Pós-Graduação em Ciência da Computação

Rodolfo Carneiro Cavalcante

AN ADAPTIVE LEARNING SYSTEM FOR TIME SERIES FORECASTING IN THE PRESENCE OF CONCEPT DRIFT

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

Rodolfo Carneiro Cavalcante

AN ADAPTIVE LEARNING SYSTEM FOR TIME SERIES FORECASTING IN THE PRESENCE OF CONCEPT DRIFT

This Ph.D thesis was presented to the Informatics Center of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Philosphy Doctor in Computer Science.

SUPERVISOR: **Prof. Adriano Lorena Inácio de Oliveira**
CO-SUPERVISOR: **Prof. Leandro Lei Minku**

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

C376a Cavalcante, Rodolfo Carneiro
An adaptive learning system for time series forecasting in the presence of concept drift / Rodolfo Carneiro Cavalcante. – 2017.
150 f.: il., fig., tab.

Orientador: Adriano Lorena Inácio de Oliveira.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências.

1. Inteligência computacional. 2. Previsão de séries temporais. I. Oliveira, Adriano Lorena Inácio de (orientador). II. Título.

006.3 CDD (23. ed.) UFPE- MEI 2017-125

Rodolfo Carneiro Cavalcante

An Adaptive Learning System for Time Series Forecasting in the Presence of Concept Drift

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de **Doutor** em Ciência da Computação.

Aprovado em: 13/03/2017.

_____

Prof. Adriano Lorena Inacio de Oliveira
Orientador do Trabalho de Tese

BANCA EXAMINADORA

_____

Profa. Dra. Teresa Bernarda Ludermir
Centro de Informática / UFPE

_____

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio
Centro de Informática / UFPE

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto
Centro de Informática / UFPE

_____

Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho
Instituto de Ciências Matemática e Computação / USP

_____

Prof. Dr. Tiago Alessandro Espinola Ferreira
Departamento de Estatística e Informática / UFRPE

I dedicate this work to my parents who, in all wisdom, have taught me right and wrong.
Without them I would not have gotten this far.

Acknowledgements

Ao grande projetista do universo, que em sua infinita sabedoria, nos provê, entre outras coisas, a justiça pelos nossos atos. Tenho sinais de Sua presença durante toda esta caminhada.

Aos meus pais, José Carneiro e Maria Elza, pela educação que me deram e por sempre me apoiarem em todas as minhas batalhas. Aos meus irmãos Ricardo e Mariana, por todo o companheirismo e amor proporcionados. Agradeço em especial a meu irmão Ricardo pelo lindo presente, minha sobrinha Maria Alice, um anjo que veio a este mundo para nos alegrar. Agradeço ao meu tio Wilson, por quem tenho muito respeito e estima.

Ao meu orientador, Adriano Oliveira, por ter contribuído diretamente com minha pesquisa e, principalmente, com minha formação. Agradeço pelo aprendizado durante este tempo e pelo caminho que trilharemos daqui em diante. Ao meu co-orientador, Leandro Minku, por sua grande dedicação com minha pesquisa e sobretudo pelo zelo com a ciência. Pude aprender várias lições a partir desta interação.

À banca, pelas valiosas contribuições para a melhoria do presente trabalho e ao Centro de Informática por criar o ambiente necessário ao desenvolvimento desta tese. Aos professores Teresa Ludermir, Francisco de Carvalho, Renata Souza e Aluízio Araújo pelos importantes fundamentos aprendidos nas disciplinas cursadas. Ao pessoal da secretaria da pós, por todo o suporte durante estes 4 últimos anos. À FACEPE (Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco) pelo suporte financeiro durante meu primeiro ano de doutorado.

Aos amigos que fiz em Recife, em especial os companheiros do ap, Dário, David e Hugo, pelas boas conversas e momentos de descontração. Aos colegas do grupo de pesquisa pela ajuda durante esse tempo. Ao amigo Tiago Lima pela amizade e solicitude.

Aos meus amigos do curso de Ciência da Computação da UFAL, que sempre me apoiaram durante o desenvolvimento deste trabalho. Vocês contribuíram direta e indiretamente para essa pesquisa.

E por fim, a minha amada esposa Joiciane, que me acompanha durante toda esta vida acadêmica, por todo o seu apoio e principalmente pela paciência durante estes anos.

"There is only one good, knowledge, and one evil, ignorance."
(Socrates)

Abstract

A time series is a collection of observations measured sequentially in time. Several real-world dynamic processes can be modeled as time series. One of the main problems of time series analysis is the forecasting of future values. As a special kind of data stream, a time series may present concept drifts, which are changes in the underlying data generation process from time to time. The concept drift phenomenon affects negatively the forecasting methods which are based on observing past behaviors of the time series to forecast future values. Despite the fact that concept drift is not a new research area, the effects of concept drifts in time series are not widely studied. Some approaches proposed in the literature to handle concept drift in time series are passive methods that successive update the learned model to the observations that arrive from the data stream. These methods present no transparency to the user and present a potential waste of computational resources. Other approaches are active methods that implement a detect-and-adapt scheme, in which the learned model is adapted just after the explicit detection of a concept drift. By using explicit detection, the learned model is updated or retrained just in the presence of drifts, which can reduce the space and computational complexity of the learning system. These methods are generally based on monitoring the residuals of a fitted model or on monitoring the raw time series observations directly. However, these two sources of information (residuals and raw observations) may not be so reliable for a concept drift detection method applied to time series. Residuals of a fitted model may be influenced by problems in training. Raw observations may present some variations that do not represent significant changes in the time series data stream. The main contribution of this work is an active adaptive learning system which is able to handle concept drift in time series. The proposed method, called Feature Extraction and Weighting for Explicit Concept Drift Detection (FW-FEDD) considers a set of time series features to detect concept drifts in time series in a more reliable way, being trustworthy and transparent to users. The features considered are weighted according to their importance to define concept drifts at each instant. A concept drift test is then used to detect drifts in a more reliable way. FW-FEDD also implements a forecasting module composed by a pool of forecasting models in which each model is specialized in a different time series concept. Several computational experiments on both artificial and real-world time series showed that the proposed method is able to improve the concept drift detection accuracy compared to methods based on monitoring raw time series observations and residual-based methods. Results also showed the superiority of FW-FEDD compared to other passive and active adaptive learning systems in terms of forecasting performance.

Keywords: Adaptive learning systems. Data streams. Concept drift. Time series. Forecasting.

Resumo

Uma série temporal é uma coleção de observações medidas sequencialmente no tempo. Diversos processos dinâmicos reais podem ser modelados como uma série temporal. Um dos principais problemas no contexto de séries temporais é a previsão de valores futuros. Sendo um tipo especial de fluxo de dados, uma série temporal pode apresentar mudança de conceito, que é a mudança no processo gerador dos dados. O fenômeno da mudança de conceito afeta negativamente os métodos de previsão baseados na observação do comportamento passado da série para prever valores futuros. Apesar de que mudança de conceito não é uma nova área, os efeitos da mudança de conceito em séries temporais ainda não foram amplamente estudados. Algumas abordagens propostas na literatura para tratar esse problema em séries temporais são métodos passivos que atualizam sucessivamente o modelo aprendido com novas observações que chegam do fluxo de dados. Estes métodos não são transparentes para o usuário e apresentam um potencial consumo de recursos computacionais. Outras abordagens são métodos ativos que implementam um esquema de detectar-e-adaptar, no qual o modelo aprendido é adaptado somente após a detecção explícita de uma mudança. Utilizando detecção explícita, o modelo aprendido é atualizado ou retreinado somente na presença de mudanças, reduzindo a complexidade computacional e de espaço do sistema de aprendizado. Estes método são geralmente baseados na monitoração dos resíduos de um modelo ajustado ou na monitoração dos dados da série diretamente. No entanto, estas duas fontes de informação (resíduos e dados crus) podem não ser tão confiáveis para um método de detecção de mudanças. Resíduos de um modelo ajustado podem ser influenciados por problemas no treinamento. Observações cruas podem apresentar variaçõs que não representam mudanças significativas no fluxo de dados. A principal contribuição deste trabalho é um sistema de aprendizado adaptativo ativo capaz de tratar mudanças de conceito em séries temporais. O método proposto, chamado de *Feature Extraction and Weighting for Explicit Concept Drift Detection* (FW-FEDD) considera um conjunto de características da série temporal para detectar mudança de conceito de uma forma mais confiável, sendo transparente ao usuário. As características consideradas são ponderadas de acordo com sua importância para a definição das mudanças em cada instante. Um teste de mudança de conceito é utilizado para detectar as mudanças de forma mais confiável. FW-FEDD também implementa um módulo de previsão composto por um conjunto de modelos de previsão onde cada modelo é especializado em um conceito diferente. Diversos experimentos computacionais usando séries reais e artificiais mostram que o método proposto é capaz de melhorar a detecção de mudança de conceito comparado com métodos baseados na monitoração de dados crus da série e métodos baseados em resíduos. Resultados também mostraram a superioridade do FW-FEDD comparado com outros métodos de aprendizado adaptativo ativos e passivos em termos de acurácia de predição.

Palavras-chave: Sistemas de aprendizado adaptativo. Fluxos de Dados. Mudança de Conceito. Séries Temporais. Previsão.

List of Figures

List of Tables

List of abbreviations and acronyms

ABC	Artificial Bee Colony
ADWIN	Adaptive Windowing
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
AR	Autoregressive Process
ARCH	Autoregressive Conditional Heteroskedastic
ARIMA	Autoregressive Integrated Moving Average
ARMAX	Autoregressive Moving Average with Exogenous Variables
AWNN	Adaptive Wavelet Neural Network
CDT	Concept Drift Test
CPM	Change-Point Method
CUSUM	Cumulative Sum
DDD	Diversity for Dealing with Drifts
DDM	Drift Detection Mechanism
DOF	Degree of Drift
DWM	Dynamic Weighted Majority
ECDD	Exponentially Weighted Moving Average for Drift Detection
EDDM	Early Drift Detection Mechanism
ELM	Extreme Learning Machine
EWMA	Exponentially Weighted Moving Average
FLANN	Functional Link Artificial Neural Network
FW-FEDD	Feature Extraction and Weighting for Explicit Concept Drift Detection
GA	Genetic Algorithms

GARCH	Generalized Autoregressive Conditional Heteroskedastic
ICA	Independent Component Analysis
ICI	Intersection of Confidence Intervals
KNN	K-Nearest Neighbor
LSSVM	Least Squared Support Vector Machine
MA	Moving Average Process
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Multilayer Perceptron
NICA	Nonlinear Independent Component Analysis
OS-ELM	Online-Sequential Extreme Learning Machine
OWE	Online Weighted Ensemble of Regressor Models
PCA	Principal Component Analysis
PL	Paired Learners
PHt	Page-Hinkley Test
PSO	Particle Swarm Optimization
RMSE	Root Mean Square Error
RBF	Radial Basis Function
SARIMA	Seasonal Autoregressive Integrated Moving Average
SLFN	Single Hidden Layer Feedforward Neural Network
SOMLP	Self-organized Multilayer Perceptron
STEPD	Statistical Test of Equal Proportions
SVD	Single Value Decomposition
SVM	Support Vector Machine
SVR	Support Vector Regression
WT	Wavelet Transform

List of symbols

S	Time series
t	Time stamp
$\hat{y}$	Predicted time series value
X	Input attributes for a time series model
k	Lagged time series observation time stamp
$\{x_t\}$	Short representation of a time series
$\{w_t\}$	Gaussian white noise time series
p	Autoregressive order
q	Moving average order
d	Differencing order
T_{x_t}	Trend component of a time series
S_{x_t}	Seasonal component of a time series
R_{x_t}	Residual component of a time series
td	Trend degree of a time series
sd	Seasonal degree of a time series
ρ_k	Correlation coefficient
τ	Skewness coefficient
κ	Kurtosis coefficient
B	Bicorrelation coefficient
I	Mutual information
m_f	Size of the window to compute features
λ	Weighting given to recent data in computing EWMA
W	Warning threshold used in ECDD and PHt

C	Drift threshold used in ECDD and PHt
δ	Discount factor used in PHt
$\mathbb{I}$	Confidence interval estimated by ICI
Γ	Confidence parameter of ICI
fv_0	Initial feature vector
fv_t	Current feature vector at time t
fv_c	Feature vector that defines the modeled concept
s	Time stamp that demarcates the init of a concept drift
M	Set of forecasting models
F	Set of feature vectors associated with the individual forecasting models
m_m	Size of the window for building a forecasting model
θ	Threshold for inserting a new forecasting model
ψ	Parameters of the forecasting algorithm
η	Parameter that indicates the need to build a new forecasting model
γ	Threshold for discretize a regression error of DWM
β	Factor for decreasing weights of DWM

Contents

1 INTRODUCTION

A time series is a collection of observation measured sequentially in time (COWPERTWAIT; METCALFE, 2009). Several dynamic processes can be modeled as a time series, such as stock price movements (CAVALCANTE; OLIVEIRA, 2014), a company payroll (OLIVEIRA; MEIRA, 2006), product sales (HAMZACEBI, 2008), daily temperature of a city (SINGH; BORAH, 2013), electricity consumption (ALVAREZ et al., 2011), exchange rates (BRITO; OLIVEIRA, 2014), among others. Time series modeling and forecasting can be considered main challenging activities in the computational intelligence literature. In the last decades, several approaches have been proposed for time series forecasting. Two major classes of these approaches are the traditional statistical models and the computational intelligence methods (WANG et al., 2011). Statistical models assume that the time series under study is generated from a parametric process (KUMAR; MURUGAN, 2013). Computational intelligence approaches, on the other hand, are data-driven, self-adaptive methods able to capture linear and nonlinear behavior of time series without the need of *a priori* specific statistical assumptions about the data (LU; LEE; CHIU, 2009).

Despite the fact that there is a vast literature on time series forecasting, the majority of the existing approaches does not take into account that a time series is a kind of data stream (CAVALCANTE et al., 2016). A data stream is a set of data observations which arrive sequentially one by one (GAMA, 2012). Dynamism is an inherent feature of data streams. This dynamism implies that patterns in a data stream may evolve over time and introduces a big challenge to traditional batch learning algorithms, which is the ability to permanently maintain an accurate decision model even in the presence of changes in the underlying data generation process. This phenomenon is referred to as concept drift (SCHLIMMER; GRANGER, 1986b; SCHLIMMER; GRANGER, 1986a; WIDMER; KUBAT, 1993; WIDMER; KUBAT, 1996), concept shift (LUGHOFER; ANGELOV, 2011), dataset shift (RAZA; PRASAD; LI, 2015).

Most of the approaches designed to time series forecasting have a learning phase which operates in offline mode. They first learn a model from the data, and then this model is used to perform forecasting without updating the learned model. Due to this, they are unaware of concept drifts. These methods are based on the main assumption that time series concept is stable in such a way that the time series observations follow a fixed and immutable probability distribution along the time. In this scenario, these methods first learn how the time series behaves and then they are used to forecast future behaviors. This assumption, however, may not hold for several industrial time series applications. For example, the time series of the sales of a product may change its behavior due to changes

in government regulations or advertising campaigns. The time series of stock prices of a company may change its behavior due to changes in political and economical factors or due to changes in the investors psychology or expectations.

Changes in the dynamic of a time series impose serious challenges to forecasting approaches. In these cases, learning methods which first learn how a time series behaves and then are used to perform forecasting without updating may become obsolete in case of explicit or implicit changes in the time series behavior. These environment with changes requires more sophisticated learning methods, able to precisely detect and adapt to changes in real-time.

1.1 Motivation

In the last decades, the computational intelligence scientific field has been devoted to design machine learning algorithms able to learn and model specific problems in order to support decision-making. The supervised learning approach attempts to learn about a knowledge domain by means of observing past cases or instances of the problem and their respective solutions. The goal of this approach is to identify and model the relationship between descriptor attributes and outputs of the past instances. The model of the relationship between inputs and outputs represents the knowledge domain and can be used for solving unseen instances of the same problem. Examples of supervised machine learning problems are spam filtering (GUZELLA; CAMINHAS, 2009), credit card fraud detection (MAES et al., 2002), credit risk evaluation (ANGELINI; TOLLO; ROLI, 2008), stock prices time series forecasting (ATSALAKIS; VALAVANIS, 2009), among others.

The classical supervised learning main assumption is that the statistical distribution of the instances of some knowledge domain is immutable, in the sense that the examples come from a fixed unchangeable probability distribution (GAMA et al., 2004; DITZLER et al., 2015). If this supposition holds for the whole machine learning prediction process then, once an algorithm has learned how to perform a task, the learned model can be used to perform this task in the future. So, after the learning phase is completed, the system would not need further improvement or change. However, in many real-world applications, data arrives in a stream and patterns evolve over time, since concepts are often not stable (FDEZ-RIVEROLA et al., 2007; MINKU; YAO, 2012). This is due to the inherent dynamism of data streams, in which the data is collected over an extended period of time. In practice, this instability implies that a set of instances has a legitimate output at one time and a different legitimate output at another time (KOLTER; MALOOF, 2007).

Several methods able to handle concept drift have already been proposed in the literature for classification problems (GONCALVES et al., 2014). According to Ditzler et al. (2015), the existing approaches can be divided into two categories: (i) the passive

adaptive approaches, and (ii) the active adaptive approaches. The passive approaches, also called implicit or blind methods are those which update the decision model at regular intervals independent of the occurrence of concept drifts. Examples of blind methods include the online and incremental learning algorithms proposed by Fdez-Riverola et al. (2007), Cohen et al. (2008) and the dynamic ensemble methods proposed by Kolter and Maloof (2007), Tsymbal et al. (2008), Brzezinski and Stefanowski (2014a), Brzezinski and Stefanowski (2014b), Soares and Araújo (2015). The main issues of these approaches are the potential resource consumption to update the learned model even when the incoming data belong to the same concept and the lack of transparency for the user. Since these approaches just adapt to changes without properly identifying it, they do not inform the user the existence or absence of changes. Informing the user about the existence of concept drifts may increase the trust in the automatic prediction process. Practitioners may make certain decisions based on the knowledge that a change has occurred in data.

The active adaptive learning approaches are those which implement some explicit drift detection method and update the learned model just after detecting a concept drift. Examples of explicit drift detection methods are the Drift Detection Mechanism (DDM) (GAMA et al., 2004), the Early Drift Detection Method (EDDM) (BAENA-GARCIA et al., 2006) and the Exponentially Weighted Moving Average for Drift Detection (ECDD) (ROSS et al., 2012). These adaptive learning systems update the learned model just after a change is detected in the data distribution. These methods rely on an explicit drift detection mechanism, reacting to changes by updating the existing model or building a new one (GAMA et al., 2014). An advantage of explicit drift detection is that this approach works as a white box, by informing the user about the occurrence of concept drifts.

Two main explicit drift detection approaches are (i) those that monitor the residuals of a fitted model and (ii) those that monitor features extracted from the data generating process (ZLIOBAITĖ; BUDKA; STAHL, 2015). Methods that monitor the residuals of a fitted model are supported by the assumption that when the distribution of the incoming data stream is stable, the residuals of a model fitted to the data are constant or decrease as the number of predicted instances increases. The first main issue of these approaches is that the residual levels may not properly reflect concept drifts. These methods rely on the accuracy of the decision model used for prediction. If a poor training process is used to build the decision model, it may result in lots of false alarms or high miss-detection rates, due to generalization problems such as overfitting. In some cases the concept may change and the error remains constant. Monitoring data distribution directly, on the other hand, may be a faster and reliable way of detecting concept drifts in data. Based on that, in this thesis, we will investigate how to detect concept drifts in time series by inspecting some features that describes data distribution, instead of monitoring residuals.

The forecasting of future values of a time series can be considered one of the main

challenges of the time series analysis and machine learning field (TAY; CAO, 2001). As a kind of data stream, time series typically present concept drift (GUAJARDO; WEBER; MIRANDA, 2010). Despite the fact that concept drift problem has been widely studied in the literature in classification problems (GAMA, 2012; DITZLER et al., 2015), little effort has been dedicated to solve this problem in regression and time series analysis (HU et al., 2015). Concept drifts in time series forecasting have a key difference with respect to classification and other regression problems, requiring separate investigation and potentially different drift detection methods. This key difference is the serial correlation characteristic, in which the time series observations present some temporal relationship, instead of being independent and identically distributed as in classification problems. This serial correlation sometimes implies in the presence of systematic changes in time series observations, such as trend and seasonality, which do not necessarily imply in changes in the learned model. Depending on how the concept drift problem is approached, these systematic changes may hinder the drift detection process.

Some approaches have been proposed to handle concept drift in time series, specifically. However, these approaches have some limitations. Those two issues of passive approaches, namely the excessive adaptation and lack of transparency, may prevent the wide applicability of these approaches in real industrial applications. Excessive adaptation may be a waste of resources and provide only incremental insignificant benefits towards the forecasting performance (ZLIOBAITĖ; BUDKA; STAHL, 2015). Some applications require efficiency in decisions and this implies in the need for efficient adaptive learning methods. In applications in which the data have no frequent changes, the successive adaptation represents a constant computational cost, but does not result in significant improvements in forecasting accuracy. The lack of transparency for user is another negative feature of these methods. For example, in financial market forecasting, traders may decide to reduce their market positions or change the market segment to invest when concept drifts are identified.

Some concept drift detection approaches proposed in the literature for detecting changes in time series are explicit mechanisms based on monitoring time series observations directly (LIU et al., 2013; ROSS, 2013; FERREIRA; LOSCHI; COSTA, 2014; KILLICK; ECKLEY, 2014). These methods analyze time series observations in order to identify points of divergence, also known as change points. Change points are time series observations that divide a time series sequence into two segments such that the null-hypothesis of no changes in the distribution of observations is rejected. Due to this, change points are also referred to as structural breaks. These approaches generally operate in a time series after removing trend and seasonality. A concept drift is then identified when there is a statistically significant change in mean or variance of the time series observations. Such methods are unable to detect important changes in the underlying generation process, such as changes in the behavior of trends, changes in the linear or nonlinear relationship

about the observations, appearance or disappearance of seasonal patterns, changes in the periodicity of seasonality, among others.

A second group of existing time series concept drift approaches tries to identify concept drifts by monitoring the error (residuals) of a forecaster model (ALIPPI; BORACCHI; ROVERI, 2013b; CAVALCANTE; OLIVEIRA, 2015). Such approaches generate a model of the stable state of the time series and then they monitor the residuals of the fitted model to the new observations. When the distribution of these residuals changes significantly, a drift is identified. The issue of this approach is the dependence of the concept drift detection process on the model created, as discussed before.

Few approaches proposed in the literature have investigated how to detect concept drifts by monitoring *statistical time series features*[1] derived from the raw time series observations. Time series features are derivative statistics able characterize some relationship about time series observations. Boracchi and Roveri (2014) used the self-similarity feature to identify concept drifts in time series. The proposed approach measures the self-similarity between time series segments and uses the values of this feature as change detector variable. A Concept Drift Test (CDT) is then used to monitor and detect changes in this feature and identify is as a change in the time series generation process. However, this approach just monitors one aspect of time series behavior and its applicability is restrict to time series that present self-similarity. A more general method should be able to monitor several aspects of time series behavior.

There are several specific time series features that may be used to define time series concepts, such as autocorrelations, partial autocorrelations, presence of trend and seasonality, periodicity, self-similarity among others. Some of these features may be able to describe some linear or nonlinear behaviors of a time series in order to accurately describe the nature of the time series and consequently they may be effectively used to describe the underlying data generation process. Some of these features have been used in the literature to solve important time series analysis problems, such as time series classification (PRUDÊNCIO; LUDERMIR; CARVALHO, 2004), time series clustering (WANG; SMITH; HYNDMAN, 2006; AGHABOZORGI; SHIRKHORSHIDI; WAH, 2015), time series meta-learning (PRUDÊNCIO; LUDERMIR, 2004; WANG; SMITH-MILES; HYNDMAN, 2009; LEMKE; GABRYS, 2010), among others. These problems are intrinsically related with the concept definition, since their main objective is to precisely characterize a time series behavior. Clustering algorithms try to identify similarity in the whole time series or time series segments in order to group them, maximizing the similarity intra-groups. Meta-learning methods try to identify similarity in time series behaviors in order to apply a successful forecasting method used in similar time series in the past, maximizing the forecasting accuracy.

[1] It is not the same as input feature attributes.

In this sense, an important work proposed by Prudêncio and Ludermir (2004) have used some time series specific features in order to choose, among a set of possible models, the best one to forecast a given series. In this approach, several models are kept in memory associated with the time series used to build them. The time series are represented by a set of statistical features. When a new time series is available to be predicted, its features are extracted and the best suited model is recovered from memory according to the similarity of the new time series with the time series features used to build that model. We can use similar ideas to build an adaptive learning system which is robust to concept drift. Since the environment is dynamic and the time series patterns may evolve over time, we can address the problem of identifying drifts by monitoring the features in order to detect concept drifts. When a concept drift occurs, we can identify the time series features of the new concept and use a previous forecasting model (or a combination of them) according to the features of the time series concept used to build it (or them, in case of combination).

In this work, we use the same idea of using a CDT as proposed by Boracchi and Roveri (2014), but to monitor and detect changes in a set of time series features that describe time series behaviors in order to build a general model of concept drift detection. The original contributions of this work are (i) the application of a CDT on an univariate signal that summarizes the information of some time series features to detect changes in the data generation process; (ii) the proposition of a weighting function to compute the importance of the features to describe concepts to improve the drift detection; and (iii) the integration of this drift detection method to a forecasting method based on a set of individual models and a model selection approach. The CDT assess the statistical significance of concept drifts in the monitored signal, reducing the number of false positive detections.

An important work related to the forecasting method implemented in this thesis was proposed by Rossi, Carvalho and Soares (2012). In that work, authors proposed a model selection approach based on some characteristics extracted from data in the context of regression. Aiming at improving the learning system performance in dynamic environments affected by concept drift, the proposed approach used the characteristics of data to select, at fixed time intervals, the more suited model to handle the incoming data. The meta-attributes are generated by extracting characteristics from data observations, such as the existence of outliers, skewness, kurtosis, average, variance, correlation between attributes and target, among others. Periodically, the approach applies a meta-classifier to predict the most appropriate learning algorithm for the unlabeled data based on that characteristics. A sliding window scheme is used to build both the base-models and to extract features and build the meta-data.

We use similar ideas of extracting features of time series to recommend the most appropriate model to forecast a time series concept, in order to improve forecasting

performance. The time series features and drift information are used to define model competences and to recommend which models should be used to handle concepts in order to improve the forecasting accuracy. We use an explicit concept drift detection scheme to identify when there is the need to create a new forecasting model. Another important difference between the proposed work and that proposed by Rossi, Carvalho and Soares (2012) is that in this work the recommendation of models are done at each instant a new forecasting need to be done, instead of performing the recommendation periodically at fixed intervals as in that work. This approach may provide a faster drift recovering. In addition, this thesis is focused in time series forecasting, while that work are applied in regression problems.

1.2 Problem Formulation

The time series forecasting problem can be defined as follows. Let $S = \{x_1, ..., x_i, ...\}$, $x_i \in \mathbb{R}$ sampled from an unknown probability distribution, be a time series. At each time stamp t, we want to forecast the regression value $\hat{y} = x_{t+n}$, which is the time series value in the next $t + n$ instant, based on a set of *input attributes* $X = \{x_{t-k}, ..., x_t\}$, i.e., the last k time series observations. So, the utmost goal of a machine learning algorithm for building a time series forecasting method is to estimate $P(y = x_{t+n}|x_{t-k}...x_t)$. Once this function is estimated on a training set, we can use it to forecast future values $\hat{y}$. However, a concept drift in the data generation process may cause a change in the relationship between the input and target variables on the target data (test set).

Gama et al. (2014) define a concept drift between time point t_0 and time point t_1 in the context of classification as $\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y)$, where p_{t_0} is the joint distribution at time t_0 between the set of input variables X and the target variable y. These changes may occur due to changes in components of this relation, i.e., changes in the prior probabilities of classes $p(y)$, and/or changes in the class conditional probability density function $p(X|y)$, which can be viewed as change in the posterior probabilities $p(y|X)$ and/or in the unconditional probability density function $p(X)$.

In this research, we are particularly interested on changes in the posterior probabilities $p(y|X)$, or in more details, on $P(y = x_{t+n}|x_{t-k}...x_t)$, as they affect the underlying function being learned. This kind of change implies in the need for updating the forecasting model. The reason for this choice is that the main goal of this thesis is to build an adaptive learning system able to keep high forecasting accuracy even in the presence of concept drifts in the time series. This change can cause a change in the decision boundary learned by the forecasting algorithm. The proposed adaptive learning system should be able to quickly detect and to adapt its decision boundary if necessary.

So, given the inputs X and the target attribute y, in stable periods (with no concept

drift), $p(X)$ and $p(y)$ are expected to be generated by the same underlying process $\mathbb{S}$. So, changes in $p(y)$ will be always reflected by some changes in $p(X)$ with some delay. Due to this, we expect a drift detection method based on monitoring time series data ($p(X)$) to provide a better understanding of how concepts evolve over time than those based on monitoring the residuals of a fitted model. However, the existing approaches that monitor the time series observations assume that any changes in $p(X)$ would also affect $p(y|X)$, which is not always the case. Examples of these situations are the systematic changes caused by seasonality. If, instead, we monitor the right features derived from the observations, we may filter out changes in $p(X)$ that are not linked to changes in $p(y|X)$, such those caused by systematic changes (trends and seasonality, for example). This kind of explicit drift detection may provide a better understanding of how concepts evolve over time than those based on monitoring the forecasting error. Since they monitor data distribution features, the drift detection process relies just in the statistical test that assesses the evolving of the data distribution and in the feature set used to describe the data.

1.3 Objectives, Research Questions and Hypothesis

The main aim of this research is to investigate and propose a new adaptive learning system designed to model and forecast time series eventually affected by concept drifts with the utmost goal of improving the forecasting accuracy in dynamic environments. Motivated to overcome the main issues of existing active and passive adaptive learning systems described in Section 1.1, we propose a learning method that falls into the class of active adaptive approaches, since it handles concept drift by explicitly detecting changes and then updating the learned model to cope with changes. The proposed method should be:

- accurate in detecting concept drifts, since it is feature-based;
- transparent to the user, since it implements an explicit drift detection mechanism;
- able to handle recurrent concept drifts;
- accurate in forecasting time series with evolving patterns, since it quickly reacts to concept drifts;
- efficient, since it updates the learned model just after detecting a drift in the time series.

In order to accomplish the objective of building the described adaptive learning system, some research questions arise and need to be answered by this thesis:

1. Several methods have been proposed in the literature for detecting changes in time series. Some of them are based on monitoring the residuals of a fitted model. The main issue of this approach is that the residuals may not properly reflect changes in the time series. Other methods are based on monitoring the raw time series observations. However not every change in these observations indicates a change in the decision function of the forecasting model. **So, could the time series drift detection be improved by using an approach that monitors a suite of time series statistical features instead of monitoring raw time series observations or monitoring the residuals of a fitted model?**

2. After choosing a set of time series features to describe time series concepts, we need to define how to use the information of these features to accurately detect concept drifts. Some variations in one or in a small subset of these features may not properly indicate changes in the whole underlying generation process. So, detecting a concept drift whenever any of these features changes in isolation is likely to make the concept drift detection method too sensitive, generating many false alarms. **So, how can we best combine the information provided by different time series features make an accurate concept drift detection?**

3. By defining a set of time series features and an approach to combine the information of these features to detect concept drift, we build a feature-based drift detection method. However, time series from different domains may present different behaviors. For different time series behaviors, some features may be more or less informative about these behaviors. Less informative features may hinder the drift detection. **Can we implement a method for determining and emphasizing the most informative features in order to build a general concept drift detector which improved detection accuracy and which may be applied on a broad set of time series of different domains?**

4. Once the concept drifts in a time series are properly identified, the adaptive learning system should be able to quickly react to them in order to improve the forecasting accuracy. **How to better build the adaptive learning system that implements feature-based drift detection in order to improve forecasting accuracy meanwhile minimizing computational costs?**

The main hypotheses formulated in this thesis in order to answer these questions are as follows:

1. We investigate a set of time series statistical features which could best describe time series concepts. Our hypothesis is that using a set of statistical time series features, namely trend degree, seasonal degree, autocorrelation, partial autocorrelation,

skewness, kurtosis, turning points rate, periodicity, standard deviation of residuals, bicorrelation and mutual information, is effective to describe time series concepts. Based on monitoring these statistical features, we can identify time series behaviors (concepts) in the stable state (*in-control*) and detects with minimum delay when these behaviors go *out-of-control*, which configures a concept drift. In order to answer the first question, we compare concept drift detection methods based on monitoring this set of time series features against methods based on monitoring time series raw observations and methods that monitor the residuals of a forecasting model. The study analyzes the drift detection accuracy of the methods using artificial and real-world data sets.

2. There are some candidate ways of combining the monitored time series features in order to detect concept drifts. Our hypothesis is that we can combine the time series in a feature vector instead of observing them in isolation. So, we monitor the differences between an initial feature vector (which describes the time series concept in the in-control state) and the current feature vector that describes the time series at each instant of the processing. If the time series concept remains stable, then it is expected that the distances between the feature vectors be constant and small. On the other hand, when a concept drift happens, it is expected the distances increase. This approach would provide two main advantages: (i) it reduces the problem of monitoring the distribution of several features individually to the problem of monitoring the distance between the feature vectors, which is just a univariate signal; and (ii) it is able to reduce the sensitivity of the drift detector and consequently the number of false positives (false alarms) caused by variations in few features individually. The second question is answered by comparing the drift detection accuracy of our approach considering time series features in combination against the use of isolated time series features. Some explicit concept drift detection (CDT) methods were applied to detect changes in the distances between feature vectors.

3. The identification of which features are more important to define time series concepts and when these concepts change is a challenging task, since this is an unsupervised task that should be performed in an online way. According to Kuncheva and Faithfull (2014), features with lowest variance are more likely to be affected by a change than features with higher variance. Our hypothesis is that we can use some heuristic based on the variability of the time series features to implement a dynamic feature weighting strategy. This method may provide a better understanding of the evolution of time series concepts, which would increase the trust of the user on the system. It may also improve the forecasting accuracy since the learned model would be updated as soon as possible to cope with concept drifts. In order to answer the third question,

we investigate two weighting strategies and evaluate which of them best improves drift detection compared to the method without weighting.

4. There are several ways of handling concept drifts to build an adaptive learning system, such as (i) creating a forecasting model and updating this model at every new data observation or chunk of observations received by the system, (ii) eliminating an existing model and building a new one when a drift is detected, (iii) keeping an ensemble of forecasting models suited to handle different concepts, (iv) keeping a pool of forecasting models and selecting one of them to properly answer a time series concept at each instant in real-time. Our hypothesis is that using a pool of forecasting models combined with an online scheme to select which model to use to handle a new concept drift faced by the learning system is more effective in time series forecasting. Since time series concepts are characterized by a feature vector that defines its behavior, the natural way of selecting a forecasting model to forecast future values in some instant is to select the one which was trained in a similar context to the current one. So, in order to answer the fourth question, we proposed an adaptive learning which keeps a pool of forecasting models and selects the more appropriate model at each instant. The proposed method is compared with some active and passive adaptive learning strategies in terms of forecasting performance.

1.4 Organization of the Thesis

This thesis is organized as follows. In this chapter, the motivations for carrying out this research were presented together with the problem formulation and the main research questions, hypothesis and objectives.

Chapter 2 discusses the three fundamental concepts for the present research, namely time series, data streams and concept drift. The main time series components and characteristics are presented. Several time series models, both statistical and intelligent methods, are described in that chapter. Following, the concept of data stream and the main challenges imposed by data streams to conventional machine learning methods are discussed. The main challenge of learning data streams, the phenomenon called concept drift, is presented. This chapter also includes a discussion of the main kinds of methods proposed in the literature to cope with concept drift: the active and passive adaptive learning methods.

Chapter 3 reviews some approaches proposed to handle concept drift in time series analysis. The proposed methods were divided into active and passive adaptive learning methods, according to its behavior. The active methods are classified into residual-based methods and methods that monitor time series raw observations directly. A discussion about the main issues and drawbacks of each class of approaches is presented in this

chapter. After the analysis of the reviewed methods, the requirements of an ideal adaptive learning system are defined.

Chapter 4 presents the main contribution of this thesis: a novel adaptive learning system for time series forecasting which employs an online explicit drift detection method. This method, called Feature Extraction and Weighting for Explicit Concept Drift Detection (FW-FEDD), monitors some statistical features of the time series in order to identify changes in the underlying data generation process. The proposed system implements a forecasting module composed by a set of individual models specialized in different time series concepts. In that chapter, FW-FEDD functioning and algorithms are described in detail.

Chapter 5 presents the experiments conducted in order to validate the research hypotheses formulated in this thesis. In that chapter the experimental setup, the data sets used, and figure of merits employed are explained and the main results obtained are presented and discussed.

Chapter 6 concludes the thesis with a summary of the main results and contributions of this research. This chapter also gives directions for further research.

Some of the material presented in this thesis were published in (or will be submitted to) the following papers:

1. **Cavalcante, R. C. and Oliveira, A. L. (2014)**. An Autonomous Trader Agent for the Stock Market Based on Online Sequential Extreme Learning Machine Ensemble. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN14), pp. 1424-1431.

2. **Cavalcante, R. C. and Oliveira, A. L. (2015)**. An Approach to Handle Concept Drift in Financial Time Series Based on Extreme Learning Machines and Explicit Drift Detection. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN15), pp. 1-8.

3. **Cavalcante, R. C., Brasileiro, R. C., Souza, V. L., Nobrega, J. P., and Oliveira, A. L. (2016)**. Computational Intelligence and Financial Markets: A Survey and Future Directions. Expert Systems with Applications, vol. 55, pp. 194-211.

4. **Cavalcante, R. C., Minku, L.L., and Oliveira, A. L. (2016)**. FEDD: Feature Extraction for Explicit Concept Drift Detection in Time Series. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN16), pp 740-747.

2 BACKGROUND

This chapter discusses some fundamental concepts for this thesis. This thesis has as main goal the construction of an adaptive learning system specifically designed for forecast time series that eventually present concept drift. A time series is a sequence of numeric observations collected over fixed sampling intervals (COWPERTWAIT; METCALFE, 2009). Several dynamic processes can be modeled as time series, such as stock price movements, monthly sales of a company, daily temperature of a city, exchange rates, among others.

In the last decades, several approaches have been proposed for time series analysis and forecasting. Two major classes of these approaches are the traditional statistical models and the computational intelligence approaches (WANG et al., 2011). Statistical models, such as linear regression, exponential smoothing and autoregressive integrated moving average (ARIMA), among others, assume that the time series under study is generated from a linear process (KUMAR; MURUGAN, 2013). Computational intelligence methods, such as expert systems, fuzzy systems and artificial neural networks (ANNs), have been applied with relative success in modeling and forecasting time series (LEE, 2009). These methods are data-driven, self-adaptive methods able to capture nonlinear behavior of time series without statistical assumptions about the data (LU; LEE; CHIU, 2009). Section 2.1 provides a formal definition of time series and explains the background literature on time series analysis.

