS-ELMs for time series forecast.

This chapter described current approaches for PM forecasting found in the literature. In this work are studied machine learning algorithms to forecast air pollution concentrations, since they do not require detailed knowledge about complex meteorological and atmospheric processes, and are easier to implement when compared to deterministic models. These approaches are good alternatives to sophisticated deterministic forecasting models, not only capable of presenting comparable performances but also requiring less computational resources (Peng et al., 2017).

Although, ANNs and MLPNNs have been used to forecast PM concentrations in many scenarios, studies suggest that new patterns may appear in PM data, thus, making limited the capability of prediction of ANNs and MLPNNs, since they are not able to adapt to changes. On the other hand, OS-ELMs has proven to be better than traditional algorithms like MLPNNs in such scenarios, since they can adapt faster to new changes.

The focus of this work is to improve the estimation accuracy of PM concentrations. Considering the dynamic behavior of PM data reported in the literature, online learning approach was chosen as the base strategy to handle concept drift presented in such data. The OS-ELM was selected the base algorithm of the experiments conducted in this work.

Additionally, considering the results found in a previous work (Bueno et al., 2017), where an ensemble of OS-ELMs implementing the updating ensemble scheme proposed by Street and Kim (2001) improved the stability of the results when compared to single OS-ELMs, this work builds on that work and propose new mechanisms to deal with concept drifts in order to improve the forecasting accuracy of ensembles. The proposed mechanisms studied here are incorporated and evaluated on EOS and DOER (Soares & Araújo, 2015), being the last chosen since it incorporates several concept drift handling strategies and shows good performance in concept drift scenarios.

4 Proposed mechanisms for dynamic ensembles

Considering the OS-ELM as the base forecasting model in the online dynamic ensemble approaches defined in the last section, in this chapter both the Dynamic and Online Ensemble Regression (DOER), proposed by Soares & Araújo (2015), and the Ensemble of Online Learners with Substitution of Models (EOS), using the adaptation scheme proposed by Street & Kim, (2001) and adapted in this work, are described in more details, together with the additional mechanisms proposed for dynamic ensembles.

The first strategy studied here, DOER, is an ensemble with dynamic inclusion and removal of models, dynamic adaptation of the model weights and online adaptation of model's parameters (Soares & Araújo, 2015). The second approach, EOS, is an online ensemble that implements the component updating scheme proposed by Street and Kim (2001) with inclusion and removal of models at a fixed frequency (Bueno et al., 2017). Finally, the additional mechanisms for dynamic ensembles proposed in this work are basically strategies to handle concept drifts in an implicit way. Such mechanisms were incorporated into DOER and EOS.

4.1 Ensemble approaches

Two ensemble learning approaches were considered in this work, according to the insights found in the literature review. In ensemble learning, the outputs of a set of models are combined to get the final prediction. Between the strategies to combine the components' outputs, the Simple Average strategy calculates the ensemble output with the average of individual components' outputs. On the other hand, in Weighted Average-based ensembles, the outputs are calculated using components weights, which determine the relative importance of a component in the ensemble. Usually, more accurate components get larger weights, making its contribution more relevant in the final output.

Ensembles can be static, in which their components remain the same throughout the operation, or dynamic, in which the ensemble's components are included, replaced or excluded by new ones according to a predefined criteria. As the incorporation and exclusion of models into ensembles has been considered an important approach for dealing with concept drifts (Gomes Soares & Araujo, 2015), this work evaluates this

approach through two ensemble learning algorithms: the Ensemble of Online Learners with Substitution of Models (EOS) and the Dynamic and On-line Ensemble Regression (DOER).

4.1.1 The Ensemble of Online Learners with Substitution of Models

The Ensemble of Online Learners with Substitution of Models (EOS) (Bueno et al., 2017) is an ensemble of online learners, which implements sliding windows and the ensemble updating scheme originally proposed by Street & Kim (2001). EOS was proposed as part of this dissertation and its application for PM_{10} forecasting in sample-based scenarios was published in Bueno et al., (2017). The EOS algorithm is described in Algorithm 3.

Algorithm 3. Ensemble of Online Learners with Substitution of Models (EOS)

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; window's size, m; number of samples for initial training phase, N_1; an online supervised learner f, maximum number of models in the ensemble R; an ensemble Σ; inclusion/replacement frequency λ

1. Initialization: set $\Sigma \leftarrow \emptyset$, $t = N_1 + 1$, and the initial training data as $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$;
2. $f_k \leftarrow$ obtain a model trained with D_1, $\Sigma \leftarrow \Sigma + f_k$, $k = 1$, and $r = 0$;
3. **while** $t \leq T$ do:
   a. slide the window: $D_t = \{(x_t, y_t)\}_{t=\, t-(m-1)}^{t} \subset S$;
   b. obtain the output prediction of Σ using x_t;
   c. retrain/update all models of Σ using D_t; $t \leftarrow t + 1$, $r = r + 1$;
   d. **if** $t \; mod \; \lambda = 0$
      $f_0 \leftarrow$ obtain a new model trained with $D_{temp} = \{(x_t, y_t)\}_{t=t-(\lambda-1)}^{t} \subset S$;
      if $k < R$
         a. include f_k to Σ: $\Sigma \leftarrow \Sigma \cup f_k$ and set $k = k + 1$;
      else
         b. obtain MSE_j with D_{temp} for each model $f_j \, \epsilon \, \Sigma$,
         c. replace the model with the worst MSE_j: $f_j \leftarrow f_0$
4. **end while**

end

EOS implements dynamic mechanisms for inclusion and exclusion of components at a fixed rate. This allows EOS to adapt to changing environments, through the incorporation

of new components trained with a batch of recent samples and the elimination of those old components with the worst performance over the past samples.

The data stream $S = \{(x_t, y_t) | x_t \in \mathbb{R}^{rx1}, y_t \in \mathbb{R}, t = 1, \dots, T\}$ is used as input of the algorithm, according to the definition at the beginning of this chapter. EOS also requires the definition of the window size m; the online supervised learner f; the number of samples for the initial training phase N_1; the maximum number of models in the ensemble R; the ensemble of online learners Σ; and the frequency of inclusion and substitution of components λ.

In Step 1 the initial training batch D_1 is defined with the first samples of the data stream and $\Sigma \leftarrow \emptyset$ denotes that the ensemble is initially empty. In the initialization phase (Step 2), a component f_k is trained with the samples in D_1, and subsequently added to the ensemble Σ. The number of components in the ensemble k is updated and the control variable that counts the number of iterations r is initialized. From Steps 3 to 4, the SW is shifted along the data stream. Step 3a incorporates the new arriving sample to the window and discards the oldest sample, according to the defined window size m. The ensemble output $F(x_t)$ is calculated in Step 3b using simple average of the individual component's outputs. All components of the ensemble are updated with the samples in the SW D_t in Step 3c, then, in Step 3d, it is evaluated whether the current iteration is equal to λ, the frequency of inclusion and substitution of models. If so, a new component f_k is trained with the last samples in D_{temp}, of size λ, and r is restarted.

If the current number of components in the ensemble k is less than the maximum number of components in the ensemble R, f_k is added to the ensemble directly and k is updated. Otherwise, the MSE_j of each component j of the ensemble is calculated with the set of samples D_{temp} and the component with the worst performance (the highest MSE) is substituted by the new component trained from scratch (f_k) with D_{temp}.

The EOS algorithm differs from others approaches in the inclusion exclusion scheme, which incorporate new models to the ensemble at a fixed rate without any implicit mechanism to determine if a drift is present. Another feature of the EOS is the size of the

window. EOS selects the most recent samples of the data stream in a window, to updated the components of the ensemble in each iteration.

4.1.2 The Dynamic and on-line ensemble regression

The Dynamic and On-line Ensemble Regression (DOER) approach (Soares & Araújo, 2015) is an online sample-based ensemble of online learners, designed for regression in changing environments. The structure of DOER is presented in Algorithm 4.

DOER offers dynamic adaptation of components' weights according to the accuracy of components' predictions on the most recent samples, assigning larger weights to the most accurate components, so inaccurate components would not degrade the ensemble's performance. This approach also enables the inclusion and removal of components with bad performance. The pruning strategy of DOER removes the components with the worst accuracy evaluated in the most recent samples, by ensuring that just the most accurate components are used to predict new instances

In order to adjust the ensemble to changes, DOER uses a sliding window with the most recent samples to train and incorporate new components when the ensemble's performance is not satisfactory. It is important to highlight that the SW used in DOER differs from the SW used in the EOS algorithm, since EOS updates the ensemble's components with the samples in the SW, while DOER uses the SW to train new components to be added to the ensemble.

The data stream $S = \{(x_t, y_t) | x_t \in \mathbb{R}^{\mathbf{rx1}}, y_t \in \mathbb{R}, t = 1, \dots, T\}$ and the size of the sliding window m are defined as inputs. This SW is used to train the new components to be added to the ensemble. It must also be defined the online supervised learner f; the number of samples for the initial training phase N_1; the factor α that controls the inclusion of models and the maximum number R of components in the ensemble.

Algorithm 4. Dynamic and on-line ensemble regression (DOER)

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; window's size, m; number of samples for initial training phase, N_1; an online supervised learner f, factor of inclusion of new models α; maximum number of models in the ensemble R; an ensemble Σ;

1. **Initialization**: set $\Sigma \leftarrow \emptyset$, $t = N_1 + 1$, and the initial training data as $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$;
2. $f_k \leftarrow$ obtain a model trained with D_1; set $life_k = 0$, $MSE_k^t = 0$, $\mathrm{w_k} = 1$, $\Sigma \leftarrow \Sigma + f_k$ and $k = 1$;
3. **while** $t \leq T$ do:
   a. slide the window: $D_t = \{(x_t, y_t)\}_{t=\, t-(m-1)}^{t} \subset S$;
   b. obtain the output prediction $F(x_t)$ of Σ as: $F(x_t) = \left(\sum_{j=1}^{k} w_j f_j(x_t)\right) / \sum_{j=1}^{k} w_j$;
   c. for all models $f_j \in \Sigma$, obtain the prediction error e_j^t on x_i as $e_j^t = (y_t - f_j(x_t))^2$, and set $life_j = life_j + 1$;
   d. obtain MSE_j^t for each $f_j \in \Sigma$ using Eq. 17;
   e. calculate the weight for each model from Σ using Eq. 18;
   f. retrain all models of Σ using D_t;
   g. $t \leftarrow t + 1$
   h. **if** $|(F(x_t) - y_t)/y_t| > \alpha$
      $f_0 \leftarrow$ obtain a new model trained with D_t; set $life_0 = 0$; $MSE_0^t = 0$; and $w_0 = 1$;
      if $k < R$
         a. include f_k to Σ: $\Sigma \leftarrow \Sigma \cup f_k$ and set $k = k + 1$;
      else
         b. replace the model f_j by f_0, where $j = \min_{v=1,\ldots,k}(MSE_v^t)$: $f_j \leftarrow f_0$
4. **end while**

end

In the initialization phase, the number of models k is set and batch D_1 organized with the first N_1 samples of the data stream. Step 2 trains the first component of the ensemble with the batch of samples in the SW and k is updated.

From Step 3 to Step 4, the SW is dislocated to add the new incoming sample to the window and remove the oldest one (Step 3a). The ensemble output is calculated using the weighted average of components' outputs (Step 3b). The error of each component f_j of

the ensemble $\Sigma, (j = 1, \dots, k)$ is calculated in Step 3c, using the current sample (x_t, y_t), according to Eq. 16.

$$e_j^t = (y_t - f_j(x_t))^2 \tag{16}$$

where $f_j(x_t)$ is the prediction of the component f_j. Once the error is calculated, the $life_j$ is incremented. Then, in Step 3d, the MSE_j^t of the current window is calculated for each component of the ensemble, as given in Eq. 17.

$$MSE_j^t = \begin{cases} 0, & \text{if } life_j = 0, \\ \dfrac{life_j - 1}{life_j} . MSE_j^{t-1} + \dfrac{1}{life_j} . e_j^t, & \text{if } 1 \leq life_j \leq m, \\ MSE_j^{t-1} + \dfrac{e_j^t}{m} - \dfrac{e_j^{t-m}}{m}, & \text{if } life_j > m \end{cases} \tag{17}$$

The goal of Eq. 17 is to estimate the average error of each component f_j on the last m samples using the mean squared error (MSE). This approach allows the estimation of the MSE of the current window, this is to say, a vector e_j with the last m prediction errors is considered in the calculation of the MSE_j^t. Step 3e calculates the weights w_j of each component f_j according to its error MSE_j^t as in Eq. 18.

$$w_j = \exp\left(-\frac{MSE_j^t - \text{med}(MSE^t)}{\text{med}(MSE^t)}\right), \tag{18}$$

where $MSE^t = [MSE_1^t, \dots, MSE_k^t]$ and $\text{med}(MSE^t)$ is the median value of the components' errors, MSE^t. Equation 18 transforms the MSE^t of each component in a way that the components with errors closer to the median obtain a weigh equal to 1, while components with MSE^t lower or higher than the median obtain weights exponentially higher or lower, respectively. This approach avoids that components with low accuracy impact negatively the ensemble's output. In Step 3f, all components are updated, and, after this, it is evaluated if a new component must be added according to α (Step 3g). In that case, a new model is trained with the current window D_t and weight equal to 1. If the current

number of components in the ensemble is smaller than the defined limit R, the new component is added directly; otherwise, the component f_j with the worst MSE_j^t is replaced by the new component f_k.

Table 4 resumes the main features of the dynamic ensembles algorithms mentioned above. Both approaches incrementally add components in the initial phase of the operation, the main difference is the frequency those components are added to the ensemble: while EOS updates the ensemble at a predefined frequency, DOER calculates in every step if an update is required. Regarding the sliding window, the EOS uses a small window to update the components of the ensemble and DOER uses a larger window to train the new components to be added to the ensemble.

Table 4. Comparative table of the EOS and DOER dynamic ensembles approaches

Features	EOS	DOER
Initial training	Incremental	Incremental
Sliding Window	For updating the ensemble	For training new components
Ensemble's Combination Strategy	Simple average	Weighted average
Incorporation of new components	Fixed frequency	Dynamic frequency

4.1.3 Proposed ensemble approaches based on DOER and EOS

New mechanisms for online dynamic ensembles were proposed and evaluated in this work. These mechanisms were incorporated into DOER and EOS algorithms, and will be described here.

4.1.3.1 Rank of components

This approach incorporates a simple rank of components to the original DOER and EOS algorithms. The MSE of the component is used to rank the components of the ensemble at each iteration. This strategy selects the most accurate components of the ensemble in $c \subset \Sigma$ to predict the next sample, sorting the components of the ensemble with a simple ranking function according to the MSE (Step 3(g) of Algorithms 5 and 6). The structures of EOS-rank and DOER-rank are presented in Algorithms 5 and 6, respectively. The main features of the EOS-Rank and DOER-Rank dynamic ensembles are summarized in tables 5 and 6 respectively.

Algorithm 5. The Ensemble of Online Learners with Substitution of Models with Ranking of Components (EOS-rank)

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; the window size, m; the number of samples for initial training phase, N_1; an online supervised learner, f; the maximum number of models in the ensemble, R; an ensemble, Σ; the inclusion/replacement frequency, λ; the size of the subset of components, l.