Despite the fact that there is a vast literature on time series analysis, the majority of the existing approaches does not take into account that a time series is a special kind of data stream (CAVALCANTE et al., 2016). In data stream process, data continuously flow at high-speed and in potentially unexpected frequency. Dynamism is an inherent feature of data streams. This dynamism implies that patterns in the data stream may evolve over time, which introduces a big challenge to traditional batch learning algorithms, which is the ability to permanently maintain an accurate decision model even in the presence of changes in the data stream. These changes are known as concept drifts (GAMA, 2012). Section 2.2 provides a formal definition of data streams and concept drift and gives an introduction to the literature in this area.

Concept drifts have been widely studied in classification problems (DITZLER et al., 2015). The methods proposed for handling concept drifts can be divided into two main groups: (i) passive adaptive methods and (ii) active adaptive methods methods. Passive methods (KOLTER; MALOOF, 2007; SOARES; ARAÚJO, 2015) are those that update the decision model in regular intervals, independent of the occurrence of concept drifts.

Active methods (GAMA et al., 2004; ROSS et al., 2012), are those that monitor some statistics of the data stream in order to detect concept drifts. However, despite there is a considerably literature on concept drift in classification problems, just few approaches addressed this problem in time series analysis.

2.1 Time Series Analysis

A time series is an ordered sequence of observations, usually ordered by time (WEI, 2006), in which the intervals of observations can be equally spaced or not. A widely studied kind of time series is that represented by $S = \{x_1, ..., x_i, ...\}$, $x_i \in \mathbb{R}$ (BROCKWELL; DAVIS, 2002), in which each observation is measured at fixed, equally spaced time intervals, called sample intervals (COWPERTWAIT; METCALFE, 2009). A time series of length n can be represented by $\{x_t\} = \{x_1, x_2, ..., x_n\}$, or simply by $\{x_t\}$, for short. Data obtained from observations collected sequentially over time are encountered in several branches of engineering, science, sociology, economics, in the analysis and forecasting of data such as interest rates (OH; HAN, 2000), stock prices (CHEN, 2010), exchange rates (D'URSO et al., 2013), sales records (DOGANIS et al., 2006), crude oil prices (BAO et al., 2011), temperature (CHEN; HWANG, 2000), precipitation (PARTAL; KISI, 2007), wind speeds (SFETSOS, 2002), demographic studies (HYNDMAN; BOOTH; YASMEEN, 2013), among others.

Due to the vast number of applications, time series analysis has attracted attention of statisticians, engineers, politicians, traders and scientists. One of the main objectives of time series analysis is to understand the dynamic process that generates the time series observations, which may help in decision-making. According to Palit and Popovic (2006), some important tasks of time series analysis are: (i) definition, classification and description of time series, (ii) model building and (iii) forecasting of future values. In the next sections, we briefly describe these activities.

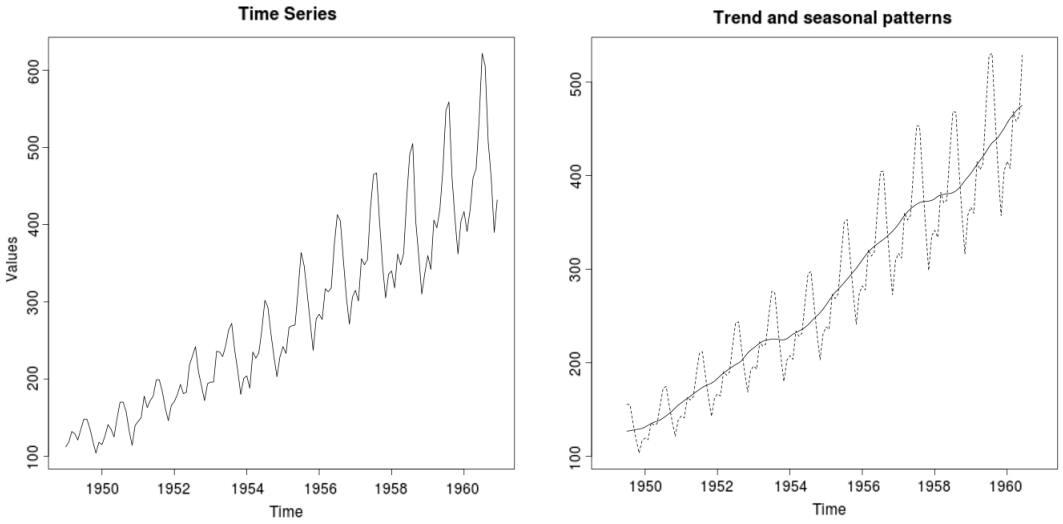
2.1.1 Time Series Definition, Classification and Description

Time series definition, classification and description are activities related to the identification of some important components of a time series or its main behavior. The main components of a time series are trend, seasonal pattern, and noise observations and outliers. A trend can be defined as a systematic change in a time series that does not appear to be periodic (COWPERTWAIT; METCALFE, 2009). It is a long-term feature which can be observed as an increase or decrease in the level of sequential data values. Trend analysis is an important task of time series analysis, since it gives an idea of global or local time series behaviors.

Seasonality is a repeating pattern that may happen within fixed our variable

periods (COWPERTWAIT; METCALFE, 2009). Many real-world time series are affected by seasonal patterns, such as climate time series and sales time series, in which a known pattern may repeat hourly, daily, weekly, monthly, yearly, etc. Seasonal analysis is another important task of time series analysis, since it helps in understanding the periodical behaviors of the underlying process. Figure 1 illustrates a time series and the identification of the trend and seasonal patterns. Trends and seasonal components are frequently interpreted as deterministic components of a time series and modeled with mathematical functions. However, in more complex environments, such as financial applications, time series frequently present a random or stochastic trend. In other environments, these components may not be visible or simply do not exist.

Figure 1 – Original time series (left) and its decomposition in trend and seasonality (right). In the figure on right, the continuous line represents the trend component and the dashed line represents the seasonal component.



Source: elaborated by the author.

Noise, outliers and missing-values are data observations often found in real-world time series. Noise and outliers are time series data observations that are very different from previous and subsequent observations of the time series. They can be considered anomalous observations. It is worth to mention that noise data in this context is a time series data observation and not the residual component resulting from the removing of trend and seasonality components of a time series. The main difference between noise observations and outliers is that noise are observations not genuinely generated by the underlying process, while outliers are genuine data points that may represent some peculiar situation in the time series processes. Missing-values are values not registered in the fixed time interval of the time series. Noise observations and missing-values are generally originated due to errors in the data collection process, such as failures in meters or to human error. For example, a daily temperature of a city may have a observation much higher than

the previous and subsequent days due to failure in the thermometer. Both noise and missing-values can difficult the time series analysis and they are typically disregarded by the analysts. Different from noise, outliers may be of particular interest and should not be excluded from the analysis.

According to the time series behavior, it can be classified as linear or non-linear, stationary or non-stationary (PALIT; POPOVIC, 2006). Linear process interpret all regular structure in a data set through linear correlations. This means that the intrinsic dynamics of the system are governed by the linear paradigm that small causes lead to small effects (KANTZ; SCHREIBER, 2004). All irregular behaviors or the system are attributed to random factors. So, linear time series are generated by a linear combination of the present and past values and a random factor and summarized by a linear model, such as a set of ordinary differential equations. Nonlinear time series, on the other hand, are those in which the time series observations result of a nonlinear combination of past values and random factors. In the case of a non-linear system analysis, no tools or methodology is universally applicable (VIDYASAGAR, 2002).

A time series is said to be stationary if its statistical properties are the same along the whole time series observation process. The underlying stochastic process is in a state of statistical equilibrium. In practical terms, the mean and variance of the time series observations are constant for $\{x_t\}$ and $\{x_{t-k}\}$, for any lag k. This kind of time series presents no trends and seasonal patterns. Non-stationary time series, on the other hand, are those with different statistical properties along the generation process. This kind of time series is more common in engineering, business and economic fields. It is important to discuss that there are some ways to estimate and remove certain non-stationarity aspects of a time series, such as the trends and seasonality, in order to make the analysis easier. The simplest way to soften the non-stationary effects of trends and seasonality in time series is by taking the differences between successive data values x_t and x_{t+1}.

An important characteristic present in some non-linear time series data generation process is the chaotic property. Chaos is a behavior of dynamical systems that are highly sensitive to initial conditions. Over time, chaotic systems can produce unpredictable, divergent and infinitely detailed and self-similar behavior (without ever actually repeating) due to that sensitivity (BOEING, 2016). The observations of a chaotic time series may be randomly repeated several times without maintaining any definite periodicity (PALIT; POPOVIC, 2006).

2.1.2 Time Series Modeling and Forecasting

A second important activity of time series analysis is the building of a model from the data. Model estimation can be used for providing a compact description of the data or to allow forecasting and simulations. Model building can be considered one of the

main time series analysis tasks, since a fitted model helps to understand the past and to predict the future, enabling managers or policy makers to make properly informed decisions (COWPERTWAIT; METCALFE, 2009). According to Shumway and Stoffer (2010), the one of the main objectives of time series analysis is to develop mathematical models able to describe the sample data. The fitted model provides a summary of the main characteristics of a time series.

Different from conventional data modeling, estimating a probability model for a time series is a particular task (SHUMWAY; STOFFER, 2010) due to the presence of a unique feature of time series, namely the serial correlation of adjacent points in time. The serial correlation restricts the applicability of many conventional statistical methods, which rely on the main assumption that the data observations are independent and identically distributed (CRYER; CHAN, 2008). So, statistical models used to describe time series need to incorporate the serial dependence in order to be adequate for forecasting future values. This serial dependence originates other characteristics that need to be addressed by the time series modeling process, such as the presence of trends and seasonality, stationary or non-stationary behaviors, linearity and non-linearity, the presence of outliers among others. Some of these features can be identified by looking carefully the time series plots. Others are identified by correlation analysis.

Due to its wide applicability, time series modeling has been investigated by both statisticians and machine learning researchers. Statisticians attempt to describe time series behavior through mathematical relations among time series components. Machine learning approaches attempt to automatically extract the relationship that define time series behaviors without prior statistical assumptions about the data. According to Shumway and Stoffer (2010), the traditional statistical time series models can be divided into two main groups: (i) time domain models, which attempt to describe the behaviors of the time series as a parametric function of current and past values; and (ii) frequency domain models, which assume that the primary characteristic of interest in time series is the periodic or sinusoidal variations. Despite the fact that time domain approaches are mostly used in engineering practice (PALIT; POPOVIC, 2006), these two approaches are not mutually exclusive and can be used in combination to improve the modeling.

A common time domain modeling approach used in literature consists in decomposing the time series into deterministic and non-deterministic components. This approach is based on quantifying these main components of the time series and the random variation or residuals (COWPERTWAIT; METCALFE, 2009). The deterministic components are the trend and seasonal patterns, which can be modeled with mathematical functions of time. When the trend and seasonal components are modeled, they can be removed from the original time series and which results in the random component. Then another model can be fitted to the residuals and the forecasting can be achieved by forecasting just the

residuals. The predicted value is then obtained by adding the forecasting residuals to the deterministic trend and seasonal models (BROCKWELL; DAVIS, 2002).

This scheme may not be properly used in time series which present no well defined trends and seasonal patterns, such as financial data, since in these cases it is unrealistic to assume any deterministic component. In these cases, the simple differencing may be useful to isolate the time series residuals, since this approach does not rely on a priori assumptions about the trend and seasonal patterns or even the assumption that these components remain fixed throughout the observation period (BROCKWELL; DAVIS, 2002).

Model building is a challenging task. Box and Jenkins (1976) proposed a general multi-step model-building strategy to guide the time series model estimation task. The three main steps of this process are: (i) model specification or identification, (ii) model fitting and (iii) model diagnostics. In the first step, a particular model needs to be chosen considering the characteristic of the time series to be modeled (the assumptions about the data). There are several families of statistical and computational intelligence models available in the literature. Examples of these main models are the simple zero-mean models, models which consider trend and seasonality, linear models, non-linear complex models, volatility models, frequency domain models and intelligent models. (PALIT; POPOVIC, 2006). Some of these models are described in more detail below in this section.

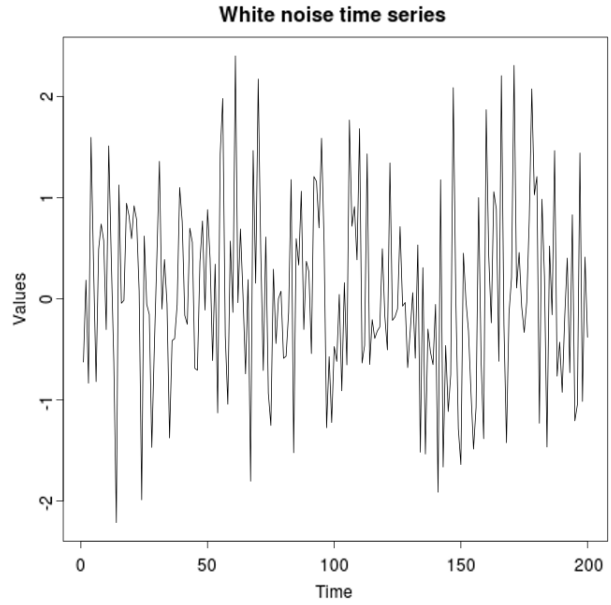
In order to help in identifying the most appropriate model, we can use some knowledge of the domain to be modeled or we can perform an analysis of the time series features through the time series plots, scatter plots, autocorrelation function, among other tools (COWPERTWAIT; METCALFE, 2009). An important requirement is that the chosen model should have few parameters to be adjusted. The adjustment of several parameters may result in an overfitting of the model to the historical data and consequently it would present poor extrapolation ability (BERGMEIR; BENÍTEZ, 2012). Methods with a high number of parameters are more sensitive to incorrect settings of the parameters, which may hinder the process of finding the true patterns in data. So, higher-order polynomials, for example, may give a good fit to the historic time series, but they should not have good accuracy in forecasting unseen time series observations.

The second and third steps of the model building are the model fitting and model diagnosis. After an appropriate model has been chosen, the model parameters are estimated. Model fitting consists in finding the best parameters to describe the observed data. Methods such as least squares and maximum likelihood are useful approaches for model fitting. The model diagnosis consists in assessing the quality of the estimated model. Some accuracy metrics should be used in order to measure this quality. If some inadequacies are found in the built method, the process should return to the first step. The process needs to restart until an adequate model is built, according to the objective criteria defined.

2.1.2.1 Traditional Statistical Models

The statistics literature proposed several mathematical time series models. Each family of statistical models has its particularities and it is designed to model specific time series behaviors. The simplest model for a time series is the white noise, which is designed to time series with no trend and seasonal components and in which the observations are independent and identically distributed (iid) random variables with zero mean and finite variance. Time series generated from uncorrelated variables can be described as a Gaussian white noise model $\{w_t\}$, in which $wt \sim N(0, \sigma_w^2)$. Figure 2 illustrates the behavior of a Gaussian white noise time series. If the stochastic behavior of a time series could be explained in terms of the white noise model, classical statistical methods would suffice to describe that time series (SHUMWAY; STOFFER, 2010). The Gaussian white noise model plays an important role in the analysis of residuals of a time series model (BROCKWELL; DAVIS, 2002).

Figure 2 – White noise time series.

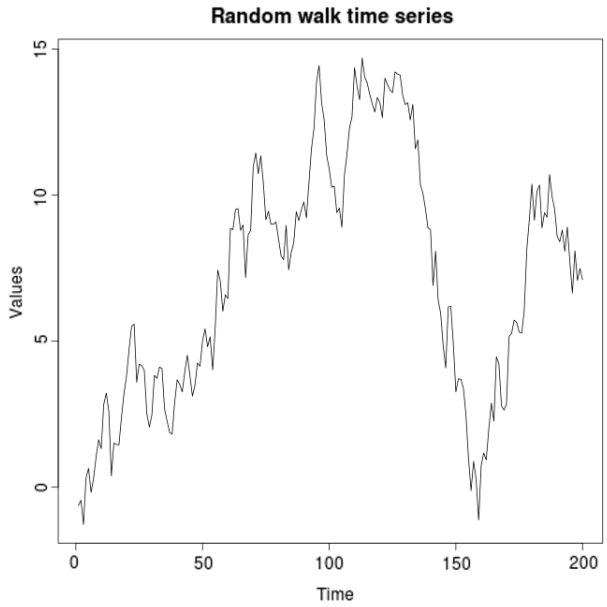


Source: elaborated by the author.

In the increasing scale of model complexity, the random walk model is obtained by cumulatively summing white noise observations. A random walk with zero mean is obtained by the eq. 2.1, where $\{w_t\}$ is a white noise. A random walk generally provides a good fit to data with stochastic trends and no defined seasonal patterns (COWPERTWAIT; METCALFE, 2009), such as financial time series. The random walk behavior can be observed in Figure 3.

$$x_t = x_{t-1} + w_t, \tag{2.1}$$

Figure 3 – Random walk time series.



Source: elaborated by the author.

Another important linear model is the autoregressive (AR) model. A time series $\{x_t\}$ can be modeled by an autoregressive process of order p, abbreviated as $AR(p)$, if its points can be defined as in eq. 2.2, where $\alpha 1, ..., \alpha_p$ are the model parameters and $\{w_t\}$ is a white noise (COWPERTWAIT; METCALFE, 2009). The random walk is an special kind of AR(1) process with $\alpha_1 = 1$. The shape of an AR(4) time series with parameters $\alpha = [0.9; -0.2; 0.8; -0.5]$ can be visualized in Figure 4.

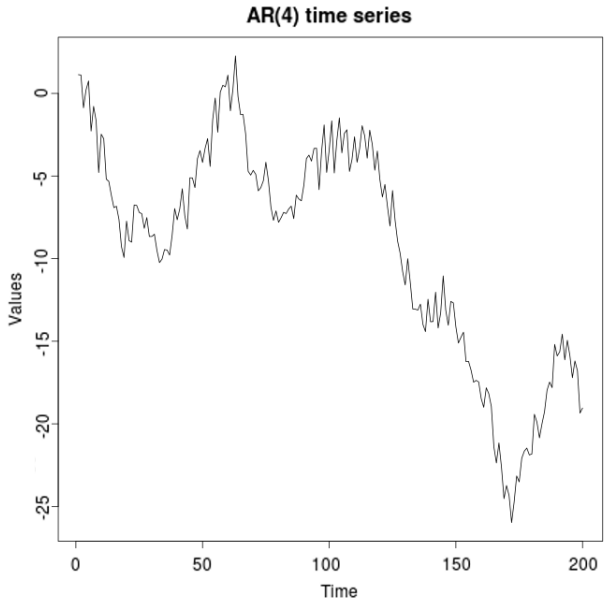
$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \alpha_p x_{t-p} + w_t \tag{2.2}$$

A linear model suitable for stationary time series is the moving average process. A time series $\{x_t\}$ is a moving average ($MA(q)$) process of order q if its observations can be obtained by a linear combination of the current white noise and the q most recent past white noise observations, as defined in eq. 2.3. The shape of a MA(3) time series with parameters $\beta = [0.8; 0.6; 0.4]$ can be visualized in Figure 5.

$$x_t = w_t + \beta_1 w_{t-1} + \beta_2 w_{t-2} + ... + \beta_q w_{t-q} \tag{2.3}$$

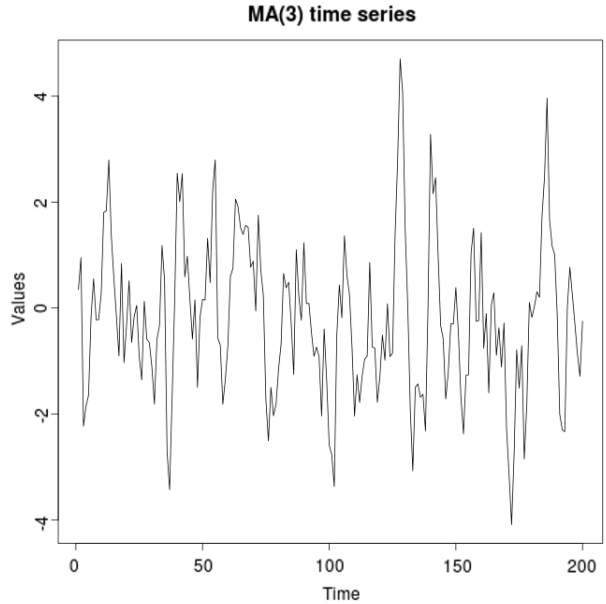
The autoregressive integrated moving average (ARIMA) model is a general class of linear models that integrates the autoregressive ($AR(p)$) and moving averages ($MA(q)$) process. A time series $\{x_t\}$ follows an $ARIMA(p, d, q)$ process if the dth differences of $\{x_t\}$ is an $ARMA(p, q)$ process. An ARIMA(1,1,1) time series can be visualized in Figure 6. This model is widely used in the literature to model and forecasting time series (WANG;

Figure 4 – AR(4) time series.



Source: elaborated by the author.

Figure 5 – MA(3) time series.

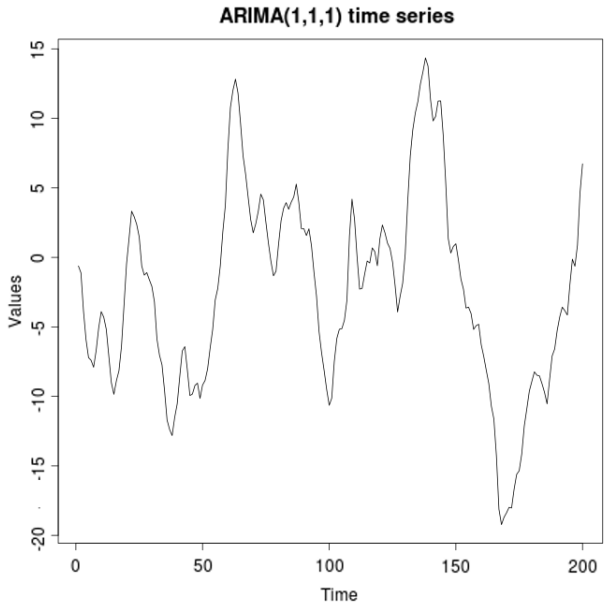


Source: elaborated by the author.

HUANG; WANG, 2012). An improvement of ARIMA for modeling time series with seasonal effects is the seasonal ARIMA (SARIMA). SARIMA model uses differencing at a lag equal to the number of seasons to remove additive seasonal effects.

A statistical time series model widely used in financial applications, for example, is the Generalized Autoregressive Conditional Heteroskedastic (GARCH) model. In some

Figure 6 – ARIMA(1,1,1) time series.



Source: elaborated by the author.

cases, it is more realistic to consider that the residuals resulting of fitting a model to a time series have no zero mean and constant variance (such as in ARIMA), but they are serially correlated and can be modeled by an AR process (GARCIA et al., 2005). In financial time series forecasting, GARCH is typically used to model volatility in financial prices. GARCH is an extension of the Autoregressive Conditional Heteroskedastic (ARCH) model, which models the conditional changes in variance (COWPERTWAIT; METCALFE, 2009). A series $\{\epsilon_t\}$ is first-order autoregressive conditional heteroskedastic, denoted $ARCH(1)$, if its observations are defined as in eq. 2.4, where $\{w_t\}$ is white noise with zero mean and unit variance and α_0 and α_1 are model parameters. The ARCH model should only be applied to a residual series $\{\epsilon_t\}$ that is uncorrelated and contains no trends or seasonal changes, such that resulting after fitting a satisfactory model. A series $\{\epsilon_t\}$ is $GARCH(p, q)$ if it can be defined as in eq. 2.5, where h_t is given by eq. 2.6.

$$\epsilon_t = w_t \sqrt{\alpha_0 + \alpha_1 \epsilon_{t-1}^2} \tag{2.4}$$

$$\epsilon_t = w_t \sqrt{h_t} \tag{2.5}$$

$$h_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{j=1}^{q} \beta_j h_{t-j} \tag{2.6}$$

In addition to the methods described herein, several other statistical models have been proposed in the literature. Despite the fact that traditional statistical time series

models have been widely used to model and forecast time series, there are some issues that prevent the use of these methods on a large number of real-world time series. One of them is the difficult to automatize the model specification task (in the first phase of the general model-building process). The task of choosing a suited model to a time series generally involves the analysis of plots and autocorrelations, making assumptions about the time series behaviors and even the need of some knowledge of the modeled field. These tasks sometimes difficult the automation of the time series modeling. Another important issue is that traditional models generally assume that the time series under study are generated from a linear process (KUMAR; MURUGAN, 2013). However, several time series, mainly in the financial and economical fields are essentially complex, highly noisy, dynamic, nonlinear, and chaotic in nature (SI; YIN, 2013). A third issue is the risk of overfitting of the model to the training data. Some complex time series requires high-order polynomials to better define the time series behaviors. However many parameters may imply in over adjustment of the model to the data, and consequently it may result in bad forecasting accuracy.

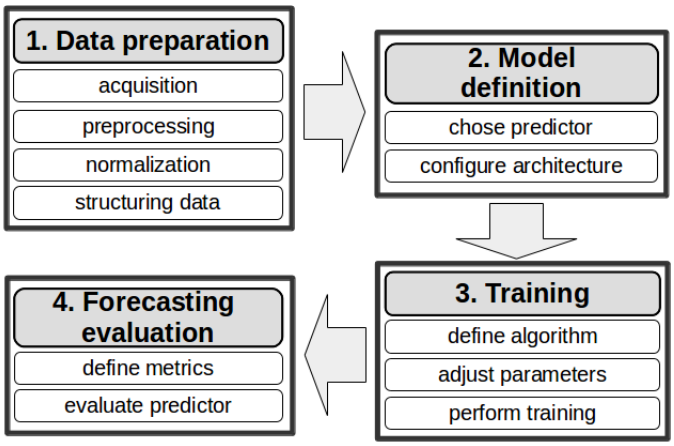
2.1.2.2 Computational Intelligence Models

In the last years, computational intelligence methods have been applied with relative success in modeling and forecasting time series (LEE, 2009). These techniques are more adaptive and flexible, since they are able to capture linear and nonlinear relationship between relevant factors with no prior knowledge about the input data (ATSALAKIS; VALAVANIS, 2009). Among these techniques, artificial neural networks (ANN), support vector machines (SVM) and hybrid methods, have been widely used in forecasting time series, since they are able to estimate time series behaviors without any prior statistical assumptions about the data (TAY; CAO, 2001; LU; LEE; CHIU, 2009). They generally exhibit high tolerance to imprecision and perform relatively well in noisy environments; they are numeric, data-driven, non-parametric and self-adaptive mechanisms; they require less historical data than traditional statistical models (CHENG; WEI, 2014). According to Liang et al. (2009), it is generally believed that non-parametric methods outperform the parametric methods in forecasting complex time series, such as financial time series, for example.

Palit and Popovic (2006) describes a systematic procedure, similar to that proposed by Box and Jenkins (1976), to forecast time series with computational intelligence methods. This procedure includes the following operational steps: (i) data preparation, (ii) algorithm definition, (iii) training and (iv) forecasting evaluation. Figure 7 summarizes these four steps of the machine learning modeling and forecasting methodology.

Data preparation is the first step of the time series modeling and forecasting process. Data preparation starts by acquiring the data to be learned and modeled. After the data

Figure 7 – Time series modeling and forecasting methodology using computational intelligent methods.



Source: elaborated by the author.

acquisition, the application of some preprocessing procedure on these data may be very useful, since they may be used to improve the accuracy of an intelligent predictor in several ways. Feature selection or extraction methods can be applied to a dataset in order to select the best representative features of the input data which reduce the input dimension and consequently minimize the training time. Preprocessing mechanisms can be used to filter irrelevant features and noise from input data or detect and correct outliers. Methods such as Principal Component Analysis (PCA) (TSAI; HSIAO, 2010; WANG; WANG, 2015), Independent Component Analysis (ICA) and its nonlinear variant (NICA) (LU; LEE; CHIU, 2009; DAI; WU; LU, 2012; KAO et al., 2013) and Wavelet Transform (WT) (GRANÉ; VEIGA, 2010; WANG et al., 2011) have been used in literature to filter noise from data, detect outliers and correct them as a preprocessing step of time series modeling.

The normalization task consists in scaling the attribute values for the use in training step of the machine learning method. Structuring data means to organize the available data for supervised learning and consists in activities such as dividing the data in training, validation and test, defining the past lags to be used as input and the future lags to be forecasted, among other activities.

Model definition consists in choosing the forecasting method to be used and configuring its architecture. Several machine learning models have been proposed in the literature to model and forecast time series, such as artificial neural networks, support vector regression, fuzzy systems, ensemble techniques, among several others (PALIT; POPOVIC, 2006). The architecture configuration is the definition of intrinsic model properties. For a multilayer perceptron (MLP), for example, in this phase the designer should define the number of input, output and hidden nodes, the number of hidden layers, the activation function to be used, among other architectural aspects.

In order to forecast future time series values, several artificial neural network models have been used in the last decades, such as the MLP trained with backpropagation (DHAR; MUKHERJEE; GHOSHAL, 2010; KAYAL, 2010; OLIVEIRA et al., 2011), Functional Link Artificial Neural Network (FLANN) (MAJHI; PANDA; SAHOO, 2009), Self-Organized MLP (SOMLP) (MAHDI; HUSSAIN; AL-JUMEILY, 2009), Radial Basis Function (RBF) Neural Network with Gaussian radial function, Legendre Neural Network (LIU; WANG, 2012), Bayesian Neura Networks (TICKNOR, 2013), among others.

Support Vector Machines (SVM) are computational intelligent learning methods that have been widely used as an alternative for ANN in pattern recognition tasks. SVM learning mechanism implements a risk function that considers the empirical error and a regularized term based on the structural risk minimization principle (CHEN, 2010). SVM constructs a hyperplane as the decision surface such that the margin of separation between points in different classes is maximized. Decisions are made based on support vectors which are data points that define the classification boundaries in the training set. In contrast to the empirical risk minimization principle, which tries to minimize the miss-classification error, the structural risk minimization principle implemented by SVM seeks to minimize an upper bound for the generalization error. According to Yuan (2013), the solution of SVM may be the global optimum, while conventional ANNs tend to produce just local optimum solution. Due to its strongly nonlinear approximation ability, SVM have been applied both in classification (SVC) and regression problems (SVR) (GUO-QIANG, 2011). In time series forecasting, several authors have been used SVM (CHEN, 2010; LUO; WU; YAN, 2010; BAO et al., 2011; CHAO; LI-LI; TING-TING, 2012) as forecasting model.

Several researchers proposed the use of hybrid mechanisms, in order to combine individual solutions of different intelligent algorithms to forecast time series (TRESP, 2001). The combination of individual approaches may allow the reduction of uncertainties in parameter adjustment and the stochasticity in training (KOURENTZES; BARROW; CRONE, 2014). By facing the forecasting problem with dividing and conquer approach, two or more intelligent approaches may contribute with their individual knowledge to improve the forecasting performance of the whole group. Liang et al. (2009) proposed a cascade hybrid method in which a statistical model is fitted to time series trend and a computational intelligence method is used to learn the residuals of the statistical model. (WU; SHAHIDEHPOUR, 2010) proposed a hybrid mechanism that combines an Autoregressive Moving Average with Exogenous Variables (ARMAX) model, GARCH and Adaptive Wavelet Neural Network (AWNN). (ZHU; WEI, 2013) proposed a hybrid approach in which an ARIMA and Least Squared Support Vector Machine (LSSVM) are used to model linear and a nonlinear components of a time series. Nayak, Mishra and Rath (2015) presented a hybrid model that integrates SVM with K-Nearest Neighbor (KNN) approach.

Several researchers have used a different strategy of combining individual expert knowledge to model and forecasting time series, called ensemble learning. The motivation to use ensembles is to combine individual learning algorithms instead of using a single algorithm to improve the modeling of an environment. There are several strategies to build ensemble methods, such as training the individual models with the same data in order to reduce training problems, or training individual models with instances located in different positions of the feature space. Ensembles can be constructed with algorithms of the same type or with different learning algorithms. Neto et al. (2010) investigated the use of ensemble of MLPs and RBF networks to forecast time series. Cavalcante and Oliveira (2014) compared the use of Extreme Learning Machine (ELM) and Online-Sequential Extreme Learning Machine (OS-ELM) ensembles in this context. Ballings et al. (2015) provided a comparative study on evaluation performance of ensemble methods, namely the random forest, adaboost and kernel factory, against single classifier models, namely ANN, logistic regression, SVM and KNN.

After choosing the intelligent model, we need to train it with the available time series data. The training phase consists in selecting the training algorithm, adjusting training parameters and executing the training procedure in order to estimate a model of the learned data. The training algorithm defines a set of rules to fit the chosen model to the training data. Since every training algorithm has a set of parameters to be set, the designer should define how to adjust these parameters in order to improve the forecasting performance. When using a MLP trained with backpropagation, for example, the designer needs to define the learning rate, the number of training epochs, the stopping criteria for the training, among other aspects. After these definitions, the training needs to be performed in order to fit the data.

The parameters to be set for a computational intelligence algorithm pose an important challenge to the use of these methods, since some of these parameters can highly influence the forecasting accuracy. In order to avoid this issue, some researchers have investigated the use of optimization algorithms to find the best parameter setting for a machine learning algorithm and improve forecasting accuracy, such as Artificial Bee Colony (ABC) (BRASILEIRO et al., 2013), Genetic Algorithms (GA) (HUANG, 2012; EVANS; PAPPAS; XHAFA, 2013), Particle Swarm Optimization (PSO) (ABDUAL-SALAM; ABDUL-KADER; ABDEL-WAHED, 2010; PULIDO; MELIN; CASTILLO, 2014), among others.

The last phase (evaluation) consists in defining the evaluation metrics and in measuring the accuracy of the results obtained by running the trained method on the test data. Forecasting evaluation can be performed by using machine learning regression error measures, such as Mean Absolute Error (MAE) (eq 2.7), Mean Absolute Percentage Error (MAPE) (eq 2.8) and Root Mean Square Error (RMSE) (eq 2.9), where y_i is the actual

value, $\hat{y}_i$ is the forecasting value and n is the number of predictions made (BERGMEIR; BENÍTEZ, 2012). These metrics are the most commonly used as figures of merit in time series forecasting studies in literature.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2.7}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{2.8}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{2.9}$$

2.2 Data Streams

The machine learning field has been dedicated to develop learning algorithms able to learn about specific problems with experience and helping people in decision-making (MITCHELL et al., 1997). Machine learning methods try to automatically acquire knowledge of a specific domain aiming at improving the solution of automatic tasks by means of experience (GAMA, 2012). The machine learning literature may be divided some groups of approaches, such supervised, unsupervised and reinforcement learning approaches (DUDA; HART; STORK, 2012), through different paradigms, such as symbolic, statistical, based on examples, evolutionary, among others. In the last decades, the supervised machine learning literature has concentrated in batch learning algorithms to model small persistent data sets (GAMA, 2010). In this learning scheme, the whole training data is available to the algorithm, and the training process can consult these data in several steps. The main assumption of batch learning algorithms is that the modeled data is independent and identically distributed and generated from an immutable distribution (GAMA; SEBASTIÃO; RODRIGUES, 2009). So, the modeling task consists in extracting some relationships about instance attributes (or descriptors) and outputs of the data set instances. The model learned is then used for solving future unseen cases of the same problem.

With the recent advances in information systems and technologies, there has been an increase in the amount of information continuously collected by these systems. In many applications, such as credit card activity analysis, financial markets, sensors network, web mining, telecommunications, network monitoring, the volume of collected data is so large that it imposes some restriction to process and store these data (AGGARWAL, 2007). In these information systems, data continuously flow possibly at high-speed and in potentially unexpected frequency. A representative example of application that handle continuous streaming of data is the detection of fraudulent credit card activities. Several

credit card operations occur simultaneously and with no regularity in the frequency of the transactions. The number of credit transactions increases indefinitely in a continuous flow and at a high-speed.

In these real-world applications, the huge volume of data makes impossible the storage of all historic data, and consequently, it makes impossible to process the data efficiently in multiple passes (AGGARWAL, 2007). Data streams introduce several challenges to machine learning methods. Besides the large volume of data, frequently data streams are dynamic and time-changing. So, these data frequently present patterns that evolve over time, which prevents the direct use of batch learning algorithms. Due to the inherent dynamics, the main assumption of stationarity may not hold for several industrial applications which deals with streaming of continuous data. As a result, a static decision model is a problem in data streams.

We can illustrate some issues of applying conventional batch learning in dynamic data streams with the modeling of a classical problem of machine learning, which is the spam filtering. Spam is an unsolicited commercial communication which causes inconveniences to email users. Due to the importance of this problem, several machine learning techniques have been applied to model the problem of deciding which is a legitimate or a spam message. In a batch learning scheme, a set of pre-classified messages is presented to a machine learning algorithm, which attempts to extract a model of the explicit and implicit the relationship among message features and the label of the message. In this problem, there is a high cost associated to false positives, which is to classify a legitimate message as a spam and block the message (FDEZ-RIVEROLA et al., 2007). However this environment is dynamic, and the true label of a message can change due to changes in the spammers activities, or even due to changes in the users interests. In case of environment changes, batch learning algorithms will commit several classification mistakes.

Gama (2010) listed several differences between processing data streams and conventional data sets. Data streams are better modeled as transient data rather than persistent tables, since the data elements arrive online. Due to the high-frequency in which data arrives, it is impossible to store all data in the disk. Besides, it is not possible to control or even known a priori the order in which data elements arrive in the system. Data streams are potentially unbound in size, or in other words, they are open-ended data sets.

These characteristics of data streams demand a different paradigm of effective mining algorithms. One of the main challenges is that the underlying regularities in data may evolve over time rather than being stable. So, the data cannot be considered identically distributed along the data stream mining process (GAMA; SEBASTIÃO; RODRIGUES, 2009). Robust data mining methods applied to data streams must have mechanisms to monitoring incoming data, detect changes and incorporate these changes in the learned model. These changes are referred in the literatures as concept drifts (GAMA, 2012).

Concept drift causes a degradation in predictive accuracy of conventional machine learning models (GAMA et al., 2004).

The high speed nature of data streams requires faster algorithms to process this kind of data. These algorithms should be able to take decisions and eventually adapt themselves as fast as the rate of data arrival (GABER; ZASLAVSKY; KRISHNASWAMY, 2007). This imposes a restriction that data cannot be scanned more than once. Once an element has been processed, it should be discarded or archived, and not retrieved easily. As a result, online and adaptive mining algorithms are required to solve both concept drift and high speed issues. However, these objectives are conflicting, since more adaptation implies in better accuracy, but requires more processing time. A tradeoff between accuracy and efficiency in terms of processing and memory should be pursued.

Learning from evolving data streams has become a hot topic in the last years. The main machine learning applications, namely classification, regression, clustering, frequent pattern mining, among others, need to address space, learning time and generalization restrictions imposed by data streams. However, despite the increasing interest in adaptive systems designed for data streams, they are still rarely deployed in real-world applications in practice (ZLIOBAITE et al., 2012). Zliobaite et al. (2012) summarize six main challenges in building practical adaptive learning systems: (i) making adaptive systems scalable, (ii) dealing with realistic data, (iii) improving usability and trust, (iv) integrating expert knowledge, (v) taking into account various applications needs, and (vi) moving from adaptive algorithms towards adaptive tools.

Among these challenges, the improvement of usability and trust can be one of the main factors that still prevents wide use of adaptive learning systems in industrial applications. Most of the proposed adaptive learning methods have a huge number of user adjustable parameters. The tunning of these parameters is often a difficult task, which affects negatively the usability of the model. It also decreases the trust of users on the system, since the model may be updated thousands of times by means of retraining and, at every retraining, the parameters need to be tuned again. Since these methods perform in an online mode, the parameters should be optimized automatically. Industrial applications strongly require models with a few adjustable parameters as possible (ZLIOBAITE et al., 2012). A possible solution is the design of self-adjusting parameters, but it is not an easy and computationally cheap task.

2.3 Concept Drift

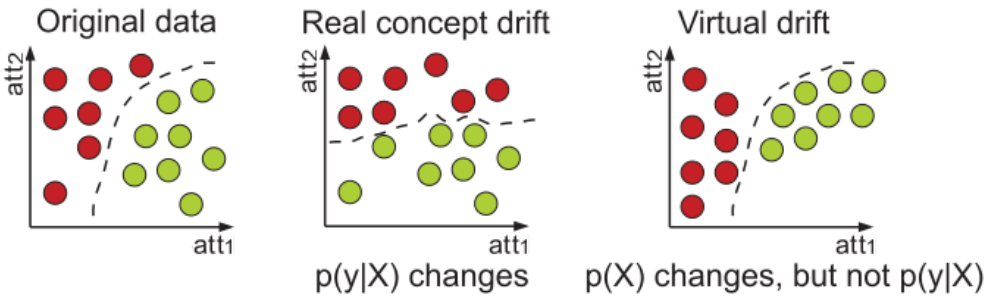
Concept drift is the problem associated with supervised learning in which the relation between input attributes and target variable changes over time (GAMA et al., 2014). Traditional machine learning and data mining have not to face this problem, since

they have been performed in an offline way and usually to model relative small data sets. In this scenario, model building consists in a process of inducing a model by several passes on the training data subset, and this model could be used to estimating outputs. The training phase, generally using a smaller testing set, is then performed to assess the model accuracy.

However, the traditional mining approach may not be effective when applied to model high-frequency, open-ended, continuous data streams. The main assumption of stationarity in data can not be sustained due to dynamism inherent to data streams. The changes in data can be due to seasonality or periodicity effects, changes in user behavior or psychology, climate changes, among others (DITZLER et al., 2015). In these environments, it is required that the learning algorithms be able to change the learned model in order incorporate the new states of the data. More than just adapt, learning algorithms need to be able to track and analyze the nature of these changes (AGGARWAL, 2007), since they can be very useful for the users.

According to what actually changes in the data stream, the literature makes a distinction among two kinds of concept drifts (GAMA et al., 2014): (i) real concept drift, which refers to changes in the conditional distribution of the target variables given the input variables ($p(y|\mathbf{X})$), and (ii) virtual concept drift, which refers to changes in the incoming data distribution ($p(\mathbf{X})$) without affect the conditional distribution ($p(y|\mathbf{X})$). Figure 8 illustrates the differences between real and virtual concept drifts in the classification context for a two-dimensional problem. It is good to mention that, in practice, changes in $p(\mathbf{X})$ may be associated to changes in $p(y|\mathbf{X})$. This may be frequently the case in classification, and can be particularly the case in time series forecasting, given that $\mathbf{X}$ is computed based on previous y values.

Figure 8 – Real and virtual concept drifts.



Source: adapted from Gama et al. (2014).

According to the speed at which the change occurs, drifts can fall into two categories: (i) abrupt concept drifts, in which the incoming data distribution changes suddenly in a few time steps, and (ii) gradual concept drift, in which the changes are slow and take more time to be complete. Abrupt concept drifts may be easier to be detected than gradual

drifts. Several researchers, such as Goncalves and Barros (2013) and Alippi, Boracchi and Roveri (2013b), investigated another class of concept drift: the recurring concept drift, in which concepts tend to repeat along the time. In these cases, some memory of past drifts may be useful to improve the system performance. In recurring concept drifts, the speed of change can be abrupt or gradual.

According to Minku, White and Yao (2010), these denominations are not sufficient to classify the kinds of drifts that can occur in data streams, since this classification results in heterogeneous categories of drifts. Authors claim that when analyzing drifts in isolation, it can be observed that different drifts can cause different amount of changes, and these changes can take more or less time to be completed. To analyze drifts in isolation, they proposed a categorization of drifts based on two criteria, namely (i) severity, which is the amount of changes that a new concept causes, and (ii) speed, which is the inverse of the time taken for a new concept to completely replace the old one. A further division is made in the level of class or feature changes. Class severity reflects the changes in the prior ($p(y)$) or posterior probabilities ($p(y|\mathbf{X})$). Feature severity criterion covers the changes in the unconditional ($p(\mathbf{X})$) and class-conditional ($p(\mathbf{x}|y)$) probabilities. In terms of class severity, a drift is considered severe when all examples in the input space change the target class when the drift occurs. On the other hand, if part of the input space has the same target class in the old and new concepts, then the drift is said intersected. In terms of feature severity, drifts are said severe when the probabilities associated to the whole input space are modified and intersected if part of the input space maintains the same probability.

Minku, White and Yao (2010) also proposed a categorization when analyzing drift sequences by means of three criteria: (i) predictability, (ii) frequency and (iii) recurrence. According to the predictability criterion, drifts can be divided into random and predictable. Predictable drifts are those in which the sequence of changes can be learned to predict future changes. The frequency defines how often concept drifts happen in data. According to the frequency criterion, drifts can be divided into periodic and non-periodic. Concept drifts that take place at every t time steps are considered periodic. Otherwise they are non-periodic. The recurrence criterion, as discussed previously, divides drifts into recurrent, if they returns to previous concepts, and non-recurrent, otherwise. Recurrent drifts may present cyclic (periodic) or unordered behavior.

There are several methods proposed to cope with concept drift in data streams. Recently, Ditzler et al. (2015) proposed a valuable discussion about the existing methods. According to these authors, the adaptive mechanisms to cope with concept drift can be divided into two main groups: (i) passive adaptive approaches, which are adaptive learning systems that continuously update the model over time, without requiring an explicit detection of change, and (ii) active adaptive approaches, which are learning systems which rely on an explicit detection of the change in data distribution to activate an adaptation

to the change. In the rest of this chapter we discuss these two main categories of methods.

2.3.1 Passive Adaptive Methods

Passive approaches, also called implicit or blind methods (KOLTER; MALOOF, 2007; FDEZ-RIVEROLA et al., 2007; TSYMBAL et al., 2008; SOARES; ARAÚJO, 2015; MIRZA; LIN; LIU, 2015) are those that update the decision model at regular intervals, independent of the occurrence of concept drifts. Two major passive learning approaches are (i) the online and incremental methods, and (ii) the dynamic ensemble approaches. The methods which rely on an online or incremental learning approach are those in which, for each new example x_t arriving from the data stream, the decision model is used to predict the class $\hat{y}_t$ of the example and, when the true label y_t of that example is available, the method updates the learning model to incorporate the new knowledge in case of error (GAMA et al., 2014).

Dynamic ensemble learning has been used to handle concept drift in data streams with relative success (DITZLER et al., 2015). Ensembles are methods that employ several individual decision models to produce a global prediction. Ensemble methods handle concept drift by dynamically adding and removing individual models according to their individual prediction performance. The dynamics of ensembles implements two important features for handling concept drift: the forgetting mechanism and the incremental learning. The removing of individual decision models which have bad performance implements the forgetting mechanism, since the disposal of models means forgetting older information, probably from a previous concept. The adding of new individual models implements the incremental learning, since new decision models are trained with the more recent examples from the data stream.

An important challenge that needs to be addressed in building adaptive methods which rely on online learning or dynamic ensembles is the definition of which recent examples from the data stream should be used for updating or training new decision models. Since these methods implement no explicit drift detection mechanisms, they cannot precise the exact instant in which a concept drift happened. So, the task of defining which examples should be used to model the new concept is critical to these approaches. In the literature, some methods have been handling this issue by means of instance weighting and instance selection strategies (FDEZ-RIVEROLA et al., 2007). The instance weighting mechanism consists in assigning weights to the examples according to their age or utility for defining the new concept (SOARES; ARAÚJO, 2015). These weights are used internally by the adaptive learning system in order to update the learning model. In ensemble methods, for example, the instance weights may be used to define the performance of the individual models and for deciding which of them will be removed from the ensemble. Models that commit mistakes to classify instances with higher weights are more likely to be excluded

from the ensemble.

Instance selection consists in defining the subset of examples that should be used for retraining. This is a very simple adaptation scheme to work in non-stationary environments (ALIPPI; BORACCHI; ROVERI, 2011). The simplest instance selection scheme is the use of a sliding window of fixed size which slides over the data stream instances. The instances in the window are used to retrain or update the learned model and older data are discarded as the window slides (ZHANG et al., 2009). Since the sliding window contains the more recent data arrived from the data stream, it is expected that concept drift would be handled implicitly by successively retraining the decision model with the more recent data. The main issue of using sliding windows of fixed size is the definition of the window size. If the window size is large, old instances, perhaps from a previous concept, may be included in the window, which can hinder the learning process. Besides, more memory is needed to store the instances in the window. On the other hand, if the window size is small, the instances in the window may be insufficient to define the new context (GU; TAN; HE, 2013). Fdez-Riverola et al. (2007) investigated how to handle concept drift in the spam filtering problem by means of an instance selection strategy based on the sliding window scheme.

There are two important issues of the passive adaptive approaches that need to be discussed. The first of them is that passive approaches handle concept drift in an implicit way, and the user is unaware of changes in the data stream. In some applications, it is useful for the user to know when changes occurs. Explicitly detecting and informing the users about concept drifts may increase the trust in the adaptive system. Users could make certain decisions based on the knowledge that a change has been occurred. The second important issue is that passive methods handle concept drift by successive adaptations in the learning model. Excessive adaptation may represent a waste of resources and provide only incremental insignificant benefits towards the prediction performance (ZLIOBAITĖ; BUDKA; STAHL, 2015). Some applications, such as the forecasting of high-frequency trading markets (HFT) (CHORDIA et al., 2013) and prediction of conditional branch outcomes in microprocessors (FERN; GIVAN, 2003), require efficiency in decisions and this implies in efficient adaptive learning methods. In other applications that have no frequent changes, the successive adaptation represents a constant computational cost, but does not result in improvements in prediction accuracy.

2.3.2 Active Adaptive Methods

Active adaptive learning methods (GAMA et al., 2004; ALIPPI; BORACCHI; ROVERI, 2011; ROSS et al., 2012; GONCALVES; BARROS, 2013) are the adaptive learning systems that rely on a explicit concept drift detection mechanism, reacting to changes by updating the existing model or building a new one. At this point it is important

to distinguish the term active adaptive learning from the term active learning. Active learning is a set of algorithms and heuristics that select which data points are used in the training set in order to reduce the number of data points used in training and/or to improve the generalization performance of a learning method (ROY; MCCALLUM, 2001; BACH, 2006). Active adaptive learning is a detect-to-react learning scheme used to cope with concept drifts in data stream learning.

An advantage of detecting-and-reacting approaches is that this adaptive learning scheme is transparent, by informing the user about the occurrence of concept drifts. The learned model may be updated just after a drift detection, which can save computational resources of the adaptive model. Two main explicit drift detection approaches are those that monitor the residuals of a fitted model and those that monitor features extracted from the data generating process. Methods that monitor the residuals of a fitted model rely on the assumption that when the distribution of incoming instances to be predicted is immutable, it is expected that the residuals of the fitted model are constant or decrease as the number of predicted instances increases (GAMA et al., 2004). On the other hand, when a change in the data distribution occurs, which configures a concept drift, it is expected that the residual level increases. So, in order to monitor changes in the residual level, these methods implement statistical tests to identify statistically significant changes before firing a concept drift. When a concept drift is identified, a drift signal is triggered, which indicates that the decision model needs to be updated to reflect the new concept.

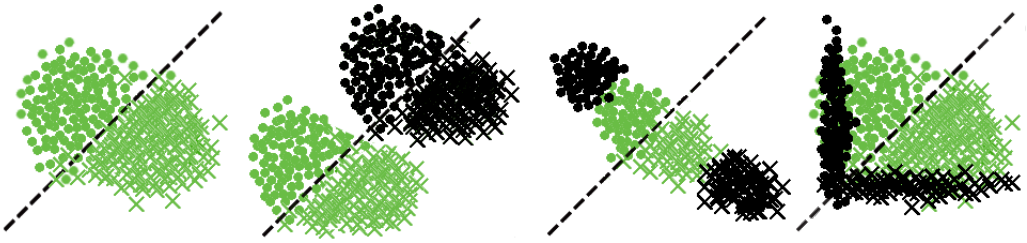
Examples of explicit concept drift detection mechanisms based on monitoring classification errors are the Drift Detection Mechanism (DDM) (GAMA et al., 2004), the Early Drift Detection Mechanism (EDDM), the Exponentially Weighted Moving Average for Concept Drift Detection (ECDD) (ROSS et al., 2012) and the Page-Hinkley test (PHt) (PAGE, 1954). The main assumption of these methods is that a change in the distribution of the online error-rate may indicate a change in the incoming data distribution. The main difference between these methods is the way of identifying a concept drift. DDM compares the mean and standard deviation of the errors with the minimum mean and standard deviation up to the considered instances. EDDM monitors the distance between two consecutive errors. ECDD compares the exponentially weighted moving average (EWMA) of the error with the simple average of the error. PHt monitors the cumulative sum of the error means. An important work, proposed by Goncalves et al. (2014), compared eight of the most cited error-based explicit drift detection methods, namely the DDM, EDDM, PHt, ECDD, Adaptive Windowing (ADWIN), Paired Learners (PL), Statistical Test of Equal Proportions (STEPD) and Degree of Drift (DOF) in the context of data stream classification. DDM was the best method in terms of accuracy.

Minku and Yao (2012) proposed an ensemble method for handling drift detection which employs a mechanism for explicit drift detection, instead of handling changes

implicitly. The main idea of the proposed method, called Diversity for Dealing with Drifts (DDD), is to maintain ensembles with different levels of diversity in order to achieve robustness to different types of concept drift. DDD is also based on the idea that drifts should be detected as soon as possible in order to obtain fast adaptation. As early drift detection can usually be achieved at the cost of an increased number of false alarms, DDD was also designed to achieve robustness to false alarms by keeping different ensembles. The drift detection method used by DDD is the EDDM drift test, even though other drift detection methods are also applicable. Experimental results performed on that paper showed that DDD presented good drift accuracy in comparison to other explicit and ensemble methods.

Despite the fact that explicit drift detection method based on residuals of a fitted model have been extensively used in the literature, they present some drawbacks. The main issue of these approaches is that the residual levels may not properly reflect concept drifts. As discussed by Kuncheva and Faithfull (2014), in some cases, the same model may have a good accuracy even when the data distribution changes. This phenomenon is illustrated in Figure 9. In other cases, if a poor training process is used to build the decision model, the accuracy may remain bad even in the presence of changes in data distribution. In these cases (Figure 10) the concept drift will not be detected either.

Figure 9 – The data changes but the classification accuracy remains good. The original data is on the left. In the other cases, the points in black appeared after a concept drift. The error-rate remains the same in all cases and does not indicate concept drift.

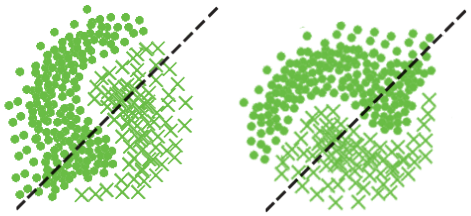


Source: adapted from Kuncheva and Faithfull (2014).

Explicit drift detection methods which monitor some features extracted from the data stream are expected to provide better drift detection accuracy in theory. This kind of explicit drift detection may provide a better understanding of how concepts evolve over time than those based on monitoring the prediction error. Since they monitor data distribution features, the drift detection process relies just in the statistical test that assess the evolving of the data distribution and in the feature set used to describe the data.

An important distinction among explicit drift detection approaches based on monitoring extracted features is related to the delay of detection presented by these methods. The majority of the proposed drift detection methods that monitor data distribution or

Figure 10 – The data changes but the classification accuracy remains bad. The original data is on the left. The error-rate remains the same in all cases and does not indicate concept drift.



Source: elaborated by the author.

statistical behaviors perform a retrospective drift detection (GOMBAY; SERBAN, 2009). In this kind of explicit detection, the whole data stream needs to be available and the drift identification process consists in search for change-points or structural breaks in the data stream behavior. These methods are very useful in applications which require some explanation about data generation process behaviors, such as in climate time series applications (WERNER et al., 2015), interest rates analysis (OH; HAN, 2000), among others.

Very few methods have been designed to detect concept drifts based on features in real-time. For example, Alippi, Boracchi and Roveri (2011) proposed an adaptive classifier which implements an explicit drift detection test based on the Intersection of Confidence Intervals (ICI) rule. In this approach, an initial data subset (training set), assumed to be stable, is used to model the initial concept. Then, the stationarity of the data generator mechanism is monitored online by means of a set of features which are independent and identically distributed and follows a Gaussian distribution. A change in this data stream is detected by inspecting each feature separately using the ICI rule. In stable conditions, it is expected that feature values are also stable. The ICI rule is used to assess variations in the expected value of each feature. During the operational phase, the test computes for each instant the zeroth-order polynomial fit for each feature, and the ICI rule is used to verify if the intersection of all confidence intervals for a feature differs from the empty set. When this intersection is an empty set, the ICI rule identifies a non-stationarity in the data stream. When a drift is detected the classifier is reconfigured to the new process state.

In a subsequent work, Alippi, Boracchi and Roveri (2013b) extended the proposed explicit drift detection method to cope with recurrent concepts. The main idea is to create representations for the concepts identified during the processing of the data stream. Once a concept drift is identified, the concept representation is created and compared with the stored representations of concepts. When a new concept is similar to a previously seen concept, its supervised samples are used to reconfigure the classifier, in order to

improve the classification of the new concept. The proposed classifier detects concept drift by means of two concept drift tests monitoring the distribution of input data and the classification error. According to the authors, this method reduces false positive detections. The ICI-based concept drift test is used to detect concept drifts in the monitored variables. A main issue of this approach is that the representation of the already seen concepts is based on storing the supervised examples and the features that characterize the concept, which requires an amount of extra memory.

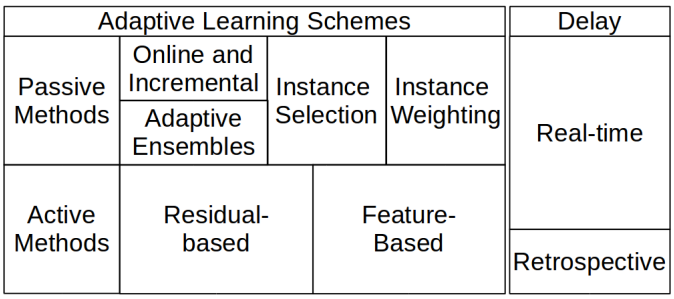
Goncalves and Barros (2013) combines both feature extraction and residual monitoring to handle recurring concept drifts. The proposed method stores a collection of individual classification models trained to different concepts and the data samples used to build them. Two error-based explicit drift detection tests, namely DDM and EDDM, were used to identify concept drifts. When a concept drift is detected, the proposed method compares the current data distribution to the several stored data distribution samples in order to verify whether the new concept was previously seen or is a new context. A multivariate non-parametric statistical test is used to check the similarity of two data distributions. In case of the current concept has already been seen above, the classifier built for that concept is recovered from the pool of stored classifiers to be used. On the other hand, if the data configures a new context, the incoming data is used to build a new classifier. Several experiments were made to compare the proposed framework with some drift mechanisms proposed in literature and results showed an equivalent performance of the proposed approach.

Figure 11 relates the different types of methods for dealing with concept drift proposed so far in the literature. As discussed before, the adaptive learning schemes can be divided into passive and active methods. The passive approaches are those that relies in an online or incremental learning or in an adaptive ensemble scheme. These passive approaches generally implement instance selection or instance weighting to define how retrain or add a new model to the system. The active methods, on the other hand, implement an explicit drift detection scheme that monitors the residuals of a fitted model or some features of the incoming data distribution. In relation to the delay in handling the changes, the methods can be divided into real-time or retrospective approaches. All the passive methods operate in real-time, since they adapt the learned model continuously. The active methods may operate in real-time or in a retrospective manner.

2.4 Summary

In this chapter we discuss the three main concepts which represent the fundamentals for this thesis. We started by discussing about time series theory, models and applications. A time series is a sequence of observations, typically collected over fixed sampling intervals.

Figure 11 – Categorization of the concept drift handling methods.



Source: elaborated by the author.

Several dynamic processes can be modeled as time series, such as stock price movements, monthly sales of a company, the temperature of a city, exchange rates, among others. Time series analysis and forecasting can be considered two of the main challenges in the computational intelligence literature. In the last decades, several approaches have been proposed for time series analysis. Two major classes of these approaches are the traditional statistical models and the computational intelligence approaches. Many statistical models generally assume that the time series under study is generated from a linear process. Computational intelligence techniques, on the other hand, are data-driven, self-adaptive methods able to capture nonlinear behavior of time series without any *a priori* statistical assumption about the data.

This chapter has also discussed that despite the fact that there is a vast literature on time series analysis, the majority of the existing approaches does not take into account that a time series is a special kind of data stream. So, this chapter briefly described the concept of data streams, which is a set of data observations that arrive sequentially item by item. Dynamism is an inherent feature of data streams. This dynamism implies that patterns in a data stream may evolve over time and introduces a big challenge to traditional batch learning algorithms, which is the ability to permanently maintain an accurate decision model even in the presence of changes in the data stream.

This chapter has also discussed that most of the approaches designed to time series analysis assume that there is no occurrence of concept drifts in the data generation process. These methods are based on the main assumption that time series concepts are stable in such a way that the time series observations follow a fixed and immutable probability distribution. This assumption, however, may not hold for several industrial time series applications. For example, the time series of the sales of a product may change its behavior due to changes in government regulations or advertising campaigns. The time series of stock prices of a company may change its behavior due to changes in political and economical factors or due to changes in the investors psychology or expectations. In the next chapter, some approaches that investigate the concept drift problem in time series

forecasting are reported and discussed.

3 CONCEPT DRIFT IN TIME SERIES

Despite the fact that there is a considerable number of studies which investigate the concept drift problem in data streams, little attention has been given to the problem of concept drift in time series. What makes time series data unique and prevents the direct application of conventional adaptive learners to time series is the temporal correlation of the data observations. In this chapter, we discuss some of the main articles that investigate this problem in time series data streams. Following the classification proposed by Ditzler et al. (2015), the existing approaches to handle concept drift in time series are divided into active and passive adaptive learning methods.

3.1 Passive Adaptive Learning Methods

The passive adaptive learning methods assume the fact that the underlying data generation process is not stable, but evolve over time. To accommodate changes in the data over time, these methods perform a continuous adaptation in the learned models, in order to keep them up-to-date and avoid loss in forecasting accuracy. The two main categories of passive approaches are the online and incremental learning algorithms and the dynamic ensemble methods. Recently some proposed approaches have been employing these techniques to handle concept drifts in time series.

Guajardo, Weber and Miranda (2010) proposed an implicit concept drift handling method for time series forecasting which is based on moving window instance selection and support vector regression (SVR) with retraining. A moving window slides through the time series data stream in order to define the training and test sets. At each step the window moves, SVR is retrained with the portion of the window reserved for training and applied to the test set. The window size is adjusted to fit seasonal patterns of the time series and slides considering these cycles. An issue of this approach is that the seasonal patterns of a time series is typically not known *a priori* in the general case. Besides, several real time series have no well defined seasonal patterns, which prevents a widespread application of this method.

Gu, Tan and He (2013) used the similar idea of having a moving window for handling concept drift in time series implicitly by updating a forecasting model. In that work, the window has a variable size which is adjusted by a probabilistic model that defines which instances should be in the retraining window. The probabilistic model gives more weight to more recent samples and to the samples more similar to the samples to be predicted. Samples with higher weights are used to fill the window. The proposed model was combined with the autoregressive (AR) model and with multi-layer feed-forward neural

networks (MLP) trained with backpropagation. In the experiments, authors compared the error-rate of the proposed adaptive learning method, which combines instance weighting with AR and with MLP against the simple AR and simple MLP. Results showed that the proposed approach was able to improve forecasting performance of the single forecasting models.

3.2 Active Adaptive Learning Methods

Active adaptive learning systems are those which update the learned model just after explicitly detecting a concept drift in the underlying time series data generation process. Change detection mechanisms designed to time series are typically carried out by inspecting the residuals of a fitted model. In these approaches, a statistical or a machine learning model is adjusted to the data and used to forecast new observations. The residuals of the forecasting are used to identify drifts in the underlying data generation process. As discussed earlier in Chapter 2, this approach presents some drawbacks that affects drift detection accuracy, since residuals may not properly reflect changes due to problems in training the model used.

Some researchers have investigated how to detect concept drifts in time series by monitoring raw time series observations directly. The proposed approaches are typically based on a retrospective statistical analysis of the time series observations in order to identify change points in the series. In a retrospective analysis, the drift detection is done just after receiving several samples from the data stream. Industrial applications typically require a real-time drift detection, since it allows a fast updating of the decision model and, consequently, the minimization of forecasting performance losses. Besides, these change-point methods (CPM) that monitor raw time series observations are able to discover changes in just the mean or variance of the time series observations in order to identify structural breaks. These changes not always imply in the need for updating a forecasting model in order to keep the forecasting accuracy. In the rest of this section, we discuss some of these two kinds of concept drift detection methods in time series.

3.2.1 Residual-Based Concept Drift Detection

Auret and Aldrich (2010) proposed a retrospective mechanism to explicitly identify change-points in time series using random forests. Random forests are ensembles of regression or classification trees in which each tree depends on a random vector sampled independently from the data. A classification or regression tree divides the feature space into recursive binary partitions and assigns a class label or a regression value for each partition. The proposed method constructs a random forest to model an initial time series segment which is considered the normal process or the known concept. This interval is

defined by a moving window of length w. For the data into the moving window, the method constructs a subspace of extracted features, represented by a lagged trajectory matrix. As the moving window slides w instances, the approach computes a test matrix for the new data and compares the distance between the reference matrix and the test matrix. The mean sum of squared residuals, measured by the Euclidean distance, is used to measure this distance. When this difference is significant, a concept drift is identified.

Alippi, Boracchi and Roveri (2013a) proposed an ensemble of CPMs to detect changes in the residuals of a fitted model in a retrospective manner. The proposed ensemble aggregates several individual estimates of the change point, each obtained by running a CPM on a subsequence of the residuals defined by random sampling. The idea behind the ensemble is to improve the change-point identification compared to a single CPM. The Lepage statistical test (LEPAGE, 1971) was used to detect drifts since it is able to detect changes in mean and variance of the residuals. Different aggregation mechanisms of the ensemble have been investigated in this work in order to combine the outputs of the CPMs.

Fokianos, Gombay and Hussein (2014) also proposed an explicit, retrospective CPM for binary time series. The proposed method searches for changes in the coefficients of a logistic regression model adjusted to time series with an autoregressive-type dependence. The statistical procedure for testing possible change-points is based on computing standardized scores of the time series observations obtained via a partial likelihood function. Experiments were performed with two time series: one composed of mortality rates after a cardiac surgery and another one composed of IBM share transactions. However the proposed method was not compared with other approaches.

Brodersen et al. (2015) proposed a retrospective approach to measure effectiveness of the introduction of discrete market events, such as the release of a new product or an advertising campaign, in modifying market behaviors. In this work, a metric of interest, such as clicks in a web site is modeled as a time series. In some instant, the intervention in the market is performed and the causal impact of that intervention needs to be confirmed. The causal impact is confirmed if the difference between the observed time series after the introduction of the event and the values that would have been obtained without the intervention is statistically significant. This is an special case of the explicit concept drift detection problem in which the probable drift point is known in advance and just needs to be confirmed. In order to confirm that the intervention causes a change in the time series behavior, the proposed approach uses three sources of information: (i) the normal behavior of the time series before intervention, (ii) other time series related to the time series under study and (iii) the prior knowledge about the Bayesian model parameters. These information are combined using a state-space time series model, where one of the components of state is a linear regression approach. The proposed approach is then fitted

to the time series prior to the intervention and used to forecast values after the intervention. The difference between the observed and predicted values gives a semiparametric Bayesian posterior distribution for the causal effect.

Werner et al. (2015) proposed an explicit, retrospective method for detecting structural break points in climate time series. In this approach, linear piecewise regression is used to model the time series data. For a given predefined number of breaking points to be located, the residual squared sum of the model fitted to data is calculated for all possible break point locations. A hypothesis test is performed in each detected break point in order to assess the significance of the residual differences in the data divided by that break point. The main issue of this approach is that it is a retrospective change detection in which the number of possible breaking points should be defined beforehand by user. In the general case, it is not possible to know this information in advance.

Some authors have investigated the use of online, explicit concept drift detection in time series based on residuals of a fitted model. Yamanishi and Takeuchi (2002) proposed an online, explicit method for both outlier and concept drift detection in time series. The proposed approach fits an AR model to the data and updates its parameters incrementally so that the effect of past examples is gradually discounted. Each data observation receives a score computed by a prediction loss function. This score is a measure of deviation of the data observation from the fitted model. Higher scores indicate a high possibility of the observation being an outlier. Concept drift detection is done by monitoring the moving average losses of the time series observations. The main issue of this approach is that it is limited to time series that present an autoregressive behavior. This supposition may not hold for several real-world time series.

Kirch and Kamgaing (2015) proposed an online, explicit drift detection method for time series which is based on function estimation. This method estimates the initial set of function parameters which defines a stable historic data set (known concept). This function estimation procedure is similar to those used to derive estimators such as maximum likelihood estimators, least-squares estimators, among others. After this initial function estimation, the new incoming observations are monitored for a change in the estimated parameters. When no change occurs in data, it is expected that the sum of residuals of the estimated function on the unseen data is close to 0, and different from 0 otherwise. When the partial sum of residuals weighted by a weighted function defined by the user is higher than a critical value, a change point is detected. The choice of the weight function determines the detection delay. The detection accuracy of the method is highly influenced by the choice of the monitoring function. Several estimating functions were tuned and tested in several linear and non-linear time series, binary models, Poisson autoregressive time series, among others.

In a recent work (CAVALCANTE; OLIVEIRA, 2015), we investigated the use of

an online, explicit drift detection method to handle concept drift in financial time series. In that work, we compared the use of DDM and ECDD drift tests in combination with ELM and OS-ELM learning methods. A first set of experiments showed that the combination of ELM with drift detection is able to improve the accuracy the batch learning algorithm when applied to time series with concept drift. In a second set of experiments, it was showed that combining OS-ELM with drift detection tests could reduce the processing time of the online learning system while maintaining the forecasting accuracy.

Gombay and Serban (2009) proposed a different online, explicit drift detection method for AR time series, which is not based on monitoring the residuals of a fitted model, but based on monitoring the parameters of the model themselves. The Page's cumulative sum (CUSUM) process for detecting changes in the mean of independent observations was adapted to detect changes in the parameters of an AR time series, namely μ, σ^2, $\alpha_1, ..., \alpha_p$. Initially, any parameter or set of parameters have a set of values θ_0 estimated by past samples. The main idea is use the CUSUM test to detect a change in these parameters by monitoring the departure of the current values of the parameters θ from the initial values θ_0. The null-hypothesis $H_0 : \theta = \theta_0$ is tested after each new observation arrives sequentially in time. A critical value is used to define a threshold for drift detection. The main issue of the proposed approach is that it is limited to drift detection in AR time series. However, several real-world time series such as economical and financial time series present a nonlinear behavior.

3.2.2 Methods Based on Monitoring Time Series Observations

Oh and Han (2000) investigated the change-point detection problem in interest rates time series. Since in every country governments monitor and heavily interfere in interest rates through monetary policies, these kind of data typically present change points. The change-point detection is performed with the Pettitt test, which is a non-parametric CPM performed in a retrospective way (PETTITT, 1980). For each observation in the time series, this test verifies if that point divides the time series into two segments in which the null-hypothesis of no changes in the distributions is rejected. The number of change-points to be searched by this method is defined by the user. After identifying the change-points, the proposed method groups the time series segments limited by the change-points and uses these intervals to model artificial neural networks specific for each group. An issue of this approach is that the number of changing points to be searched should be defined by the user, which may require a priori knowledge about the analyzed time series.

Liu et al. (2013) proposed an explicit, retrospective change-point detection for time series which relies on a statistical test based on non-parametric divergence estimation between time series samples from two consecutive segments. The relative Pearson divergence dissimilarity measure is used to measure the differences between two segments and identify

drift points. This dissimilarity measure is estimated by a direct density-ratio estimation method called the unconstrained least-squares importance fitting (uLSIF). Authors also improved the drift detection method by using an extension of the uLSIF called relative uLSIF (RuLSIF), which corrects a weakness of uLSIF, which is the possibility of having unbounded density-ratio when the denominator density is not well-defined. Authors compare the uLSIF-based change point detection with the RuLSIF version as well as with the Kullback-Leibler importance estimation procedure (KLIEP), which is another direct density estimation proposed in the literature. Experiments with artificial and real-world time series showed that RuLSIF based change-point detection method presented better change-point detection accuracy.

Yamada et al. (2013) proposed an explicit, retrospective change-point detection for high-dimensional time series which incorporates feature selection to improve the drift detection. In this work, authors propose a supervised drift detection method that works as follows. Let two non-overlapping windows $x_w(t) = \{x_{t-n+1}, ..., x_t\}$ and $x_w(t + n) = \{x_{t+1}, ..., x_{t+n}\}$ of size n extracted from time series $\{x_t\}$. A pseudo binary label $y_i \in \{-1, +1\}$ is assigned to time series observations. The observations x_i within $x_w(t)$ receive label $y = 1$ and observations x_i within $x_w(t + n)$ receive $y = -1$. The change-point detection strategy consists in computing a dependency score between input x_i and output y of the data in $\mathbb{Z}(t) = \{x_w(t) \cup x_w(t + n)\}$. In the case where the dependency measure takes a large value, it means the data in $x_w(t)$ and $x_w(t + n)$ are separable, or in other words, their samples come from different distributions, so a concept drift is detected. Authors claim that in high-dimensional environments, noise and outliers may disrupt the drift detection. So, in order to cope with high-dimensionality, they propose the use of a feature selection approach, in order to improve the drift detection.

Ross (2013) describes some parametric and nonparametric online change detection methods implemented in the R language. In this work, the author describes how to identify multiple change points in a foreign exchange time series. The time series is pre-processed with the first order differencing to remove the correlation between the observations. Then the Mood change point method (MOOD, 1954), a nonparametric change detector, is applied to the differenced time series observations. A similar work, which also describes an R package to discover change points in time series, is proposed by Killick and Eckley (2014). The main issues of these methods is that they are retrospective change detection methods and the fact that they are limited to find changes in the mean and/or variance of the time series, disregarding other important sources of changes in the time series.

Blythe et al. (2012) proposed a retrospective change-point detection approach based on Stationary Subspace Analysis (SSA) for multivariate time series. According to these authors, not all directions in the high-dimensional signal space are informative for change-point detection, since there is a subspace in which the distribution of the data remains

immutable over time. So, in this work, authors use SSA to reduce the dimensionality of the data by identifying which are the most non-stationary directions, since they are considered more informative for detecting changes in the time series. SSA receives a multivariate signal and factorizes it into stationary and non-stationary sources. Then the proposed approach finds the projection to the most non-stationary sources and considers just them to find changing points. Results obtained by the performed computational experiments showed that the SSA preprocessing was able to improve the change-point identification.

3.3 Summary and Discussion

In the literature, we can find some approaches dedicated to handle concept drift in regression problems and in time series forecasting. In this chapter we review some of these work. Table 1 summarizes the main approaches designed to handle the concept drift in time series analysis. In this table, the approaches were classified according to the following aspects: (i) the transparency for the user in handling drift, which can be passive or active; (ii) the drift detection delay, which can be online or retrospective; (iii) the method employed to handle drift, which can be just passively retraining/updating, the monitoring of statistical features of data or the residuals of a fitted model; and (iv) the main goal of approach, which can be just the detection of change-points or improving the forecasting performance.

As we discussed throughout this chapter, some of these approaches handle concept drifts in an implicit way, by means of online or incremental learning schemes which generally rely on instance selection or instance weighting. As discussed previously, there are some disadvantages of using implicit methods for handling concept drift. The main issue of these approaches is the lack of transparency for the user. Explicit drift detections can be themselves useful for decision-support. Another issue of implicit methods is the potential resource consumption to update the learned model successive times. Excessive adaptation may be a waste of computational resources (ZLIOBAITĖ; BUDKA; STAHL, 2015).Since these methods are unaware of drifts, the successive retraining continues even when the new instances comes from the same data distribution.

Table 1 shows that some proposed approaches implement an explicit, retrospective drift detection. Since these methods generally work with the complete time series under study, they implement a drift detection method that search for possible breaking points, which are points that divide the data into subsequences with different distributions. So, these methods rely on monitoring the differences in data distributions separated by possible changing points. Despite that these methods are very useful to help in understanding time series, they are not suitable for real-time applications. According to Liu et al. (2013), the retrospective drift detection tends to provide better detection accuracy than the online

Table 1 – Comparison of drift handling methods in time series analysis.

Work	Transparency	Delay	Method	Main goal
Guajardo, Weber and Miranda (2010)	Passive	Online	Instance selection and retraining	Forecasting
Gu, Tan and He (2013)	Passive	Online	Instance weighting and retraining	Forecasting
Auret and Aldrich (2010)	Active	Retrospective	Residual-based	Change-point detection
Alippi, Boracchi and Roveri (2013a)	Active	Retrospective	Residual-based	Change-point detection
Fokianos, Gombay and Hussein (2014)	Active	Retrospective	Residual-based	Change-point detection
Werner et al. (2015)	Active	Retrospective	Residual-based	Change-point detection
Brodersen et al. (2015)	Active	Retrospective	Residual-based	Causal Impact
Yamanishi and Takeuchi (2002)	Active	Online	Residual-based	Change-point detection
Gombay and Serban (2009)	Active	Online	Residual-based	Change-point detection
Kirch and Kamgaing (2015)	Active	Online	Residual-based	Change-point detection
Cavalcante and Oliveira (2015)	Active	Online	Residual-based	Forecasting
Oh and Han (2000)	Active	Retrospective	Monitoring observations	Change-point detection
Liu et al. (2013)	Active	Retrospective	Monitoring observations	Change-point detection
Yamada et al. (2013)	Active	Retrospective	Monitoring observations	Change-point detection
Ross (2013)	Active	Retrospective	Monitoring observations	Change-point detection
Killick and Eckley (2014)	Active	Retrospective	Monitoring observations	Change-point detection
Boracchi and Roveri (2014)	Active	Online	Feature-based	Change-point detection
This thesis	**Active**	**Online**	**Feature extraction and weighting**	**Forecasting**

Source: elaborated by the author.

drift detection, since the approach knows the entire time series behavior. However, as stated by Yamanishi and Takeuchi (2002), one important requirement of drift detection methods is that it should be online, in such a way that the drift point should be detected immediately after a change happens in data. Real-time systems, such as autonomous trading systems, robot control, among others, require a drift detection which is able to minimize the delay of detection. In these applications, the drift detection mechanism is usually embedded in adaptive learning systems, which should be able to adapt quickly and adequately to accommodate possible changes (MINKU; YAO, 2012).