1. **Initialization:** set $\Sigma \leftarrow \emptyset$, $t = N_1 + 1$, and the initial training data as $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$; $t = N_1$;
2. obtain a model, f_k, trained with D_1; set $life_k = 0$, $MSE_k^t = 0$, $\mathrm{w_k} = 1$, $\Sigma \leftarrow \Sigma + f_k$, $k = 1$; and $c = \Sigma$;
3. **while** $t \leq T$ do:
   a. slide the window: $D_t = \{(x_t, y_t)\}_{t=\,t-(m-1)}^{t} \subset S$;
   b. obtain the output prediction $F(x_t)$ of c as: $F(x_t) = \left(\sum_{j=1}^{k} w_j f_j(x_t)\right) / \sum_{j=1}^{k} w_j$;
   c. for all models $f_j \in \Sigma$, obtain the prediction error e_j^t on x_i as $e_j^t = (y_t - f_j(x_t))^2$, and set $life_j = life_j + 1$;
   d. obtain MSE_j^t for each $f_j \in \Sigma$ using Eq. (14);
   e. calculate the weight for each model from Σ using Eq. (15);
   f. update all models of Σ using D_t; $t \leftarrow t + 1$;
   g. rank the components of the ensemble according to their MSE and obtain the subset $c = rank(\Sigma, l)$, of size l;
   h. **if** $t \; mod \; \lambda = 0$
      obtain a new model, f_0, trained with $D_{temp} = \{(x_t, y_t)\}_{t=t-(\lambda-1)}^{t} \subset S$; set $life_0 = 0$; $MSE_0^t = 0$; and $w_0 = 1$;
      if $k < R$
         a. include f_k to Σ: , $\Sigma \leftarrow \Sigma \cup f_k$ and set $k = k + 1$;
      else
         b. obtain MSE_j with D_{temp} for each model $f_j \, \epsilon \, \Sigma$;
         c. replace the model with the worst MSE_j: $f_j \leftarrow f_0$;
4. **end while**

end

Table 5. Comparative table of the EOS and EOS-Rank dynamic ensemble approaches

Features	EOS	EOS-Rank
Initial training	Incremental	Incremental
Sliding Window	For updating the ensemble	For updating the ensemble
Ensemble's Combination Strategy	Simple average	Weighted average
Incorporation of new components	Fixed frequency	Fixed frequency
Dynamic Ensemble Mechanism		Rank of components

The goal of this approach is to enable a faster inclusion of components that contribute to the ensemble's final output, since only the predictions of components in c are combined. This allows the ensemble to exclude more than one low accuracy component of the ensemble's output at once, differently from DOER and EOS algorithms, where no more than one component can be replaced at once.This approach also allows that components with a relatively bad performance remain for a longer time into the ensemble, since those components can be kept out of c until their individual performance improves and, then, included again into c, thus maintaining the previously acquired knowledge for more time. In changing environments, it can be risky to remove a component that may be important in the future, especially in scenarios with recurring drifts, where the knowledge of a component can be relevant when that concept is restored (Soares & Araújo, 2015).

Table 6. Comparative table of the DOER and DOER-Rank dynamic ensemble approaches

Features	DOER	DOER-Rank
Initial training	Incremental	Incremental
Sliding Window	For training new components	For training new components
Ensemble's Combination Strategy	Weighted average	Weighted average
Incorporation of new components	Dynamic frequency	Dynamic frequency
Dynamic Ensemble Mechanism		Rank of components

EOS-rank incorporates the dynamic parameterization of components as in DOER algorithm. The MSE calculated in each iteration for each component is used to rank the ensemble, in order to select the components with the lowest errors in the subset c, as described in Step 3(g) of Algorithm 5. DOER-rank also incorporates the ranking mechanism using the MSE of each component. The rank is applied after updating the ensemble, as shown in Algorithm 5, Step 3(g).

Algorithm 6. The Dynamic and Online Ensemble Regression with Ranking of Components (DOER-rank);

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; the window size, m; the number of samples for initial training phase, N_1; an online supervised learner, f; the factor of inclusion of new models, α; the maximum number of models in the ensemble, R; an ensemble Σ;

1. **Initialization**: set $\Sigma \leftarrow \emptyset$, $t = N_1 + 1$, and the initial training data as $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$; $t = N_1$;
2. obtain a model, f_k, trained with D_1; set $life_k = 0$, $MSE_k^t = 0$, $\mathrm{w_k} = 1$, $\Sigma \leftarrow \Sigma + f_k$, $k = 1$;
3. **while** $t \leq T$ do:
   a. slide the window: $D_t = \{(x_t, y_t)\}_{t=\,t-(m-1)}^{t} \subset S$;
   b. obtain the output prediction $F(x_t)$ of Σ as: $F(x_t) = \left(\sum_{j=1}^{k} w_j f_j(x_t)\right) / \sum_{j=1}^{k} w_j$;
   c. for all models $f_j \in \Sigma$, obtain the prediction error e_j^t on x_i as $e_j^t = (y_t - f_j(x_t))^2$, and set $life_j = life_j + 1$;
   d. obtain MSE_j^t for each $f_j \in \Sigma$ using Eq. (14);
   e. calculate the weight for each model from Σ using Eq. (15);
   f. update all models of Σ using D_t;
   g. rank the components of the ensemble according to their MSE and obtain the subset $c = rank(\Sigma, MSE, l)$, of size l; $t \leftarrow t + 1$;
   h. **if** $|(F(x_t) - y_t)/y_t| > \alpha$
      obtain a new model, f_0, trained with D_t; set $life_0 = 0$; $MSE_0^t = 0$; and $w_0 = 1$;
      if $k < R$
         a. include f_k to Σ: $\Sigma \leftarrow \Sigma \cup f_k$ and set $k = k + 1$;
      else
         b. replace the model f_j by f_0, where $j = \min_{v=1,\ldots,k}(MSE_v^t)$: $f_j \leftarrow f_0$;
4. **end while**

end

4.1.3.2Initial Ensemble and Weighted Average

The original EOS incorporates components to the ensemble incrementally along its operation, until the limit of components is reached. Then, inaccurate components are substituted. This allows the ensemble to better adapt to changes, incorporating new data patterns that may emerge, through new components trained with the most recent samples.

Nevertheless, in the early stages of EOS operation, low accuracy components may affect the ensemble's accuracy if the number of components is small, thus affecting the overall performance of the algorithm. It is important to consider that EOS can operate for more time with fewer components than DOER, since the frequency of incorporation/substitution of components is lower in EOS than in DOER, as DOER evaluates, in each iteration, whether it is possible to incorporate a new component. In order to mitigate this effect, EOS-D is proposed. Algorithm 7 presents the structure of EOS-D.

The original EOS was modified to incorporate an initial ensemble of components and DOER's weighted average aggregation strategy. Hence, R components are trained in the initial stage of this approach with D_1, and the component's weights are calculated according to Eq. (15). Components are also initialized with weights $w_j = 1$ and $life_j = 0$, as in DOER. The main features of the EOS-D are summarized in Table 7.

Table 7. Comparative table of the EOS and EOS-D dynamic ensemble approaches

Features	**EOS**	**EOS-D**
Initial training	Incremental	All components trained at the beginning
Sliding Window	For updating the ensemble	For updating the ensemble
Ensemble's Combination Strategy	Simple average	Weighted average
Incorporation of new components	Fixed frequency	Dynamic frequency
Dynamic Ensemble Mechanism		Rank of components

Algorithm 7. The Dynamic Ensemble of Online Learners with Substitution of Models using weighted average (EOS-D).

input: a data stream $S = \{(x_t, y_t)\}_{t=1}^{T}$; the window size, m; the number of samples for initial training phase, N_1; an online supervised learner, f; the maximum number of models in the ensemble, R; an ensemble Σ; the inclusion/replacement frequency, λ;

1. **Initialization:** set $\Sigma \leftarrow \emptyset$, $t = N_1 + 1$, and the initial training data as $D_1 = \{(x_t, y_t)\}_{t=1}^{N_1} \subset S$;
2. **for** k=1 **to** R
   a. obtain a model, f_k, trained with D_1; set $life_k = 0$, $MSE_k^t = 0$, $w_k = 1$, $\Sigma \leftarrow \Sigma + f_k$;
3. **while** $t \leq T$ do:
   a. slide the window: $D_t = \{(x_t, y_t)\}_{t=\, t-(m-1)}^{t} \subset S$;
   b. obtain the output prediction $F(x_t)$ of c as: $F(x_t) = \left(\sum_{j=1}^{k} w_j f_j(x_t)\right) / \sum_{j=1}^{k} w_j$;
   c. for all models $f_j \in \Sigma$, obtain the prediction error e_j^t on x_i as $e_j^t = (y_t - f_j(x_t))^2$, and set $life_j = life_j + 1$;
   d. obtain MSE_j^t for each $f_j \in \Sigma$ using Eq. (14);
   e. calculate the weight for each model from Σ using Eq. (15);
   f. update all models of Σ using D_t; $t \leftarrow t + 1$;
   g. **if** $t \; mod \; \lambda = 0$
      $f_0 \leftarrow$ obtain a new model trained with $D_{temp} = \{(x_t, y_t)\}_{t=t-(\lambda-1)}^{t} \subset S$; set $r = 0$; set $life_0 = 0$; $MSE_0^t = 0$;
      if $k < R$
         a. include f_k to Σ: , $\Sigma \leftarrow \Sigma \cup f_k$ and set $k = k + 1$;
      else
         b. obtain MSE_j with D_{temp} for each model $f_j \in \Sigma$, for $j = 1, \ldots, R$;
         c. replace the model with the worst MSE_j: $f_j \leftarrow f_0$, for $j = 1, \ldots, R$;
4. **end while**

end

4.2 Final Remarks

Considering the algorithms and strategies described above, six different configurations based on OS-ELM, DOER and EOS were evaluated in the experiments that will be discussed in Chapter 5:

- The OS-ELM with Sliding Window (OS-ELMsw);
- The original Ensemble of Online Learners with Substitution of Models (EOS);
- The Dynamic and on-line ensemble regression (DOER);
- The Ensemble of Online Learners with Substitution of Models with ranking of components (EOS-rank);
- The Dynamic and on-line ensemble regression with ranking of components (DOER-rank);
- The Dynamic Ensemble of Online Learners with Substitution of Models using weighted average (EOS-D).

These algorithms will enable a comparative study of the strategies when evaluated in real and artificial sample-based scenarios. The goal of this comparison is to identify the most suitable approach for different types of concept drift and, ultimately, the best approach for PM_{10} forecasting scenarios. The novelty of this proposal and main contribution of this work is to incorporate mechanisms capable of dealing with concept drifts in the forecasting of PM_{10} concentration.

5 Experimental Methodology and Results

Having defined the algorithms capable of handling concept drifts in the last chapter, this chapter describes the methodology adopted to evaluate such algorithms in different scenarios. First, real-world and artificial datasets with known dynamic behaviors were applied to the algorithms. Then, Particulate Matter datasets were applied and the results compared in order to try to verify of possible dynamic behaviors in PM data. The datasets used in the simulations and the preprocessing methods are described in this chapter, together with the experimental configurations, the parameter setup and the experimental results.

5.1 Data description

Two artificial and six real-world datasets were employed in the experiments in order to study the performance of the algorithms in different scenarios. The datasets are listed in Table 8.

Table 8. Specifications of datasets used in the experiments

Dataset	Type	Number of Attributes	Outliers	Missing Values	Number of Samples
Friedman	Artificial	6	88	-	1992
Hyperplane 500	Artificial	6	30	-	500
Hyperplane 1000	Artificial	6	32	-	1000
Hyperplane 2000	Artificial	6	107	-	2000
Hyperplane 3000	Artificial	6	146	-	3000
Debutanizer Column	Real-world	7	113	-	2394
Sulfur Recovery Unit - H20 (SRU1)	Real-world	5	522	-	10081
Sulfur Recovery Unit - S2O (SRU2)	Real-world	5	497	-	10081
PM_{10} Concentration - Campinas	Real-world	6	748	1049	35064
PM_{10} Concentration - Jundiaí	Real-world	6	941	1139	35064
PM_{10} Concentration - São Caetano do Sul	Real-world	6	1218	570	35064

5.1.1 Artificial datasets

Artificial datasets were used to simulate the problems and scenarios that the algorithms are expected to deal with. In this work, the hyperplane dataset, which has been used as benchmark for concept drift algorithms (Kolter & Maloof, 2005) together with the Friedman's function (Ikonomovska, 2012) were adopted.

The hyperplane dataset involves noise, gradual drifts and non-recurring drifts. It was created by Kolter (2005) to evaluate the AddExp algorithm for regression. Feature vectors consist of 10 input variables $x \in [0,1]$ with uniform distribution, the output variable $y \in [0,1]$ and a number of samples T. Four target concepts $[C_1; C_2; C_3; C_4]$ are introduced, each one lasting $T/4$ samples. The output for each concept y_t is given by:

- $concept\ C_1: y_t = (x_1 + x_2 + x_3)/3\,,\ \text{for } t = 1, \dots, \frac{T}{4};$ (19)
- $concept\ C_2: y_t = (x_2 + x_3 + x_4)/3\,,\ \text{for } t = \left(\frac{T}{4} + 1\right), \dots, \frac{T}{2};$ (20)
- $concept\ C_3: y_t = (x_4 + x_5 + x_6)/3\,,\ \text{for } t = \left(\frac{T}{2} + 1\right), \dots, \frac{3T}{4};$ (21)
- $concept\ C_4: y_t = (x_7 + x_8 + x_9)/3\,,\ \text{for } t = \left(\frac{3T}{4} + 1\right), \dots, T;$ (22)

where T is the size of the dataset. As T varies in each experiment with $T \in [500, 1000, 2000, 3000\ ]$, four datasets were obtained, as shown in Table 8. The smaller the value of T, the larger is the rate of concept drift.

Friedmans' dataset is generated from the Friedmans' function. It contains 5 continuous features $x \in [0,1]$ independently distributed according to a uniform distribution. The target value is given by Eq. (23):

$$y = 10 * \sin(\pi * x_1 * x_2) + 20 * (x_3 - 0.5)^2 + 10 * x_4 + 5 * x_5 + \sigma(0,1) \qquad (23)$$

where $\sigma(0,1)$ is a random number generated from a normal distribution with mean 0 and variance 1.

5.1.2 Real-world datasets

Six real-world datasets were considered in the simulations; three corresponding to concentration values of PM_{10} from different cities and three associated with industrial processes.

The Sulfur Recovery Unit (SRU) and Debutanizer Column datasets correspond to industrial processes. In the case of the SRU dataset, two outputs where considered; the H_2O concentration (output 1) and S_2O concentration (output 2), referred as SRU1 and SRU2 respectively. For the Debutanizer Column dataset, the output corresponds to the

butane concentration. Since most of the industrial processes tend to present some kind of time-varying behavior, those datasets are suitable to evaluate the proposed algorithms (Gomes Soares & Araujo, 2015). The detailed information about these industrial derived datasets can be found in (Fortuna et al., 2007).

5.1.2.1Particulate matter datasets

Particulate Matter datasets are composed of samples of PM_{10} concentration, hourly collected from 01-Jan-2012 to 01-Jan-2015 in the cities of Jundiaí, São Caetano do Sul and Campinas (all in the State of São Paulo, Brazil) by São Paulo's Environmental Agency (CETESB). From the cities monitored by CETESB, these three presented the lowest number of missing values, so they were adopted in this study.

The city of Jundiaí has a population of approximately 370,000 inhabitants (IBGE, 2017),

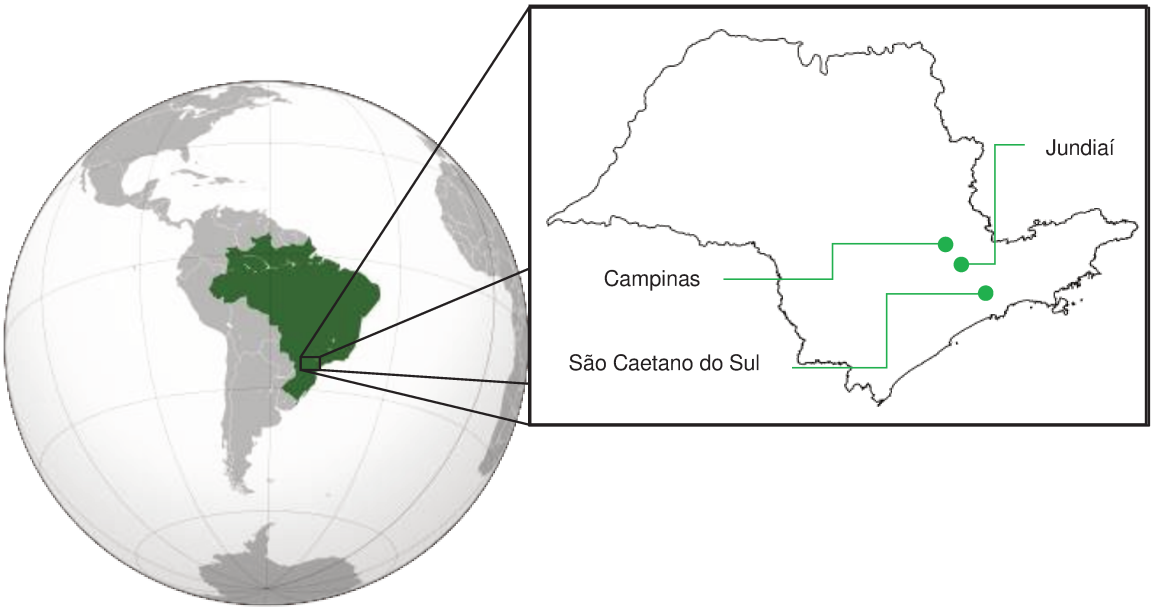

Figure 5. Geographical locations of the cities of the State of São Paulo whose data were employed in the experiments

an area of 431.21 km^2 and a population density of 930 inhabitants/km^2. São Caetano do Sul, which is part of the Metropolitan Region of São Paulo, has a population of approximately 149,000 inhabitants (IBGE, 2017), an area of 15.33 km^2 and a density of 10,000 inhabitants/km^2. Finally, the city of Campinas has a population of approximately 1,080,000 inhabitants (IBGE, 2017), 795,667 km^2 of area and a population density of 1,358 inhabitants/km^2. Geographical locations of these cities are presented in Figure 5.