The online explicit drift detection methods are ideal to be used in adaptive learning systems applied in real-time problems. The majority of these approaches discussed in

this chapter relies on monitoring the residuals of a fitted model. In these approaches, a statistical model or a machine learning model is fitted to a subset of time series observations that represent the reference to the known concept. The main assumption is that if there is no concept drift, these residuals tend to be stationary. In case of drifts, these residuals tend to increase, since the learned model is outdated and is not effective to take decisions about the new concept.

There are some disadvantages of using residual-based explicit drift detection methods. Since these methods rely on the forecasting accuracy of the model, problems in the parameter adjustment process of the model or in the training process, in case of machine learning modeling, may imply in a bad drift detection. In some cases, the concept may change and the residual level keep high and constant due to generalization problems such as overfitting. In these cases, the residuals may not properly reflect concept drifts. According to Alippi, Boracchi and Roveri (2013b), in practice, the residuals of a fitted model are not independent and identically distributed, even before a concept drift. Instead, they are correlated and influenced by the dynamic of the original signal. In theory, monitoring some data features directly may provide more robust concept drift identification in time series. However the majority of discussed work monitor time series raw data directly. Although this approach is useful to identify breaking points in time series, it is not useful to identify when there is the need to update the time series model used to forecast future values.

Just a few studies consider time series features to identify concept drifts. Boracchi and Roveri (2014) proposed an online concept drift detection for time series that is based on a time series feature, namely the self-similarity. In the proposed approach, at each time instant, a data sequence of fixed size is extracted from the incoming data and the most similar previously seen data sequence is recovered from memory. A change indicator variable $x(t)$ is computed as the difference between the two time series sequences. When the arriving data sequence differs significantly from previously seen sequences, the distribution of $x(t)$ changes, and a drift is detected. The intersection of confidence intervals (ICI) drift detection test (ALIPPI; BORACCHI; ROVERI, 2011) is used to detect changes in the distribution of $x(t)$. The main issue of this approach is that it is specifically designed to time series that present self-similarity and periodicity. So, the proposed approach addresses the problem of detecting structural changes in the time series. Besides, this approach presents considerable memory consumption to store the data sequences which define the regular or stable time series behavior.

In Table 1 one can see that in terms of the main goal of the reviewed methods, just a few of them has as the main objective the improvement of the forecasting accuracy. In real-time adaptive systems, the main goal of the learning method is to adapt to changes as soon as possible in order to keep the performance of the system. In this approach, the drift detection is just an intermediate activity that indicates when a change occurs in the

modeled environment, triggering an event for retraining or updating the learned model.

So, an ideal adaptive learning method able to handle concept drift in time series should:

- be transparent to the user, in the sense of explicitly detect and inform about the occurrence of drifts (active method). In financial applications, for example, in which the trader uses forecasting to take investment decisions of buying or selling stocks, this information may indicate the need to reduce market positions or to avoid buying new stocks;
- operate in an online way, reducing the delay to update the forecasting model in order to keep the forecasting accuracy even in face of concept changes. In high-frequency data streams, this requirement is crucial;
- be based on time series features that properly reflect the time series concepts, which helps to improve the concept drift detection;
- combine drift detection techniques with forecasting techniques, acting to improve the adaptability and consequently the accuracy of the method.

As we can see from Table 1, none of the approaches satisfy these requirements. This thesis is an attempt to fill this gap in the literature. In this work we investigate how to detect concept drifts based on time series specific features and how to improve the forecasting by integrating this drift detection mechanism with an adaptive forecasting model, as explained in Section 1.3. We believe that monitoring time series specific features would provide a better description of the concept drifts in the time series and allow faster recovery from these changes keeping the forecasting accuracy of the system.

4 THE PROPOSED ADAPTIVE LEARNING SYSTEM

In Chapter 3, we discussed several approaches proposed to handle concept drift in time series forecasting, such as the passive approaches, and the active approaches based on monitoring the residuals of a fitted model and those which monitor the time series raw data. In that chapter we also discussed the main issues of each approach and gave directions on how to better address the concept drift detection problem in order to overcome the issues of the existing methods. Based on the assumption that $p(X)$ and $p(y)$ are expected to be generated by the same underlying process $\mathbb{S}$, we believe that changes in $p(y)$ will be always reflected by some changes in $p(X)$ with some delay. The methods that monitor raw data consider that every change in $p(X)$ will imply in changes in $p(y|X)$, however this is not always true. On the other hand, if we monitor some statistical time series features that describe the behaviors of the inputs X and their temporal correlation, we are able to better characterize time series concepts. The time series features are pre-defined descriptive statistics automatically calculated from the time series data which describe several aspects of the time series underlying data generation process. With this approach, we can detect changes in these concepts in a faster and more reliable way.

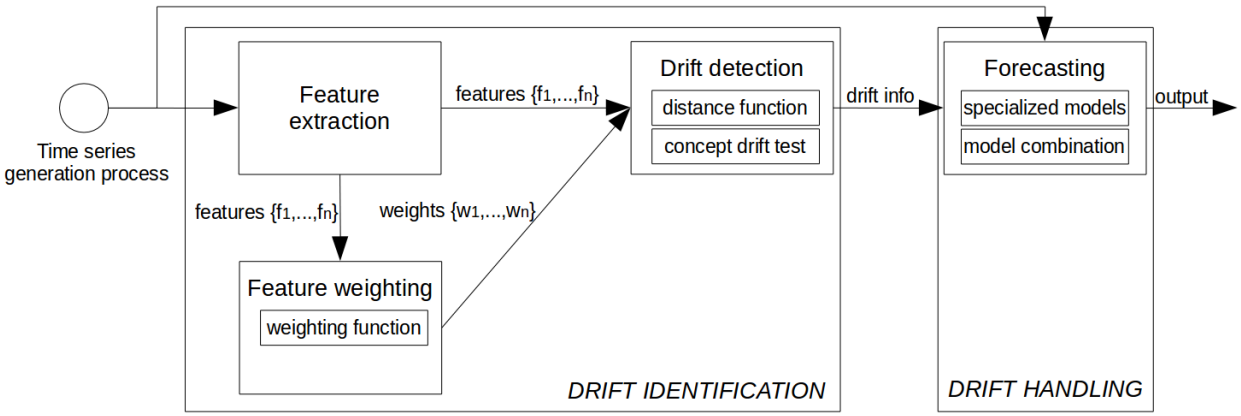
The main contribution of this thesis is an adaptive learning system for time series forecasting that employs an online, explicit concept drift detection method based on time series features. The proposed system, called Feature Extraction and Weighting for Explicit Concept Drift Detection (FW-FEDD), identifies the occurrence of concept drifts by monitoring the evolution of these features. An unsupervised weighting strategy is used to compute the importance of the features monitored in order to improve the drift detection accuracy. The forecasting method of the system is composed of a pool of forecasting models in which each individual model is designed to represent a particular time series concept.

4.1 General Architecture

The Figure 12 presents the general architecture of the proposed adaptive learning system. The proposed FW-FEDD has four main modules: (i) the Feature Extraction (FE) module, (ii) the Feature Weighting (FW) module, (iii) the Drift Detection (DD) module and (iv) the Forecasting Module (FM). We can organize these modules in two packages, one responsible for drift identification, composed of the FE, FW and DD modules, and one for drift handling, composed of the forecasting module.

The drift identification package receives the time series observations sequentially

Figure 12 – General architecture of the proposed system.



Source: elaborated by the author.

and processes them in order to identify changes in the time series generation process. The FE module is responsible for extracting time series features. The FW module is responsible for computing the importance of the features to the drift detection process. The DD module monitors the evolution of the time series features along the process and tests the occurrence of concept drifts, considering the importance of the features. Initially, a feature vector is extracted from the available time series observations. It is assumed that these initial time series observations have no concept drifts. This initial vector describes the in-control state of the time series, that represents the known concept. A moving window that slides whenever a new sample is available is used by the FE module. Features are computed on the current window and the feature vector on that window is compared with the initial feature vector. A weighted distance function is used to compute the dissimilarity between these two vectors. When the two feature vectors present a statistically significant difference, the DD module identifies it as a concept drift.

FW-FEDD is an online method since it is able to sequentially inspect incoming time series observations, one at time, in order to decide whether or not there is a change. Even though the sequential processing starts only after the initial feature vector is extracted, this is different from retrospective methods, where changes can only be detected in a past fixed-length sequence that has already been received as a whole.

After the drift identification, the forecasting module is responsible for handling eventual concept drifts. This module is composed of a set of forecasting models. Each individual model is represents the knowledge of a different time series concept. Initially, just one model is trained with the time series data available for the system. This individual model is used to forecast new observations until a concept drift be identified. In this point, the system identifies whether the existing models are able to handle the new concept or whether there is the need to create a new one to be specialized in this new concept faced

by the system. This module implements a function to decide how the existing models should be combined to better handling the time series concepts in order to improve the forecasting accuracy of the system. In the next sections, we describe the modules of the proposed system, following the division in drift identification and drift handling packages.

4.2 Drift Identification

4.2.1 The Feature Extraction Module

Several statistical features can be used to characterize time series behaviors. In the literature, the analysis of statistical descriptive time series features has been used to solve important machine learning problems, such as time series classification (PRUDÊNCIO; LUDERMIR; CARVALHO, 2004), time series clustering (WANG; SMITH; HYNDMAN, 2006; AGHABOZORGI; SHIRKHORSHIDI; WAH, 2015), time series meta-learning (PRUDÊNCIO; LUDERMIR, 2004), among others. Different from conventional input attributes used in a machine learning algorithm, time series features are derivative statistics able characterize some relationship about time series observations. We believe that some time series features can be used to define time series concepts. In stable conditions, the time series feature values are expected to be constant. Whenever these features evolve over time, it can be interpreted as a concept drift. In this context, it is worth to make a distinction between the terms feature extraction and feature selection. Feature selection mechanisms, also called feature subset selection, identify the input variables that are not relevant for modeling the data to be learned. Feature extraction mechanisms, on the other hand, try to find a transformation from the original data to a different feature space (WEBB, 2003).

In Chapter 2, we discussed that time series can be classified according to their behavior in stationary or non-stationary, linear or non-linear, among others. We also discussed some characteristics that describe important aspects of the series, such as the presence of trends and/or seasonal patterns, outliers, among others. In order to build a general drift detection method, and capture powerful characteristics to describe concepts in time series, the FE module uses feature extraction to compute eleven time series features from the original univariate time series data to describe the time series concepts. These features can be divided into four groups: (i) deterministic pattern features, (ii) linear features, (iii) descriptive statistic features and (iv) non-linear features. The deterministic pattern features are as follows:

1. Trend degree, which describes the degree of presence of trend in the time series;
2. Seasonal degree, which describes the degree of presence of seasonality in the time series.

Trend and seasonality are common features of time series (WANG; SMITH; HYNDMAN, 2006). The degree of trend and seasonality are very informative about changes in time series behavior. Typically these features are referred in literature as deterministic components, since they represent systematic behaviors in the time series. When trend and seasonal patterns are removed from the time series, the result is a residual time series, which represents the stochastic component of the original time series.

In the literature, there are several ways of estimating trend (T_{x_t}), seasonal (S_{x_t}) and residual (R_{x_t}) components of a time series $\{x_t\}$. For estimating trend, we used local regression smoothing process which implements weighted linear last squares to smooth the data. After estimating the trend component, we additively remove the trend and apply a seasonal filter to the detrended series in order to identify the seasonal factors of the detrended series. Finally, the residuals are calculated by removing the estimated seasonal effects from the detrended time series. It is important to note that these residuals are the random variations after removing trend and seasonality and should not be confused with the residuals that define the error of a fitted model.

Since we need a univariate signal to define each time series feature, we compute the trend degree and seasonal degree of the time series. The trend degree (td) is computed as $td = 1 - \frac{var(R_{x_t})}{var(x_t - S_{x_t})}$ and the seasonal degree (sd) is computed as $sd = 1 - \frac{var(R_{x_t})}{var(x_t - T_{x_t})}$. These features give an indication of the presence of these deterministic patterns in a time series. These features can assume values in the range [0,1]. Values close to 1 indicate higher presence of the corresponding feature. Time series generated by relatively controlled physical phenomena, such as temperatures, sales and behaviors of some industrial machines, may present trend and seasonal degrees close to 1. On the other hand, series of financial markets and exchange rates, for example, should have lower trend and seasonal degrees and more influence of the stochastic component.

The linear features used in this work are:

1. Autocorrelation, which describes the similarity between observations in function of some lag;

2. Partial autocorrelation of the residuals, which describes the correlation that results after removing the effects of any correlations due to terms at shorter lags;

The correlation of a variable with itself at different time instants is called autocorrelation or serial correlation. The number of time steps separating them is called lag (COWPERTWAIT; METCALFE, 2009). The autocorrelation is a measure of similarity between observations x_t and x_{t+k} as a function of the lags k between them. The autocorrelation at the lag 0 is 1 (maximum value), since it gives the autocorrelation between the observations with themselves. A high correlation in the first lags indicates that there

is a high dependence of each time series point with the recent past points (PRUDÊNCIO; LUDERMIR, 2004). The formula for estimating the autocorrelation at lag k is given by $\rho_k = \frac{c_k}{c_0}$, c_k is computed as in eq. 4.1, where c_0 is the sample variance of the time series and n is the time series size. The autocorrelations at the first five lags were used by FW-FEDD. This number was chosen empirically and not optimized.

$$c_k = \frac{1}{n-1}\sum_{t=1}^{n-k}(x_t - \overline{x})(x_{t+k} - \overline{x}) \tag{4.1}$$

The partial autocorrelation function may be defined as the correlation between x_t and x_{t-k} after removing the effect of the intervening variables x_{t-1}, x_{t-2}, x_{t-3},..., x_{t-k+1} (CRYER; CHAN, 2008). In general, the partial autocorrelation at lag k is the kth coefficient of a fitted AR(k) model. If the underlying process is AR(p), then the coefficients α_k will be zero for all $k > p$ (COWPERTWAIT; METCALFE, 2009). So, the partial autocorrelation function can be used to estimate the order of autoregressive models. The partial autocorrelations of the residuals resulting after removing trend and seasonality at the first five lags were used in FW-FEDD.

The descriptive statistic features subset extracted by FW-FEDD are:

1. Standard deviation of the residuals, which describes the degree of instability of the residual time series;
2. Skewness coefficient of the residuals, which describes the asymmetry of the data around the sample mean;
3. Kurtosis coefficient of the residuals, which describes how outlier-prone a data distribution is;
4. Periodicity, which estimates the length of a seasonal pattern;
5. Turning points rate, which describes the degree of oscillation of the time series.

The skewness coefficient is a measure of the asymmetry of data around the sample mean. A data set is symmetric if it looks the same to the left and the right of the center point (WANG; SMITH; HYNDMAN, 2006). The skewness of any perfectly symmetric distribution, such as the normal distribution, is zero. If skewness is negative, the left tail of the distribution is long relative to the right tail. In other words, the data are spread out more to the left of the mean than to the right. If skewness is positive, then, the right tail of the distribution longer than the left tail. The skewness measure is computed as in eq 4.2.

$$\tau = \frac{\mu^3}{\sigma^3} = \frac{E[(x-\mu)^3]}{E[(x-\mu)^2]^{3/2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^3}{(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2})^3} \tag{4.2}$$

Similarly to skewness, the kurtosis coefficient is a description of the shape of the probability distribution of the data. The kurtosis coefficient is informative about the tail behavior of a series (BAI; NG, 2005). A dataset with high kurtosis tends to have heavy tails. Datasets with low kurtosis tend to have a flat top near the mean rather than a sharp peak (WANG; SMITH; HYNDMAN, 2006). The kurtosis of the normal distribution is 3. Distributions which are more outlier-prone than the normal distribution have kurtosis greater than 3. Kurtosis is the fourth standardized moment of a variable and is computed as in eq. 4.3.

$$\kappa = \frac{\mu^4}{\sigma^4} = \frac{E[(X-\mu)^4]}{(E[(X-\mu)^2])^2} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^4}{(\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2)^2} \tag{4.3}$$

The periodicity and turning points rate features describe the oscillation behavior of the time series. Periodicity is the length of a seasonal pattern. Wang, Smith and Hyndman (2006) state that seasonality is different from periodicity. According to these authors, periodicity varies in frequency length, but seasonality has fixed length over each period. FW-FEDD estimates the periodicity using the scheme proposed by Wang, Smith-Miles and Hyndman (2009): find the first autocorrelations of the residuals and then find the first peak in the autocorrelations which (i) has a trough before it; (ii) the difference between peak and trough is at least 0.1; and (iii) the peak corresponds to positive correlation.

The turning points rate measures the degree of oscillation of a time series (PRUDÊNCIO; LUDERMIR, 2004). A time series point x_t is a turning point if $x_{t-1} < x_t > x_{t+1}$, or $x_{t-1} > x_t < x_{t+1}$. The presence of a very high or a very low number of turning points in a series suggests that the series is not generated by a purely random process. The percentage of turning points in the time series is computed as $100 * n_z$, where, n_z is number of turning points divided by the length of the series.

The nonlinear features attempt to describe the nonlinear behavior of a time series, which are common in real-world time series. The computed nonlinear features are as follows:

1. The bicorrelation, or three-point autocorrelation;
2. The mutual information.

The bicorrelation, also called three-point autocorrelation, is a measure of nonlinear autocorrelation (KUGIUMTZIS, 2008). This metric is an extension of the standard autocorrelation and estimates a high order correlation feature which is described by the joint moment of three variables formed from the time series in terms of two delays s_1 and s_2 (KUGIUMTZIS; TSIMPIRIS, 2010). The bicorrelation at two positive lags k_1 and k_2 ($k_1 < k_2$) is given by the eq. 4.4. A simplified scenario for the delays is when $k_2 = 2k_1$, in

which the measure becomes a function of the delay k_1 The bicorrelations of the residuals at the first three lags were used in the proposed approach.

$$B(k_1, k_2) = (n - k_2)^{-1} \sum_{t=1}^{(n-k_2)} x_t * x_{t+k_1} * x_{t+k_2} \tag{4.4}$$

The mutual information for two variables $\mathbf{X}$ and $\mathbf{Y}$ is defined as the amount of information that is known of one variable when the other is given (KUGIUMTZIS; TSIMPIRIS, 2010). The mutual information is an entropy-based measure that estimates the general correlation between x_t and x_{t-k} for different lags k (KUGIUMTZIS, 2008). This measure can be considered as a correlation metric that measures the linear and nonlinear autocorrelation of a time series. There are several ways to estimate mutual information. The histogram-based estimate, which was used in this work, can be computed as in eq. 4.5, where i is the number of bins of the partition of the data, p_i is the estimated probability that a data point x_t is in bin i, p_j is the estimated probability that a data point x_{t+k} is in bin j, and $p_{i,j}$ is the estimated joint probability that that x_t is in bin i and x_{t-k} is in bin j. The mutual information of the series at the first three lags were used in this work.

$$I(k) = \sum_{i,j} p_i log \frac{p_{i,j}}{p_i p_j} \tag{4.5}$$

The FE module receives a time series segment $\{x_{t-m_f}, ..., x_t\}$, of size m_f, as input and computes a feature vector containing the values for all eleven features, which totals 23 attributes: trend degree, seasonal degree, 5 first autocorrelations, 5 first partial autocorrelations, periodicity, standard deviation, skewness, kurtosis, turning points rate, bicorrelation at 3 first lags and mutual information at 3 first lags. These values 3 and 5 were empirically chosen and not optimized.

4.2.2 The Feature Weighting Module

Time series from different domains may present different behaviors. So, for different time series behaviors, some features may be informative about these behaviors and some features may be uninformative. One example is the periodicity, which is a feature that identifies the size of the time series seasonality or cycles. Some time series present no seasonality and, in these cases, the periodicity feature is not informative to describe its behavior. Time series from different domains may also present different concept drifts. In these cases, some features may be more important than others in describing the concept drifts that can eventually happen. Besides this, the importance of the features may also change due to concept drifts. A time series that has no periodicity may start to present some seasonal pattern after a concept drift. So, the periodicity feature becomes very important

for identifying the concept drift. Due to these situations, it would be of utmost importance if we could, automatically and in an online way, be able to detect the importance of the features to the detection of concept drifts.

According to Kuncheva and Faithfull (2014), features with lowest variance are more likely to be affected by a change than features with higher variance. In that work, authors intended to detect concept drifts in multivariate time series, so the term feature in that work refers to the input dimensions of the time series. Their proposed approach monitors raw observations in each dimension of the time series to discover concept drifts. We can bring this idea to our context of monitoring statistical descriptive features of a univariate time series. For example, a time series that presents no seasonal pattern has seasonal degree equal zero on average with zero variance. However, if it suddenly starts to present some cyclic pattern, the seasonal degree tends to increase. This change in a feature that was presenting values close to zero variance is very important to define the concept drift.

Based on this statement, we propose a heuristic weighting strategy to consider the variability of the features during the computation of the distances between the feature vectors. The idea is to give more importance to the features with lower variability, believing that these features will be more informative about eventual concept drifts in the time series. We also believe that with this strategy we can build a general approach for handling concept drifts in time series of every domain. This approach also allows the addiction of more features without loss of generality of the method. This unsupervised feature weighting is independent of external information or human intervention.

Kuncheva and Faithfull (2014) proposed to compute the principal components of the feature set using Principal Component Analysis (PCA) and consider the least important components as the more indicative of concept drifts. We use the same idea, but applied to the features extracted from the time series. Single Value Decomposition (SVD) was used to compute PCA since it calculates both the principal components and the coefficients. At every instant we compute the norms of the eigenvector coefficients in the new space. The principal components are those with high norms (and high variation). The less important components, which have lower norms (lower variation), are considered more important to indicate concept drifts. Based on this, the weights of the features are given by the inverse of the norms computed by the PCA.

A problem with this weighting function is that PCA has a high computational cost, which may become an impediment in real-time applications with a high set of features to be considered or in which time series observations arrive at a high speed. So, in this work, we propose another heuristic weighting function which considers the weight of a feature as the inverse of the standard deviation of the feature. This strategy is computationally cheap, since the variance and standard deviation can be computed in $O(m_f)$, where m_f is the number of initial observations, and it can updated online in $O(1)$ time. With this

strategy, features with high standard deviations have less influence in the computation of the distances between the feature vectors. On the other hand, features with low standard deviation, which are more influenced by eventual concept drifts, have higher weights since they are more effective to indicate concept drifts. We expect that this approach is able to improve the concept drift detection with less computational cost.

4.2.3 The Drift Detection Module

The DD module is responsible for monitoring the time series features and for identifying when a concept drift happens. It receives a feature vector from the FE module, the feature weights from the FW module, and analyzes whether there is a change in the time series. The main problem of applying a CDT on features individually (even considering the importance of the features) is the high number of false positives that will be potentially detected. If each feature is monitored separately, a change in at least one feature causes the test to identify a change in the underlying time series generation process. As discussed by Alippi, Boracchi and Roveri (2011), increasing the number of features analyzed also increases the probability of having false positives.

In order to tackle this problem and improve the concept drift detection, we propose to consider the time series features in combination instead of individually. The idea is to detect concept drifts only when there is stronger evidence, supported by more than one time series feature. So, the proposed method monitors the differences between an initial feature vector, extracted from the time series when it is *in-control* state, and the current feature vector extracted at each time instant of the time series processing. This approach has two main advantages: (i) it reduces the problem of monitoring the distribution of several features to the problem of monitoring just a univariate signal, which is the distances between vectors; (ii) it is expected to reduce the false alarms caused by variations in few features individually.

As shown in Figure 12, the DD module consists of two main components: (i) a distance function, which computes the dissimilarity among two feature vectors, and (ii) a CDT, which tests the occurrence of significant changes in the distance level over time. In a time series without concept drift, the distance between feature vectors is expected to be small and constant. When a concept drift happens, the time series is expected to behave differently from before the change, and this different behavior will be reflected in its features. So, it is expected that the distance between the feature vectors increase from the drift point.

In this work, we use the Pearson correlation distance metric (STREHL; GHOSH; MOONEY, 2000) to compute the distances between the feature vectors. This distance metric was chosen because it is not influenced by the range of possible values each feature can assume. This is a requirement of the method, because the time series features have

different scales. Since FW-FEDD is designed to perform in open-ended data streams, it is not possible to use conventional normalization methods for rescaling the attributes, due to impossibility of finding maximum and minimum values, or the mean and standard deviations for the whole time series.

The Pearson distance is a dissimilarity metric based on Pearson's product-momentum correlation coefficient of two vectors. This metric describes the similarity in shape between the two vectors. In practical terms, it measures the degree of correlation between two vectors. When two vectors are very similar, the correlation is high. As we need a weighted distance metric, we implemented the weighted Pearson correlation distance metric to take into consideration the importance of each feature. The weighted Pearson correlation coefficient for two vectors $\mathbf{x}$ and $\mathbf{y}$ of size d and weights $\mathbf{w}$ is implemented as follows:

$$corr(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \frac{cov(\mathbf{x}, \mathbf{y}, \mathbf{w})}{\sqrt{cov(\mathbf{x}, \mathbf{x}, \mathbf{w}) cov(\mathbf{y}, \mathbf{y}, \mathbf{w})}},$$

where

$$cov(\mathbf{x}, \mathbf{y}, \mathbf{w}) = \frac{\sum_i^d w_i (x_i - m(\mathbf{x}, \mathbf{w}))(y_i - m(\mathbf{y}, \mathbf{w}))}{\sum_i^d w_i}$$

and

$$m(\mathbf{x}, \mathbf{w}) = \frac{\sum_i^d w_i x_i}{\sum_i^d w_i}.$$

The second component of the DD module is the drift detection test. In the literature there are several CDTs. The majority of them was proposed to solve the concept drift detection problem in the context of classification. In this work, we investigate three CDTs which are widely used in the literature, namely the Exponentially Weighted Moving Average for Concept Drift Detection (ECDD) (ROSS et al., 2012), the Page-Hinkley test (PHt) (PAGE, 1954) and the Intersection of Confidence Intervals (ICI) based CDT (ALIPPI; BORACCHI; ROVERI, 2010; ALIPPI; BORACCHI; ROVERI, 2011). These tests were chosen because they operate in a sequential mode, since we want to handle drifts in an online way. The DD module uses one of these tests to monitor the distances between an initial feature vector and the current feature vector. When the test identifies a change in the distances, the process is restarted in order to handle future drifts.

The ECDD analyses the exponentially weighted moving average (EWMA) of a variable to identify changes in its values. EWMA is an estimator of the mean of a sequence of values of a variable which gives more importance to recent data, whereas older data is being progressively downweighted. Suppose a set of values for a variable $\{x_1, ..., x_n\}$ which presents a mean μ_0 and standard deviation σ_x. The EWMA estimators for the variable are $Z_0 = \mu_0$ and $Z_t = (1 - \lambda)Z_{t-1} + \lambda x_t$, $t > 0$. The parameter λ indicates the weight given to

recent data when compared to older data. The mean and standard deviations of Zt are μ_{Z_t} and $\sigma_{Z_t} = \sqrt{(\frac{\lambda}{2-\lambda}(1-(1-\lambda)^{2t})\sigma_x}$, respectively. Ross et al. (2012) defined two rules to monitor concept drift based on EWMA. When $Zt > \mu_0 + W\sigma_{Z_t}$, a warning signal is triggered. When $Zt > \mu_0 + C\sigma_{Z_t}$, a change signal is triggered. The warning threshold and control limit, W and C respectively, are parameters of the method.

The Page-Hinkley test (PHt) is a sequential analysis technique originally used for change detection in signal processing (GAMA et al., 2014). PHt monitors the difference between the observation at each time instant and the mean of observations up to the current time. It is expected that if the incoming data distribution is immutable, the current data observations are close to the data average. On the other hand, if a concept drift occurs, the data values increase and the current values move away from the average, increasing the cumulative difference between these two values. Mathematically, at each instant t, PHt computes the cumulative difference between the value x_t and its mean $\overline{x}_T$, given by $m_t = \sum_{t=1}^{T}(x_t - \overline{x}_T - \delta)$, where δ is a discount factor. The minimum m_t, defined as $M_T = min(m_t, t = 1...T)$, found during the process is kept in memory. PHt tests for the difference between M_T and m_t and uses this difference to define a threshold. When $m_t - M_T > C$, a change signal is triggered, and the learned model needs to be retrained with recent data. The constant δ is a parameter of the method.

The ICI-based CDT is an online drift detection test that makes no a priori assumptions about the distribution of the process generating data. This test is based on the ICI rule, which operates on sequences of data having expectation $\mu(t)$ and standard deviation σ. At each instant, the ICI rule adaptively identifies an optimal neighborhood $U_i^+(t)$ and regularizes the data with an estimate $\hat{\mu}(t)$ of $\mu(t)$, obtained by least squared error polynomial fit of the data belonging to the optimal neighborhood. The ICI rule estimates the confidence intervals $\mathbb{I}_i$ of $\hat{\mu}_i(t)$ as $\mathbb{I}_i = [\hat{\mu}_i(t) - \Gamma\sigma_i(t); \hat{\mu}_i(t) + \Gamma\sigma_i(t)]$, where $\Gamma > 0$ is a parameter of the method. In the operational phase, the ICI rule identifies a change in the observed data when the intersection of confidence intervals is an empty set.

4.2.4 The Drift Detection Algorithm

Since the drift identification process proposed in this thesis can be used with any forecasting method or even alone (in cases where we just want to discover concept drifts in time series), we present the algorithm that describes this process separately from the forecasting method. The steps of the drift identification algorithm are detailed in Algorithm 1. The inputs of the algorithm are (Step 1): an initial subset of observations of the time series $S_0 = \{x_1, ..., xi, ..., x_{m_f}\}$, where $x_i \in \mathbb{R}$ is a time series sample, the window size m_f, the drift thresholds for the test Δ, which vary according to the test used. In Step 2, the initialization of the variable s, which denotes the start of the known time series concept is done. Steps 3 to 17 is repeated for every instant a new sample from the time series

becomes available. If the difference between the current time instant t and the start of the current known concept is less than m_f observations, the algorithm does nothing, since the moving window is not yet filled. When the difference $t - s$ is equal to the window size m_f (Step 4), then FE computes the initial feature vector fv_0 for the time series samples in the moving window $\{x_s, ..., x_t\}$, where $s = 1$ for this initial vector (Step 5). The feature vector fv_0 is used as a reference feature vector for the drift test, since it represents the known concept.

Algorithm 1: The FW-FEDD algorithm.

1: **Inputs:** $S_0 = \{x_1, ..., x_{m_f}\}$, m_f, Δ.
2: $s = 1$
3: **for** (each instant t a new instance x_t arrives) **do**
4: **if** $(t - s == m_f)$ **then**
5: $fv_0 =$ FE.extractFetures($\{x_s, ..., x_t\}$)
6: **else if** $(t - s > m_f)$ **then**
7: $fv_t =$ FE.extractFetures($\{x_{t-m_f}..., x_t\}$)
8: $w =$ FW.computeFeatureWeigths(fv_t)
9: $d_t =$ DD.weightedPearsonDistance(fv_0, fv_t, w)
10: $statistics =$ DD.computeTestStatistics(d_t)
11: **if** (DD.cdt($statistics$,Δ) detects drift) **then**
12: trigger a drift signal
13: $s = t + 1$
14: $fv_0 = [\,]$
15: **end if**
16: **end if**
17: **end for**

When $t - s > m_f$ (Step 6), the algorithm starts the online processing of the time series. In Step 7, the current feature vector fv_t is firstly computed by the FE module for the instances in the moving window. In Step 8, the weights w of the features are computed by the FW module. In Step 9, the weighted distance between the initial feature vector and the current feature vector in time t (d_t) is computed using the weighted Pearson correlation distance.

In Step 10, the statistics used by the chosen test are computed. This step is specific for each of the three tests investigated in this work. For ECDD, the statistics are the averaged distances (μ_d), the EWMA estimator of the distances (Z_t^d) and the standard deviation of Z_t^d, denoted by $\sigma_{Z_t^d}$. For PHt, the statistics are the averaged distances (μ_d), the cumulative differences m_t and the minimum cumulative difference M_T. For the ICI, the statistics are the confidence intervals $\mathbb{I}_i$ of the averaged mean of distances μ_d.

In Step 11, the CDT is run over the computed statistics in order to detect whether a concept drift happen in the distribution of distances. In case of drift detection, a drift signal is triggered and the process is restarted. The algorithm sets the start of the new

concept as $t+1$ and reset the feature vector that represents the current concept (fv_0), which will be calculated when the algorithm receives m_f new observations of the new concept (Steps 4 and 5). The online processing of the time series is repeated until no more time series observations are received.

4.3 Forecasting and Handling Drifts

There are several ways of handling a concept drift in a data stream in order to keep the forecasting performance of the system. As discussed early in Chapter 2, this handling can be explicit or implicit. The simpler implicit handling approach is to use an online or incremental learning method, which updates the learned model at every new data or chunk of data received (GU; TAN; HE, 2013). The main advantage of this approach is the ability to handle drifts as fast as possible, however at a high computational cost to update the model at every instant a new observation is available. Another implicit way to handle concept drifts is to keep a weighted ensemble of methods, in which the weights are computed dynamically based on the accuracy of methods (SOARES; ARAÚJO, 2015). These ensembles generally use some strategies to add and remove individual models in order to handle changes in data. This approach may improve the forecasting accuracy, but it can also increase the computational costs.

The simpler way to actively handle concept drifts is to create an individual model of data with a learning algorithm and use an explicit drift detection method. Whenever a drift is detected, the learned model is dropped and a new one is built to model the new concept (GAMA et al., 2004). Towards building an active adaptive learning system with minimum requirements of processing costs, we proposed a method that combines both active and passive drift handling approaches. The idea is to build a forecasting mechanism composed of a set of individual forecasting models in which each model is designed to handle a particular time series concept. This approach builds a new forecasting model just when none of the existing models in the system is able to handle a new time series concept faced by the system. The individual models employ online learning in order to specialize in a different concept drift and handle small changes in a concept.

There are two main design questions to be answered in order to build this forecasting module: (i) the intelligent forecasting method and the learning algorithm used to train it and (ii) the scheme to combine the individual models in order to make forecasting. In this work, we used a single hidden layer feedforward network (SLFN) trained with the Online-Sequential Extreme Learning Machines (OS-ELM) (LIANG et al., 2006). SLFNs have been widely used in the literature, being applied in many fields such as pattern recognition, signal processing short-term forecasting and so on (HUANG; ZHU; SIEW, 2004). OS-ELM is a variant of ELM that can learn data one-by-one or chunk-by-chunk

(blocks of data) with fixed or varying chunk size. OS-ELM combines the advantages of ELM, such as speed and generalization performance, with a sequential learning process. This algorithm randomly chooses the input weight matrix (which links input and hidden nodes) and the biases and analytically determines the output weight matrix of the SLFN. In the online processing, at any time, the newly arrived single or chunk data is used for updating the initial learned model. It is important to argue that other intelligent methods could be used instead of this.

So, initially, an individual SLFN is built from an initial amount of time series observations available using the OS-ELM training algorithm. This model is associated to the feature vector that describes those time series observations in order to define the context (concept) in which the model was created. In the online processing, the FW-FEDD monitors the time series and, when a concept drift is detected, the feature vector that defines the new concept is extracted and the forecasting module tries to identify if there is a forecasting model able to handle that concept. This is done by comparing the similarity between feature vectors associated with the models created and the current feature vector. In positive case, the forecasting module will use the existing models to make the forecasting according to the combination scheme used (as described further). On the other hand, if none of the models are able to answer that concept properly, a new SLFN is built to model the new concept. This decision on whether there is a forecasting model able to answer for a new concept is done by comparing the similarity of the feature vectors associated with the existing forecasting models with a threshold θ defined by the user.

In this work, we investigated two ways of combining the individual forecasting models created during the processing of a time series. The first and simpler scheme is to select the individual model which was created in the more similar context (concept) of the current time series observations. In this scheme, just the model with the feature vector closest to the current feature vector is able to perform the forecasting. The second strategy investigated was to consider the forecasting outputs of all the forecasting models created and compute the importance of each output as the inverse of the distance of the feature vector of each forecasting model to the current feature vector at each instant. So, models created in contexts more similar to the current one receive more weights than models created in more dissimilar contexts.

The functioning of the proposed adaptive learning method, now integrated with the drift identification is described in Algorithm 2. The inputs of the algorithm (Step 1) are: the window size that indicates time series observations used for feature extraction m_f, the window size that indicates time series observations used for building a forecasting model m_m, the drift thresholds for the test Δ, the threshold for inserting a new forecasting model θ, the lag that defines the forecasting inputs of the SLFN k, the forecasting algorithm Φ and the forecasting algorithm parameters ψ. In Step 2, the initialization of some variables

is done. Variable s marks the start of the current concept. $\mathbf{M}$ is the set of forecasting models in the pool, $\mathbf{F}$ is the set of feature vectors associated with models in $\mathbf{M}$, both $\mathbf{M}$ and $\mathbf{F}$ are initially empty. Variable η stores a boolean value that indicates the need to build a new forecasting model and starts with value true to indicate the need for including a new model when it receives the minimum number of time series observations required for model building.

Algorithm 2: The adaptive learning system algorithm.

Inputs: m_f, m_m, Δ, θ, k, Φ, ψ
s=1, $\mathbf{M} = \emptyset$, $\mathbf{F} = \emptyset$, $\eta = 1$;
for (each instant t a new instance x_t arrives) **do**
  if $(t - s == m_f)$ **then**
    fv_c = FE.extractFeatures($\{x_{t-s}, ..., x_t\}$)
  else if $(t - s > m_f)$ **then**
    fv_t = FE.extractFeatures($\{x_{t-m_f}, ..., x_t\}$)
    if $(t - s == mm)$ **then**
      if $(\eta == 1)$ **then**
        ϕ_i = FM.train(Φ,$\{x_{t-m_m}, ..., x_t\}$,ψ)
        $fv_{\phi_i} = FE.extractFeatures(\{x_{t-m_m}, ..., x_t\})$
        M.addModel(ϕ_i)
        F.addFeatureVector(ϕ_i)
        $\eta = 0$
      end if
    end if
    w = FW.computeFeatureWeigths($\{x_{t-s}, ..., x_t\}$)
    d_t = DD.weightedPearsonDistance(fv_c, fv_t, w)
    statistics = DD.computeCDTStatistics(d_t)
    if (DD.cdt(statistics, Δ) == drift) **then**
      trigger a drift signal
      $s = t + 1$
      for (each feature vector i in **F**) **do**
        $dists$(i) = FM.PearsonDistance(fv_t, fv_i)
      end for
      min_{dist} = argmin$\{dists\}$
      if $(min_{dist} > \theta)$ **then**
        $\eta = 1$;
      end if
    end if
  end if
  if (**M** != $\emptyset$) **then**
    for (each feature vector i in **F**) **do**
      $dists$(i) = FM.PearsonDistance(fv_t, fv_i)
    end for
    $\hat{y}(t + 1)$ = FM.forecasting(**M**, $dists$, $\{x_{t-k}, ..., x_t\}$)
  end if
end for

Step 3 is repeated when a new sample from the time series becomes available. When the method has received at m_f observations (Step 4), then the feature vector that describes the known time series concept fv_c is computed (Step 5). When the method has received more m_f observations (Step 6), it computes the feature vector that defines the time series observations on the current window (Step 7). When m_m observations are received, the algorithm verifies whether there is the need to include a new forecasting model (Step 8). In case of needing a new model, then a new model ϕ_i is created with the time series observations in the model window and with the model parameters (Step 10), the feature vector that defines the concept used to train the new model is computed (Step 11), the new model and the associated feature vector are stored (Steps 12 and 13). In Step 14, the variable η, which indicates the need to add a new model is set to 0.

In Step 17, the weights w of the features are computed based on one of the two weighting strategies explained in Section 4.2.2. In Step 18, the weighted distance (d_t) between the reference feature vector, which defines the known concept (fv_c), and the current feature vector in time t (fv_t) is computed using the weighted Pearson correlation distance. In Step 19, the statistics used by the CDT are updated with the new distance received, and in Step 20 the algorithm verifies if the statistics monitored are higher then the drift threshold Δ. This step is different for the three CDTs investigated. If a drift is detected, then a drift signal is triggered (Step 21) and the start of the new concept is updated to $t + 1$ (Step 22).

In Steps 23 to 25, the algorithm computes the minimum distance between the current feature vector and the feature vectors associated to the models already created. If the distance between the current feature vector and the closest feature vector stored is higher than a threshold θ (Step 27), then the algorithm identifies that there is the need to add a new forecasting model (Step 28), since the existing ones are not able to handle the new concept faced properly. So, as soon as the method received m_m observations of the new concept, a new forecasting model should be created and stored in $\mathbf{M}$.

In Step 32, the algorithm verifies if there is at least one forecasting model to make the prediction of the future observation $\hat{y}$. In positive case, the algorithm computes the distances between the feature vectors associated to the existing forecasting models (Step 33 to 35) and makes the forecast of $\hat{y}$ based on the existing models and using one of the two investigated schemes of output combination (Step 36). The distances computed are used by this function to compute the outputs.

4.4 Summary

In this chapter we introduced the proposed adaptive learning system for time series forecasting, FW-FEDD, which is composed by a novel explicit concept drift detection

module and a forecasting module, which is a pool of OS-ELMs. FW-FEDD is an online feature-based drift detection method which detects changes in time series by monitoring the features that describe concepts. The forecasting module is composed by several individual models associated with feature vectors that describe the concepts in which the models were created. The main idea is that each forecasting model is responsible for forecasting a particular time series concept. This approach allows handling recurring concepts without increasing the computational complexity of the method. In case of a recurring concept appears, the model created in the past to handle that concept is ready for use.

The main contributions of this thesis are (i) the use of statistical time series features to explicitly detect concept drifts in time series, which attempts to answer the first and second research questions described in Chapter 1; (ii) a heuristic feature weighting function to determine the importance of the features to the drift detection process, which attempts to answer the second research question; and (iii) a forecasting method that uses a pool of forecasting models specialized in different concept drifts, which attempts to answer the fourth research question.

5 COMPUTATIONAL EXPERIMENTS

In this chapter we describe the computational experiments performed in order to evaluate the proposed method discussed in Chapter 4 in both artificial and real-world datasets. In this chapter, we discuss the experimental objectives, design, parameter setting, figures of merit and the data sets used in order to make this investigation reproducible. Then, we present and analyze the main results obtained in these experiments and how they induce the validation of the hypotheses formulated in Chapter 1.

5.1 Experimental Objectives, Design and Measures Analyzed

The objectives of the experiments described in this chapter are to answer the four research questions investigated in this thesis and validate the formulated hypothesis, as presented before in Section 1.3. The first question concerns whether using time series features to identify concept drifts is more effective than using the time series data directly or monitoring the error of a forecasting model. In order to do so, in Section 5.3.1 we analyze the concept drift detection accuracy achieved by applying the ICI-based CDT on the set of features described in Section 4.2.1 individually in comparison to (i) approaches based on monitoring time series raw data, namely the approach proposed by Ross (2013), which used the Mood non-parametric statistical test (MOOD, 1954) as a change point method, and with a similar approach but using Lepage statistical test (LEPAGE, 1971) instead Mood; and (ii) approaches based on monitoring the residuals of a fitted model, namely at the ECDD, the PHt, and the ICI-based CDT applied on the error of the Extreme Learning Machine (ELM). The choice of ELM as the algorithm used to build regression models is due to the fact that ELM has been widely used in regression and time series forecasting with good generalization performance, besides presenting a very fast training (HUANG et al., 2012). The concept drift tests were adapted to work with regression errors of time series forecasting, instead of classification.

The second objective of the experiments is to evaluate whether we can improve the feature-based drift detection by applying a concept drift test on the features in combination, as described in Section 4.2.3 instead of individually. We expect that this approach is able to reduce the number of false alarms of the method, by reducing the sensitivity of the drift detection. In order to do so, in Section 5.3.2 we investigate the application of ECDD, PHt and ICI on the distances between feature vectors against the approach that applies ICI on features in isolation and against the methods based on forecasting error.

The third objective is to evaluate if the proposed feature weighting strategies described in Section 4.2.2 are able to improve the concept drift detection of the methods

which analyze features in combination. The weighting strategy heuristic gives more importance to features that best describes the time series concepts. So, in Section 5.3.3 we compare each concept drift test applied on combined features with and without the weighting strategies.

The fourth objective of the experiments is to evaluate if the proposed forecasting approach integrated with the proposed explicit drift detection which uses feature extraction and feature weighting to compose the adaptive learning system is effective in forecasting time series. In order to do so, in Section 5.4 we perform two sets of experiments, one to evaluate the best scheme of combination of individual forecasters and another two evaluate the performance of the proposed method in comparison to passive and active adaptive learning systems.

Since the three first questions are related to drift detection performance and the fourth is related to forecasting accuracy, we divided the experiments into two parts: (i) drift detection evaluation (Section 5.3) and (ii) forecasting evaluation (Section 5.4).

The performance metrics used to evaluate the drift detection accuracy of the compared methods are: (i) the number of false alarms, (ii) the number of miss-detections and (iii) the drift detection delay. These metrics are computed for each method in each time series individually. A false alarm is a false positive detection (type-I error), and consists in the detection of a drift in an instant that there is no drift occurrence. A false alarm is computed when the method detects a concept drift before the instant a real drift occurs in data. A miss-detection is a failure in detecting a drift when it actually happens in the data stream, configuring a type-II error. The drift detection delay is the amount of time instants the algorithm needed until detecting the occurrence of a drift. The drift delay is calculated as the average of the delays presented by the method considering all drifts that exist in the time series. When a miss-detection occurs, all the time series observations that belong to the missed concept are counted as delay of the method. It is worth noting that, despite the fact that miss-detection and detection delay are related metrics, they describe different aspects of the detection process. The metric used to evaluate the forecasting accuracy is the Mean Absolute Percentage Error (MAPE) which measures the percentage regression error. MAPE was chosen because it is a popular regression error measure in the literature. We also evaluate the number of individual forecasting models created during the time series processing. This metric is an indicative of the computational complexity of approaches that use more than one forecasting model.

5.2 Data Sets

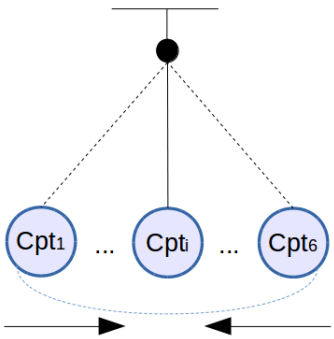
Despite the fact that concept drift is not a new research area, the effects of concept drift in time series are not widely studied. There is a lack of appropriate data sets aimed

for studies of concept drift in regression tasks (SOARES; ARAÚJO, 2015) as well as in time series analysis. By working with real-world data sets, it is not possible to know exactly when a drift effectively occurs, the kind of drift or even if there is a drift on the data (MINKU; YAO, 2012). Artificial data sets, on the other hand, allow an effective analysis of the drift detection performance. In order to evaluate the drift detection accuracy of the FW-FEDD, we first used artificial datasets. Then, in order to reaffirm the analysis of FW-FEDD, we performed experiments using seven real-world time series. Section 5.2.1 describes the artificial data sets created in this thesis and Section 5.2.2 describes the real-world data sets.

5.2.1 Artificial Data Sets

In this work, we create artificial datasets which comprise time series with linear and nonlinear behaviors affected by abrupt concept drifts (Table 2). The simulated time series have 11.000 data points and present 10 concept drifts in the instants $T^* \in \{1000, 2000, 3000, ..., 9000, 10000\}$. These artificial time series are grouped in four datasets: (i) autoregressive time series (AR), (ii) linear seasonal time series (LS), (iii) nonlinear time series (NL) and (iv) random walk-autoregressive-nonlinear time series (RW-AR-NL). The different types of time series were briefly described in Chapter 2. Each time series group is composed of 20 time series randomly generated by the same time series model. For each time series group, six different concepts are simulated and the changes follow a pendulum motion scheme, in which concepts changes from the first to the second to the third, until the sixth concept, and then from the sixth to the fifth and to fourth, and so one. The idea of using the pendulum motion scheme (Figure 13) is to evaluate the behavior of the methods in the presence of recurrent concept drifts. This scheme was used by Nasiri, Meybodi and Ebadzadeh (2016) to evaluate a particle swarm optimization (PSO) method with memory in changing environments.

Figure 13 – Pendulum motion of concepts.

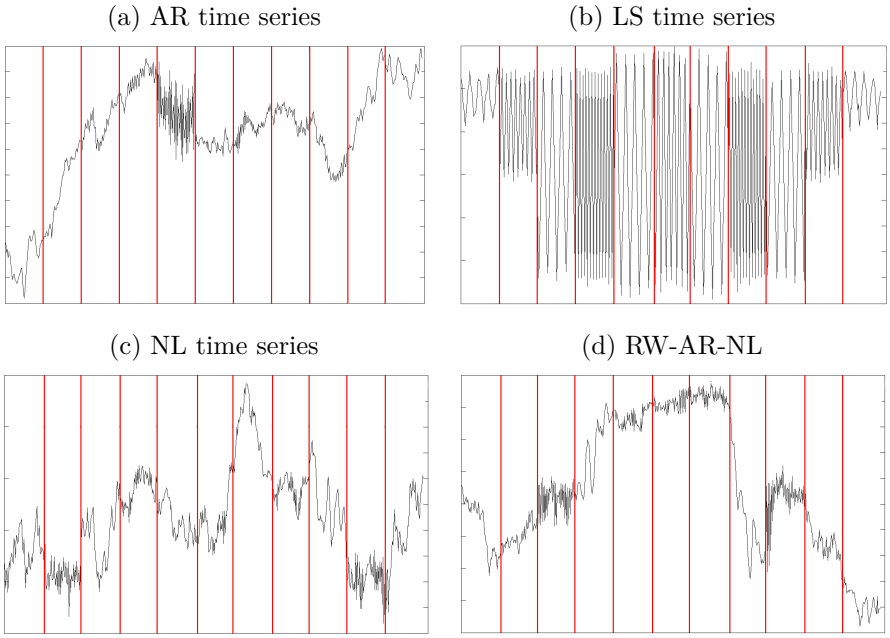


Source: adapted from Nasiri, Meybodi and Ebadzadeh (2016).

The AR time series group is composed of time series simulated by an autoregressive

process to generate the time series points. A time series $\{x_t\}$ is an autoregressive process of order p, abbreviated as $AR(p)$, if its points can be defined as $x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + ... + \alpha_p x_{t-p} + w_t$, where $\alpha_1, ..., \alpha_p$ are the model parameters and $\{w_t : t = 1, ..., n\}$ is a Gaussian white noise time series where the variables $w_1, ..., w_n$ are independent and identically distributed and follow a normal distribution ($w_t \sim N(0, \sigma^2)$) (COWPERTWAIT; METCALFE, 2009). Concept drifts were simulated by (i) changing the parameters α_i of the AR process and (ii) changing the order p of the process. The white noise random factor makes the 20 time series of this group be different from each other. The parameters of the AR that defines each concept are described in Table 2. Figure 14a illustrates an example of time series of this group. The vertical red bars indicate concept drift points.

Figure 14 – Artificial time series.



Source: elaborated by the author.

The LS time series group is composed of time series simulated by a linear model with seasonal variables of s seasons. A linear seasonal model containing s seasons can be defined as $x_t = m_t + s_t + w_t$, where m_t is the trend component, s_t is the seasonal factors and w_t is a Gaussian noise. The model used to generate the 20 time series of this group has trend $m_t = 0$ and $s_t = \beta_i$. We can rewrite this model as $x_t = m_t + \beta_{1+mod(t-1,s)} + w_t$ $(t = 1, ..., n; i = 1, ..., s)$. Concept drifts were simulated by (i) changing the parameters β_i and (ii) changing the number of seasons s of the model, as described in Table 2. The white noise random factor makes the 20 time series of this group be different from each other. Figure 14b illustrates an example of time series of this group.

The NL time series group is composed of time series simulated by two smooth autoregressive nonlinear time series models adapted from Zhang, Patuwo and Hu (2001).

Concept drifts were simulated by changing (i) the parameters α and the nonlinear model. The first three concepts were generated by smooth nonlinear model 1 (eq. 5.1) defined by and the other three concepts were generated by smooth nonlinear model 2 (eq. 5.2). The white noise random factor makes the 20 time series of this group be different from each other. The parameters of the NL that defines each concept are described in Table 2. Figure 14c illustrates an example of time series of this group.

$$x_t = [\alpha_1 x_{t-1} + \alpha_2 x_{t-2} + \alpha_3 x_{t-3} + \alpha_4 x_{t-4}] * [1 - exp(-10 x_{t-1})]^{-1} + w_t \tag{5.1}$$

$$x_t = \alpha_1 x_{t-1} + \alpha_2 x_{t-2} + [\alpha_3 x_{t-1} + \alpha_4 x_{t-2}] * [1 - exp(-10 x_{t-1})]^{-1} + w_t \tag{5.2}$$

The RW-AR-NL time series group is composed of time series simulated by a combination of random walk (RW), AR of order p and nonlinear time series models. The random walk is an AR(1) time series model. Concept drifts were simulated by changing the time series model and the parameters of the model. So, the data points of the first concept is generated by an AR(1) proces. Then the second concept is generated by an AR(4) process. The the third concept is generated by the smooth nonlinear model 1 and so on, as described in Table 2. Figure 14d illustrates an example of time series of this group.

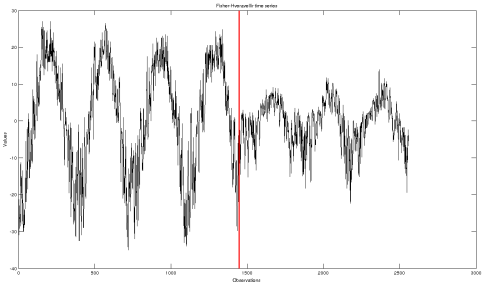
5.2.2 Real-World Data Sets

In the literature there is a lack of open real-world datasets of time series with concept drifts. Most of the existing real-world time series used in concept drift studies are time series in which the drifts consist in simple changes in the mean and/or variance of the data points. In this work, we use two real-world datasets: (i) daily temperatures time series and (ii) stock indices time series. The first data set was inspired by the ideas proposed by Boracchi and Roveri (2014). In that work, authors investigated a specific kind of real world time series which present well defined seasonal pattern. Those time series describe the water demand in Barcelona city. Concept drifts were simulated in those time series with some different strategies: (i) simulating leaks in pipes or junction by adding a factor to the time series observations; (ii) sensor degradation, by adding a Gaussian noise to time series observations and (iii) source changes, by juxtaposing two different time series. We use this third strategy, the source changes, to build the daily temperatures time series dataset. To do so, we chose some time series from the same domain and which are similar in behavior, but not identical, and we juxtaposed them. With this approach, we know in advance where the concept drifts happen in the time series and the drift detection methods can be properly evaluated. The chosen series are of daily temperatures obtained in DataMarket Time Series Data Library[1]. Figure 15 shows some examples of the resulting time series dataset after juxtaposition. This dataset is composed by 13 time series.

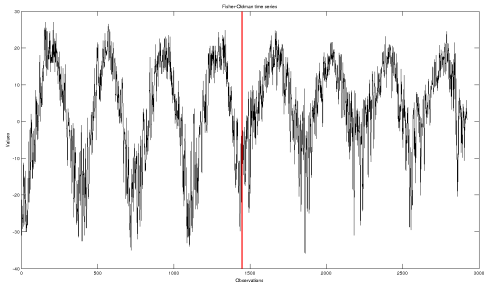
[1] https://datamarket.com/data/list/?q=provider:tsdl

Figure 15 – Daily temperature time series.

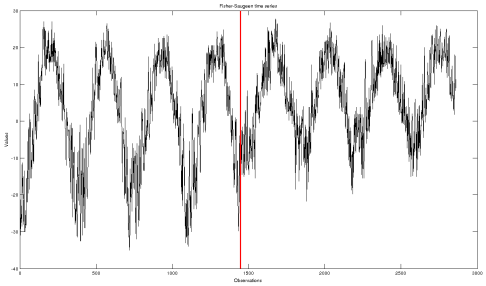
(a) Fisher River and Hveravellir



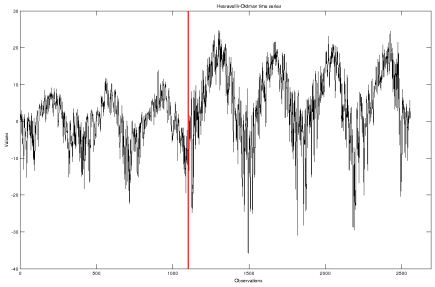
(b) Fisher River and Oldman River



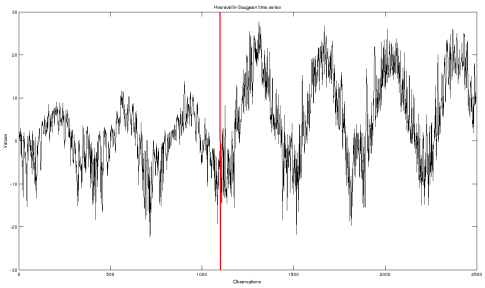
(c) Fisher River and Saugeen River



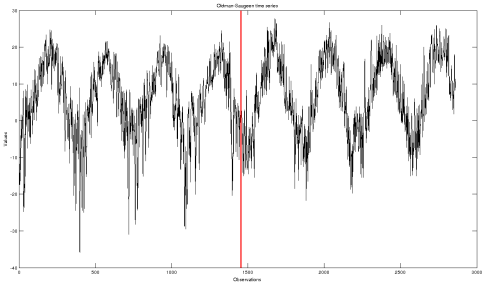
(d) Hveravellir and Oldman River



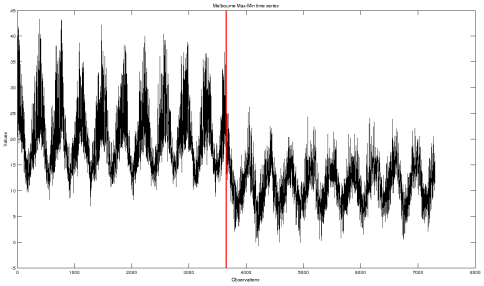
(e) Hveravellir and Saugeen River



(f) Oldman River and Saugeen River



(g) Melbourne maximum and minimum temperatures



Source: elaborated by the author.

Table 2 – Artificial time series data set description.

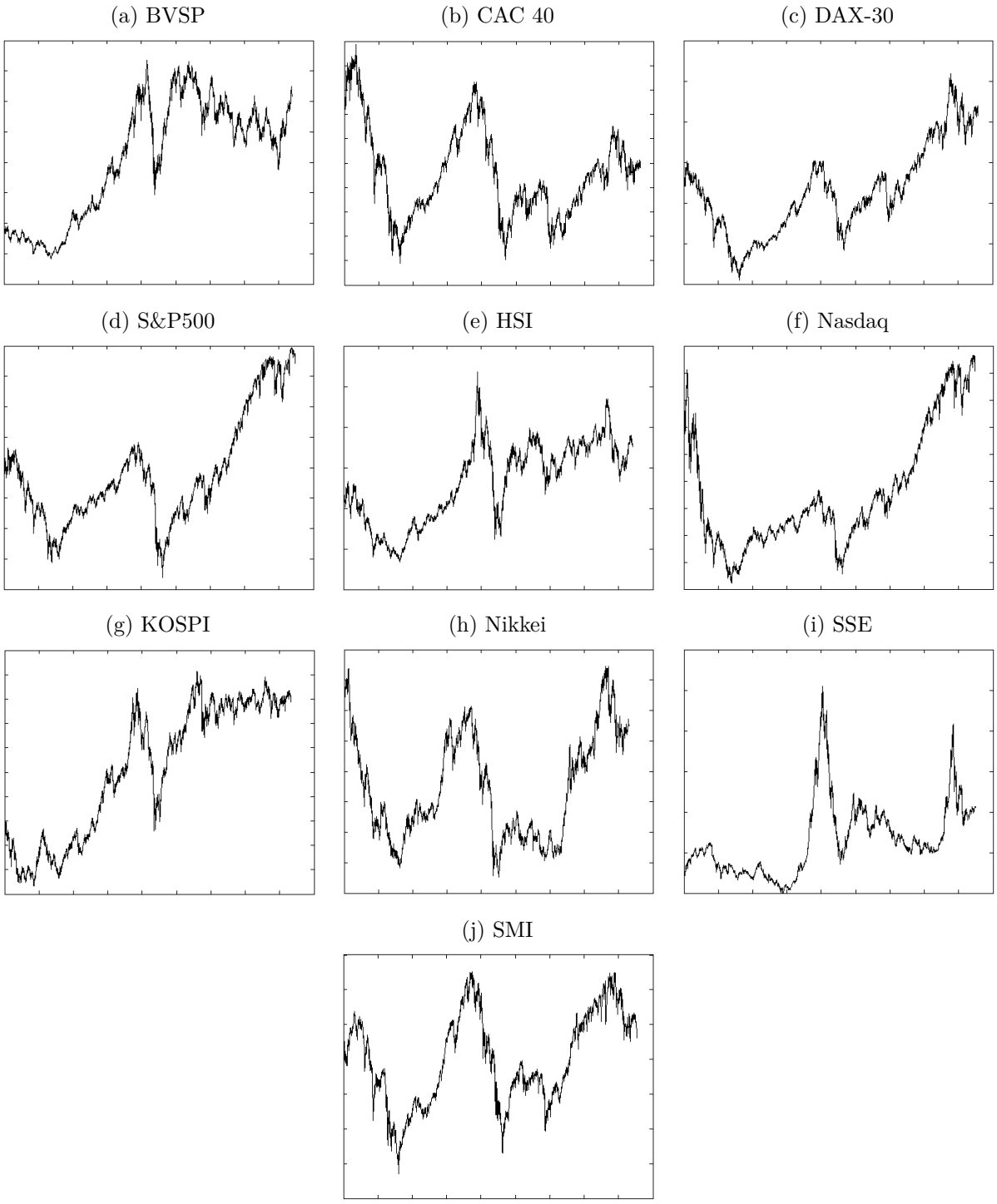
Group	**Cpt**	**Parameters**	
		α	p
AR	1	$\{1.5, -0.4, -0.3, -0.2\}$	4
	2	$\{1.2, -0.3, 0.1\}$	3
	3	$\{0.9, -0.2, 0.8, -0.5\}$	4
	4	$\{1.1, -0.6, 0.8, -0.5, -0.1, 0.3\}$	6
	5	$\{-0.1, 1.4, 0.4, -0.7\}$	4
	6	$\{0.9, 0.1\}$	2
		β	s
LS	1	$\{34, 32, 30, 28, 26, 24, 22, 24, 26, 28, 30, 32\}$	12
	2	$\{34, 26, 18, 10, 18, 26\}$	6
	3	$\{34, 26, 18, 10, 2, -6, -14, -6, 2, 10, 18, 26\}$	12
	4	$\{34, 10, -14, 10\}$	4
	5	$\{38, 28, 18, 8, 0, -8, -18, -8, 0, 8, 18, 28\}$	12
	6	$\{38, 18, 0, -16, 0, 16, 32\}$	6
		α	Model
NL	1	$\{0.9, -0.2, 0.8, -0.5\}$	smooth1
	2	$\{-0.3, 1.4, 0.4, -0.5\}$	smooth1
	3	$\{1.5, -0.4, -0.3, 0.2\}$	smooth1
	4	$\{-0.1, 1.4, 0.4, -0.7\}$	smooth2
	5	$\{0.2, 0.3, 0.6, -0.1\}$	smooth2
	6	$\{-0.1, 0.8, 0.5, -0.2\}$	smooth2
		α	Model
RW-AR-NL	1	$\{1\}$	RW
	2	$\{0.9, -0.2, 0.8, -0.5\}$	AR
	3	$\{-0.3, 1.4, 0.4, -0.5\}$	smooth1
	4	$\{1.5, -0.4, -0.3, 0.2\}$	AR
	5	$\{-0.1, 1.4, 0.4, -0.7\}$	smooth2
	6	$\{0.2, 0.3, 0.6, -0.1\}$	smooth1

Source: elaborated by the author.

The second dataset is composed by 10 time series of the main stock indices in the world, namely the BVSP (São Paulo), CAC 40 (Paris), DAX-30 (Frankfurt), S&P500, HSI (Hong Kong), Nasdaq index, KOSPI (Korea), Nikkei 225 (Tokio), SSE (Shanghai), SMI (Swiss). All time series data were collected in Yahoo Finance repository[2] from 03/01/2000 to 04/11/2016. Each time series have approximately 4300 data observations (they have not exactly the same number of data points due to differences in the number of working days in each country). These time series are plotted in Figure 16. In these time series, we cannot know in advance when the drifts happen, an even whether there is concept drifts in these time series. So these series were used just in the evaluation of the forecasting module.

[2] https://finance.yahoo.com

Figure 16 – Stock indices time series.



Source: elaborated by the author.

Since we cannot known in advance the existence or absence of concept drifts in these real-world time series, we simulated 3 changes in the behaviors of these series to create 10 more time series with known concept drifts. These changes happen in instants $t = \{1000, 2000, 3000\}$. The simulated changes are: (i) the addition of a seasonal factor to the time series points, computed as in eq 5.3, where x_t are the original time series points, x_t^{seas} are the modified time series points, $\beta = \{20, 15, 10, 5, 0, -5, -1, -5, 0, 5, 10, 15\}$ and $s = 12$; (ii) the addition of a nonlinear time series behavior x^{nlin} to the time series points, where x^{nlin} is computed as in eq 5.1 using $\alpha = \{0.9, -0.2, 0.8, -0.5\}$; and (iii) the addition of a harmonic time series behavior to the time series points, as computed in eq. 5.4. The resulting time series are shown in Figure 17. Vertical bars indicate the concept drift instants. These time series were also used just to evaluate the forecasting module, due to the fact that we do not known if there are more drifts other than the artificially introduced ones.

$$x_t^{seas} = x_t + \beta_{1+mod(t-1,s)}, t = 1, ..., n; i = 1, ...s) \tag{5.3}$$

$$x_t^{harm} = x_t + sin(2 * \pi * t/12) \tag{5.4}$$

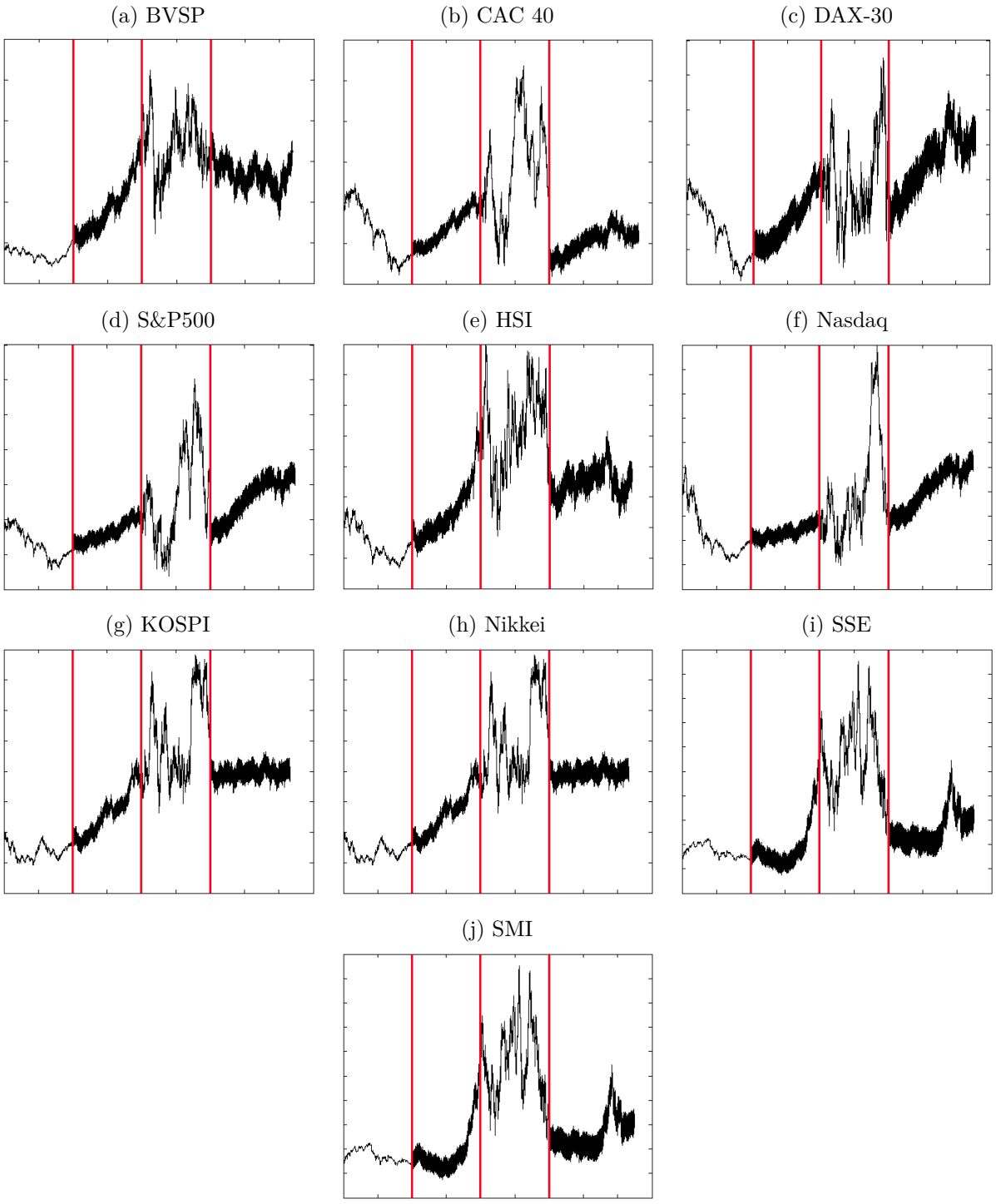
5.3 Experimental Results of the Drift Detection Evaluation

5.3.1 Using Features to Detect Concept Drift

In these experiments, we evaluate whether the concept drift approach based on monitoring time series features is more accurate than those methods based on monitoring time series raw data or based on monitoring the forecasting error of a forecasting model. In order to do so, we compare the ICI-based CDT applied on features in isolation (ICI$_{_ind-feat}$) with Mood, Lepage and with the application of ECDD, PHt and ICI on the forecasting errors of ELM (ECDD$_{_ELM}$, PHt$_{_ELM}$, and ICI$_{_ELM}$, respectively). Since Mood is a non-parametric statistical test which assesses just changes in the variance, we also include a similar approach but using the Lepage test, which assesses both changes affecting the mean and variance. The Lepage test was used by Alippi, Boracchi and Roveri (2013b) to identify change-points using an ensemble of change-point methods. The Mood and Lepage-based approaches are applied to the de-trended time series (using first order differencing), since they are based on the assumption that the monitored signal is independent and identically distributed. This is the same approach as used by Ross (2013). ICI$_{_ind-feat}$ detects a concept drift as soon as the first feature fires a change.

In order to assess the statistical significance of the results, we use the Friedman non-parametric test (FRIEDMAN, 1940), with confidence level $\alpha = 0.05$, according to the

Figure 17 – Stock indices time series with simulated concept drifts.



Source: elaborated by the author.

Table 3 – Parameter values used in grid search of drift detection.

Parameter	**Methods**	**Values**
ARL_0	Mood	$\{100, 200, 300, 400, 500\}$
γ	ICI_{ind_feat}, ICI_{feat}	$\{2.0, 2.5, 3.0\}$
TS	ICI_{ind_feat}, ICI_{feat}, ICI_{ELM}	$\{100, 200, 300, 400\}$
λ	ECDD_{ELM}, ECDD_{feat}	$\{0.1, 0.2, 0.3\}$
δ	PHt_{ELM},PHt_{feat}	$\{0.005, 0.01, 0.03, 0.05\}$
W	ECDD_{ELM}, PHt_{ELM}, ECDD_{feat}, PHt_{feat}	$\{0.5, 1.0, 1.5, 2.0\}$
C	ECDD_{ELM}, PHt_{ELM}, ECDD_{feat}, PHt_{feat}	$\{1.0, 1.5, 2.0, 2.5\}$
m_f	ICI_{ind_feat}, ICI_{feat} ECDD_{feat}, PHt_{feat}	$\{100, 150, 200, 250\}$
mm	ECDD_{ELM}, PHt_{ELM}, ICI_{ELM}	$\{100, 200, 300\}$

Source: elaborated by the author.

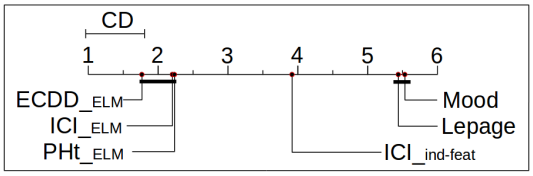
approach proposed by (DEMSAR, 2006). This evaluation approach allows the simultaneous comparison of several methods considering different data sets. The null-hypothesis is that there is no significant differences between the approaches across datasets. If the null-hypothesis is rejected, the Nemenyi *post hoc* test (NEMENYI, 1962) with 95% confidence is used to identify the best results.

We performed a grid search to identify the best parameter settings for each method compared. The best parameters for a method are those which minimize the number of drift detection errors, computed as the sum of false alarms and miss-detections. In case of ties, the parameter setting which provides the lower number of miss-detections is chosen. If the tie still remains, then the lowest drift detection delay is used in the tiebreaker. The sets of parameter ranges considered for the grid search are shown in Table 3. ARL_0, used in Mood and Lepage, corresponds to the average number of observations before a false positive occurs. The parameter γ is the confidence parameter used in ICI-based methods. TS defines the amount of observations used to model the initial confidence intervals of ICI. λ indicates the weight given to recent data when compared to older data in computing the EWMA on ECDD. δ is the discount factor is computing the cumulative differences of PHt. W and C are the warning and drift threshold, respectively, used in both ECDD and PHt. m_f is the window size used in the feature-based methods. mm is the amount of observations used to model building in the error-based methods.

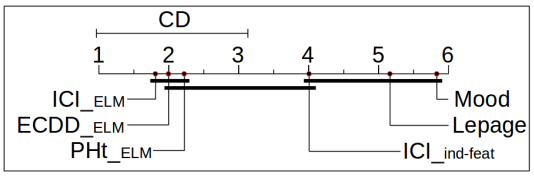
Figure 18 presents the Friedman ranks with the Nemenyi critical difference for the three metrics evaluated for the artificial time series dataset (on left) and real-world time

Figure 18 – Comparison of ICI$_{_ind-feat}$, Mood, Lepage, ECDD$_{_ELM}$, PHt$_{_ELM}$ and ICI$_{_ELM}$ against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

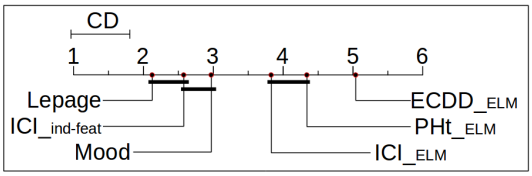
(a) False Alarms: artificial



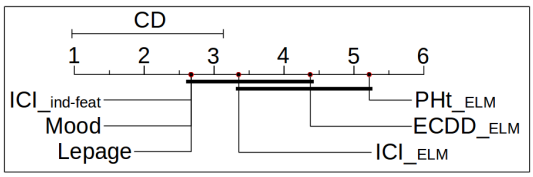
(b) False Alarms: real-world



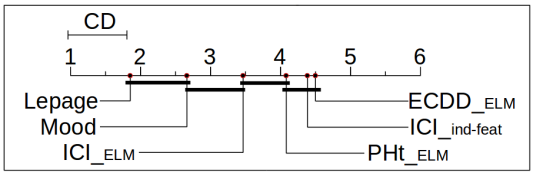
(c) Miss-detections: artificial



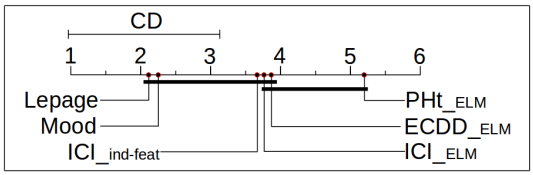
(d) Miss-detections: real-world



(e) Delay: artificial



(f) Delay: real-world



Source: elaborated by the author.

series (on the right). Methods that are not significantly different (at $p = 0.05$) have ranks which differ by at least the critical difference (CD). In terms of false alarms, the ELM-based methods presented better results in both artificial and real-world datasets. Mood and Lepage presented the highest number of false alarms, followed by the ICI$_{_ind-feat}$. The high number of false alarms causes a decrease in the number of miss-detections, since some of these false alarms may coincide with legitimate drifts. So, ICI$_{_ind-feat}$, Mood and Lepage presented the best ranks in terms of miss-detections. In terms of drift detection delay, Lepage presented the best result, followed by Mood and ICI$_{_ELM}$. The residual-based presented a high delay. This is due to the fact that these methods are based on the forecasting error, that needs to reach certain levels before firing a concept drift, which may increase the detection delay. It is important to note that in the real-world time series, the results are almost always statistically equivalent to each other. This is due to the fact that the CD is higher in these tests due to the lesser number of series evaluated.

These results show that the ICI applied on individual features present a higher number of false alarms compared to the error-based methods. This is due to the fact that this approach is very sensitive to changes in one or in a small set of features. The Mood and Lepage methods presented the lowest delay, but with the highest number of false alarms. Among the ELM-based drift detection methods, the ICI$_{_ELM}$ presented the best

trade-off between false alarms and miss-detections, so it can be considered the method with the best results over all datasets. We can conclude from these results that the ICI applied on features individually provided better results than Mood and Lepage, which tries to detect drifts directly on time series data, but was not better than monitoring forecasting error.