The cities employed in the experiments are located in the southeast region of Brazil, in the State of São Paulo, which has an area of approximately 249,000 km^2. Among the federal states of Brazil, the state of São Paulo has the largest territorial occupation, the largest population, about 44.7 millions inhabitants (IBGE, 2017), the greatest economic development (strong agricultural – standing out the sugar-cane-alcohol –, industrial and services areas) and the largest automobile fleet. As a result, the state presents major alterations in the air quality, mainly in the metropolitan areas of São Paulo and Campinas (CETESB 2016). These factors make the state of São Paulo interesting to study the behavior of air pollutants like PM, which is the aim of this work.

The hourly collected samples of PM_{10} concentration are provided by CETESB's website and were sequentially stored in an ordered vector of samples, where the PM_{10} concentration value for the current hour corresponds to s^i, the value for the last hour to s^{i-1} and so on. The samples were organized in such a way that the last five samples of the data stream, $s^{i-4}, s^{i-3}, s^{i-2}, s^{i-1}$ and s^i, were used as the inputs x_t of each forecasting model, which were configured to predict the concentration value y_t for the next hour. Hence, the t –th input-output instance $(x_t, y_t) \in S$ is given by:

$$x_t = \left[s^{i-4}, s^{i-3}, s^{i-2}, s^{i-1}, s^i\right] \tag{24}$$

$$y_t = s^{i+1} \tag{25}$$

5.2 Data preprocessing

Missing values were replaced by the average between the two nearest data points. Outlier analysis was performed using the *Hampel filter* (Pearson, Neuvo, Astola, & Gabbouj, 2015): for each sample s^i, this method computes the median m^i of a window composed of the surrounding k samples ($k/2$ per side), including the sample s^i. This method also

estimates the standard deviation S_k of s^i with respect to its window median, through the

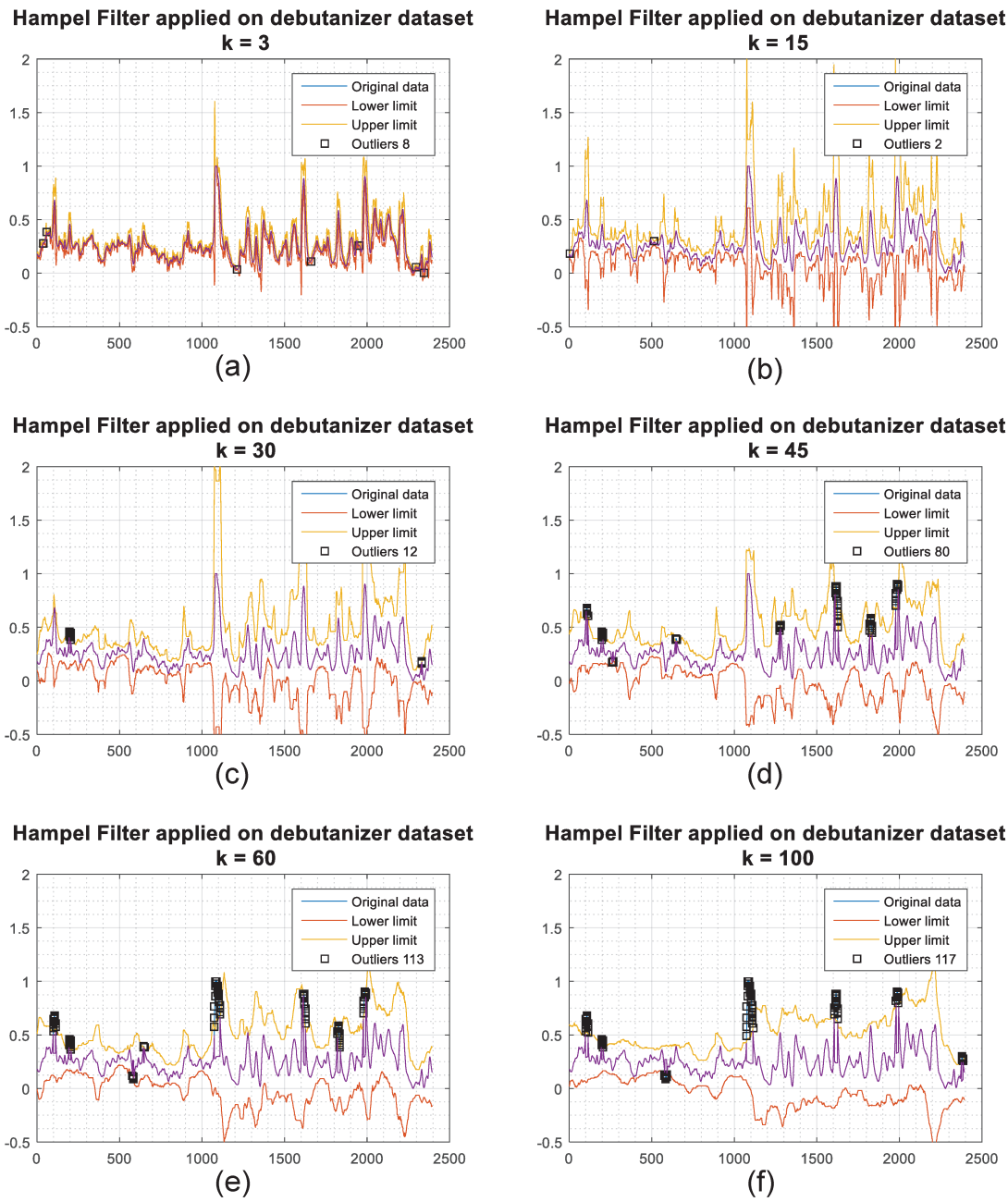

Figure 6. Hampel filter applied to Debutanizer Column and different values of k

median absolute deviation. If s^i differs from the window median by more than three standard deviations, it is replaced with the median. Figure 6 shows an example of the Hampel filter applied to the Debutanizer Column dataset.

From Figure 6, it can be seen that, when k increases, more samples are considered in the window, thus, the window median and standard deviation increase as well, changing the upper and lower limits. This process was applied to every dataset, until a certain value of k is obtained, where the number of identified outliers no further increases with k. In this work, the number of samples marked as outliers was the criterion for choosing the value of k: the larger the number of outliers identified and normalized, the more appropriate the value of k.

Historical reports of PM monitored in the cities of Campinas, Jundiaí and São Caetano do Sul, provided by CETESB (CETESB, 2017), were also considered in the outlier analysis. The air quality trends observed during the last years, presented in the reports, allowed the identification of the maximum values of PM concentration for each city. Those values were used to define the limits of the Hampel filter, considering the following rule: the product of the window median by three times the standard deviation of each sample of the dataset cannot be larger than the maximum value of PM_{10} concentration PM_{max} reported by CETESB for each city. Finally, the output of the Hampel filter is calculated as follows:

$$y_t = \begin{cases} s^i, & \textbf{if } \left|s^i - m^i\right| < 3S^i \\ m^i, & otherwise \end{cases} \tag{26}$$

where $3S^i \leq \mathrm{PM}_{max} \;\; for\; i = 0, \ldots T$ and $S^i = \mu * \mathrm{med}(\left|s^{i-k} - m^i\right|, \ldots, \left|s^{i+k} - m^i\right|)$ where $\mu = \frac{1}{\sqrt{2}\,\mathrm{erfc}^{-1}\,1/2} \approx 1{,}4826$

5.3 Experimental setup

In order to evaluate the proposed algorithms in different scenarios and to identify the most suitable approach to forecast PM_{10} concentrations, the experiments were divided into two parts. In the first part, all algorithms were evaluated in real and artificial scenarios with dynamic behaviors (SRU, debutanizer, Friedman and hyperplane datasets), as described in the last section.

The second part aimed to evaluate how the best algorithms, identified in the first part of the experiments, behave when applied to forecast PM_{10} concentration. The objective of this methodology is to obtain a more in-depth evaluation of the proposed algorithms,

considering multiple scenarios, and compare the results of the algorithms when applied to datasets with known dynamic behaviors and the results when applied to PM datasets, in order to identify if possible ongoing changes are occurring in underlying distributions of PM data.

Since all the algorithms evaluated here are based on OS-ELMs, they share some parameters, such as the activation functions, the number of hidden neurons L (hidden nodes), the sliding window size and the number of samples used for the initial training phase. The Logistic activation function, defined in Eq. (27), was used in all OS-ELMs trained here.

$$g(a, b, x) = \frac{1}{1 + e^{-(ax+b)}} \tag{27}$$

where a, b and x correspond to the weight, bias and input values, respectively.

Input and output attributes were normalized in [0,1] and the weights and biases of the OS-ELMs were randomly chosen from the range [-1,1]. The methodology to select the best configuration of hidden neurons L and sliding window size m (as referred in the algorithms in Chapter 2) for each experiment is described in the next subsections.

5.4 Sensitivity Analysis

To evaluate the impact of the parameters in the OS-ELM models and define the boundaries of the inputs, two experiments were performed. This study allows to identify patterns in datasets to setup the prediction models properly.

In the first experiment, 30% of each dataset was used to find the best setup for each algorithm. The first parameters explored were L (number of neurons) and m (sliding window size) for the OS-ELM, which is the base algorithm of all ensembles. To do so, the model was initially trained with a block of $N_1 = 120$ samples, and each experiment was repeated 10 times. Here, L was evaluated in the interval [1,25] and $m \in \{3, 5, 10, 20, 30, 60, 90\}$, as proposed in (Soares & Araújo, 2015). The performance was

evaluated through the Mean Squared Error (MSE). Figure 7 presents the results of the OS-ELM parameter exploration.

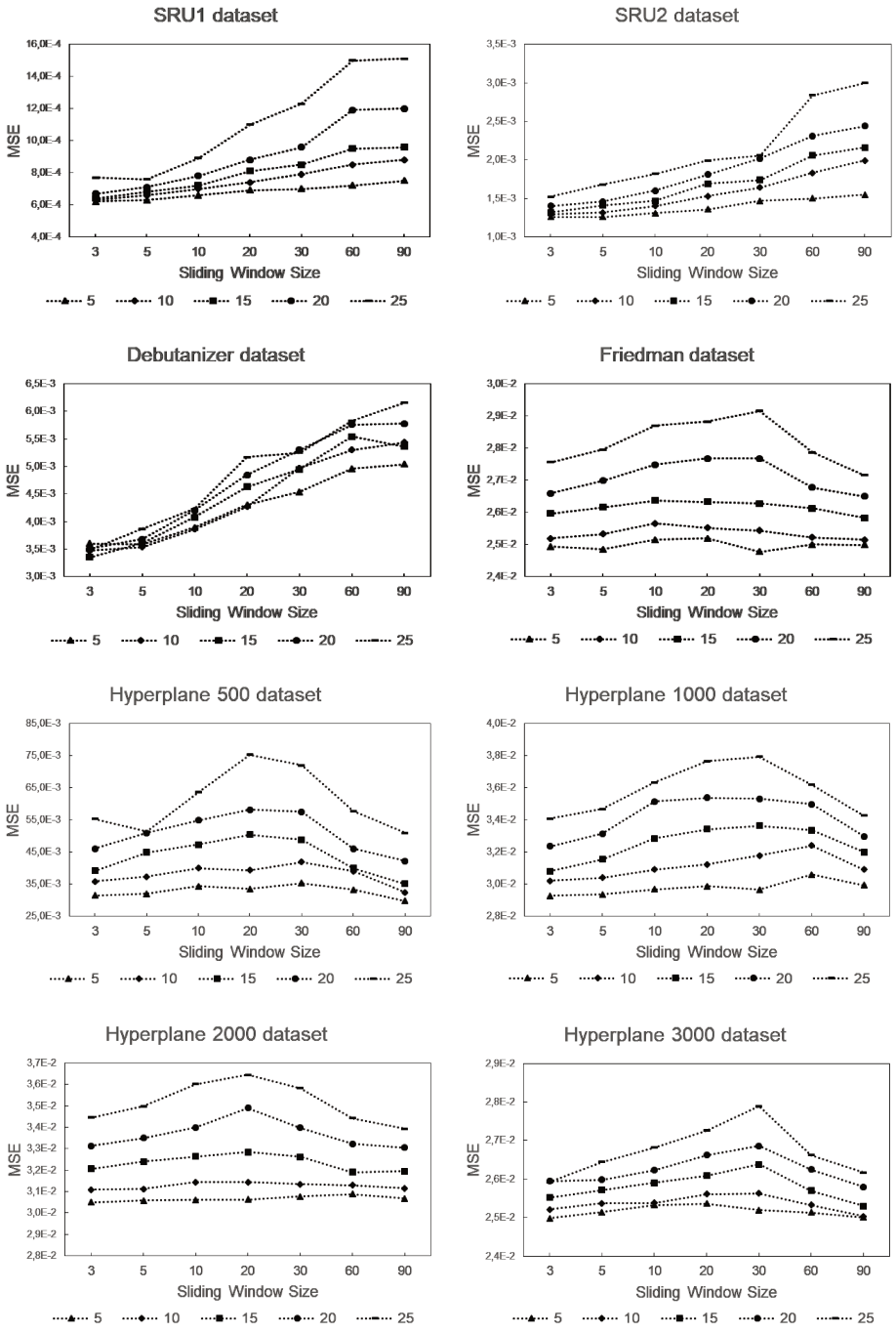

Figure 7. Errors of the OS-ELM algorithm on artificial and real-world datasets using different values of sliding window and hidden neurons

Results in Figure 7 allow the identification of the most suitable parameters L and m for the OS-ELMs. In general, the increase of L led to higher errors in all cases, but especially for the SRU1 and SRU2 datasets, where the highest increment in the MSE was noticed. Regarding the sliding window size m, OS-ELM use it to update the model, and smaller windows offered better performances in almost all the cases. For example, in the Debutanizer, SRU1 and SRU1 datasets the use of larger windows increased the error, especially with larger values of L. This is often associated with the presence of higher rates of concept drift, which leads small windows to present better performances. In general, considering the real-world datasets, it was observed that, when the size of m increases, the MSE increases as well. This behavior was not observed in the artificial datasets, where values of $m < 5$ and $m > 60$ presented the best performances.

In order to make the evaluation more equitable, the number of hidden neurons $L = 5$ and the sliding window size $m = 3$ were defined for all scenarios, since they presented the best performances according to MSE in all datasets.

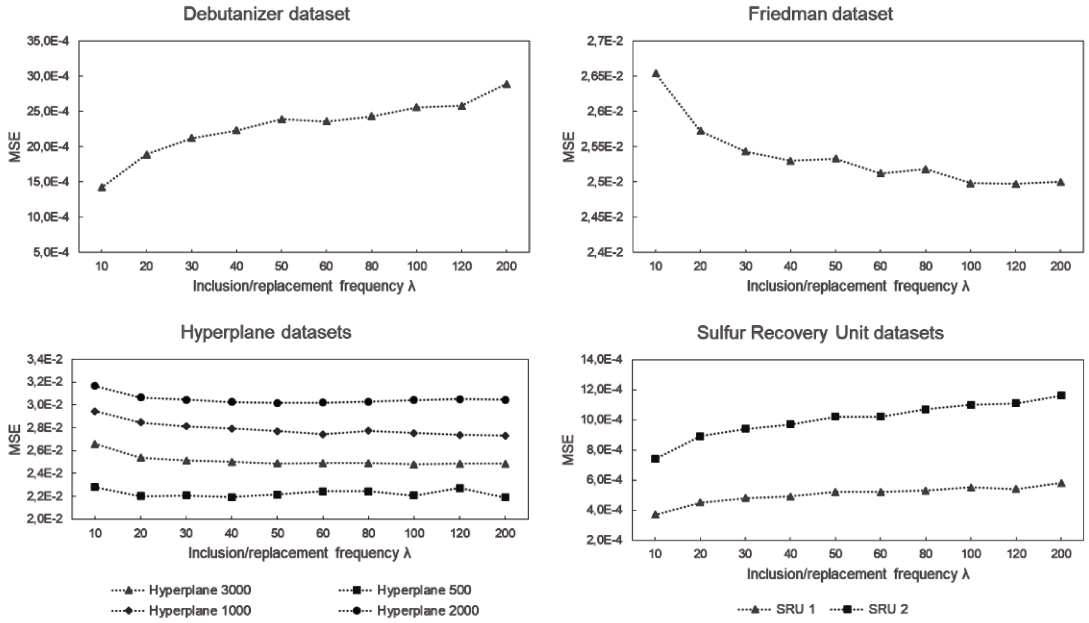

Figure 8. Errors of the EOS Algorithm varying the inclusion/replacement frequency λ on artificial and real-world datasets

Based on these parameters, the next step is to set up the EOS-based algorithms (EOS and EOS-rank). The main parameter of this approach is the inclusion/replacement

frequency λ. As described in Chapter 2, λ determines the frequency with which new components are incorporated into the ensemble or replaced, according to the size of the ensemble. This parameter also determines the number of samples used to train the new component, as a block composed of the last λ samples is used to train the new component to be incorporated into the ensemble. Results of the evaluation of different values of λ are presented in Figure 8. Finally, values of $\lambda = 10$ were assigned for real-world datasets (Debutanizer, SRU1 and SRU2), $\lambda = 100$ for the Friedman and Hyperplane 3000 datasets and $\lambda = [40, 120, 50]$ for the Hyperplane 500, Hyperplane 1000 and Hyperplane 2000 respectively.

The inclusion and exclusion of models can be an important factor that influence the adaptation of the ensemble, thus affecting its prediction performance (Soares & Araújo, 2015). An ideal high frequency inclusion of new components into the ensemble may indicate that the environment is changing rapidly, so the new components trained with the most recent samples represent the current state of the system to be predicted. Results presented in figures 7 and 8 seem to corroborate this behavior, as real-world datasets performed better with small window sizes and larger inclusion/replacement frequencies. In contrast, artificial datasets do not seem to be affected by this factor, nevertheless, it can be observed that the MSE tend to decrease when λ is increased.

Regarding the DOER-based approaches (DOER and DOER-rank) two parameters where considered; the factor of inclusion of a new model α and the sliding window size m. Notice that DOER sliding window is used to train the new models to be incorporated into the ensemble, and not to update the current components, as EOS-based approaches do. Here, $\alpha = 0.04$ was set for all the scenarios as proposed in (Soares & Araújo, 2015), since no further improvement was observed when α increased or decreased. On the order hand, DOER algorithms used the same values of the frequency of inclusion/replacement of components λ of the EOS algorithms, that is, $m = \lambda$ for each corresponding dataset. This makes the comparisons between EOS and DOER fairer, since in both cases λ and m define the size of the batch of samples used to train the new component to be added to the ensemble.

The same methodology adopted for the real-world and artificial datasets with known dynamic behaviors was also applied for PM datasets Here, just the 20% of PM_{10} concentration data were used to explore the best configuration of the algorithms, considering that PM datasets are larger than the datasets used for real-world and artificial datasets with known dynamic behaviors The evaluation of the sliding window size m for different values of L is presented in Figure 9 for the OS-ELM algorithm, the base algorithm for all approaches studied here.

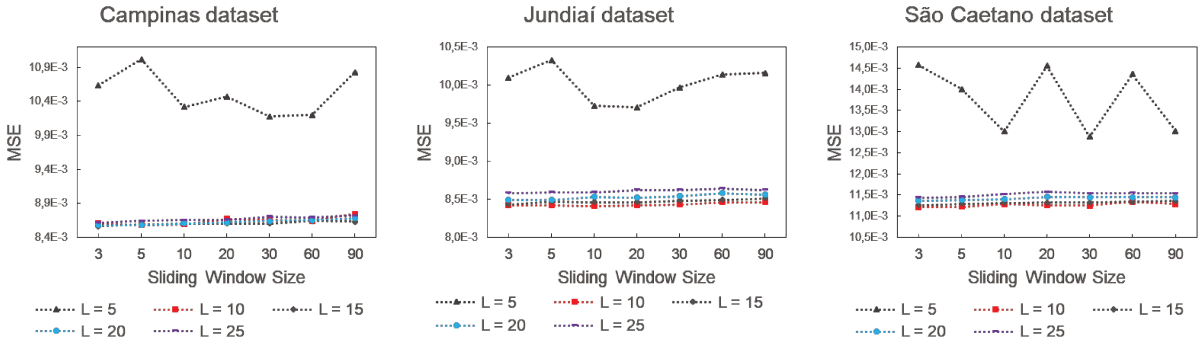

Figure 9. Errors of the OS-ELM algorithm on all particulate matter datasets using different values of sliding window and hidden neurons

Results in Figure 9 show a slightly better performance when using 10 hidden neurons and the worst performance was observed when $L = 5$ for all cases. Therefore, $L = 10$, was defined as the number of hidden neurons for all scenarios for PM datasets. Regarding the window size m, since no reasonable improvement or deterioration of the accuracy was observed when varying the size of m and, considering that smaller windows require less processing time, a window of size $m = 3$ was set for all cases.

Moreover, the evaluation of λ for the EOS algorithm is presented in Figure 10. Here, it can be observed that the performance of the algorithm improves when λ increases until $\lambda = 60$, where the lowest MSE is presented. As a result, EOS-based algorithms were set with $\lambda = 60$ for all datasets and, following the methodology applied in in the first part of the sensitivity analysis, the sliding window m of DOER-based approaches was set with the same value $m = \lambda$.

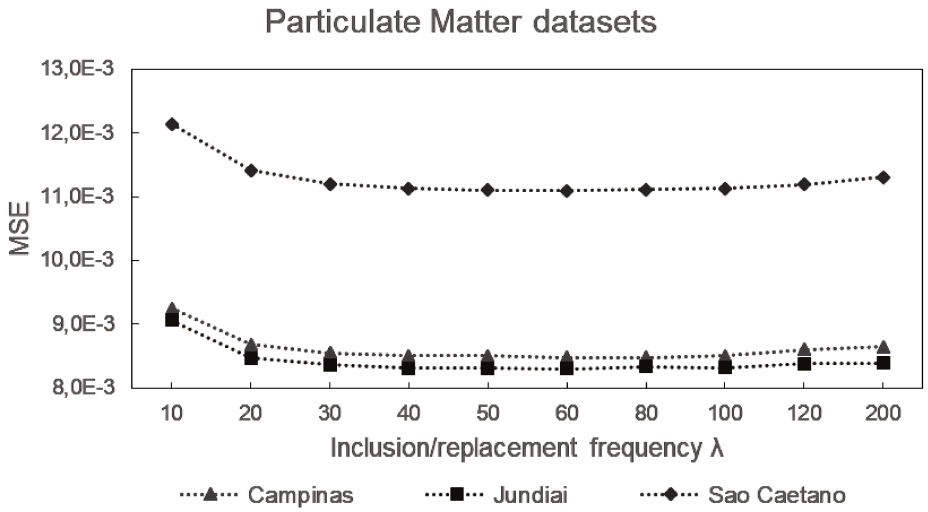

Figure 10. Errors of the EOS Algorithm varying the inclusion/replacement frequency λ on particulate matter datasets

The sensitive analyses allowed the identification of patters presented in datasets with dynamic behavior. The size of the sliding window is one of them, small sliding windows led to lower errors in scenarios changing rapidly, since the most recent samples contain the information of the current concept. On the other hand, it was identified that high updating frequencies in ensembles enables faster adaptation to changing environments.

5.5 Experimental Results

Results of the experiments are presented through the average and standard deviation of the Mean-Squared Errors (MSEs), together with the computational time required for processing the whole dataset, MSE box plots and online accumulative MSE plots for the experiments. Each model was initially trained with the first $N_1 = 120$ samples of every dataset, considering that all the approaches evaluated here are online sequential learning approaches, so they are theoretically capable of operating in scenarios where it is impossible to have all the training data available before the learning process. The remaining samples were used to simulate a data stream.

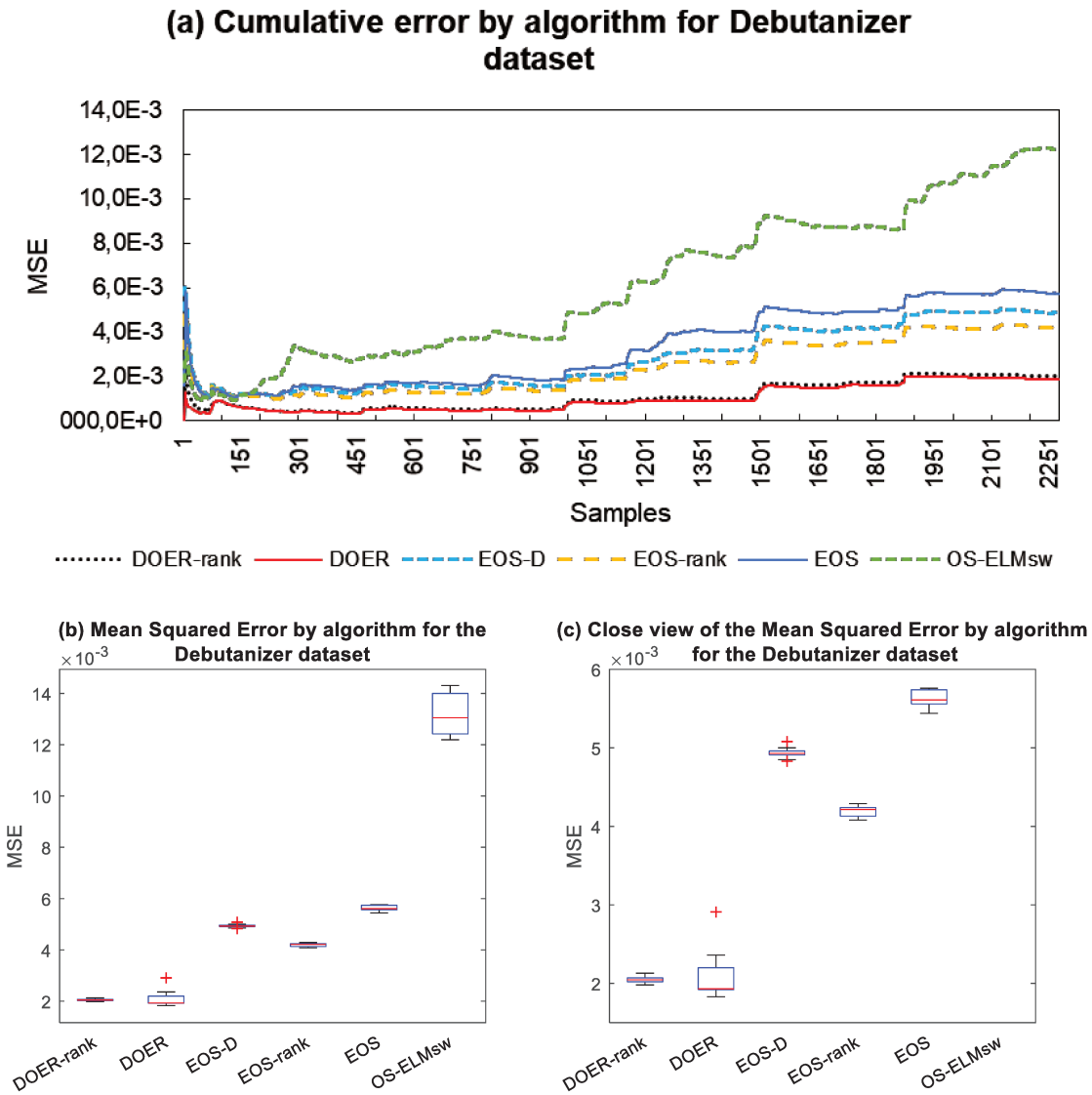

Figure 11. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Debutanizer dataset

The EOS and DOER ensembles were formed by a maximum of 10 components, since no significant improvement in the ensemble's performance for a larger number of models was observed. On the other hand, the EOS-rank and DOER-rank threshold was set to 20 components, allowing models to live more time into the ensemble before being replaced. The accuracy was evaluated through the mean and standard deviation of the Mean Square Error (MSE) between real and predicted outputs. The algorithms were implemented in Python, the experiments were repeated for 10 trials and the tests were conducted in the Scientific Python Development Environment – Spyder 2.2.3, running on a PC with Intel Core i5-4210U 1.70 GHz CPU, 6 GB of RAM and Windows 10.

5.5.1 Experimental results for the real-world and artificial datasets with known dynamic behaviors

Results for of the real-world and artificial datasets with known dynamic behaviors are summarized in tables 9 and 10. The best results according to the MSE, are highlighted in bold. It can be seen, from tables 9 and 10, that DOER-based algorithms were superior for all real-world datasets (Debutanizer, SRU1 and SRU2) while EOS-based algorithms presented the best results for artificial datasets (Friedman and Hyperplane).

Table 9. Performance comparison of the proposed approaches for the Debutanizer, Friedman, SRU1 and SRU2 datasets

Dataset	**Algorithm**	**Results**	
		Mean Squared Error	**Time** $[s]$
Debutanizer	DOER	0.00208 ± 0.00032	5.904 ± 0.15140
	DOER-rank	**0.00205 ± 0.00005**	10.97 ± 0.19840
	EOS	0.00563 ± 0.00010	5.856 ± 0.17130
	EOS-rank	0.00420 ± 0.00007	8.444 ± 0.10758
	EOS-D	0.00494 ± 0,00007	4.963 ± 0.07820
	OS-ELMsw	0.01317 ± 0.00076	0.402 ± 0.01355
Friedman	DOER	0.02751 ± 0.00006	8.325 ± 0.13097
	DOER-rank	0.02744 ± 0.00006	13.60 ± 1.04498
	EOS	0.02687 ± 0.00005	3.469 ± 0.05503
	EOS-rank	**0.02682 ± 0.00003**	6.557 ± 0.11528
	EOS-D	0.02688 ± 0.00003	4.493 ± 0.07195
	OS-ELMsw	0.02694 ± 0.00009	0.349 ± 0.01394
SRU1	DOER	0.00027 ± 0.00001	26.12 ± 1.43020
	DOER-rank	**0.00027 ± 0.00000**	47.25 ± 0.29920
	EOS	0.00042 ± 0.00000	25.82 ± 0.19807
	EOS-rank	0.00037 ± 0.00000	37.17 ± 0.38034
	EOS-D	0.00041 ± 0.00000	21.19 ± 0.19076
	OS-ELMsw	0.00067 ± 0.00001	1.531 ± 0.01771
SRU2	**DOER**	**0.00057 ± 0.00001**	25.41 ± 0.43370
	DOER-rank	0.00061 ± 0.00000	47.20 ± 0.18500
	EOS	0.00103 ± 0.00000	25.91 ± 0.37625
	EOS-rank	0.00089 ± 0.00001	37.07 ± 0.14109
	EOS-D	0.00098 ± 0.00001	21.37 ± 0.14662
	OS-ELMsw	0.00165 ± 0.00001	1.537 ± 0.03829

Ranking approaches implemented in DOER and EOS algorithms (DOER-rank and EOS-rank) presented higher accuracies in most of the cases than their original approaches (DOER and EOS). EOS only performed better than EOS-rank for the Hyperplane 2000 and Hyperplane 1000 datasets. On the other hand, DOER was superior to DOER-rank for Friedman and SRU2 datasets. Results also show that this method can reduce the standard deviation of the original approaches.