5.3.2 Using features in Combination Instead of Individually

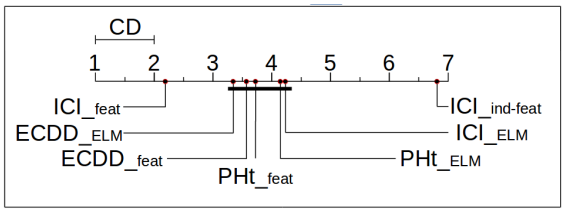
The results presented in previous section confirmed what was expected about applying a CDT on features individually: it fires a high number of false alarms in comparison with monitoring the error, but it is better than monitoring the raw time series observations. In order to tackle this problem, we propose to consider time series features in combination instead of individually, as described in Section 4.2.3. In order to answer the question whether monitoring features in combination is better then monitoring features individually, we compare the application of concept drift tests on the distances between feature vectors against the version applied on features individually. As a second goal of these experiments, we also compared this approach with the error-based drift detection methods. Since by working with distances we have just a univariate signal, we applied other concept drift tests on the distances besides the ICI, namely the ECDD and PHt, in order to make a fair comparison with the error-based methods. These tests applied on distances between feature vectors are referred as ICI$_{_feat}$, ELM$_{_feat}$ and PHt$_{_feat}$.

Figure 19 presents the Friedman ranks with the Nemenyi critical difference for the three metrics evaluated in both artificial and real-world time series. Methods that are significantly different (at $p = 0.05$) have ranks which differ by at least the critical difference. In terms of false alarms, the ICI$_{_feat}$ presented statistically the best overall results in the artificial time series and statistically equivalent results in the real-world ones. The other feature-based methods applied on features in combination and the error-based methods presented statistically equivalent results to each other in both artificial and real-world cases. The ICI$_{_ind-feat}$ presented the overall worst results. In terms of miss-detections, the feature-based methods presented the best results. Again, the ICI$_{_ind-feat}$ presented small number of miss-detections because of the high number of false alarms. ICI$_{_feat}$ and ECDD$_{_feat}$ presented the lower drift detection delay among all compared methods in the artificial datasets. In the real-world datasets, ICI$_{_feat}$ presented the best ranks, but statistically equivalent to the other methods.

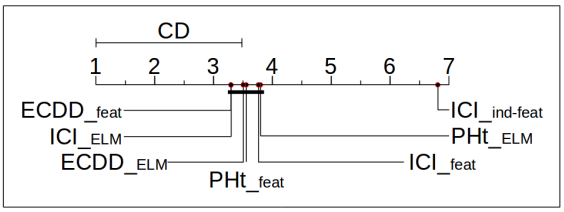
These results show that, as we expected, the application of a concept drift test to features in combination is able to reduce the number of false alarms in comparison to the approach that monitors features in isolation. The use of concept drift tests on features in combination presented better results than the error-based drift detection methods. The features that describe time series are more appropriate to indicate changes in the

Figure 19 – Comparison of ICI$_{_ind-feat}$,ECDD$_{_feat}$, PHt$_{_feat}$, and ICI$_{_feat}$, ECDD$_{_ELM}$, PHt$_{_ELM}$, and ICI$_{_ELM}$ against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

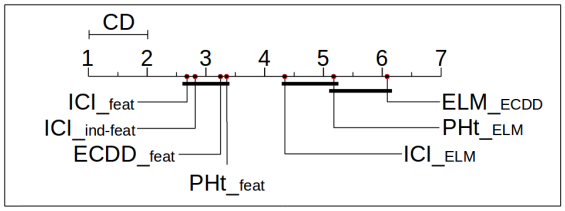
(a) False Alarms: artificial



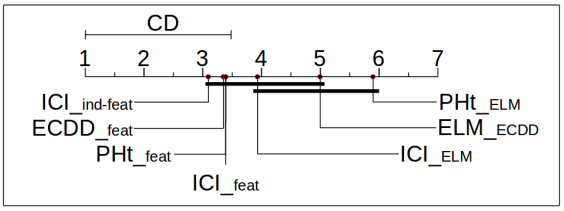
(b) False Alarms: real-world



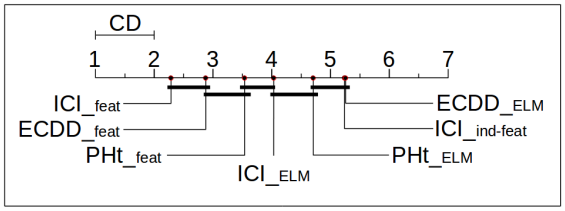
(c) Miss-detections: artificial



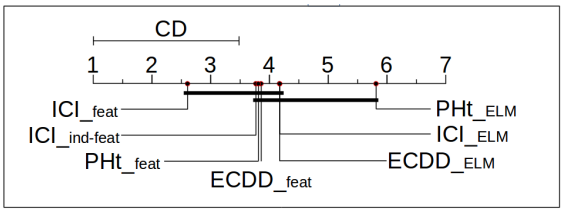
(d) Miss-detections: real-world



(e) Delay: artificial



(f) Delay: real-world



Source: elaborated by the author.

underlying data distribution than the error of a forecasting method. It also allows the reduction of the detection delay compared to error-based methods. These results also show that the ICI applied on features in combination presented the best overall ranks for all three metrics in the artificial datasets, and good ranks in the real-world datasets. Among the error-based methods, the one using ICI-based CDT present a better trade-off between false alarms, miss-detections and drift detection delay than the other methods.

5.3.3 Feature Weighting Improvement

The results on Figure 19, showed that applying a concept drift test on features in combination is more effective than applying the test on features individually or on error of a forecasting method. However, as discussed before (see Section 4.2.2), not all features are informative about concept drifts all the time. Seeking to further improve the concept drift detection, we apply the two feature weighting methods, namely the feature weighting based on PCA and the one based on standard deviation described in Section 4.2.2 to the concept drift tests investigated in this work, (ECDD, PHt, and ICI) in order to monitor features in combination. The goal of these experiments is to identify whether

the feature weighting is able to improve the concept drift detection of these tests in one or more metrics evaluated (delay, false alarms and miss-detections).

5.3.3.1 ECDD and Feature Weighting

Figure 20 present the Friedman ranks with the Nemenyi critical difference for the comparison of ECDD with and without feature weighting for the three metrics evaluated. Methods that are not significantly different (at $p = 0.05$) have ranks which differ by at least the critical difference. In terms of false alarms, the weighting strategies presented statistically better results than the no weighting strategy in the artificial datasets and exactly the same results in the real-world datasets. In terms of miss-detections, the results provided by the weighting strategies and the no weighting strategy were statistically equivalent. In terms of delay, the ECDD with no weighting strategy and with the std weighting presented statistically equivalent results in the artificial time series, and the results were statistically equivalent for the real-world time series.

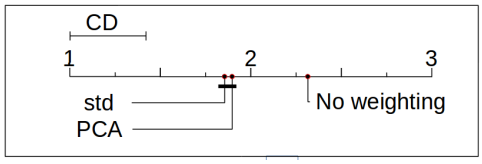
The hypothesis tests indicate that the std weighting was able to reduce the number of false alarms while keeping the number of miss-detections and drift detection delay equivalent to the no weighting strategy. The PCA weighting strategy was able to reduce the number of false alarms compared to the no weighting strategy, however it increased the drift detection delay of the CDT. Two facts may contribute to the results in the real-world time series be equivalent for all three metrics. The first of them is that this data-set contains less time series, which increase the CD of the Nemenyi test. The second one is the time series in this group have just one drift. In time series with more drifts, the wrong identification of a concept drift (false alarm or miss-detection) lead to more errors in the next drifts which happens in the time series. Figure 21 illustrates an example of what a wrong drift detection can cause in datasets sequential drifts. So, a bad drift detection method present very different results than a good one.

The ECDD test compares the differences between the EWMA and the simple mean of the distances of feature vectors. When the differences between these values are higher than a threshold, a concept drift is detected. The weighting strategies give more weights to features with lower variance. These mechanisms reduce the variance of the distances and avoid suddenly divergences among EWMA and the simple mean of distances, which consequently reduces the number of false alarms. Figure 22 illustrates the effects of std and PCA weighting strategies on distances between vectors when applied with ECDD test on the first time series of the AR time series group. The dashed lines indicates the concept drift instants. The effects of the weighting strategies on the distances are the same for the other tests investigated in this research.

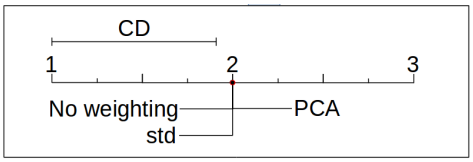
Figure 23 allows a more detailed analysis of the improvements of the weighting std strategy for each time series ($x-$axis) of each dataset (AR, LS, NL, RW-AR-NL and

Figure 20 – Comparison of ECDD without weighting and with PCA feature weighting and with std feature weighting against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

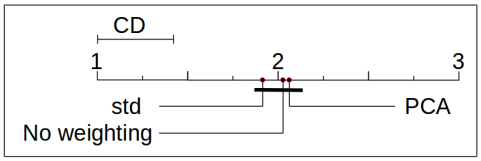
(a) False Alarms: artificial



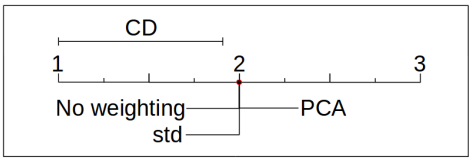
(b) False Alarms: real



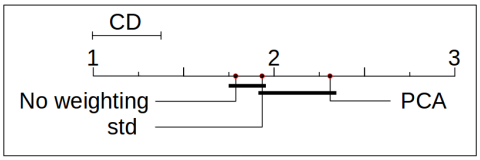
(c) Miss-detections: artificial



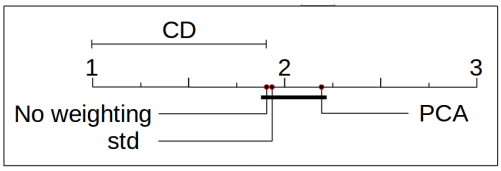
(d) Miss-detections: real



(e) Delay: artificial



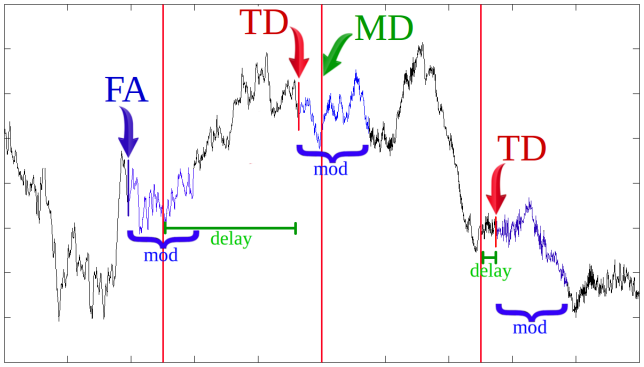
(f) Delay: real



Source: elaborated by the author.

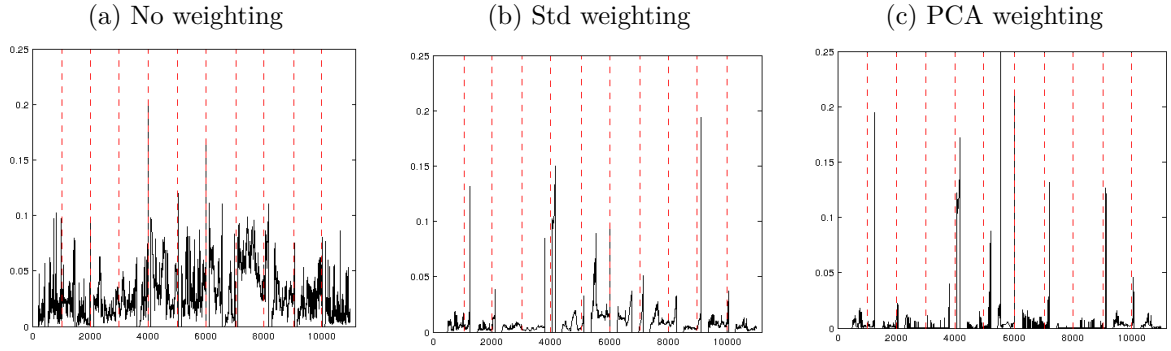
real-world) in terms of the three metrics. These plots show the differences between the results provided by the weighting strategy and the no weighting strategy. In case of the std weighting strategy have presented better result for a metric in a time series, the difference is negative, which indicates that std weighting was able to reduce the value for that metric.

Figure 21 – Example of cascade errors due to a erroneous drift identification. The vertical bars indicate concept drift instants. A false alarm (FA) causes a wrong modeling (mod) of the next concept and consequently an increase in the delay of the true detection (TD), then another wrong modeling, then a miss-detection (MD).



Source: elaborated by the author.

Figure 22 – Distances between feature vectors during time series processing with ECDD (a) without, (b) with std and (c) with PCA feature weighting strategy. The dashed lines indicate concept drift instants. The weighting strategies smooths the distances between vectors, making easier the drift detection.



Source: elaborated by the author.

If the bar is positive, it indicates that the std weighting strategy worsened the results compared to the no weighting strategy. If the difference is 0, then the result for both strategies were equal.

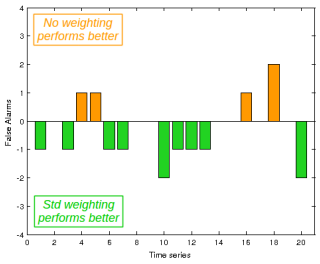
In the AR time series, std the weighting strategy triggered equal or less false alarms in 80% of the time series (Figure 23a) and presented equal or less miss-detections in 75% of the cases (Figure 23b). In terms of drift detection delay, the std weighting strategy was able to improve the results in 65% of the time series(Figure 23c). The magnitudes of the improvement provided by the std strategy in most of cases are higher than in cases where it worsened the delay.

In the LS time series, the weighting strategy provided the same number of false alarms (Figure 23d) and miss-detections (Figure 23e) than the no weighting strategy, but with a little increase in the delay in all time series of this group. This increase in the delay may be explained by the fact that, since the weighting smooths the variance of the distances, and consequently the differences between EWMA and the simple mean of the distances, so more observations are needed to confirm the concept drift.

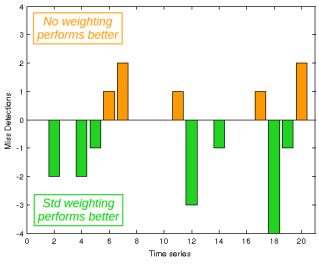
In the NL and in the RW-AR-NL time series, the weighting strategy presented equal or better number of false alarms in 95% of the time series (Figures 23g and 23j), and in terms of miss-detections, the weighting strategy present equal or better results in 90% of the cases (Figures 23h and 23k). The drift delay was increased in some cases of both time series groups due to the smooth in the distances, as explained before. The std weighting strategy provided a higher reduction in the number of false alarms in these groups of time series, compared to the AR time series. This is due to the fact that concept drifts are more well defined in these time series, which changes not just the parameters of the model that generates the time series observations but also the time series model

Figure 23 – Differences between the results provided by ECDD with std weighting and with no weighting for all time series. Values below 0 indicate an improvement of the weighting strategy.

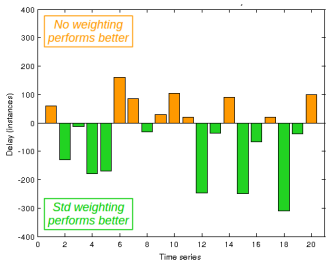
(a) AR: False alarms



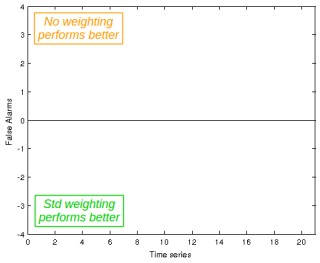
(b) AR: Miss-detections



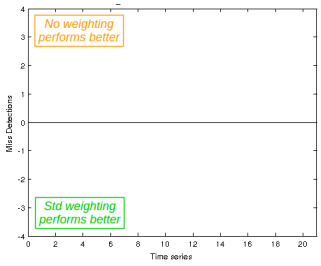
(c) AR: Delay



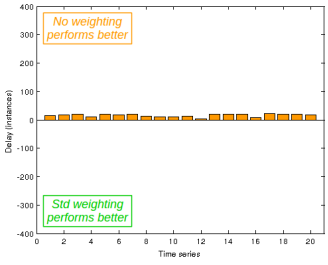
(d) LS: False alarms



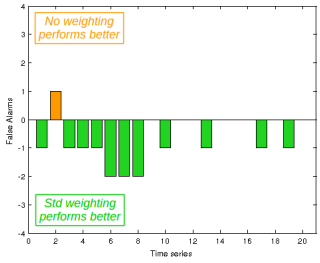
(e) LS: Miss-detections



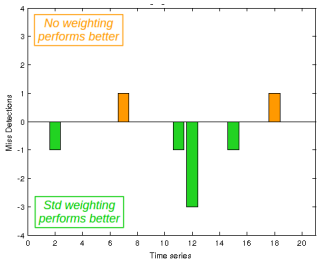
(f) LS: Delay



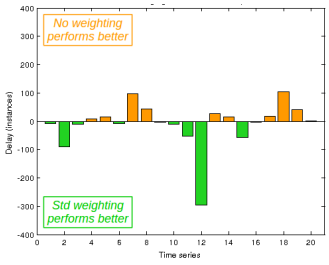
(g) NL: False alarms



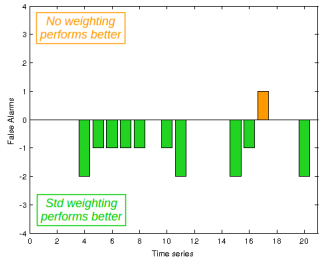
(h) : Miss-detections



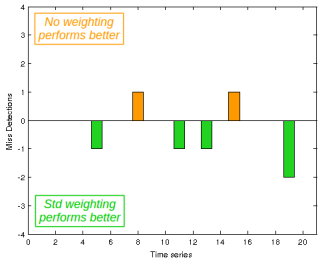
(i) NL: Delay



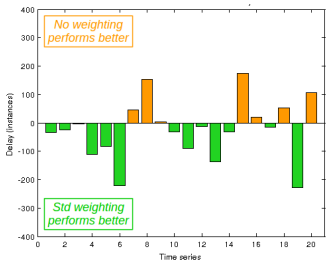
(j) RW-AR-NL: False alarms



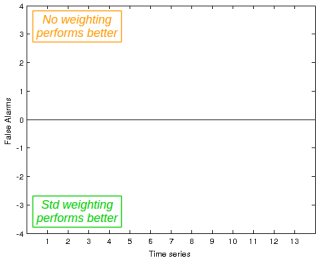
(k) RW-AR-NL: Miss-detections



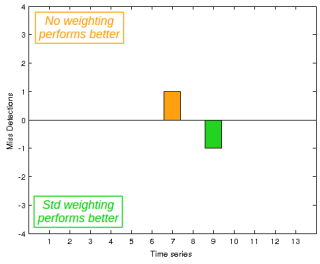
(l) RW-AR-NL: Delay



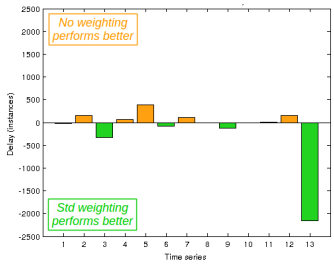
(m) Real: False alarms



(n) Real: Miss-detections



(o) Real: Delay



Source: elaborated by the author.

itself. So, features may present some changes that do not effectively imply in a concept drift. The weighting strategy then is able to identify which features are really important to identify changes.

In the real-world time series, the weighting strategy presented the same number of false alarms than the no weighting strategy (Figure 23m), and the same number of miss-detections (Figure 23n). The weighting strategy was able to improve the drift detection delay in 5 of the 7 time series. For one series, the magnitudes of the improvements was much higher than in the cases where results were worse.

Figure 24 illustrates the improvement done by the PCA weighting strategy compared to the no weighting strategy for each time series of each dataset. In the AR time series, the PCA weighting strategy was able to reduce the number of false alarms in 90% of the time series (Figure 24a), but increased the number of miss-detections (Figure 24b) and drift detection delay (Figure 24c) in almost all of them.

In the LS time series, the PCA provided equal or worse results than the no weighting strategy for all three metrics. In terms of false alarms (Figure 24d), the results were worse using PCA in 50% of the cases. In terms of miss-detections (Figure 24e), in just one time series the PCA weighting increased the number of false alarms. In terms of drift detection delay (Figure 24f), in all time series, the PCA weighting strategy increased the delay.

For the NL time series, the results achieved by PCA were equal or better than the no weighting strategy in 95% of the cases for both the number of false alarms (Figure 24g) and miss-detections (Figure 24h). However, the drift detection delay was increased in in 60% of the cases.

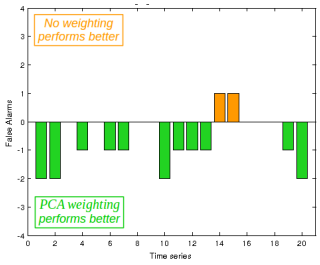
In the RW-AR-NL data set, the PCA weighting strategy presented its better results than in other time series group. In terms of false alarms (Figure 24j), it presented equal or better results than the no weighting strategy in all the time series. In terms of miss-detections (Figure 24k) it was worse than the no weighting strategy in just one case. And in terms of drift detection delay (Figure 24l), PCA provided equal or better results in 75% of the cases.

In the real-world time series, one can see that PCA was not able to improve the number of false alarms compared to the no weighting strategy (Figure 24m), but it was able to reduce the number of miss-detections in one case (Figure 24h). In terms of detection delay, the PCA strategy improved the results in four time series, but increased the delay in three.

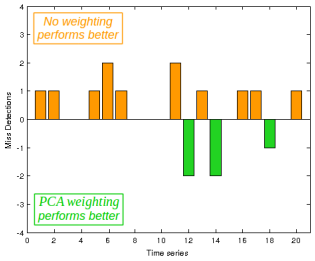
These experiments showed that the std weighting strategy was able to improvement the concept drift detection of the ECDD, being able to reduce the number of false alarms meanwhile keeping the number of miss-detections and drift detection delay constant compared to the no weighting strategy. As explained before, the std weighting strategy is

Figure 24 – Differences between the results provided by ECDD with PCA weighting and with no weighting for all time series. Values below 0 indicate an improvement of the weighting strategy.

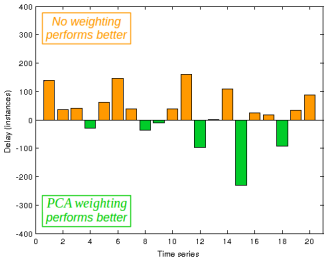
(a) AR: False alarms



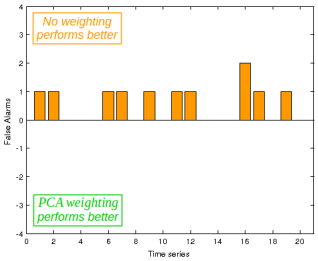
(b) AR: Miss-detections



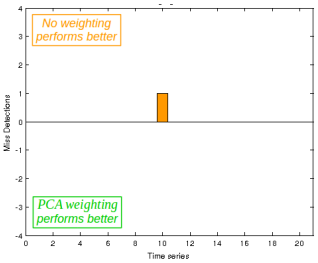
(c) AR: Delay



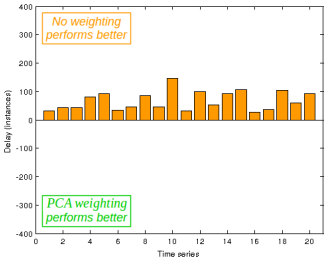
(d) LS: False alarms



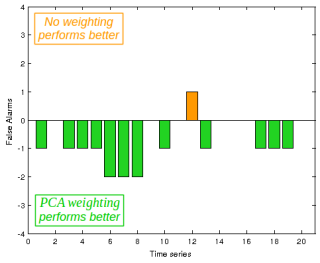
(e) LS: Miss-detections



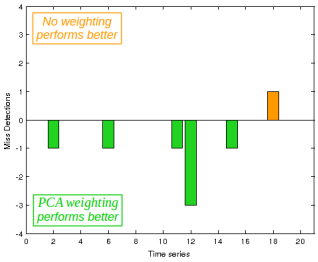
(f) LS: Delay



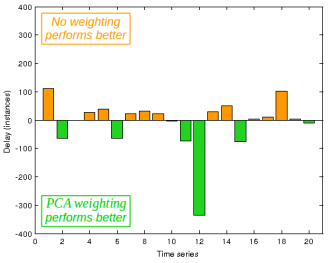
(g) NL: False alarms



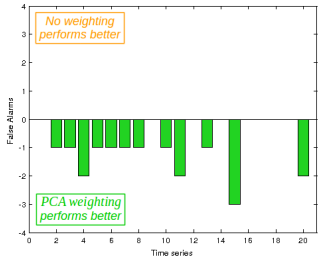
(h) NL: Miss-detections



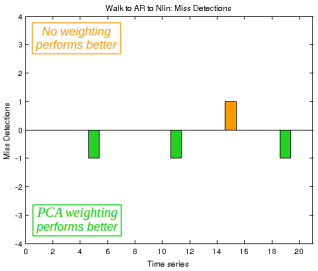
(i) NL: Delay



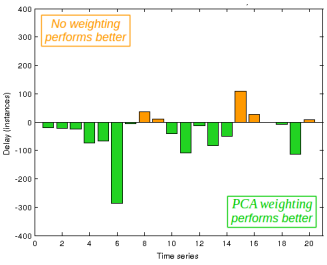
(j) RW-AR-NL: False alarms



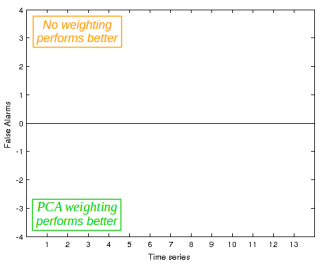
(k) RW-AR-NL: Miss-detections



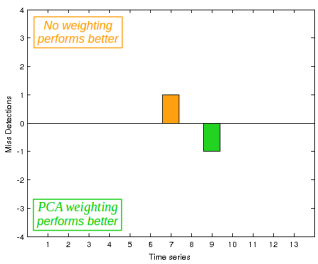
(l) RW-AR-NL: Delay



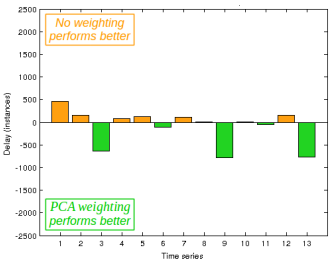
(m) Real: False alarms



(n) Real: Miss-detections



(o) Real: Delay



Source: elaborated by the author.

able to reduce the variability of the distances, which avoids some false positive detections. However, as a consequence, it may increase the drift detection delay in some cases and eventually cause miss-detections. Some false alarms may be introduced by some error in the online modeling of the new concept after a change. The PCA weighting strategy was not effective in improving the results mainly in the AR and LS time series, where the strategy worsened the results. For the NL and RW-AR-NL, the PCA strategy was able to improve the results.

5.3.3.2 PHt and Feature Weighting

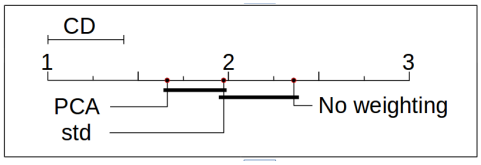
In this section we describe similar experiments to the last section but analyzing the improvement of the weighting strategies combined with PHt CDT. Figure 25 presents the Friedman ranks with the Nemenyi critical difference for the comparison of PHT with and without feature weighting for the three metrics evaluated. Methods that are not significantly different (at $p = 0.05$) have ranks which differ by at least the critical difference. In terms of false alarms, the weighting strategies presented statistically better results than the no weighting strategy in artificial data sets, and statistically equivalent results in the real-world datasets. In terms of miss-detections, the weighting strategies and the no weighting strategy presented statistically similar results for both artificial and real-world datasets. But, in terms of drift detection delay, the test indicates that the results achieved by the std weighting strategy and the no weighting strategy are statistically equivalent and both are better than the PCA weighting strategy for the artificial datasets. In the real-world datasets, the results were equivalent. Similarly to the case with ECDD CDT, the hypothesis tests indicate that the std weighting was able to improve the number of false alarms while keeping the number of miss-detections and drift detection delay equivalent to the no weighting strategy. The PCA weighting strategy was able to improve the number of false alarms compared to the no weighting strategy, however it increased the drift detection delay of the CDT.

The PHt monitors the cumulative differences between the average of distances and the minimum average distances. When this cumulative differences increase and is higher than a threshold, then the test identifies a concept drift. The weighting strategy smooths the variation of the cumulative distances, which can reduce the number of false alarms, similarly to ECDD. However, similarly, it can introduce some delay and miss-detections.

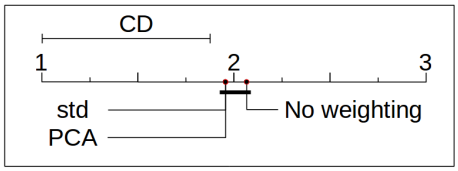
Figure 26 allows a more detailed analysis of the improvements of the std weighting strategy for each time series in each dataset in terms of the three metrics. In the AR time series, the weighting strategy provided equal or better number of false alarms in 80% of the time series (Figure 26a) and equal or better number of miss-detections in 85% of the cases (Figure 26b). The weighting strategy improved the detection delay in 65% of the time series (Figure 26c). The magnitudes of the improvements were higher than in cases

Figure 25 – Comparison of PHT without weighting and with PCA feature weighting and with std feature weighting against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

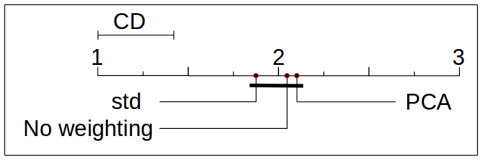
(a) False Alarms: artificial



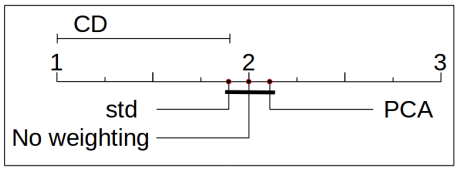
(b) False Alarms: real



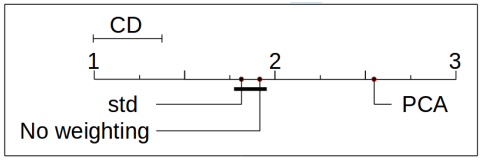
(c) Miss-detections: artificial



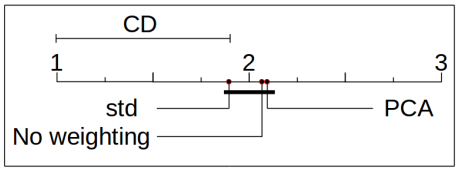
(d) Miss-detections: real



(e) Delay: artificial



(f) Delay: real



Source: elaborated by the author.

where there was deterioration in results.

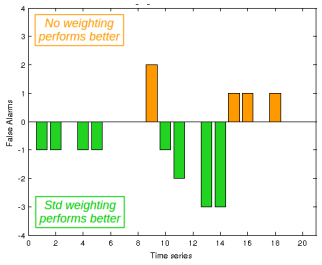
In the LS time series, the weighting strategy presented a small improvement in the drift delay in almost all time series (Figure 26f). This little reduction in the delay can be explained by the fact that the PHt compares averaged distances with the minimum averaged distances. The LS time series present well defined trend, seasonality and periodicity, and these features are very important to define drifts. Since the weighting strategy gives more importance to these features and less importance to non informative features, any changes in these features cause a significant difference between the averaged distances and the minimum averaged distances, and consequently a reduction in the drift detection delay. In terms of false alarms and drift detection delay, the results were equal.

In the NL time series, the weighting strategy provided equal or better number of false alarms in 95% time series (Figure 26g), meanwhile keeping equal number of miss-detections (Figure 26h). However, it increased the drift detection delay in almost all time series (Figure 26i). Since these time series are not so well defined as the LS, the weighting strategy needs some more observations to confirm a concept drift.

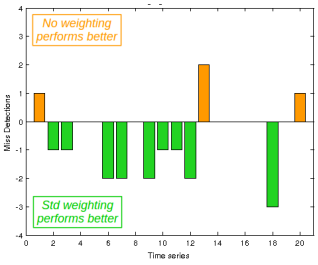
In the RW-AR-NL time series, in terms of false-alarms (Figure 26j) and miss-detections (Figure 26k), the weighting strategy was able to provide equal or better results in 85% and 100% of the time series, respectively. The improvement in the delay was higher

Figure 26 – Differences between the results provided by PHt with std weighting and with no weighting for all time series. Values below 0 indicate an improvement of the weighting strategy.

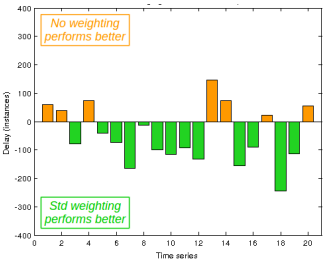
(a) AR: False alarms



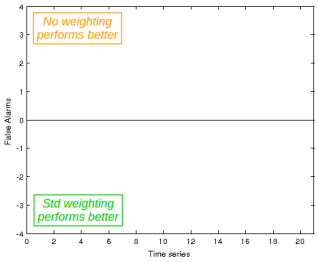
(b) AR: Miss-detections



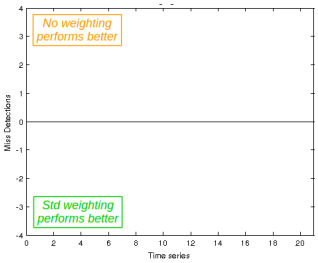
(c) AR: Delay



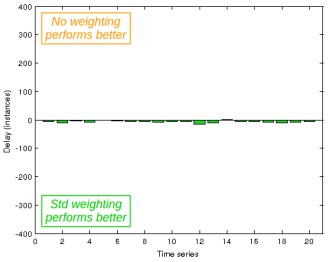
(d) LS: False alarms



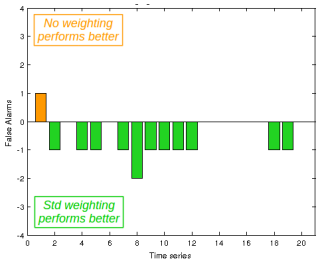
(e) LS: Miss-detections



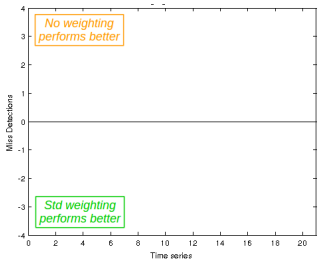
(f) LS: Delay



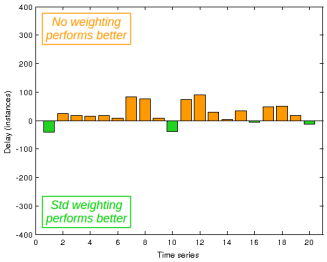
(g) NL: False alarms



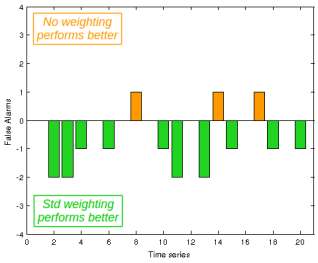
(h) NL: Miss-detections



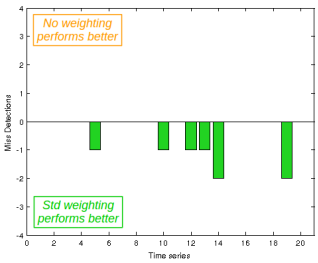
(i) NL: Delay



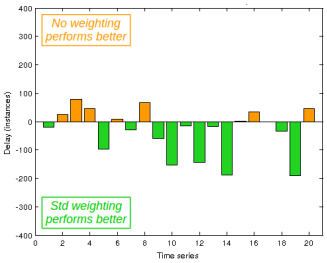
(j) RW-AR-NL: False alarms



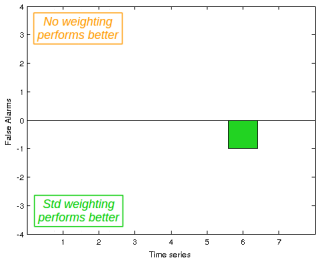
(k) RW-AR-NL: Miss-detections



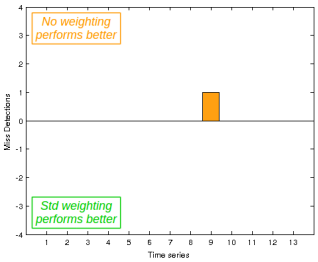
(l) RW-AR-NL: Delay



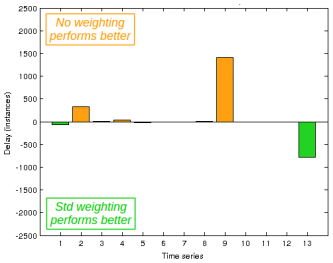
(m) Real: False alarms



(n) Real: Miss-detections



(o) Real: Delay



Source: elaborated by the author.

than in the NL time series (Figure 26l). The reason for this is the fact that concept drifts in these time series are easier to detect, since there is a change in both parameters of the time series models as in the models themselves. In this case, the weighting strategy is able to identify the more important features and detect changes more precisely.

In the real-world time series, the weighting strategy presented no significant improvements in the drift detection. In the time series 6, it was able to reduce the number of false alarms (Figure 26m), but added a miss-detection (Figure 26n). This miss-detection increased the drift detection delay in time series 6. In the other time series, the results were equal in terms of false alarms and miss-detections.

Figure 27 illustrates a similar analysis but applying PCA weighting strategy to PHt CDT. In the AR time series, the PCA weighting strategy was able to reduce the number of false alarms compared to the no weighting strategy in almost all time series (Figure 27a), however it introduced some miss-detections in several time series (Figure 27b). The PCA strategy also increased the drift detection delay in most of the cases. These results show that PCA delayed or made it difficult to detect the changes.

In the LS time series, the PCA strategy was not effective in improving the detection of PHt. In terms of false alarms (Figure 27d) and miss-detections (Figure 27b), the results were very similar, with no improvement. However, in terms of delay (Figure 27f), the weighting strategy worsened the results, increasing the drift detection delay.

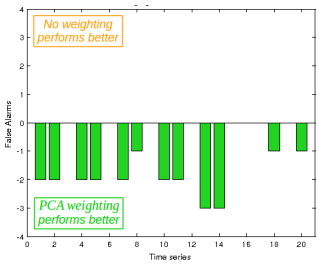
For the NL and RW-AR-NL time series, the PCA weighting strategy behaves in a similar way. It was able to reduce the number of false alarms (Figures 27g and 27j) and increased the drift detection delay (Figures 27i and 27l) in most of time series of both groups. In terms of miss-detections, in the NL time series, the results provided by PCA and the no weighting strategy were exactly the same. In the RW-AR-NL, on the other hand, some miss-detections were introduced by the PCA strategy in some series and it reduced this number in other series.