Table 10. Performance comparison of the proposed approaches for the Hyperplane datasets.

Dataset	Algorithm	Results	
		Mean Squared Error	**Time** $[s]$
Hyperplane 500	DOER	0.03165 ± 0.00018	1.304 ± 0.02400
	DOER-rank	0.03124 ± 0.00015	2.266 ± 0.04660
	EOS	0.02967 ± 0.00016	0.410 ± 0.01708
	EOS-rank	**0.02958 ± 0.00012**	1.149 ± 0.03834
	EOS-D	0.02973 ± 0.00009	0.948 ± 0.03100
	OS-ELMsw	0.02976 ± 0.00037	0.074 ± 0.00774
Hyperplane 1000	DOER	0.02766 ± 0.00004	4.147 ± 0.09250
	DOER-rank	0.02764 ± 0.00011	6.457 ± 0.00007
	EOS	0.02725 ± 0.00016	0.733 ± 0.02605
	EOS-rank	0.02727 ± 0.00005	2.348 ± 0.03305
	EOS-D	**0.02724 ± 0.00008**	2.111 ± 0.08495
	OS-ELMsw	0.02750 ± 0.00045	0.167 ± 0.01069
Hyperplane 2000	DOER	0.03113 ± 0.00005	6.833 ± 0.10960
	DOER-rank	0.03102 ± 0.00008	11.76 ± 0.16910
	EOS	**0.03002 ± 0.00004**	4.499 ± 0.08797
	EOS-rank	0.03006 ± 0.00004	7.392 ± 0.14369
	EOS-D	0.03005 ± 0.00004	4.578 ± 0.00004
	OS-ELMsw	0.03016 ± 0.00023	0.342 ± 0.01719
Hyperplane 3000	DOER	0.02664 ± 0.00005	12.93 ± 0.19080
	DOER-rank	0.02654 ± 0.00005	20.40 ± 0.31410
	EOS	0.02605 ± 0.00002	6.242 ± 0.07685
	EOS-rank	**0.02602 ± 0.00002**	10.59 ± 0.12375
	EOS-D	**0.02602 ± 0.00002**	6.825 ± 0.07331
	OS-ELMsw	0.02627 ± 0.00045	0.489 ± 0.01731

Although ranking approaches presented the more accurate results in almost all scenarios, when compared with the original ensemble algorithms, they required more processing time in all cases. DOER and EOS-based algorithms that implement ranking methods

(DOER-rank and EOS-rank) required the highest computational times, followed by DOER, EOS, EOS-D and, finally, the OS-ELMsw.

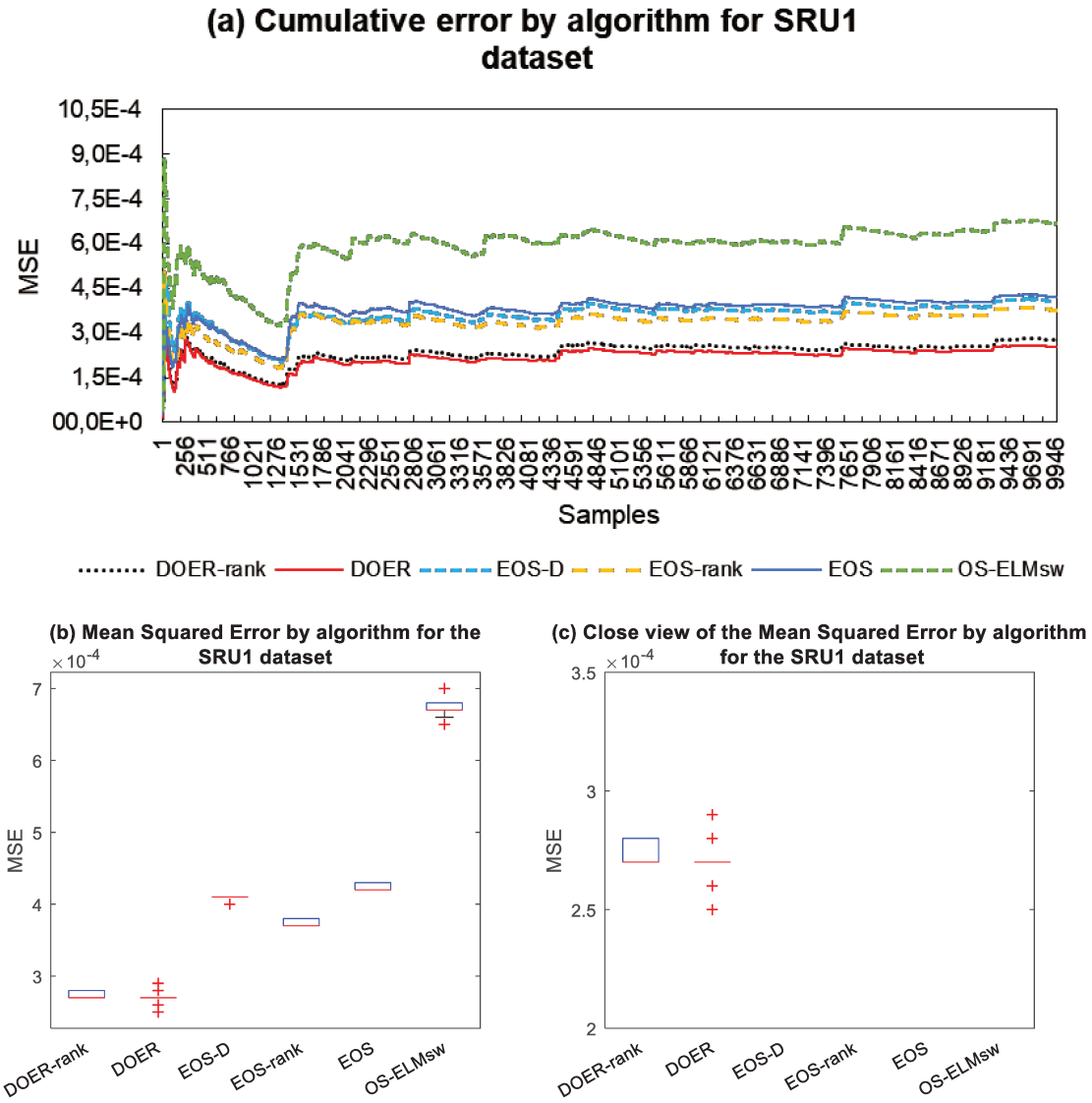

Figure 12. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for SRU1 dataset

This indicates that ranking components significantly increases the computational cost of DOER-rank and EOS-rank. Besides that, DOER spends more time than EOS since DOER includes/removes components at a higher frequency than EOS. This frequency is determined by the factor of inclusion of models α, which tends to update more frequently the components of the ensemble, while EOS works with a fixed frequency, which keeps the ensemble without modifications for more time.

Figures 11 to 13, present the online cumulative error and the box plots of the MSE for the real-world datasets (Debutanizer, SRU1 and SRU2). The online cumulative error is calculated through the squared error of each prediction, divided by the number of predicted samples. Data presented in the online cumulative error plots correspond to a single simulation of the scenarios, selected randomly. This plot enables the analysis of the behavior of the predictors over time.

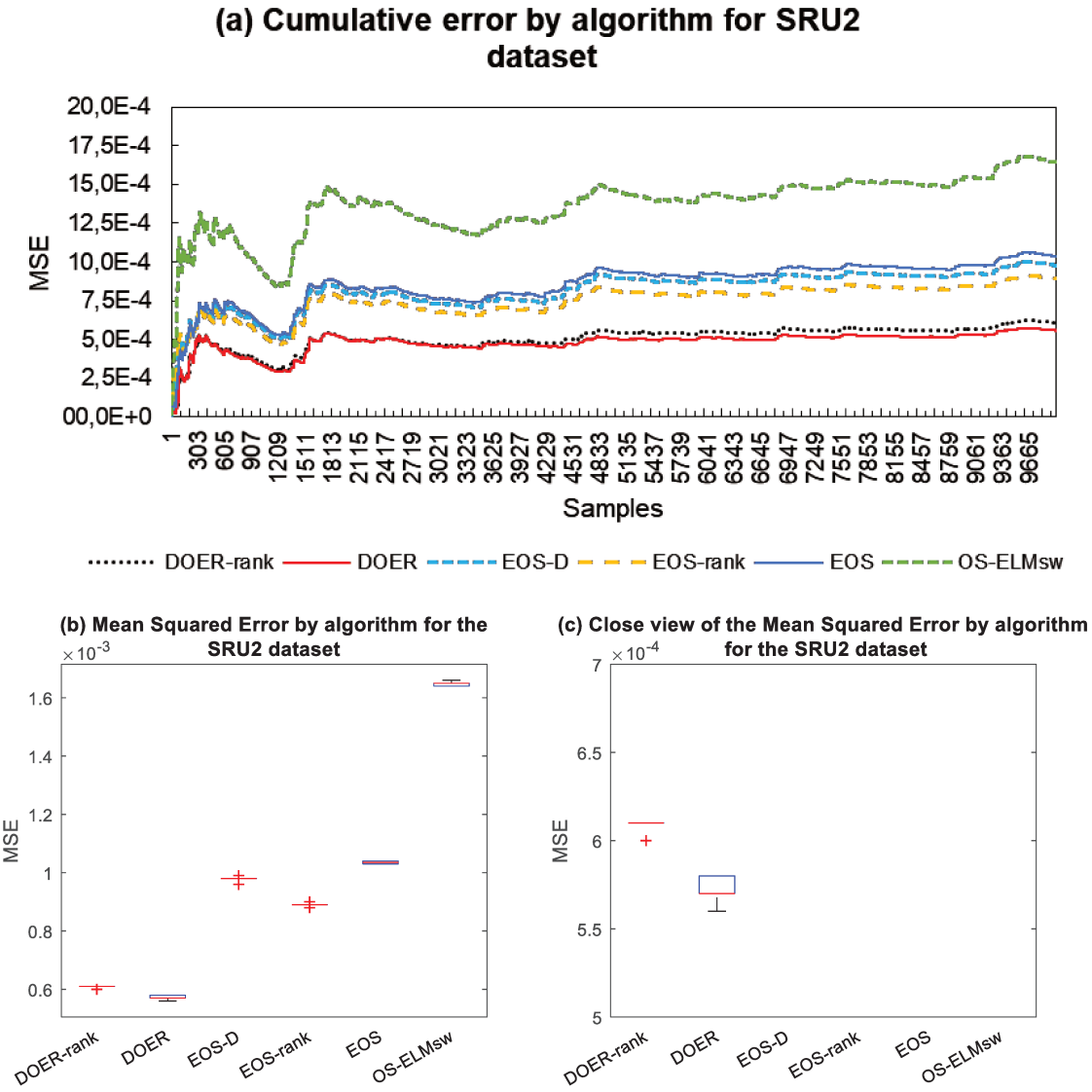

Figure 13. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for SRU2 dataset

The box plots allow the visualization of the results distribution after 10 repetitions of the experiments. The median is marked as a line within the box. The vertical lines outside the

boxes, in the edges (called whiskers), extend to the smallest (Minimum) and the largest (Maximum) observations. Outliers are represented by crosses above and below the whiskers.

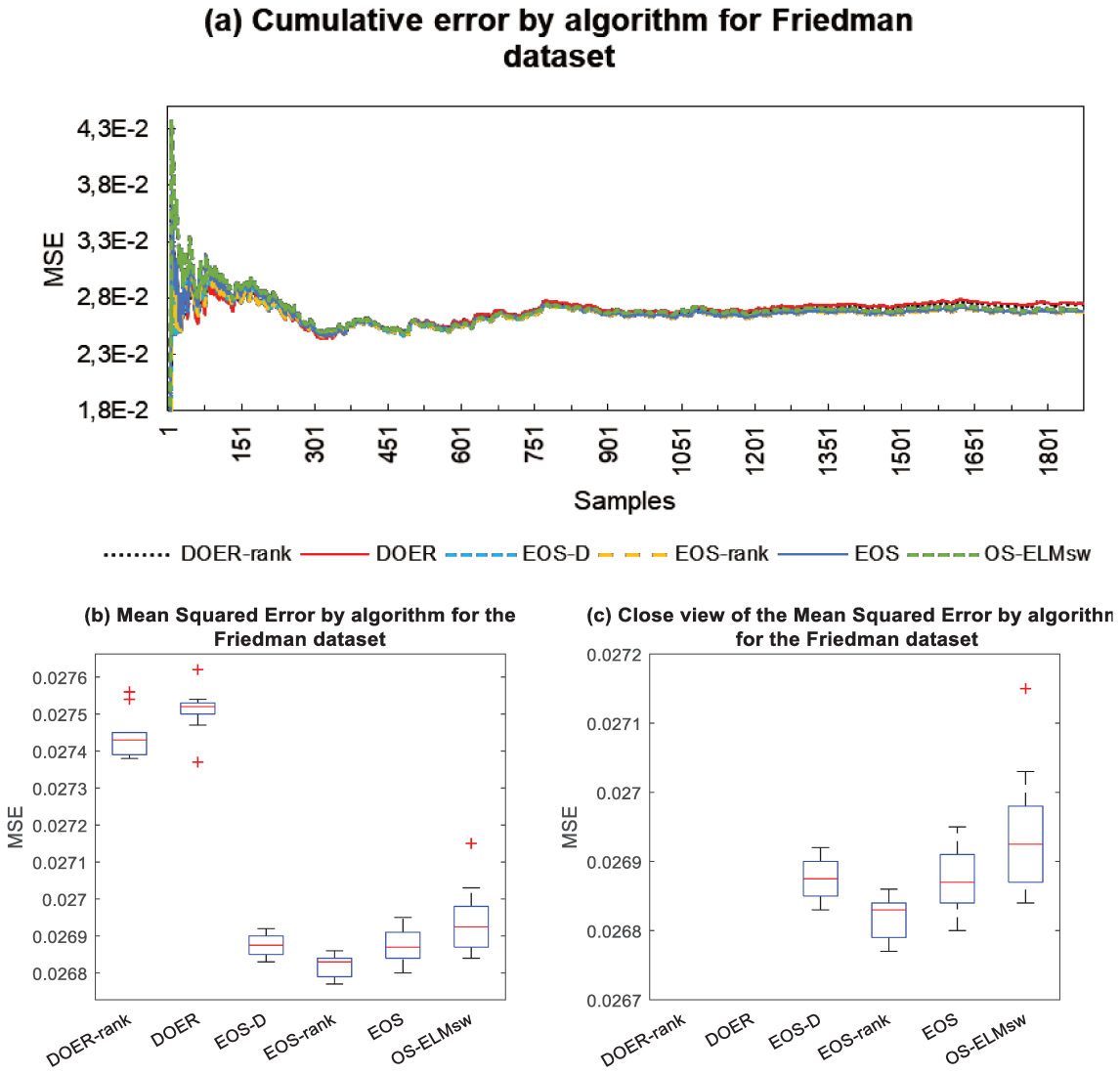

Figure 14. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Friedman dataset

Figures 11 to 13 evidence the superior performance of DOER-based algorithms over the other approaches. As observed in subplot (a) of figures 11 to 13, DOER presents a more stable behavior over time, compared with the other approaches, which present larger errors in small intervals of time, thus, degrading their performance. This may occur by the sudden changes in data distributions appearing at high rate. DOER can adapt faster to those changes, due to its higher frequency of inclusion/removal of models, which keep

only the most up-to-date models in the ensemble. Results reported in figures 11 to 13 are also coherent with the results found in the parameter analysis in figures 7 and 8, where the best performances on real-world datasets were obtained using small sliding windows and high inclusion/removal frequencies.

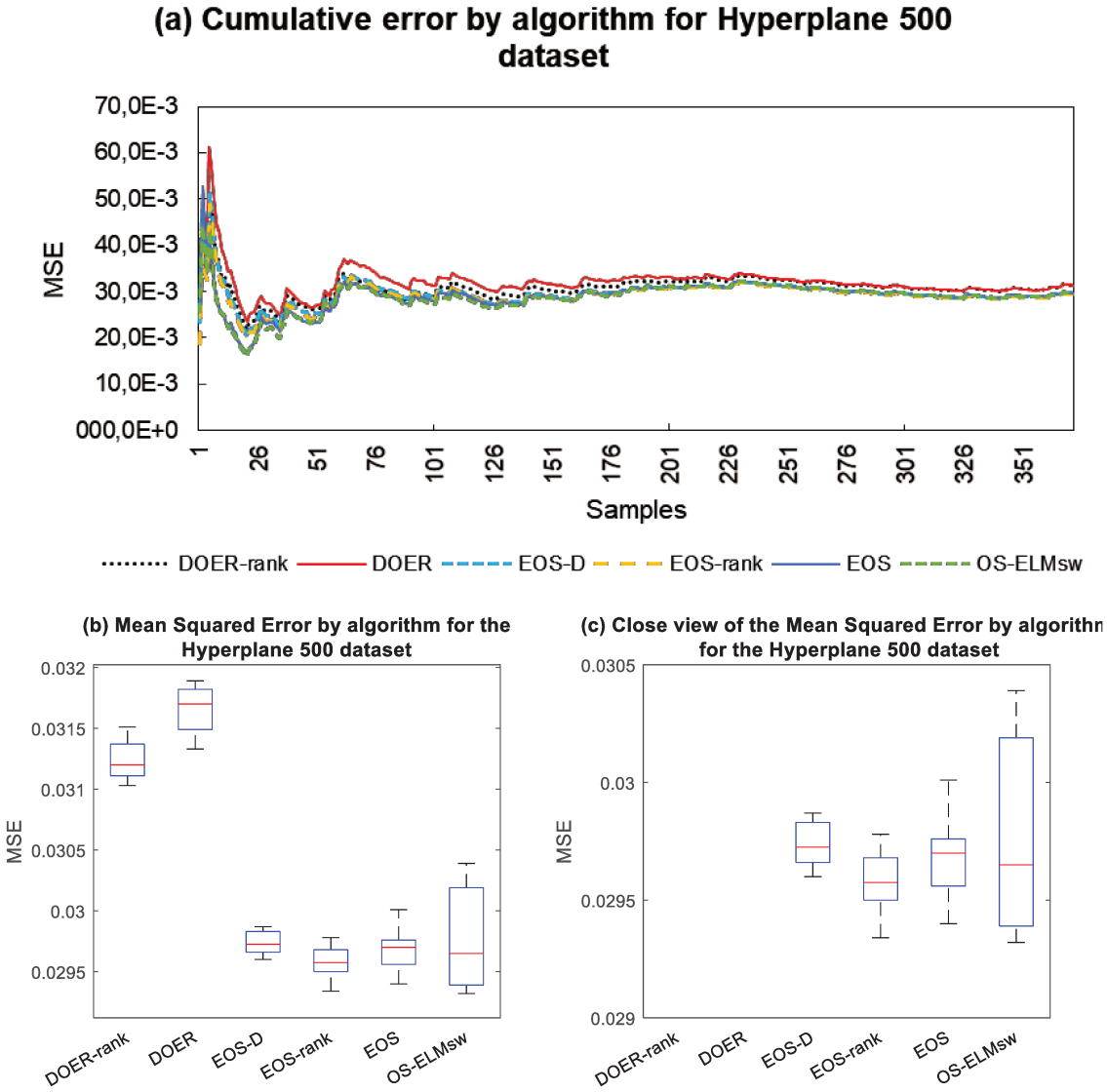

Figure 15. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Hyperplane 500 dataset

This may indicate that real-world datasets present high rates of changes in data distributions, therefore, require faster adaptation capability as the one offered by DOER-based algorithms.

EOS-based algorithms, in contrast, take more time to update the ensemble, even with low values of λ, thus, components trained with past concepts fail to properly predict new samples of the new concept. EOS does not count with mechanisms to adapt the ensemble quickly to the new concept, therefore, its performance is affected by scenarios with high rates of concept drift.

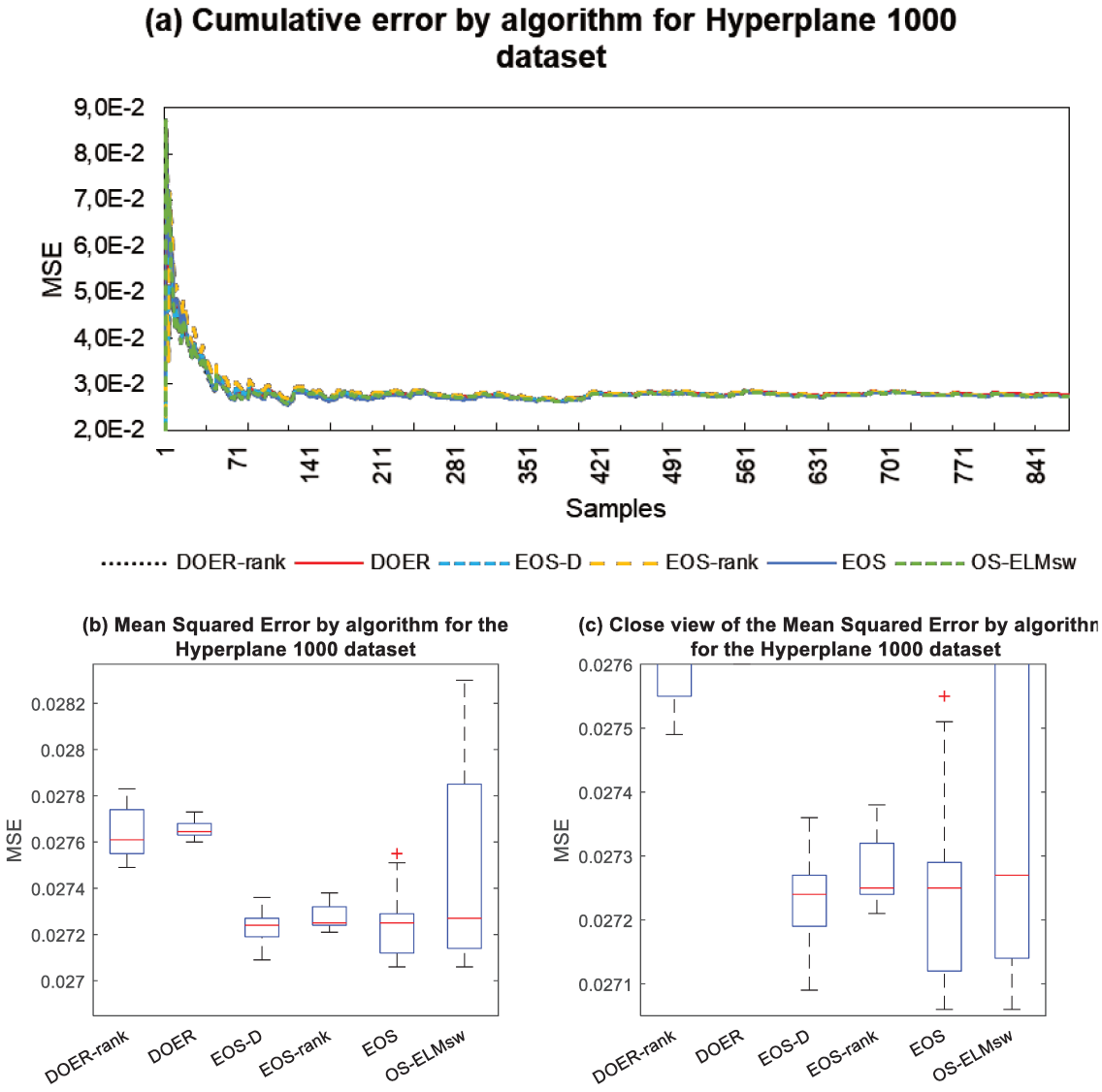

Figure 16. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Hyperplane 1000 dataset

Box plots in subplots (b) and (c) of figures 11 to 13 show the distribution of the results for the 10 repetitions of the experiment for each algorithm. Results show that DOER-rank obtained more stable results than DOER, even when DOER had better performance.

Figures 14 to 18 present the online cumulative error and the box plots of the MSE for the artificial datasets (Friedman and Hyperplane datasets).

As mentioned before, EOS-based algorithms performed better than DOER-based approaches in artificial scenarios. Nevertheless, as can be seen in figures 14 to 18, subplots (a), all the approaches presented a comparable performance in each scenario.

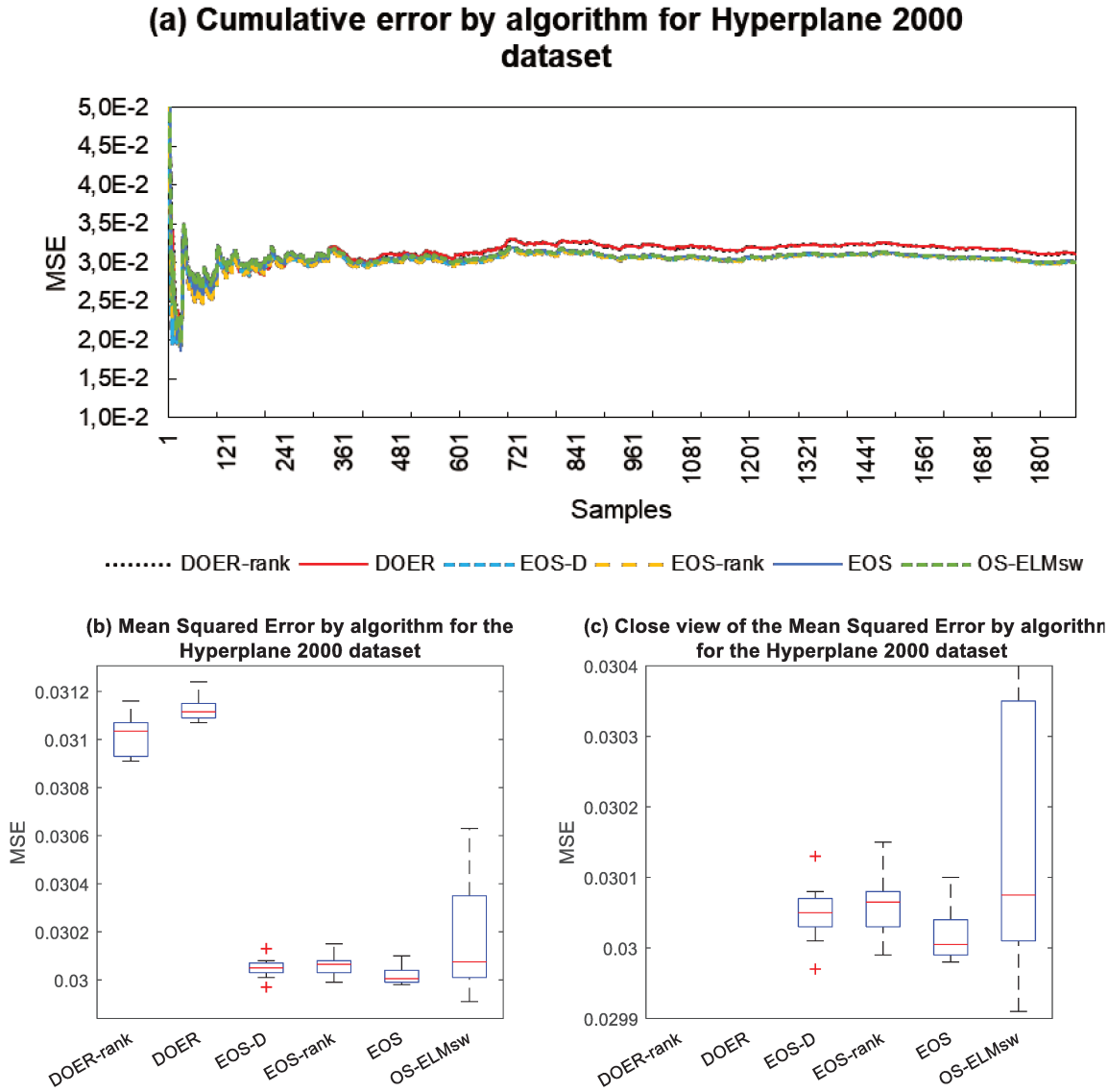

Figure 17. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Hyperplane 2000 dataset

As the artificial datasets studied here present concepts that remain for larger intervals (i.e., Hyperlpane datasets have 4 concepts that appear at each $T/4$ samples), algorithms that use SW tend to perform well with larger windows (Soares & Araújo, 2015). As

observed in Figure 7, for Friedman and Hyperplane datasets the OS-ELMsw obtained its lowest MSE when very small and very large windows were adopted. On the other hand, in Figure 8, for Friedman and Hyperplane datasets, it can be observed that when the frequency of inclusion/replacement of components increase, the MSE decrease.

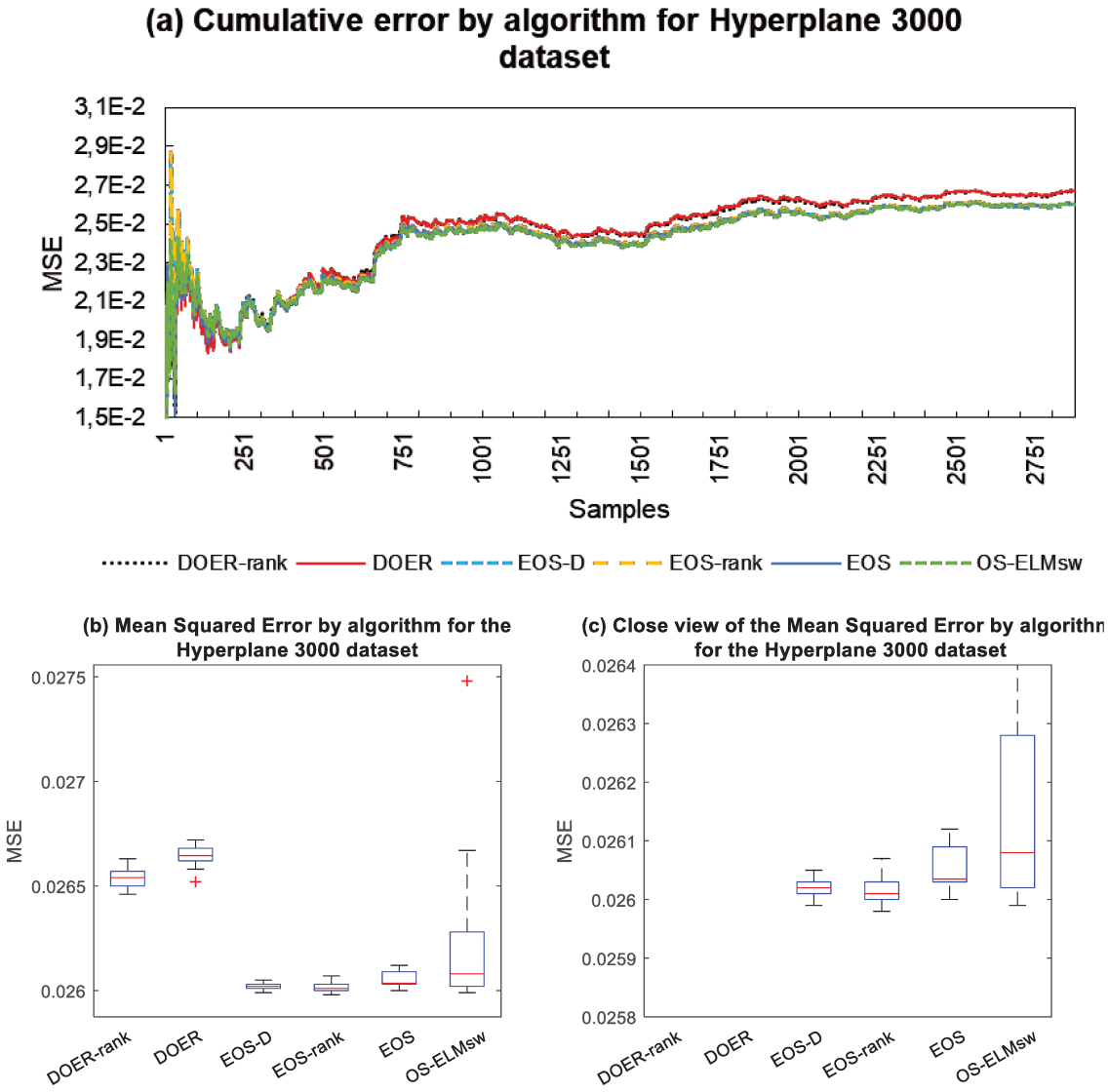

Figure 18. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Hyperplane 3000 dataset

This indicates that the environment is not changing rapidly, so high frequencies of inclusion/replacement of models are not needed since components of the ensemble are trained with data of the current concept and the new concept can be introduced to the ensemble slowly. Otherwise, the replacement of the ensemble's components would have

led to the loss of important information about the current concept. This can explain why DOER-based algorithms perform slightly worse than EOS approaches.