In the real-world time series, the PCA weighting strategy reduced the number of false alarms in one of the seven time series (Figure 27g), however it increased the number of false alarms in two time series (Figure 27n). In terms of delay, the PCA weighting strategy increased the delay in four of the seven time series.

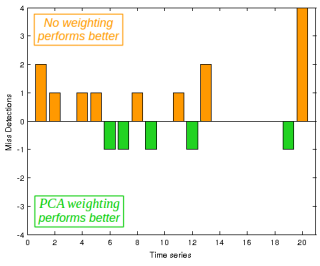
These experiments showed that the std weighting strategy was able to improve the concept drift detection of PHt, mainly in terms of false alarms and miss-detections. The identification of important features done by the weighting is able to improve the computation of differences between the averaged distances and minimum averaged distances monitored by PHt, resulting in the reduction of the number of false alarms. The PCA weighting strategy, on the other hand, was not able to improve the concept drift detection of PHt, similarly to what happened with ECDD CDT.

Figure 27 – Differences between the results provided by PHt with PCA weighting and with no weighting for all time series. Values below 0 indicate an improvement of the weighting strategy.

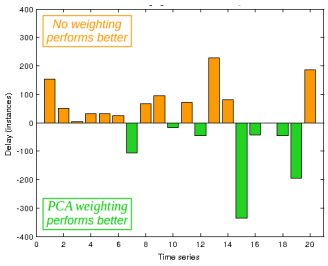
(a) AR: False alarms



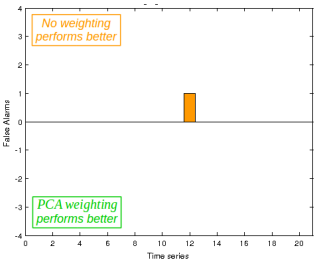
(b) AR: Miss-detections



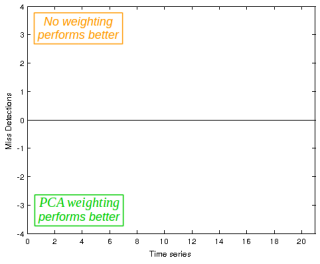
(c) AR: Delay



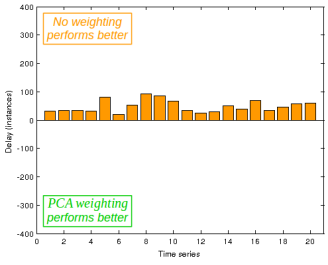
(d) LS: False alarms



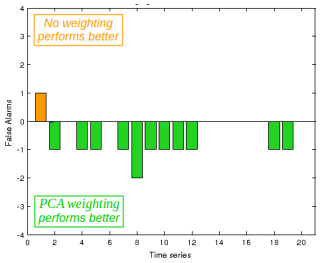
(e) LS: Miss-detections



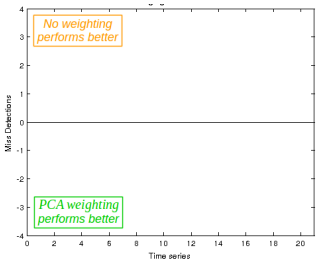
(f) LS: Delay



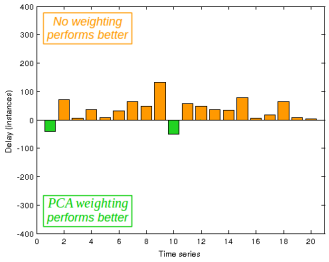
(g) NL: False alarms



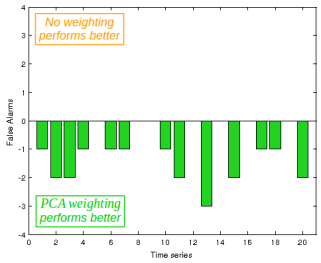
(h) NL: Miss-detections



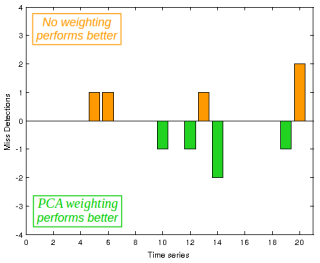
(i) NL: Delay



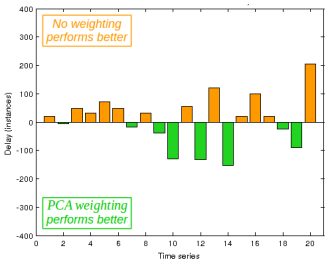
(j) RW-AR-NL: False alarms



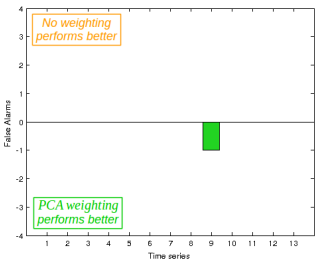
(k) RW-AR-NL: Miss-detections



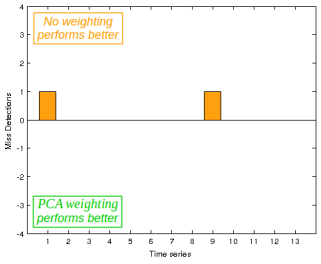
(l) RW-AR-NL: Delay



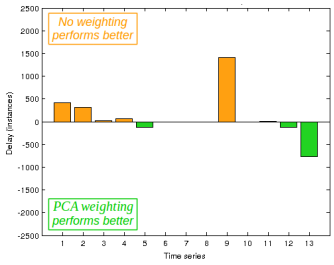
(m) Real: False alarms



(n) Real: Miss-detections



(o) Real: Delay



Source: elaborated by the author.

5.3.3.3 ICI and Feature Weighting

In this section we describe the same analysis of applying the weighting strategies but with the ICI-based CDT. Figure 28 presents the Friedman ranks with the Nemenyi critical difference for the comparison of comparison of ICI with and without feature weighting for the three metrics evaluated. Methods that are not significantly different (at $p = 0.05$) have ranks which differ by at least the critical difference.

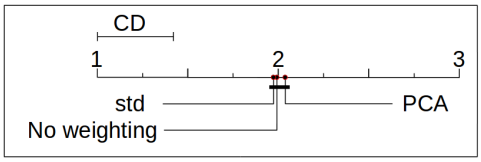
Different from ECDD and PHt, in this case, the weighting strategies provided statistically equivalent number of false-alarms and miss-detections than the no weighting strategy. However, the std weighting strategy was able to improve the drift detection delay of the CDT in the real-world time series. The PCA was not able to improve the results compared to the no weighting strategy. One reason for the lower improvement in terms of false alarms and miss-detections provided by the weighting strategies is that the ICI-based concept drift test in its original version already presents a very good detection accuracy, with numbers of false alarms and miss-detections close to 0. Maybe, the false alarms and miss-detections that occur in this case are unavoidable considering the feature set we are using in this work. The ICI models the confidence intervals for the distances using an initial set of observations in an offline mode and in the online processing, it tries to identify when the confidence intervals for these distances have no intersection with the initial one. When this happens, a concept drift is detected. The std weighting strategy is able to reduce the drift detection delay because, since it smooths the variance of the distances between feature vectors, the intervals become more narrowed and the concept drifts are detected faster. Although PCA is another way of smoothing the variance of the distances, it was not able to provide a general improvement in any of the three CDTs investigated.

Figure 29 allows a more detailed analysis of the comparison of the weighting strategy and the no weighting strategy for each time series in each dataset in terms of the three metrics. In the AR time series, the weighting strategy provided equal or better results in 85% of the time series for the number of false alarms (Figure 29a) and equal or better results in 80% of the time series in terms of miss-detections (Figure 29b). Some false alarms may be introduced by the weighting strategy due to problems in modeling the initial confidence intervals or due to spurious attribution of weights to features. In terms of detection delay, the weighting strategy was able to improve the results in 65% of the time series (Figure 29c).

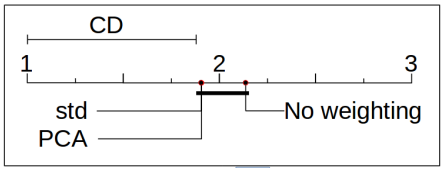
In the LS time series, the weighting strategy provided the same number of false alarms (Figure 29d) and miss-detections (Figure 29e) than the no weighting strategy, but with a small improvement in the delay in all time series (Figure 29f). This behavior was similar to the case of PHt.

Figure 28 – Comparison of ICI without weighting and with PCA feature weighting and with std feature weighting against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

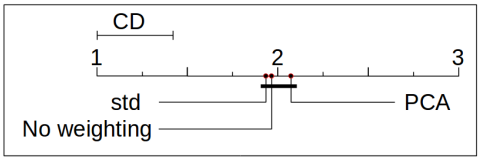
(a) False Alarms: artificial



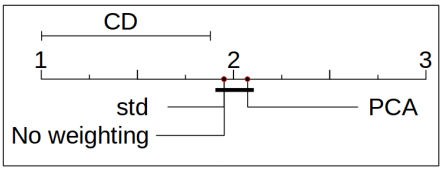
(b) False Alarms: real



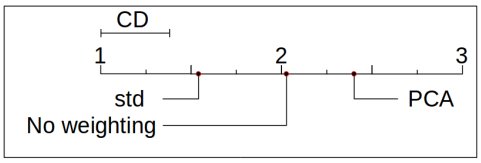
(c) Miss-detections: artificial



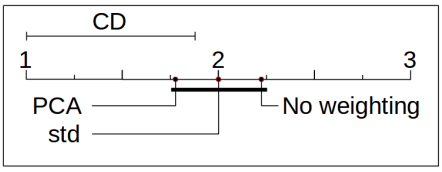
(d) Miss-detections: real



(e) Delay: artificial



(f) Delay: real



Source: elaborated by the author.

In the NL time series, in terms of false alarms, in just one case, the weighting strategy presented worse results than no weighting (Figure 29g), and in terms of miss-detections, in just two cases, the weighting strategy was not able to provide equal or better results (Figure 29h). In terms of delay, the weighting strategy improved the results in 55% of the cases (Figure 29i).

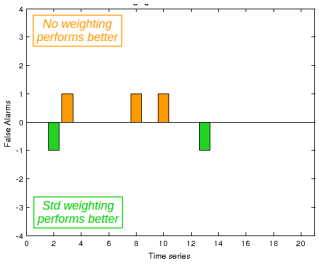
In the RW-AR-NL time series, in terms of false alarms, in one case the weighting strategy worsened the results compared to the no weighting strategy (Figure 29j). In terms of miss-detections, in three cases, the weighting strategy introduced one miss-detection (Figure 29k). The weighting strategy presented equal or better drift detection delay in 60% of the time series (Figure 29l). However, in some cases such as the time series 3 and 9 it increased the drift delay. This was due to the addiction of a miss-detection occurred in these time series.

In the real-world time series, the weighting strategy was able to provide equal or better results in terms of false alarms in all time series, reducing the false alarms in two of them (Figure 29m). In terms of miss-detections, the results were the same for both strategies (Figure 29n). The weighting strategy was able to reduce the drift detection delay in four of the seven time series (Figure 29o).

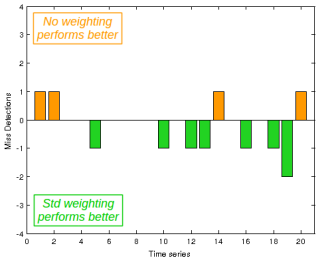
Figure 30 reports the results of a similar analysis but applying PCA weighting

Figure 29 – Differences between the results provided by ICI with std weighting and with no weighting for all time series. Values below 0 indicate an improvement of the weighting strategy.

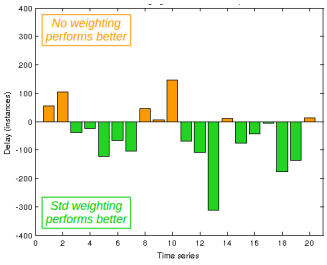
(a) AR: False alarms



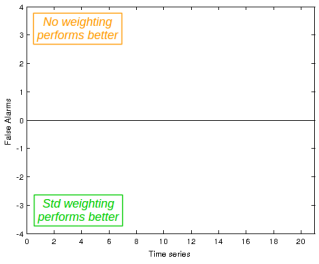
(b) AR: Miss-detections



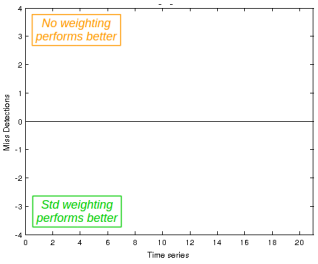
(c) AR: Delay



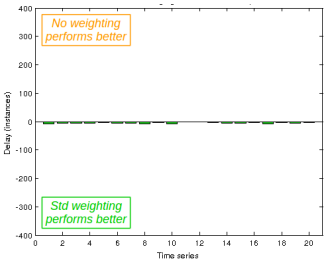
(d) LS: False alarms



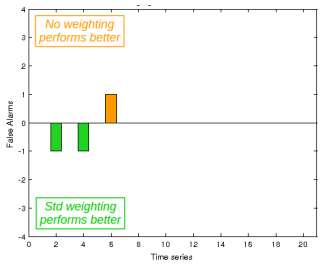
(e) LS: Miss-detections



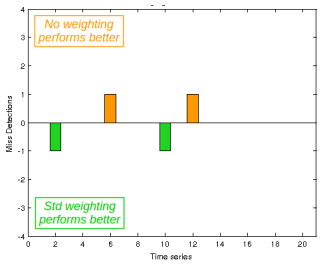
(f) LS: Delay



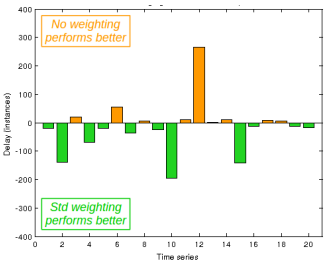
(g) NL: False alarms



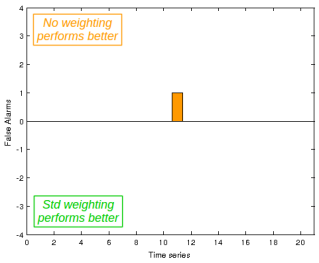
(h) NL: Miss-detections



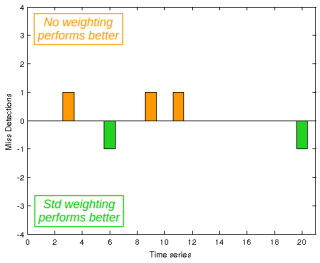
(i) NL: Delay



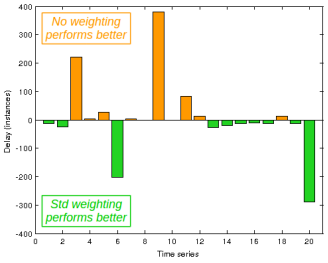
(j) RW-AR-NL: False alarms



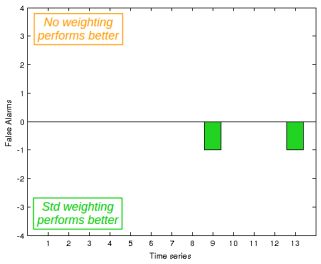
(k) RW-AR-NL: Miss-detections



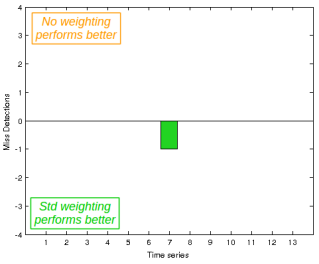
(l) RW-AR-NL: Delay



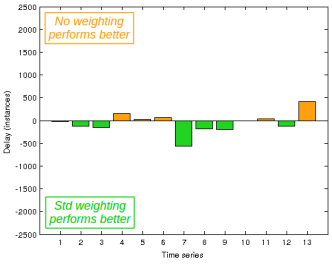
(m) Real: False alarms



(n) Real: Miss-detections



(o) Real: Delay



Source: elaborated by the author.

strategy to ICI-based CDT. Similarly to cases with ECDD and PHt CDT, the weighting strategy based on PCA was not able to improve the drift detection, but it worsened the results in most cases. In the AR time series PCA weighting was able to improve the results in terms of false alarms (Figure 30a) in just three time series and the miss-detections in just six time series, meanwhile it increased the drift detection delay in 65% of the time series.

In the LS time series, there was no improvement done by PCA strategy. In terms of false alarms (Figure 30d) and miss-detections (Figure 30e) results provided by PCA were almost the same than the no weighting strategy, but PCA strategy increased the drift detection delay in almost all time series (Figure 30f). For the NL and RW-AR-NL time series, the behavior of PCA weighting strategy was similar. No relevant improvement was made in terms of false alarms and miss-detections, but the drift detection delay was increased by the weighting strategy when compared to the no weighting strategy. In the real-world time series a small improvement was done in terms of false alarms, but the overall results in terms of miss-detections and drift detection delay was similar to the no weighting strategy, which indicates no improvement by using the PCA weighting strategy.

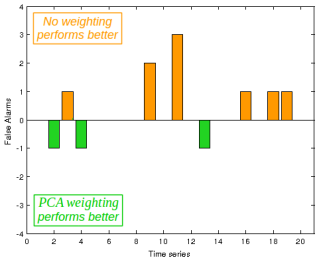
These results indicate that the std weighting strategy is able to improve the results of the ICI applied to features in combination, since it provides equivalent number of false-alarms and miss-detections, but with lower drift detection delay. Reducing the detection delay is important since it allows a faster handling of concept drifts in a real-world forecasting application. Again, the PCA weighting strategy was not able to improve the drift detection process.

5.3.4 Feature-based CDTs with Weighting Strategies and Error-based CDTs

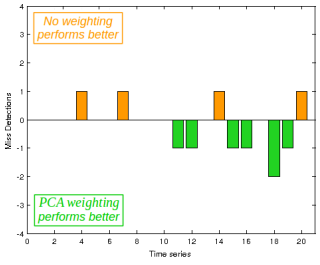
We further compare the results of these methods with the std weighting strategy and without weighting against each other and with the error-based CDTs discussed before. The goal of this last comparison is to evaluate which is the overall best concept drift detection method over all datasets. Since the PCA weighting strategy was not able to provide a significant improvement, but on the contrary, it worsened the results, it was not included in this comparison. Figure 31 shows the Friedman ranks and associated Nemenyi critical distances of the results. The Friedman test showed statistically significant differences among the results for all the three metrics evaluated. The ICI-based CDT applied on features in combination with weighting strategy (ICI$_{feat-std}$) presented the best ranks for all three metrics. In terms of false alarms, the ICI$_{feat-std}$ and the version without weighting (ICI$_{feat}$) presented statistically the best results. In terms of miss-detections, the methods based on features presented statistically better results than the other methods. In terms of drift detection delay, the ICI$_{feat-std}$ and ICI$_{feat}$ presented the better results.

Figure 30 – Differences between the results provided by ICI with PCA weighting and with no weighting for all time series. Values below 0 indicate an improvement of the weighting strategy.

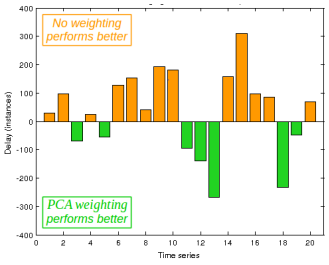
(a) AR: False alarms



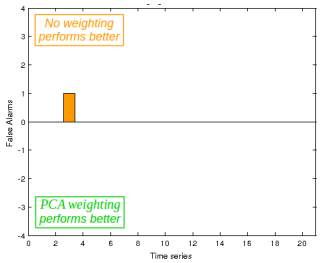
(b) AR: Miss-detections



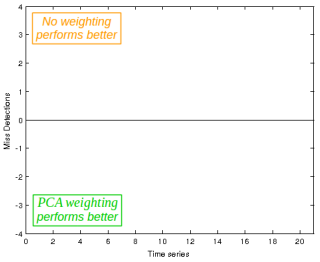
(c) AR: Delay



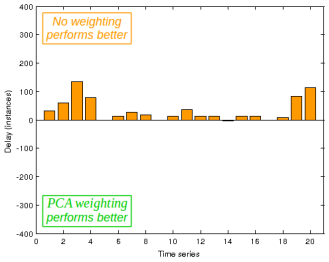
(d) LS: False alarms



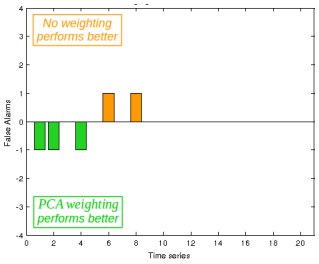
(e) LS: Miss-detections



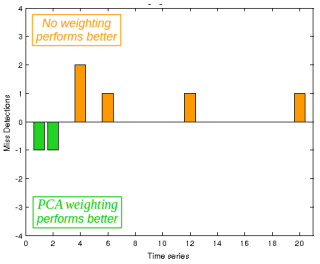
(f) LS: Delay



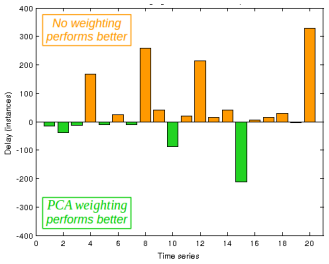
(g) NL: False alarms



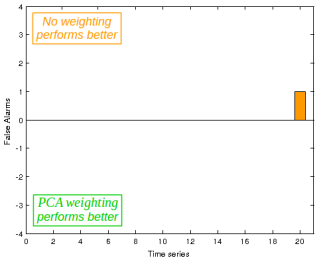
(h) NL: Miss-detections



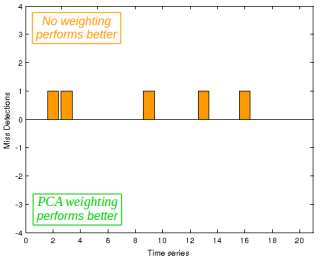
(i) NL: Delay



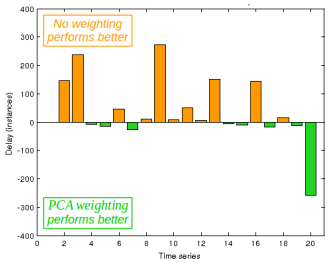
(j) RW-AR-NL: False alarms



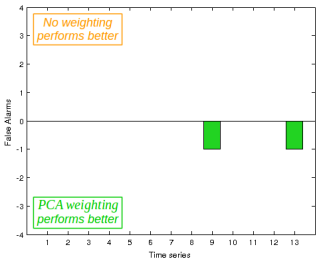
(k) RW-AR-NL: Miss-detections



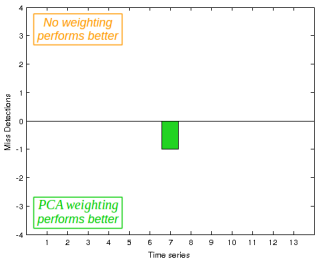
(l) RW-AR-NL: Delay



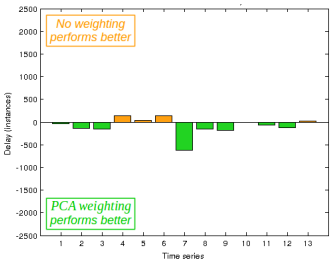
(m) Real: False alarms



(n) Real: Miss-detections



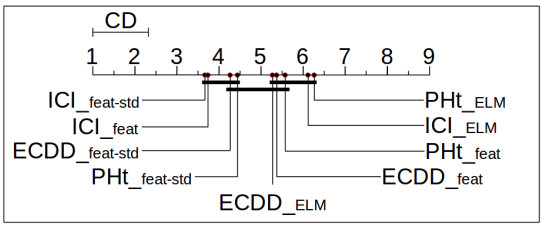
(o) Real: Delay



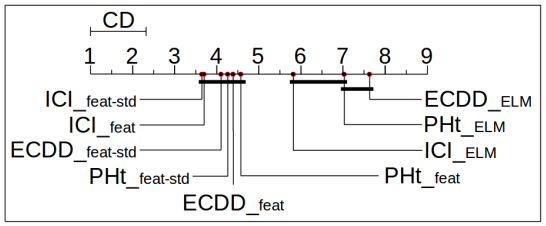
Source: elaborated by the author.

Figure 31 – Comparison of ECDD$_{_feat}$, PHt$_{_feat}$, and ICI$_{_feat}$ with and without weighting, ECDD$_{_ELM}$, PHt$_{_ELM}$, and ICI$_{_ELM}$ against each other with the Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

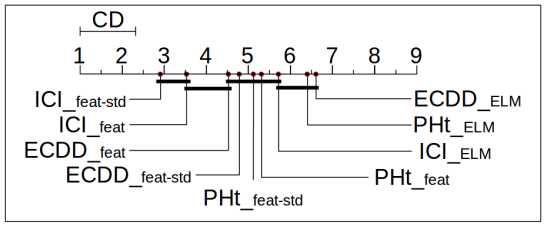
(a) False Alarms



(b) Miss-detections



(c) Delay



Source: elaborated by the author.

5.4 Forecasting Evaluation

The experiments reported in this section have the goal of answering the fourth research question formulated in Chapter 1.3 which asks how to better build the active adaptive learning system which implements a feature-based drift detection in order to improve the forecasting accuracy. Our hypothesis is that we can use a pool of individual models in which each model is built to represent a different concept. This approach combines the advantages of active and passive methods meanwhile it avoids some drawbacks of these approaches. So, there are three main subgoals of these experiments with the forecasting module. The first is to investigate which is the best way of combining the individual forecasting models, whether is to select just the "most appropriate" method to make the forecasting or combine the existing individual models to make the forecasting based on their suitability to make that forecasting. In order to do so, we compared the two combining schemes described in Section 4.3. These experiments are described in Section 5.4.1. After choosing the best way of combining the set of individual models, a second subgoal of these

experiments is performed in order to understand the importance of the parameters of the proposed model. So, we performed a sensitivity analysis of the parameters in order to study the effects of the parameter setting on the predictive performance of the method. The results of these experiments are presented and discussed in Section 5.4.2. The third subgoal of these experiments is to evaluate if the proposed forecasting module is more accurate than the other ways of handling concept drifts. In order to do so, we compare the proposed method with passive and active adaptive approaches existing in literature. Results of these experiments are discussed in Section 5.4.3.

5.4.1 Combining Individual Models

In these experiments we investigated what is the best way of combining the individual models in order to improve the forecasting accuracy. So, to ensure a reliable comparison, we simulate an oracle drift detection method in order to compare just the forecasting combination scheme of the forecasting module independently of the drift detection performance. This oracle method knows exactly where the drifts occurs. Two combination approaches were evaluated as possible candidates: (i) a dynamic weighted ensemble approach, in which the individual models are specialized in a different time series concepts and, at each instant a new observation needs to be predicted, all methods make the forecasting and the final output is weighted by considering how the models are specialized in the current time series concept; and (ii) a pool of individual models, also specialized on different time series concepts and, at every instant a new observation needs to be predicted, just the model which is more specialized in the current concept makes the forecasting. These methods are hereinafter referred to as *ens_oracle* and *closer_oracle*, respectively.

We performed a grid search to identify the best parameter settings for each method compared. Both methods have just two parameters: the threshold to include new models in the set of forecasting models Θ and the window size that defines the time series observations considered to define the current concept m_f. Table 4 show the parameter values considered for the grid search. The best set of parameters for a method is the one that minimizes the forecasting error. For each parameter setting, the methods were executed 10 times and the mean of each execution was computed. The window that defines which time series observations are used for building new models m_m was kept fixed with value $m_m = 200$. So, after a concept drift is detected by the oracle, the system waits for receiving 200 new time series observations to create a new model, if it is necessary. During this time, the existing models make forecasting according to the combination scheme implemented by the system. Just the artificial time series were used in this comparison, since we know in advance where the concept drifts happen, so we can simulate the oracle drift detector. The following parameters were defined empirically and not optimized: the lag which defines

the number of inputs used in the forecasting was set to $k = 5$, and the number of hidden neurons of OS-ELM was set to $hid = 10$.

Table 4 – Parameter values used in grid search.

Parameter	**Values**
θ	$\{0.0, 0.01, 0.05, 0.10, 0.2, 0.3\}$
m_f	$\{100, 150, 200\}$

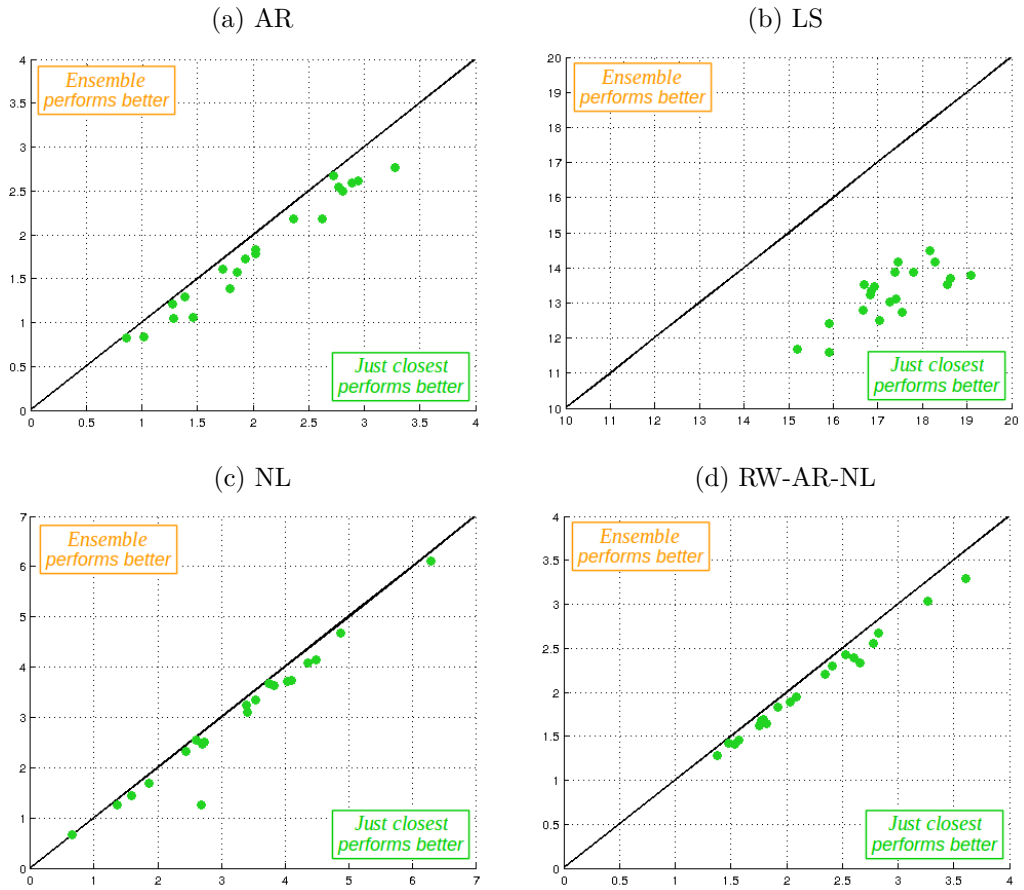
Source: elaborated by the author.

Figure 32 shows the comparison of the results in terms of MAPE provided by the *ens_oracle* and the *closest_oracle* for each time series group. These plots show the paired results of each method for each time series. If the point is on the left of the diagonal line, it indicates that the forecasting error of the *ens_oracle* was higher and the *closest_oracle* performed better. On the other hand, *closest_oracle* presented higher forecasting error and consequently the *ens_oracle* performed better. As one can see, the *closest_oracle*, presented better results for all time series of each time series group. The differences were higher in the LS time series, were the *closest_oracle* performed much better than the *ens_oracle*.

In order to assess the statistical validity of these results, we run a hypothesis test on these results. Since the methods are compared pairwise, we follow the recommendation of Demsar (2006) and use the Wilcoxon signed-ranks test, to compare two methods over multiple datasets. The p-value of the test was $8.1524e$-15, which indicates that with confidence level $\alpha = 0.05$, the results are statistically different. So, we can conclude that the approach that uses just the closest method to make the forecasting is statistically superior than the weighted ensemble combination of the existing methods. One explanation for this is that in this approach, a new model is created just when the new time series concept faced by the method is very different from the concept drifts already seen. So, each model is very specialized in the concept it was fitted to. Therefore, using the forecasting answer of different models, even they are weighted, increases the error, since a specialist gives an opinion on a context entirely different from his specialty.

With these results obtained until this section, we tested the best way of building the four main modules of the proposed adaptive learning system: (i) the feature extracted from the time series are considered in combination instead in isolation; (ii) the std feature weighting is used to improve the drift detection; (iii) the ICI-based CDT is used to detect changes; and (iv) the forecasting module uses a pool of specialized forecasting models and just the more specialized models is responsible for answering for a faced time series concept. Hereinafter, the proposed method with these characteristics is referred to as FW-FEDD.

Figure 32 – Comparison between *ens_oracle* and the *closest_oracle* in terms of MAPE.



Source: elaborated by the author.

5.4.2 Sensitivity Analysis of the FW-FEDD Parameters

In this section, we investigate the effect of the parameters on the learning ability of the proposed method. The FW-FEDD has four parameters, namely θ, which is used in the forecasting module, and three parameters used in the $ICI_{feat-std}$ drift detection as explained in Section 5.3 (Γ, TS and m_f). In order to investigate the sensibility of the method to these parameters, we performed analysis of variance (ANOVA) (MONTGOMERY, 2004) to analyze the influence of each parameter mentioned on the forecasting performance, measured through the MAPE. ANOVA is a set of statistical methods that can be used to test the hypothesis about the effect of different factors on a response variable. ANOVA was chosen instead other tests, since it allows the investigation of multiple factors and interactions at the same time. Tests such as Friedman are not appropriate for this kind of investigation.

In the ANOVA, the parameters of the method are called factors, and the different values that these parameters assume on the experiment are called factor levels. The statistical analysis can use the null hypothesis that there is no difference in the response

when using different factor levels and the alternative hypothesis that there is difference. Factors can be classified as within-subject or between-subject. Within-subject factors involve comparisons of the same subjects under different conditions (factor-levels). Between-subject factors are factors in which different groups of subjects are used for each factor level. When more than one kind of factor is used, we have a split-plot (mixed) design, which involves both between-subject and withing-subject factors. This type of ANOVA is used in this thesis.

The main assumption done by split-plot ANOVA is the sphericity (DEMSAR, 2006). If the sphericity assumption is violated, the split-plot ANOVA can get high type I error (reject the null hypothesis when it was true) (DEMSAR, 2006). Mauchly's test (MAUCHLY, 1940) can be used to detect violations of the spphericity assumption. If violations are detected, corrections such as Greenhouse-Geisser (GREENHOUSE; GEISSER, 1959) can be applied to the ANOVA's p-value so that the type I error wil not be increased. In addition to ANOVA, measures of effect size such as partial eta-squared can be used to determine whether the effect of a certain factor or interaction between factors is higher than the effect of another factor or interaction. The higher the eta-squared, the greater the relevance of the corresponding factor or interaction on the response.

In this study, the between-subject factor is the type of the time series, which varies among four different levels (AR, LS, NL and RW-AR-NL). The within-subject factors are θ, Γ, TS and m_f. All factors vary among three levels: $\theta = \{0.0, 0.01, 0.10\}$, $\Gamma = \{2.0, 2.5, 3.0\}$, $ts = \{100, 200, 300\}$, $mf = \{100, 150, 200\}$. Mauchly's tests of sphericity detected violations of the sphericity assumption (null hypothesis always rejected with p-value less than 0.001), as can be seen in Table 5. So Greenhouse-Geisser corrections were used.

Table 6 shows the results of the ANOVA for the within-subjects and for the between subjects using the Greenhouse-Geisser corrections, since Mauchly's tests of sphericity detected violations of this assumption. The table presents the type III sum of squares (SS), degrees of freedom (DF), mean squares (MS), test F statistics, the p-value (Sig.) and eta-squared (ETA). Interactions between 2 factors are represented by factor1*factor2. P-values less than 0.01 represent rejection of the null hypothesis that the average response is statistically equal at all the levels of the corresponding factors, considering significance level of 1%. Factors/interactions with large effect size (eta-squared higher than 0.10) are shown in boldface. Results ordered in decreasing order of effect size.

The interaction θ and the time series type presented the larger effect size on the forecasting error followed by the factor θ. The factor TS, which defines the number of distance observations used to model the initial confidence interval of ICI also has a large effect size when interacting with the between-subject factor type and alone. Some interactions which evolve θ, m_f, TS and type has also a significant effect size on the

Table 5 – Mauchly's test of sphericity.

Within Subject Effect	**Mauchly's W**	**Approx. Chi-Squared**	**df**	**Sig.**	**Greenhouse-Geisser**
θ	0.877	315.337	2	0.000	0.890
Γ	0.894	267.374	2	0.000	0.904
TS	0.926	183.521	2	0.000	0.931
m_f	0.885	291.567	2	0.000	0.897
$\theta * \Gamma$	0.935	160.851	9	0.000	0.967
$\theta * TS$	0.929	177.163	9	0.000	0.963
$\Gamma * TS$	0.728	759.798	9	0.000	0.874
$\theta * \Gamma * TS$	0.709	822.831	35	0.000	0.914
$\theta * m_f$	0.542	1467.950	9	0.000	0.788
$\Gamma * m_f$	0.662	987.930	9	0.000	0.844
$\theta * \Gamma * m_f$	0.646	1045.049	35	0.000	0.886
$TS * m_f$	0.687	899.232	9	0.000	0.848
$\theta * TS * m_f$	0.661	991.893	35	0.000	0.904
$\Gamma * TS * m_f$	0.576	1319.767	35	0.000	0.887
$\theta * \Gamma * TS * m_f$	0.420	2073.281	135	0.000	0.902

Source: elaborated by the author.

forecasting error of the method. The parameter Γ, was the factor with the smallest effect size on the response. The table also shows that the between-subject type has a high effect size on the response.

Since the effects of the interaction θ*m_f*type on the forecasting error has a large effect size, we generated plots of marginal means of this interaction in order to analyze the 2-way interactions and single factors involving these factors. The plots are shown in Figure 33. According to the plots, as the value of θ increases, there is an increase in the forecasting error, except in the case of type=3 (NL time series), in which $\theta = 0.01$ was better than $\theta = 0$ on average. The parameter θ indicates the minimum distances for including new models in the pool of forecasters. So, these results indicate that the best results are achieved by including a new model every time a new concept drift happens, rather than reusing an existing model specialized on a similar concept. This is the ideal scenario. However, when facing real-world problems with limited computational resources, the user needs to balance the trade-off between accuracy and computational costs.