DOER operates with higher frequencies of inclusion/replacement of models, therefore, more modifications to the ensemble are made, compared with EOS-based approaches. Therefore, important information about the current concept may be lost when a component is replaced. This behavior is evidenced when DOER-rank is compared with DOER. DOER-rank outperforms DOER in all cases, maybe due to the fact that their components are kept for more time into the ensemble, since the ranking mechanism temporarily excludes a component of the ensemble's output.

For the Friedman dataset, all algorithms presented larger errors in the beginning of the dataset, then, the error was gradually decreased and kept stable until the end. EOS-rank outperformed the other approaches with respect to the accuracy and standard deviation.

In general, EOS-based algorithms reported a similar performance over Hyperplane datasets. For Hyperplane 500, EOS-rank presented the best performance, followed by EOS-D, EOS and, finally, the OS-ELMsw. It is important to highlight that, for artificial datasets, OS-ELMsw performed relatively better compared with the results obtained for the real-world datasets. Nevertheless, as shown in figures 14 to 18, subplots (b) and (c), OS-ELMsw presents larger standard deviations, compared with the ensemble approaches. Ensembles of OS-ELMs present smaller standard deviations than single model approaches, which means that, ensemble approaches improve the stability of the predictions. This is an important issue considering the random nature of ELMs since it randomly assigns the parameters of the hidden nodes and input weights (Bueno et al., 2016).

For Hyperplane 1000, 2000 and 300 the results were similar. EOS-based algorithms performed better than DOER-based approaches. The EOS-D obtained the best performance for Hyperplane 1000 and Hyperplane 3000, where EOS-rank presented the same accuracy and standard deviation. For Hyperplane 2000, EOS outperformed the other approaches. In general, all EOS-based approaches obtained a similar performance for artificial datasets.

5.5.2 Experimental results for the Particulate Matter datasets

Table 11 reports the experimental results obtained for the Particulate Matter datasets. In this stage, EOS was not considered in the evaluation, since EOS-D and EOS-rank showed better accuracy than EOS in most of the experiments presented in the previous section.

Table 11. Performance comparison of the proposed approaches for the, Campinas, Jundiaí and São Caetano do Sul datasets

Dataset	Algorithm	Results	
		Mean Squared Error	Time
Campinas	DOER	0.01540 ± 0.00230	25.35 ± 16.2550
	DOER-rank	0.01378 ± 0.00002	27.87 ± 17.6800
	EOS-rank	**0.01155 ± 0.00001**	51.79 ± 0.26186
	EOS-D	0.01168 ± 0.00001	17.64 ± 3.69608
	OS-ELMsw	0.01193 ± 0.00007	8.014 ± 0.16395
Jundiaí	DOER	0.04560 ± 0.03396	39.15 ± 8.63620
	DOER-rank	0.00834 ± 0.00003	24.33 ± 13.8520
	EOS-rank	**0.00673 ± 0.00001**	45.34 ± 11.7002
	EOS-D	0.00684 ± 0.00001	15.90 ± 4.14645
	OS-ELMsw	0.00685 ± 0.00000	7.953 ± 0.17596
São Caetano do Sul	DOER	0.01392 ± 0.00061	41.74 ± 8.88056
	DOER-rank	0.01301 ± 0.00004	34.96 ± 13.6878
	EOS-rank	**0.01034 ± 0.00001**	31.47 ± 19.5576
	EOS-D	0.01051 ± 0.00001	15.39 ± 0.77898
	OS-ELMsw	0.01050 ± 0.00002	7.962 ± 0.26436

Results in Table 11 indicate that EOS-rank outperformed the other approaches in all scenarios. The second-best algorithm was EOS-D, followed by the single model OS-ELMsw and, finally, DOER-rank and DOER respectively.

Figures 19 to 21 present the online cumulative error and the box plots of the MSE for the PM datasets (Campinas, Jundiaí and São Caetano do Sul). As can be observed, EOS-based approaches not only perform better than DOER-based algorithms, but were also more stable, since DOER-based approaches presented abrupt errors in all scenarios.

Results obtained for PM datasets are similar with the results obtained for artificial datasets where the environment did not change too fast, so high frequencies of inclusion/replacement of models to adapt the model to new concepts (as DOER-based

approaches) are not needed. DOER-based algorithms may lose important information of the current concept by replacing components of the ensemble very fast.

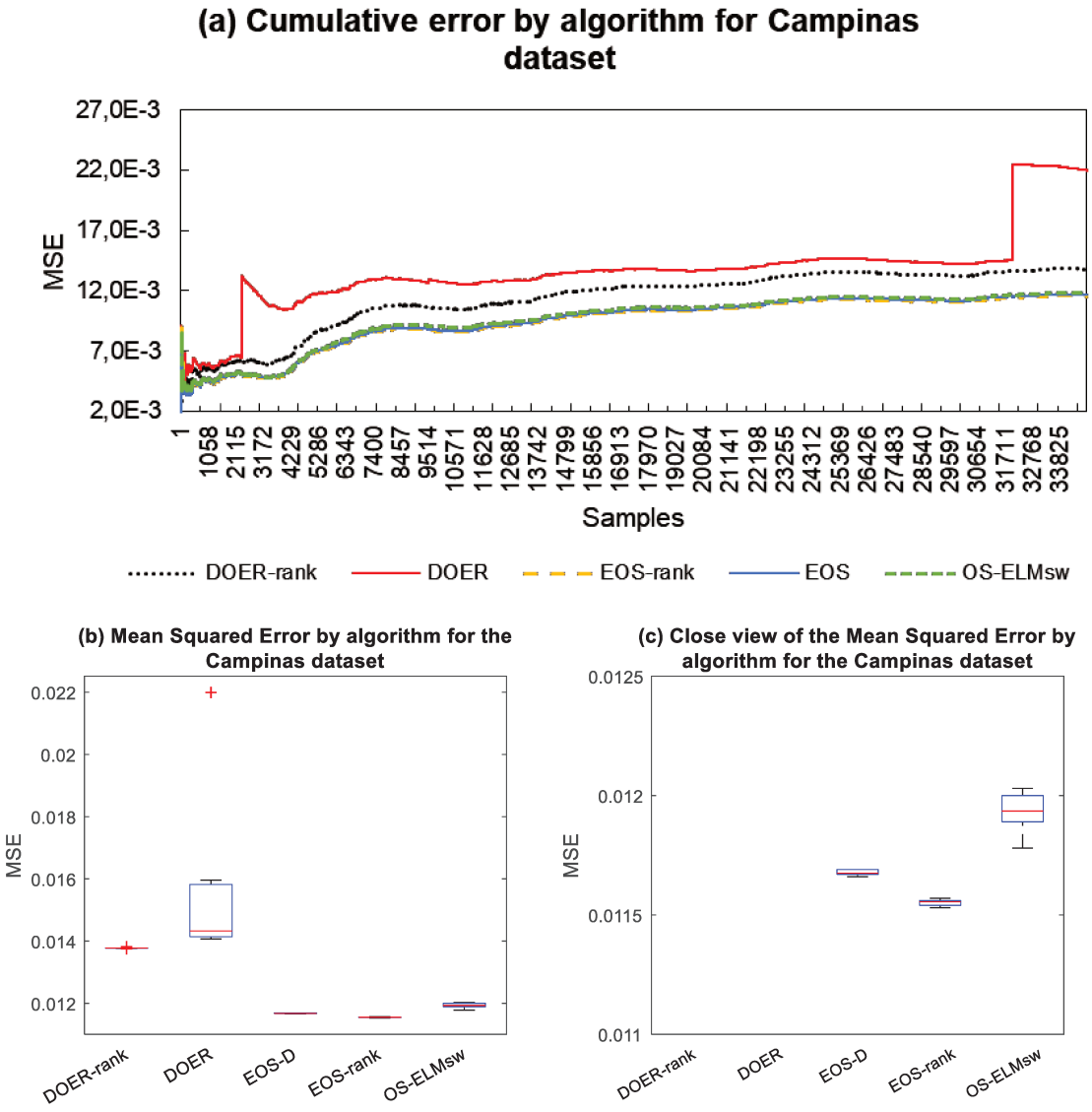

Figure 19. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Campinas dataset

Additionally, DOER does not work well on recurring drifts, due to the high frequency of incorporation/replacement of models. Its mechanisms rapidly adapt the ensemble to new concepts in such a way that the ensemble only contains information about the current

concept. Nevertheless, when recurring concepts appears again, DOER takes some time to reintroduce the recurring concept to the ensemble (Soares & Araújo, 2015).

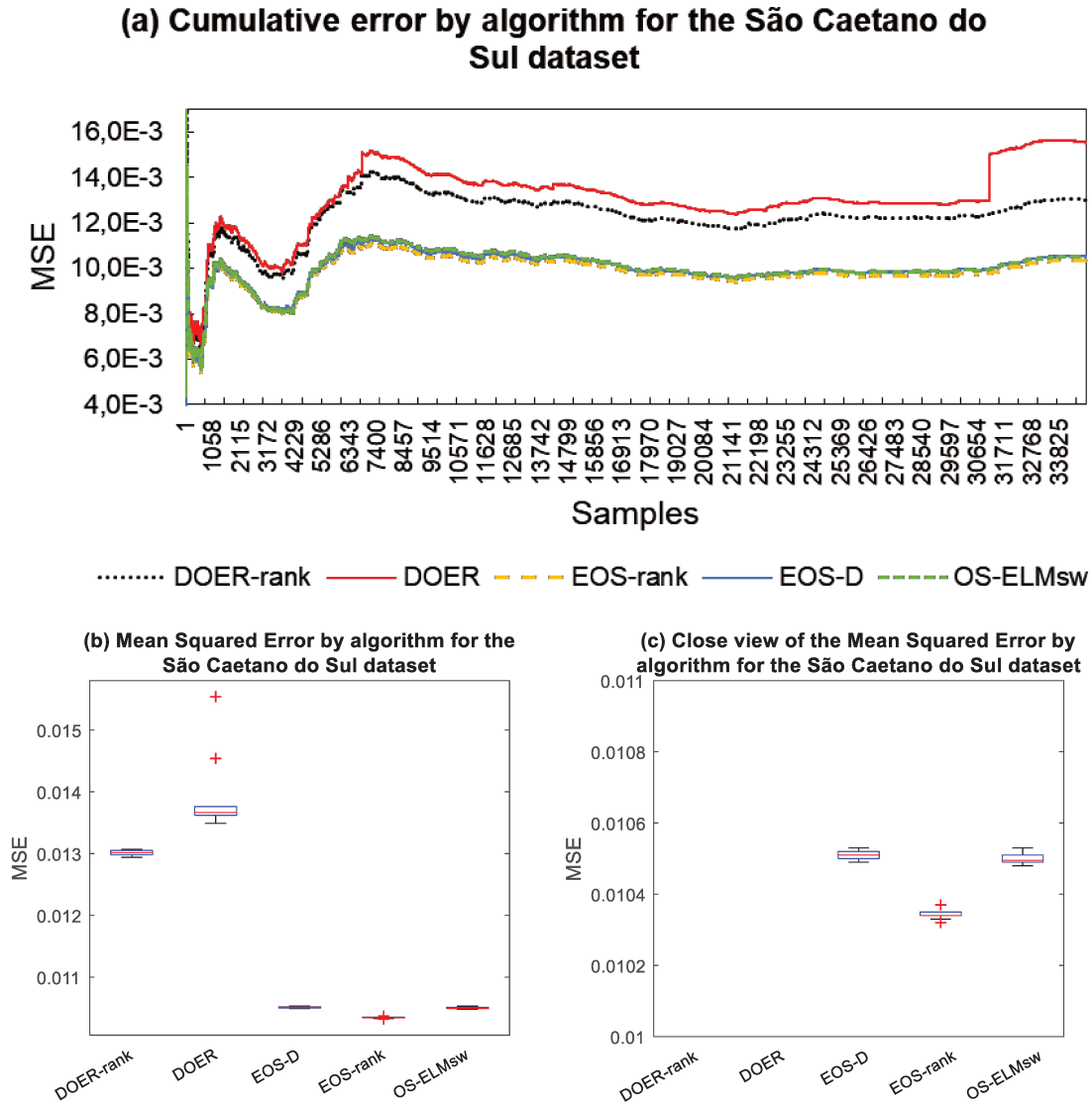

Figure 20. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for São Caetano dataset

In contrast, EOS-based approaches transition from one concept to the other more slowly, thus, keeping the information of the recurring concept and being capable of handling the information of both concepts at the same time. EOS-rank can exclude components with old information from the calculation of the ensemble's output until that information becomes useful again. In other words, when that concept appears again, the model is re-

incorporated into the ensemble's output to predict the new incoming instances of the recurring concept. This may explain why DOER-based approaches present abrupt errors over time in all PM scenarios, especially for Jundiaí dataset (Figure 21). DOER operates with higher frequencies of inclusion/replacement of models, therefore, more modifications in the ensemble are done compared with EOS approaches.

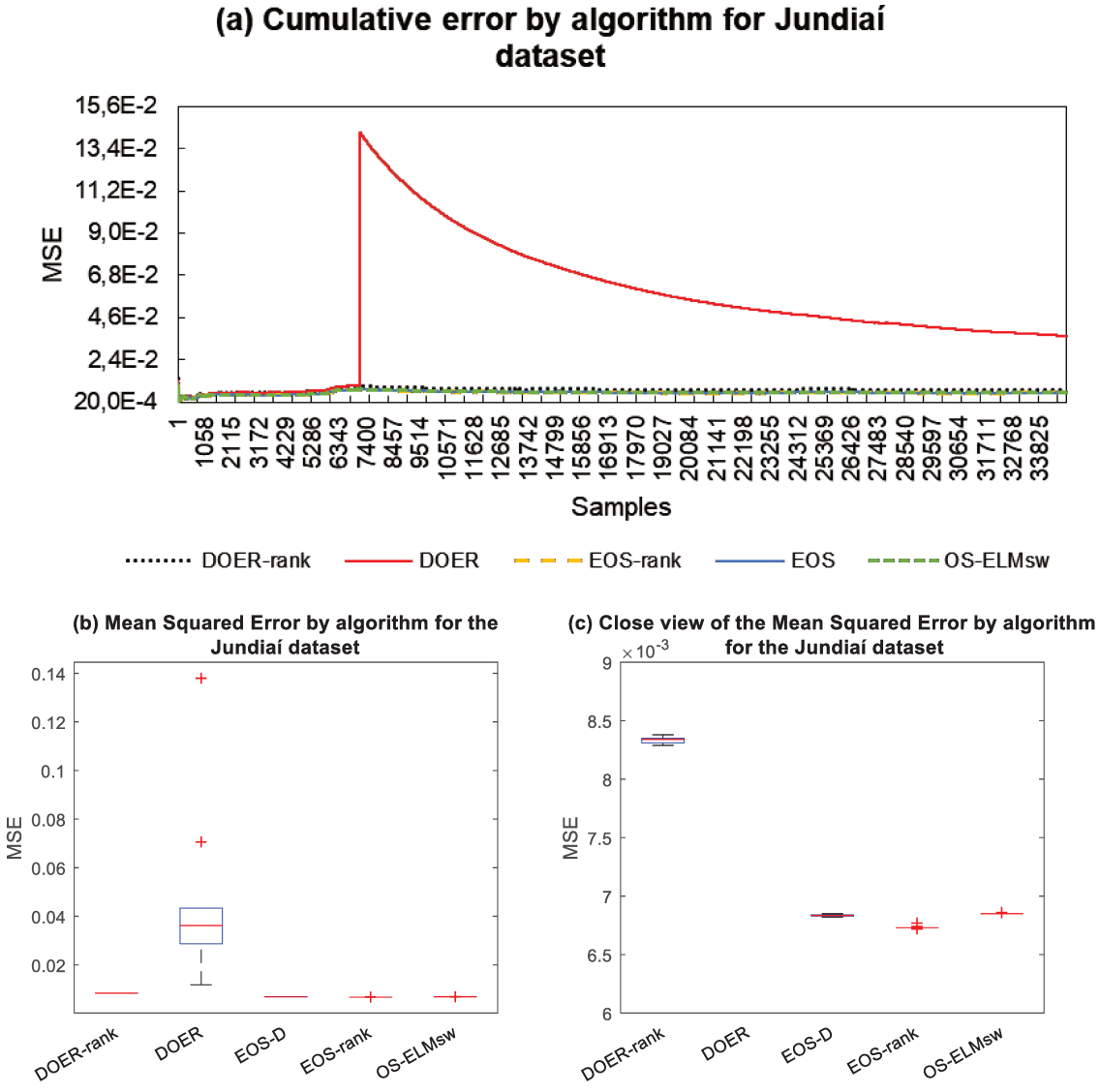

Figure 21. Online cumulative error and box plots of the Mean Squared Errors of each algorithm, for Jundiaí dataset

On the other hand, DOER-rank seems to minimize those abrupt errors that affect DOER, maybe because models live for more time in DOER-rank than in DOER, thus, keeping important information of different concepts for more time.

Forecasting in PM scenarios is not a trivial task due to the complexity of the processes involved and the influence of many factors that affect the forecasting models' performance (Bianco et al., 2017; Peng et al., 2017). Between those factors, meteorological parameters have shown to have a great influence over the concentrations of PM, especially those related with the seasons of the year, that may produce cyclical patterns on PM concentrations (Mao et al., 2017; Peng et al., 2017).

Additionally, strong correlations between carbon monoxide (CO) and concentrations of PM_{10} have been found, suggesting common sources for PM_{10} and CO (Bianco et al., 2017). The emissions of this pollutant can be related, for example, with fossil fuel combustion and may present different patterns according to the traffic in urban areas, which can be repeated along the time. That evidence, together with the results presented above, may indicate that recurring drifts with low rates of change are present in PM datasets. Therefore, approaches that slowly adapt to changes, and keep information of past concepts, as EOS-based algorithms and the OS-ELMsw, tend to perform better in such scenarios, as observed in the experimental results.

The results presented in this section indicate that the frequency of inclusion/removal of components is an important issue in concept drift scenarios. Specifically, the results indicate that low frequencies of inclusion/removal of models, like in EOS-based approaches, tend to offer better results in scenarios where the underlying distributions do not change very fast. On the other hand, in scenarios when the underlying patterns evolve fast, high frequencies of inclusion/removal of models, like in DOER-based approaches, are more suitable. Regarding PM scenarios, the results showed better performances of EOS-based approaches, thus, indicating that those scenarios do not change very fast.

6 Conclusions

The main contribution of this work is the incorporation of mechanisms to deal with concept drift in PM_{10} concentration forecasting. To do so, this work proposed an ensemble of online learners implementing the dynamic inclusion and exclusion of components scheme created by (Street & Kim, 2001), the Ensemble of Online Learners with Substitution of Models (EOS). This approach incorporates sliding windows to update the online learners that compose the ensemble, since those mechanisms (sliding windows, ensemble learning) have shown to be effective to deal with changing environments. Additionally, two derived approaches based on EOS were proposed; the EOS-rank, which adds a ranking mechanism for online selection of the best subset of components, and the EOS-D, which incorporates an initial ensemble of components and a weighted average aggregation strategy.

The proposed approaches were evaluated and compared with artificial and real-world datasets and with one of the state-of-the-art algorithms for concept drift scenarios: the Dynamic and On-line Ensemble for Regression (DOER).

Five artificial datasets that present long-lasting concepts, and three industrial application datasets with multiple concepts and high rate of drifts were used in the first part of the experiments. The results showed that DOER-based algorithms performed better in scenarios with high rates of drifts, due the high frequency of ensemble updates. On the other hand, the proposed EOS-based algorithms, together with the OS-ELMsw, performed better in scenarios with low rates of drifts, since they are capable of retaining information of past concepts for more time, which becomes useful when recurring concepts appear again.

Real-world datasets collected by CETESB in three cities of the State of São Paulo, Brazil, were used to evaluate the behavior of the best approaches identified in the first part of the experiments when applied to forecast future concentration of PM_{10}. The aim of these experiments was to determine if the incorporation of mechanisms to deal with concept drift could enhance PM_{10} concentration forecasting. The observed results indicate that particulate matter scenarios may present recurrent drifts with low rates of change. EOS-

based approaches and OS-ELMsw presented the best performances in each PM scenario. EOS-rank obtained the best results in all cases.

The obtained results are coherent with those reported in the literature, which suggest that meteorological factors, including those associated with seasons of the year, together with the concentration of other pollutants in the atmosphere, influence the behavior of particulate matter concentration. Thus, PM data patterns may evolve over time, making online approaches, such as OS-ELM, and techniques capable of dealing with concept drift (like all EOS and DOER variants considered here) more suitable to deal with this problem.

As future works, the incorporation of meteorological information (i.e., wind speed and humidity) together with concentration values of other pollutants (i.e., CO and CO_2), highly correlated with PM concentrations, is recommended. This approach has shown to be effective in works reported in the literature and can be enhanced using the mechanism proposed in this work. Another approach to be explored is the dynamic switch between DOER and EOS adaptation mechanisms. This will enable the predictor to use faster adaptation mechanism (DOER) when the environment is changing quickly, incorporating components at a faster frequency. On the other hand, when the environment is not changing fast, turn on the EOS adaptation mechanisms to retain the information of the environment for more time. To do so, Drift Detection Mechanism as proposed by Baena-García et al. (2006) may support the dynamic selection of the adaptation mechanism according to the current behavior of the environment.

References

Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gadalvà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Proceedings of 4th International Workshop on Knowledge Discovery from Data Streams* (pp. 77–86). Palo Alto, CA.

Bell, M. L., Samet, J. M., & Dominici, F. (2004). Time-series studies of particulate matter. *Annual Review of Public Health*, *25*, 247–280. http://doi.org/10.1146/annurev.publhealth.25.102802.124329

Bhattacharjee, H., Drescher, M., Good, T., Hartley, Z., Leza, J., Lin, B., Moss, J., Massey, R., Nishino, T., Ryder, S., Sachs, N., Tozan, Y., Taylor, C., Wu, D. (1999). Environmental effects of particulate matter. In Particulate Matter in New Jersey (5-1, - 5-25). Princeton, NJ: Princeton University.

Biancofiore, F., Busilacchio, M., Verdecchia, M., Tomassetti, B., Aruffo, E., & Bianco, S. et al. (2017). Recursive neural network model for analysis and forecast of PM10 and PM2.5. *Atmospheric Pollution Research*, 8(4), 652-659. http://dx.doi.org/10.1016/j.apr.2016.12.014

Bueno, A., Coelho, G. P., & Bertini, J. R. (2017). Online Sequential Learning Based on Extreme Learning Machines for Particulate Matter Forecasting. In *2017 Brazilian Conference on Intelligent Systems (BRACIS)* (pp. 169–174). IEEE. http://doi.org/10.1109/BRACIS.2017.25

Calderón-Garcidueñas, L., Kulesza, R. J., Doty, R. L., D'Angiulli, A., & Torres-Jardón, R. (2015). Megacities air pollution problems: Mexico City Metropolitan Area critical issues on the central nervous system pediatric impact. *Environmental Research*, *137*, 157–169. http://doi.org/10.1016/j.envres.2014.12.012

Cavalcante, R. C., & Oliveira, A. L. I. (2015). An approach to handle concept drift in financial time series based on Extreme Learning Machines and explicit Drift Detection. *Proceedings of the International Joint Conference on Neural Networks*, *2015–Septe*. http://doi.org/10.1109/IJCNN.2015.7280721

CETESB - Publications / Reports Environmental Agency of the State of São Paulo - CETESB, Air Quality. [in Portuguese]. http://ar.cetesb.sp.gov.br/publicacoes-relatorios/. 2017 (Accessed: 26 December 2017).

CETESB – São Paulo Environmental Agency, 2014, 2013 Air Quality Report <www.cetesb.sp.gov.br/ar/qualidade- do-ar/31-publicacoes-e-relatorios>, accessed in 24.10.2016

Elwell, R., & Polikar, R. (2011). Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Transactions on Neural Networks*, *22*(10), 1517–1531. http://doi.org/10.1109/TNN.2011.2160459

Fdez-Riverola, F., Iglesias, E. L., Díaz, F., Méndez, J. R., & Corchado, J. M. (2007). Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications*, *33*(1), 36–48. http://doi.org/10.1016/j.eswa.2006.04.011

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. *Advances in Artificial Intelligence–SBIA 2004*, (October 2017), 286–295. http://doi.org/10.1007/978-3-540-28645-5_29

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys, 46*(4), 1–37. http://doi.org/10.1145/2523813

Gomes Soares, S., & Araújo, R. (2015). An on-line weighted ensemble of regressor models to handle concept drifts. *Engineering Applications of Artificial Intelligence*, *37*, 392–406. http://doi.org/10.1016/j.engappai.2014.10.003

Gonçalves, P. M., De Carvalho Santos, S. G. T., Barros, R. S. M., & Vieira, D. C. L. (2014). A comparative study on concept drift detectors. *Expert Systems with Applications*, *41*(18), 8144–8156. http://doi.org/10.1016/j.eswa.2014.07.019

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques*. (Morgan Kaufmann Publishers, Ed.) (Vol. 3). Waltham, Mass. http://doi.org/10.1088/1751-8113/44/8/085201

Huang G., Qin-Yu Zhu, & Chee-Kheong Siew. Extreme learning machine: a new learning

scheme of feedforward neural networks. *2004 IEEE International Joint Conference On Neural Networks* (IEEE Cat. No.04CH37541). http://dx.doi.org/10.1109/ijcnn.2004.1380068

IBGE : Instituto Brasileiro de Geografia e Estatística. (2017). Ibge.gov.br. Retrieved 24 October 2016, from http://www.ibge.gov.br/home/

Ikonomovska, E. (2012). *Algorithms for Learning Regression Trees and Ensembles on Evolving Data Streams (Phd's thesis)*. Ljubljana, Slovenia: Jo_zef Stefan International Postgraduate School.

Kadlec, P. & Gabrys, B. (2010). Local learning-based adaptive soft sensor for catalyst activation prediction. *Aiche Journal*, 57(5), 1288-1301. http://dx.doi.org/10.1002/aic.12346

Kolter, J. & Maloof, M. (2005). Using additive expert ensembles to cope with concept drift. *Proceedings Of The 22Nd International Conference On Machine Learning - ICML* '05. http://dx.doi.org/10.1145/1102351.1102408

Liang, C. S., Duan, F. K., He, K. Bin, & Ma, Y. L. (2016). Review on recent progress in observations, source identifications and countermeasures of PM2.5. *Environment International*, *86*, 150–170. http://doi.org/10.1016/j.envint.2015.10.016

Liang, N.-Y., Huang, G.-B., Saratchandran, P., & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council*, *17*(6), 1411–1423. http://doi.org/10.1109/TNN.2006.880583

Liao, X., Member, S., & Carin, L. (2009). Concept Drift Between Two Data Sets With Application to UXO Sensing, *47*(5), 1454–1466.

Liu, Y., He, B., Dong, D., Shen, Y., & Yan, T. (2015). ROS-ELM: A Robust Online Sequential Extreme Learning Machine for Big Data Analytics. *Proceedings of ELM-2014 Volume 1, Algorthims and Theories*, *3*, 325–344. http://doi.org/10.1007/978-3-319-14063-6

Mao, X., Shen, T., & Feng, X. (2017). Prediction of hourly ground-level PM 2 . 5 concentrations 3 days in advance using neural networks with satellite data in eastern China. *Atmospheric Pollution Research*. http://doi.org/10.1016/j.apr.2017.04.002

Oprea, M., Ianache, C., Mihalache, S. F., Dragomir, E. G., Dunea, D., Iordache, S., & Savu, T. (2015). On the development of an intelligent system for particulate matter air pollution monitoring, analysis and forecasting in urban regions. In *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)* (pp. 711–716). IEEE. http://doi.org/10.1109/ICSTCC.2015.7321377

Pearson, R. K., Neuvo, Y., Astola, J., & Gabbouj, M. (2015). The class of generalized hampel filters. *2015 23rd European Signal Processing Conference, EUSIPCO 2015*, 2501–2505. http://doi.org/10.1109/EUSIPCO.2015.7362835

Peng, H., Lima, A. R., Teakles, A., Jin, J., Cannon, A. J., & Hsieh, W. W. (2017). Evaluating hourly air quality forecasting in Canada with nonlinear updatable machine learning methods, 195–211. http://doi.org/10.1007/s11869-016-0414-3

Pope III, C. A., Burnett, R. T., Thun, M. J., Calle, E. E., Krewski, D., & Thurston, G. D. (2002). Lung Cancer, Cardiopulmonary Mortality, and Long-term Exposure to Fine Particulate Air Pollution. *The Journal of the American Medical Association*, *287*(9), 1132–1141. http://doi.org/10.1001/jama.287.9.1132

Pozza, S. A. (2009). *Características Temporais da Concentração de Material Particulado na Atmosfera da Cidade de São Carlos – SP*. (Thesis Doctoral). São Carlos – SP. Universidade Federal de São Carlos.

Qualar. (2013). Qualidade do Ar. Retrieved 24 October 2016, from http://ar.cetesb.sp.gov.br/qualar/

Raimondo, G., Montuori, A., Moniaci, W., Pasero, E., & Almkvist, E. (2007). Data-driven models to forecast PM10 concentration. *IEEE International Conference on Neural Networks - Conference Proceedings*, 190–194. http://doi.org/10.1109/IJCNN.2007.4370953

Ross, G. J., Adams, N. M., Tasoulis, D. K., & Hand, D. J. (2012). Exponentially weighted

moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2), 191–198. http://doi.org/10.1016/j.patrec.2011.08.019

Shaban, K. B., Kadri, A., & Rezk, E. (2016). Urban Air Pollution Monitoring System With Forecasting Models. *IEEE Sensors Journal*, *16*(8), 2598–2606. http://doi.org/10.1109/JSEN.2016.2514378

Soares, S. G., & Araújo, R. (2015). A dynamic and on-line ensemble regression for changing environments. *Expert Systems with Applications*, *42*(6), 2935–2948. http://doi.org/10.1016/j.eswa.2014.11.053

Soares, S. G., & Araújo, R. (2016). An adaptive ensemble of on-line Extreme Learning Machines with variable forgetting factor for dynamic system prediction. *Neurocomputing*, *171*, 693–707. http://doi.org/10.1016/j.neucom.2015.07.035

Souza, R. M. S., Coelho, G. P., Estela, A., Silva, A., & Simone, A. (2015). Using Ensembles of Artificial Neural Networks to Improve PM 10 Forecasts. *Chemical Engineering Transactions*, *43*, 2161–2166. http://doi.org/10.3303/CET1543361

Street, W. N., & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '01*, 377–382. http://doi.org/10.1145/502512.502568

Taylor, P. (2008). Airborne Particulate Matter and Human Health : Toxicological Assessment and Importance of Size and Composition of Particles for Oxidative Damage and Carcinogenic Mechanisms, (October 2015), 37–41. http://doi.org/10.1080/10590500802494538

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, *4*(C), 2004–15. http://doi.org/10.1.1.58.9085

WHO Air quality guidelines for particulate matter, ozone, nitrogen dioxide and sulfur dioxide : global update 2005 : summary of risk assessment. (2018). *Apps.who.int*. Retrieved 25 March 2018, from http://apps.who.int/iris/handle/10665/69477

Xuejun Liao, & Carin, L. (2009). Migratory Logistic Regression for Learning Concept Drift Between Two Data Sets With Application to UXO Sensing.*IEEE Trans. Geosci. Remote Sensing*, *47*(5), 1454-1466. http://dx.doi.org/10.1109/tgrs.2008.2005268

Yadav, B., Ch, S., Mathur, S., & Adamowski, J. (2016). Discharge forecasting using an Online Sequential Extreme Learning Machine (OS-ELM) model: A case study in Neckar River, Germany. Measurement, 92, 433-445. http://dx.doi.org/10.1016/j.measurement.2016.06.042