As it can be observed in these plots, the effects of m_f varies according to the type of the series. The parameter m_f indicates the size of the window of data used to extract time series features at each instant in order to detect drifts. In the AR time series (Figure 33a), $m_f = 150$ provide lower forecasting error. In the LS time series (Figure 33b) setting $m_f = 100$, combined with $\theta = 0$ provide better forecasting accuracy on average. This may be due to the fact that these time series present well defined features, such as

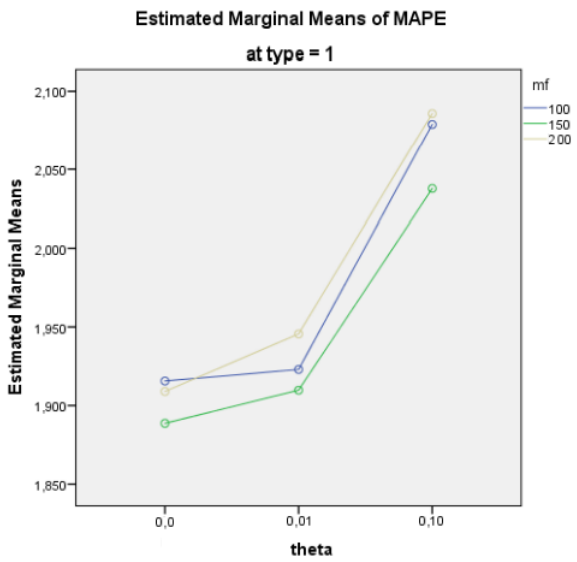
Table 6 – Tests of within-subjects and between-subjects effects.

Factor	**SS**	**DF**	**MS**	**F**	**Sig.**	**Eta**
Within-subjects effects						
θ * type	42116.090	5.341	7885.292	9364.609	0.000	**0.921**
θ	20536.775	1.780	11535.150	13699.197	0.000	**0.851**
TS * type	11292.891	5.588	2020.984	806.492	0.000	**0.502**
TS	4259.805	1.863	2287.014	912.653	0.000	**0.276**
θ*m_f*type	2943.871	9.461	311.155	262.588	0.000	**0.247**
m_f * type	5311.420	5.383	986.704	253.359	0.000	**0.241**
TS*m_f * type	1884.206	10.172	185.233	98.094	0.000	**0.109**
θ*m_f	999.384	3.154	316.892	267.430	0.000	**0.100**
m_f	1681.771	1.794	937.269	240.666	0.000	0.091
Γ*TS*m_f* type	1186.327	21.277	55.756	49.455	0.000	0.058
TS*m_f	698.300	3.391	205.946	109.064	0.000	0.044
θ*TS*m_f*type	456.458	21.707	21.029	32.902	0.000	0.040
θ*TS*type	168.040	11.551	14.548	25.906	0.000	0.031
Γ*TS*type	368.381	10.491	35.115	24.413	0.000	0.030
Γ*TS*m_f	390.336	7.092	55.036	48.816	0.000	0.020
Γ*m_f*type	227.237	10.125	22.444	15.597	0.000	0.019
θ*Γ*m_f*type	191.713	21.264	9.016	14.139	0.000	0.017
θ*Γ*TS*m_f*type	253.786	43.300	5.861	10.280	0.000	0.013
θ*TS*m_f	123.919	7.236	17.127	26.797	0.000	0.011
θ*Γ*TS*type	114.111	21.947	5.200	8.514	0.000	0.011
Γ*TS	117.320	3.497	33.550	23.325	0.000	0.010
θ*TS	49.407	3.850	12.832	22.851	0.000	0.009
θ*Γ*m_f	99.139	7.088	13.987	21.935	0.000	0.009
θ*Γ*type	42.404	11.603	3.655	6.751	0.000	0.008
Γ * type	30.520	5.427	5.624	3.725	0.002	0.005
Γ*m_f	59.096	3.375	17.510	12.169	0.000	0.005
θ*Γ*TS	57.680	7.316	7.885	12.911	0.000	0.005
θ*Γ*TS*m_f	93.717	14.433	6.493	11.389	0.000	0.005
θ*Γ	16.416	3.868	4.244	7.841	0.000	0.003
Γ	13.812	1.809	7.635	5.057	0.008	0.002
Between-subjects effects						
type	6604480.626	3	2201493.542	34389.538	0.000	0.977

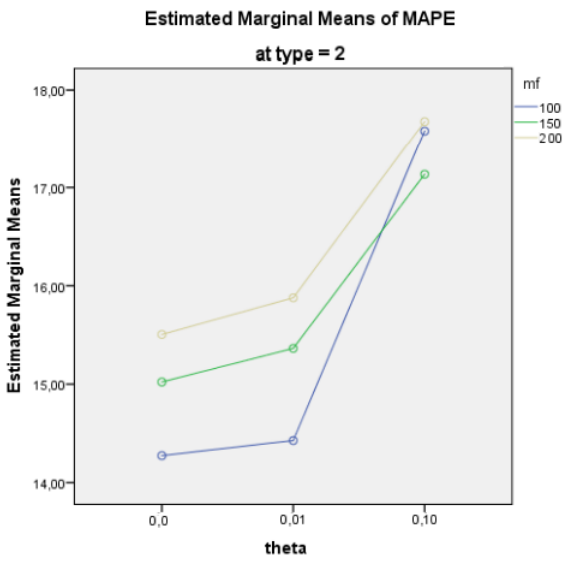
Source: elaborated by the author.

Figure 33 – Plots of marginal means for the effect of θ*m_f*type on the forecasting error.

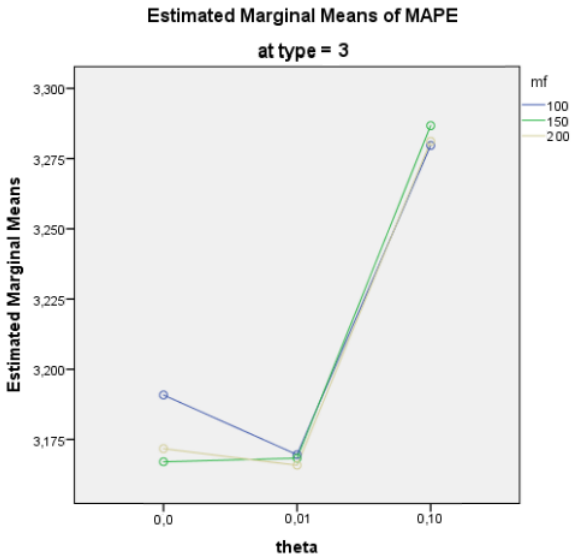
(a) AR



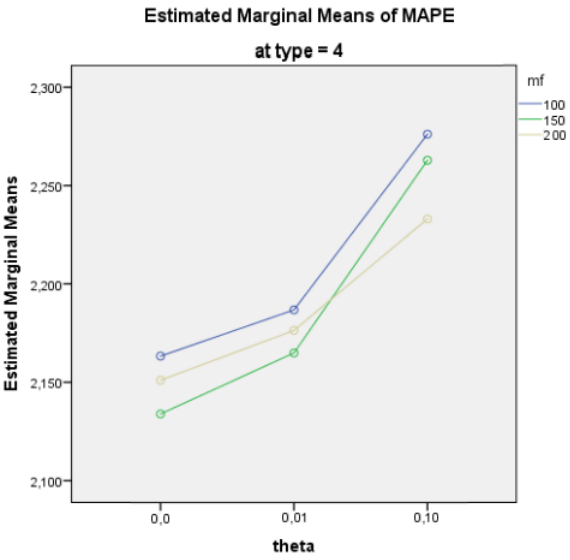
(b) LS



(c) NL



(d) RW-AR-NL



Source: elaborated by the author.

seasonality and periodicity, so the method does not need much observations to identify changes. Therefore, it does not need a wide sliding window. In the NL and RW-AR-NL time series, best results are achieved by setting $m_f = 150$ and $\theta = 0$. The explanation for these results may be the fact that $m_f = 150$ is an intermediate size for the window of observations used to extract features, which is neither so wide that can cause drift detection delay, nor so narrow that prevents a reliable characterization of the current time series concept.

The plots on Figure 33a also show that the factor type has a large effect size. The

average forecasting error is higher in the LS time series series type than in the others, on average. This may be due to the fact that these time series have concept drifts that change the periodicity of the time series. Since the lagged inputs used in the forecasting model are fixed, maybe these time series concepts become difficult to model in these conditions. The error is small on AR and RW-AR-NL time series.

We also further investigate the effects of the interaction TS*m_f*type on the forecasting error (Figure 34), due to its large effect size. As it can be observed from these plots, excepting for the LS time series type, the differences of forecasting error achieved by the different combinations of values for the factors TS and m_f are not so high as in the interaction θ*m_f*type. One can also see that the interactions between TS and m_f do not demonstrates a common pattern over all time series types. In the AR time series, the forecasting error is better by using $TS = 100$ in combination with $m_f = 150$. In the other cases, $TS = 300$ presents the lowest forecasting error compared to the other factor levels. The TS defines the number of distance observations used to estimate the initial confidence interval used by the ICI-based CDT. So, these results show that the higher the number of observations used to model the initial interval, the better is the drift detection and consequently the forecasting accuracy. In the LS time series, is better to set $m_f = 100$ as explained before. In the NL and in RW-AR-NL, is better using $m_f = 150$ or $m_f = 200$.

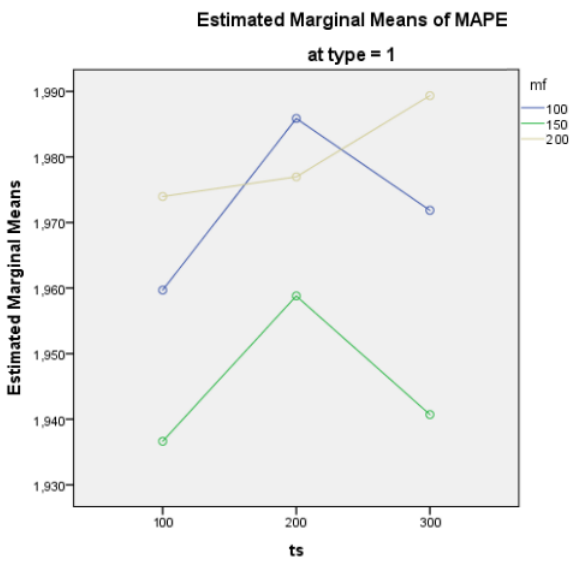
5.4.3 Comparing FW-FEDD with Passive and Active Adaptive Approaches

The experiments in this section are performed in order to evaluate if and when the proposed method is better than some existing passive and active adaptive learning systems and with methods which are unaware of concept drifts. In order to do so, we compare the proposed method with: (i) the ELM without drift detection, which allows we evaluate how the methods perform compared to a batch learning algorithm unaware of drift occurrence; (ii) the random-walk (RW) method, also called naive forecasting, in which the forecast $y_t = x_{t-1}$, i.e, the last observation; (iii) the OS-ELM, which is an implicit online learning method; (iv) the Dynamic Weighted Majority (DWM) (KOLTER; MALOOF, 2007), which is an online ensemble learning method; (v) the Online Weighted Ensemble of Regressor Models (OWE) (SOARES; ARAÚJO, 2015), another online ensemble learning method; and (vi) ELM combined with ICI (ICI_{ELM}), which was the active adaptive learning method which provided the best drift detection accuracy in the experiments on Section 5.3.

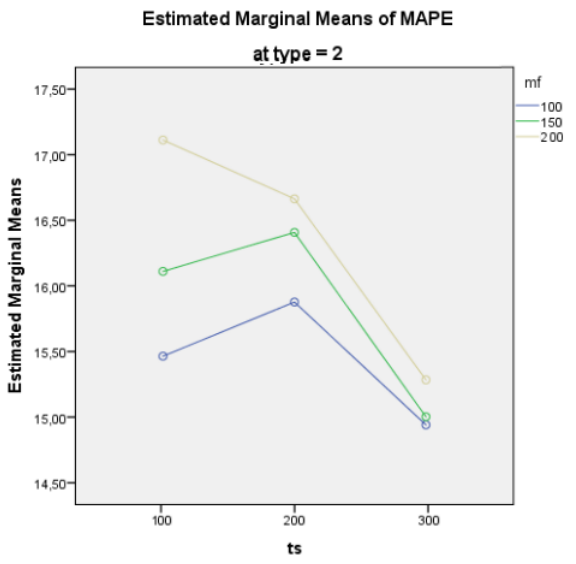
DWM is an ensemble learning approach originally proposed for classification problems. The basic idea of DWM is to keep a set of online classification models and assign weights to these models based on their individual performance. The majority voting is used to combine the outputs of the individual models. DWM also implements a mechanism to add and remove individual learners based on the global performance of the ensemble. In order to handle regression problems, we changed the original DWM in some ways. Firstly,

Figure 34 – Plots of marginal means for the effect of θ*TS*type on the forecasting error.

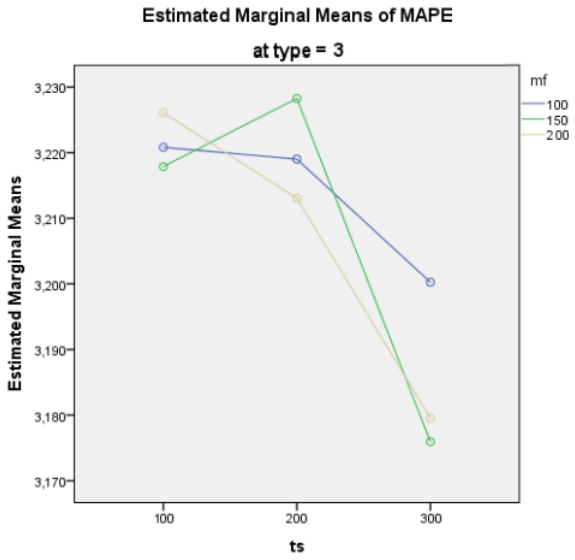
(a) AR



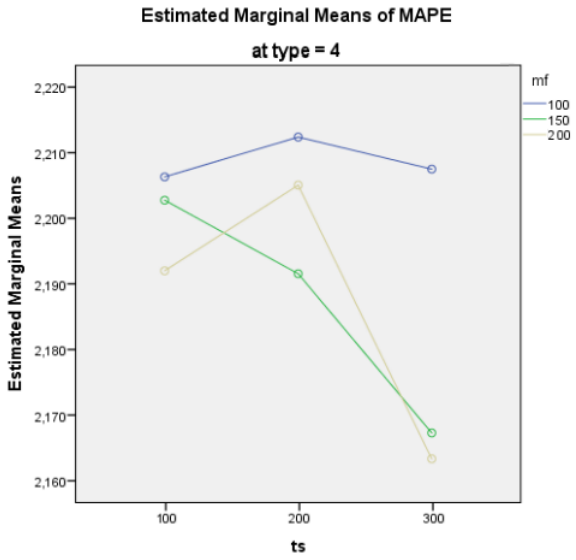
(b) LS



(c) NL



(d) RW-AR-NL



Source: elaborated by the author.

we changed the classification models for OS-ELMs, which are also online learning methods and can be used for regression. Secondly, we changed the majority voting combination scheme for the weighted averaging of the individual regression outputs. The original DWM uses the classification error of the ensemble to decide when to add a new individual model to the ensemble. We implement a discretization scheme based on a threshold γ in order to handle regression. If the absolute regression error is higher than γ, a new model is added to the ensemble. This γ becomes a parameter of the method. The remain parameters of DWM are the factor for decreasing weights (β), the threshold for deleting individual

models (δ), the period between individual models removal, creation and weight update (P).

The OWE is a more recent ensemble method for handling concept drift, but originally proposed for regression problems. OWE implements three adaptive mechanisms to handle drift: instance selection, instance weighting and ensemble learning itself. A set of individual regressor models are kept in the ensemble. Each regressor has an associated weight and this weight represents the contribution of the individual outputs for the overall output of the ensemble. These weights are dynamically updated according to the individual regression accuracy. OWE also implements a dynamic inclusion and exclusion of individual regressors to the ensemble. The original implementation made available by the authors was used in this comparison. The parameters of this method are the window size m_m, the factor for demarcating correct and incorrect predictions (γ), the factor to control the inclusion of new models (α), the discount factor (κ) and the maximum number of models in the ensemble (B).

Random-walk model was included in this comparison since it is usually used as a benchmark in time series forecasting (HYNDMAN; KOEHLER, 2006). The random-walk model can be considered a passive adaptive model, since every change is automatically incorporated by the model. We performed a grid search to identify the best parameter settings for each method compared. The number of hidden neurons of ELM and OS-ELM was fixed to $hid = 10$ and not optimized. Table 7 show the parameter ranges considered for the grid search of each compared method. For each parameter setting in each time series, we ran the methods 10 times. Both the artificial and real-world time series were used in these experiments.

Table 7 – Parameter values used in grid search.

Parameter	**Methods**	**Values**
β	DWM	$\{0.6, 0.7, 0.8, 0.9\}$
δ	DWM	$\{0.1, 0.2, 0.3, 0.4\}$
γ	DWM and OWE	$\{0.03, 0.05, 0.07\}$
κ	OWE	$\{0.1, 0.2, 0.3\}$
α	OWE	$\{0.2, 0.3, 0.4\}$
m_m	OWE and ICI$_{ELM}$	$\{100, 200, 300\}$
ts	ICI$_{ELM}$ and FW-FEDD	$\{100, 150, 200\}$
Γ	ICI$_{ELM}$ and FW-FEDD	$\{2.0, 2.5, 3.0\}$
m_f	FW-FEDD	$\{100, 150, 200\}$

Source: elaborated by the author.

Table 8 shows the averaged results in terms of MAPE with associated standard deviation for each time series group. Best results are in bold. These results show that, on average, the proposed FW-FEDD presented the best results for every time series group,

Table 8 – Average MAPE of the methods in each time series group.

Time Series	ELM	RW	OS-ELM	DWM	OWE	ICI$_{ELM}$	FW-FEDD
AR	37.61(19.3)	3.93(1.9)	2.45(0.8)	1.84(0.6)	2.31(0.7)	3.70(1.1)	**1.75(0.6)**
LS	110.4(21.2)	53.0(2.2)	27.9(1.3)	16.9(0.7)	19.6(1.1)	17.8(0.8)	**13.1(0.8)**
NL	41.70(26.8)	4.95(1.5)	3.50(1.3)	3.06(1.2)	3.73(1.3)	5.22(2.0)	**3.03(1.2)**
RW-AR-NL	38.04(25.6)	3.13(1.1)	2.42(0.6)	2.13(0.5)	2.50(0.5)	4.09(1.1)	**2.04(0.5)**
Temperatures	13.07(4.1)	**10.82(2.2)**	11.07(2.0)	10.94(1.9)	10.91(1.9)	11.45(1.9)	10.95(2.7)
Indices	13.87(13.6)	2.38 (1.5)	2.09(1.1)	1.84(0.9)	2.07(1.0)	3.16(1.3)	**1.80(0.8)**

Source: elaborated by the author.

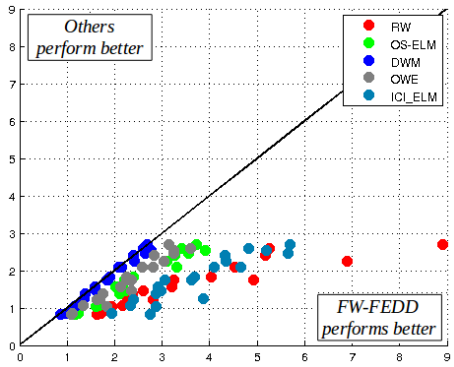
excepting the temperatures time series dataset, followed by the DWM. The simple ELM, which is unaware of concept drifts presented very high forecasting errors, mainly in the artificial time series. Since ELM is a batch learning method which never updates its learned model, the high number of drifts in these artificial time series degrades severely the forecasting performance of this method. The random-walk forecasting model, which is a simpled naive passive adaptive method, which uses the last observation as the forecast, presented better results than ELM. The OS-ELM presented better results than ELM and RW and even better results than ICI$_{ELM}$ in most of the cases. Among the ensemble methods, DWM was better than OWE in almost all time series groups.

Figure 35 shows the paired comparison of forecasting error of each method against the FW-FEDD in each time series group. These plots allow a more detailed comparison among the methods in each time series individually. The ELM results were omitted since they are much higher than the others and its inclusion would difficult the visualization of these results. In the AR time series (Figure 35a), none of the methods presented better results than FW-FEDD. DWM presented the closest results to FW-FEDD, while ICI$_{ELM}$ and random-walk presented the worst results.

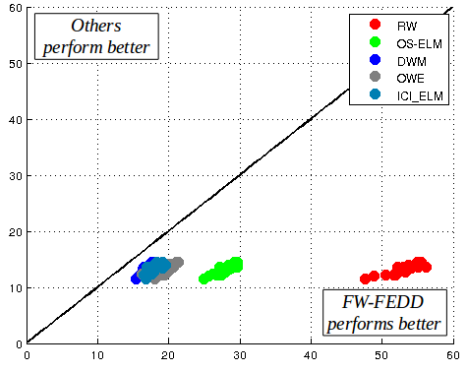
In the LS time series (Figure 35b), the forecasting error of all the methods were higher than in the AR time series. This may be due to problems in the parameters of the learning algorithm which models the time series. Since the periodicity of linear time series model that simulates these time series is variable for different concepts, the fixed lagged inputs $k = 5$ may not not appropriate for predicting every concept. In this case, the difference between the results provided by FW-FEDD and the other methods were also higher than in the AR time series. So, our proposed method presented higher accuracy even when there are some modeling problems. In this case, the RW presented the worst results, followed by the OS-ELM. RW presented the worst results due to the differences between the high differences between the adjacent values of the series, which increase the error. OS-ELM presented bad results due to the fact that OS-ELM has no forgetting mechanism. So, in the presence of more severe drifts, it presents the worst recovery from changes.

Figure 35 – Comparison of RW, OS-ELM, DWM, OWE, ICI_{ELM} against FW-FEDD regarding forecasting error (MAPE). Results on the right of the diagonal line indicate that the FW-FEDD presented smaller error. Results on the left of the diagonal line indicate that the other methods presented smaller error.

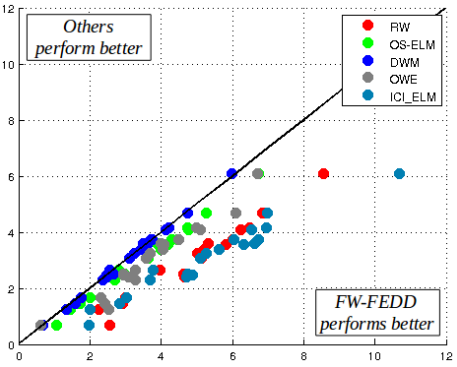
(a) AR



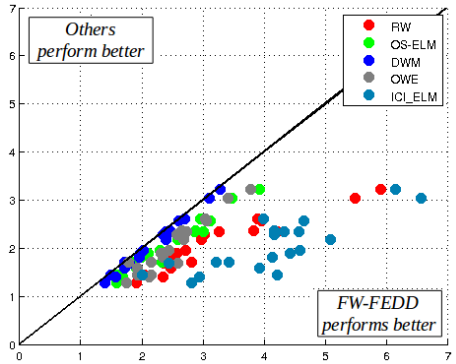
(b) LS



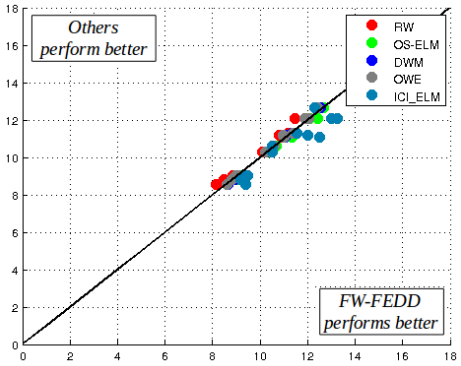
(c) NL



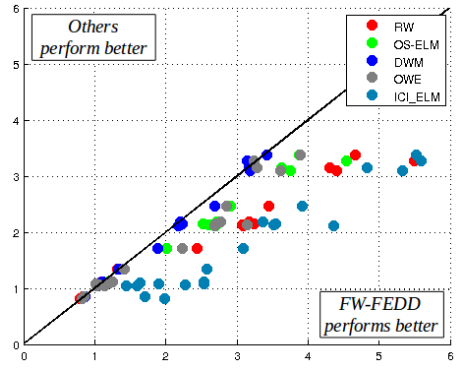
(d) RW-AR-NL



(e) Daily temperatures



(f) Financial indices



Source: elaborated by the author.

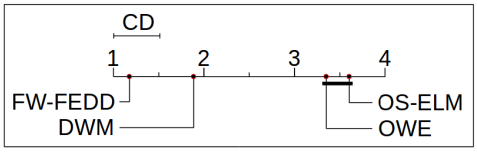
In the NL (Figure 35c) and RW-AR-NL (Figure 35d) time series, the methods presented similar behavior regarding the forecasting error. In these time series, the differences between the results were smaller than in the LS time series. FW-FEDD presented better results, followed by the DWM, which presented very similar results, being better in a few cases. The ICI_{ELM} presented the worst results, which can be justified by the inaccurate drift detection process of this method, that may introduce some drift detection delay and miss-detections.

We plot the results of the real-world time series separately due to the different natures of the real-world time series used in this work. In the daily-temperatures time series (Figure 35e), the methods presented very similar results and, in some cases, even better results than the proposed FW-FEDD. This is due to the fact that the approaches needed just one or two models to provide the best results they can do, since there is just one concept drift in these series. So, in this case use one online/incremental learning algorithm or a drop-and-retrain active adaptive method is enough to achieve good accuracy. In the stock indices time series (Figure 35e), the proposed method presented the best results. DWM presented very similar results to the proposed FW-FEDD, followed by OWE. Again, the ICI_{ELM} presented the worst results.

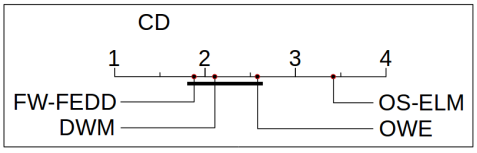
Figure 36 presents the Friedman ranks with the Nemenyi critical difference for the forecasting error of the four methods with best overall results, namely the OS-ELM, OWE, DWM and FW-FEDD. Methods that are significantly different (at $\alpha = 0.05$) have ranks which differ by at least the critical difference. In the artificial datasets, the hypothesis test indicates the statistically signficant superiority of the proposed FW-FEDD, compared to the other three methods, followed by the DWM. In the real-world dataset, the proposed FW-FEDD was statistically equivalent to DWM and OWE. The OS-ELM presented the worst ranks in both datasets.

Figure 36 – Comparison of OS-ELM DWM, OWE and FW-FEDD against each other with the Nenenyi test regarding the forecasting error. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

(a) Artificial dataset



(b) Real-world dataset



Source: elaborated by the author.

We also evaluated the models in terms of the number of models created during the processing of the time series. As discussed before, this metric is an indicative of the computational cost of the method, since more training implies in the need of more

computations. Table 9 shows the number of models, on average, created by each model. The ELM, random-walk and OS-ELM create just one model during the process of a time series. As can be observed, the proposed FW-FEDD needed to create less models than DWM or OWE, on average.

Table 9 – Average number of models created by the methods in each time series group.

Time Series	**ELM**	**RW**	**OS-ELM**	**DWM**	**OWE**	**ICI$_{ELM}$**	**FW-FEDD**
AR	**1(0)**	**1(0)**	**1(0)**	17.8(9.4)	43.3(37.3)	13.5(3.0)	11.75(3.1)
LS	**1(0)**	**1(0)**	**1(0)**	14.9(1.1)	489.9(44.9)	11.4(0.7)	12.1(2.4)
NL	**1(0)**	**1(0)**	**1(0)**	25.3(13.3)	82.7(62.8)	12.0(2.17)	10.85(3.5)
RW-AR-NL	**1(0)**	**1(0)**	**1(0)**	18.4(10.2)	49.2(77.9)	12.8(3.0)	12.4(2.6)
Temperatures	**1(0)**	**1(0)**	**1(0)**	4.6(11.4)	87.0(39.2)	1.6(1.1)	1.0(1.6)
Indices	**1(0)**	**1(0)**	**1(0)**	7.09(6.6)	59.1(80.1)	4.85(1.4)	3.25(1.9)

Source: elaborated by the author.

Figure 37 shows the paired comparison of number of models created by each method against the FW-FEDD in each time series group. The OWE results were omitted since they are very higher than the others and it difficult the visualization of these results. The plots show that RW, ELM and OS-ELM always present better results, since they use just one model to process a time series. Plots also show that in a small number of cases the DWM and the ICI$_{ELM}$ present smaller number of models than the FW-FEDD, but in the majority of the cases, DWM requires much more models than FW-FEDD.

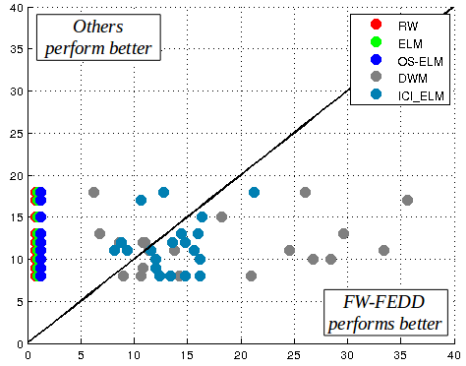
Figure 38 presents the Friedman ranks with the Nemenyi critical differences for the number of models created by the methods which the best forecasting performance, namely the OS-ELM, OWE, DWM and FW-FEDD. The test indicates that FW-FEDD presents statistically better results than all the other methods, excepting the OS-ELM, which uses just one model. So, these results show that the proposed method was able to provide the best forecasting accuracy while requiring a small number of forecasting models, which demonstrates the superiority of the proposed method.

5.5 Summary

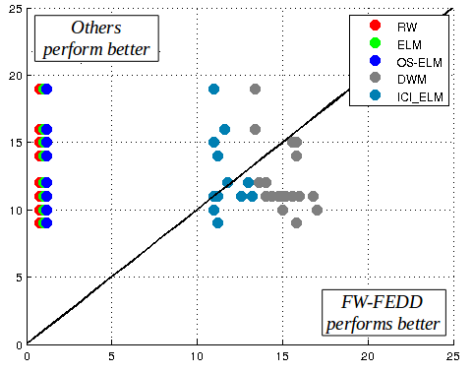
The experiments in this chapter were performed to answer the research questions described in Chapter 1. The first research question asked whether a set of time series features is a better source of information for drift detection than residuals of a fitted model or the raw time series data. In Section 5.3.1, we first compared the application of the ICI-based CDT on features individually against the application of some CDTs to error of an ELM forecasting method and against the Mood and Lepage approaches. The results showed that the use of features individually presented a high number of false alarms compared to the other methods. This is due to the fact that the method becomes

Figure 37 – Comparison of ELM, RW, OS-ELM, DWM and ICI_{ELM} against FW-FEDD regarding number of models created. Results on the right of the diagonal line indicate that the FW-FEDD presented smaller number of models. Results on the left of the diagonal line indicate that the other methods presented smaller number of models.

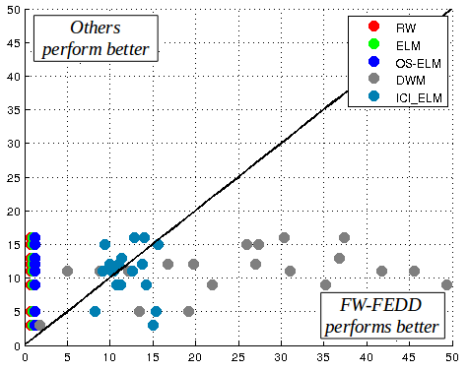
(a) AR



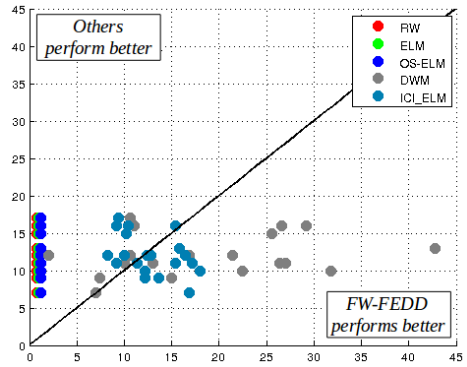
(b) LS



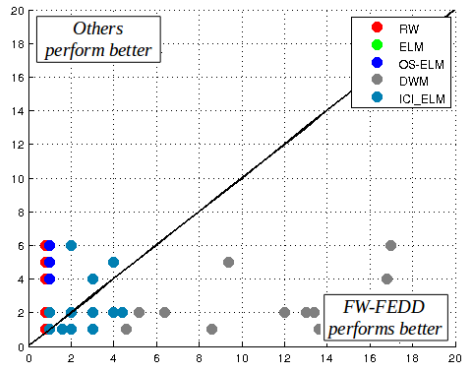
(c) NL



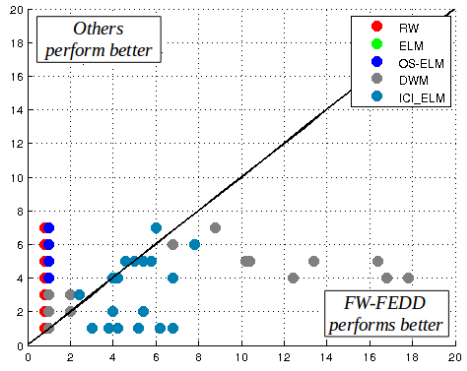
(d) RW-AR-NL



(e) Temperature time series

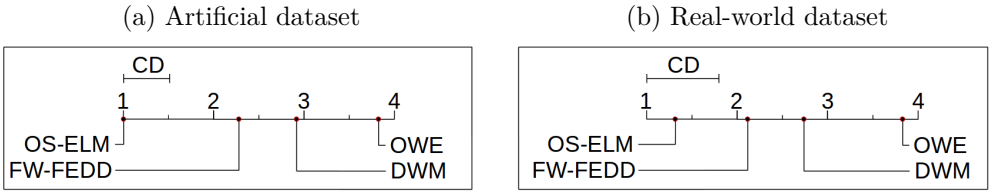


(f) Indices time series



Source: elaborated by the author.

Figure 38 – Comparison of the forecasting methods against each other with the Nenenyi test regarding the number of created models. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.



Source: elaborated by the author.

very sensitive to changes in a small subset of features. The Mood and Lepage methods presented the worst performance in terms of false alarms.

Our second research question was whether the use of features in combination could reduce the number of false alarms and improve the overall performance of this approach. To answer that question we proposed the use of features in combination by monitoring the distances among feature vectors that describes the time series during the online processing. In Section 5.3.2 we compared the use of some CDTs applied on distances between feature vectors with error-based approaches. The results indicated that our hypothesis that using features in combination improves concept drift detection can be accepted.

Our third research question was whether we could improve drift detection by identifying the importance of features to detect drifts. We then proposed a weighting strategy that considers the variance of features to compute the importance of these features. The experiments in Section 5.3.3 showed that the weighting strategy gets usually similar or better accuracy than the no weighting strategy for the three concept drift tests investigated. In case of ECDD, the weighting strategy was able to reduce the number of false alarms of the drift detection. For PHt, the proposed weighting strategy reduced both false alarms and miss-detections. For ICI-based CDT, it reduced the drift detection delay. The last comparison among the methods (Section 5.3.4) showed that the monitoring of features combined with weighting strategy presented better performance than the no weighting strategies and than the error-based drift detection methods.

Our fourth research question was how to build a forecasting method that used the feature-based drift detection information in order to improve the forecasting accuracy. Our hypothesis was that we could use a set of individual forecasting models in which each model is specialized in a different time series concept. We then compared two ways of combining the individual forecasting models and found out that using just the more specialized model to answer a time series concept is better than combining the forecasting of the existing models in the model set (Section 5.4.1). So, we built an adaptive learning system which uses the ICI-based CDT in combination to the proposed std feature weighting based on

the distance between feature vectors as the explicit drift detection and a pool of individual specialized forecasting methods. Section 5.4.3 showed that the proposed method provides better forecasting accuracy than some implicit and explicit forecasting methods while using a relatively low number of model creating during the processing of the time series.

6 CONCLUSION AND FUTURE WORK

This thesis introduced a new adaptive learning system to forecast time series which are eventually affected by concept drifts: the feature selection and weighting for drift detection in time series (FW-FEDD). The proposed adaptive learning system employs an online, explicit drift detection method which monitors a set of statistical time series features in order to detect concept drifts in real-time. An unsupervised feature weighting heuristic was proposed in order to identify the more relevant features at each time instant and refine the drift detection process. A forecasting module composed by a set of individual forecasting models is responsible for handle the drifts identified. Each individual model is specialized in a different time series concept. So, at each instant a new forecasting should be performed, the method identifies the current time series concept and chooses the more appropriate method to perform that forecasting.

The main contribution of this thesis is the new explicit drift detection based on feature extraction and feature weighting for time series. The majority of existing approaches are based on identification of change points, which are time instants in which the mean and/or variance of the time series observations change. Although these methods are very useful for posterior identification of structural breaks in time series, they are not suitable for the identification of changes that affect $p(y|X)$, which implies in the need for retraining or updating the forecasting model. Other approaches are based on monitoring the forecasting error of a forecaster method. Although these methods are focused on when the forecasting model is outdated and is no longer effective for make forecasting, they may not provide a reliable comprehension about the time series concepts. Problems such as poor generalization may make the drift detection process not reliable.

The proposed FW-FEDD explores the comprehension about time series concepts to provide a reliable concept drift detection. To do so, it is based on monitoring of a set of linear and non-linear time series features that describe the relationship about the time series temporal observations. However, not all of these features are informative about concept drifts for every kind of time series and for every possible concept drifts. So, the proposed feature weighting strategy is used as an attempt to identify which of these features are more informative about concept drifts at a given moment. This feature weighting strategy is an attempt to build a general time series drift detection method.

Several experiments were performed to identify the best way to build the proposed method and to compare it with existing concept drift detection methods proposed for time series. The experiments had the goal of answering four main research questions: (i) monitoring time series specific features is better than monitoring the raw time series

observations or the forecasting error to detect concept drifts? (ii) using features in combination is better than using features individually? (iii) there is a way of identifying the time series features more informative for drift detection? (iv) how is the best way of build a forecasting module based on a set of individual models? We formulate some hypotheses for each of these research questions and the computational results showed an indication of the validity of our hypotheses. Table 10 summarizes the main conclusions obtained in this study.

Table 10 – Summary of the research questions and answers obtained in this study.

Research Questions and Answers
I. Monitoring time series features is better than monitoring the raw time series observations or the forecasting error to detect concept drifts? *Features are reliable sources of information for drift detection, but when they are inspected individually it generates lots of false alarms (false positives).*
II. Using features in combination is better than using features individually? *Yes. It allows a more accurate detection than using features individually and than using raw time series observations or residuals of a fitted model.*
III. There is a way of identifying the time series features more informative for drift detection? *Experiments showed that the proposed feature weighting function that considers features variance was able to improve drift detection.*
IV. How is the best way of building a forecasting module based on a set of individual models? *Results showed that using a pool of forecasting models specialized in different concept drifts presented better performance than some passive and active adaptive methods.*

Source: elaborated by the author.

6.1 Limitations of the Proposed Method

Although the results showed superior performance of the proposed method, there are some limitations that prevents it to perform better in terms of both drift detection accuracy, forecasting accuracy and computational cost. One of these limitations is the problem of sequential drifts. In some cases, an erroneous drift identification (a false alarm or a miss-detection) can cause the erroneous identification of subsequent drifts. An example of this situation is when a miss-detection happens and due to this, the method identifies the drift with some delay. If that drift is identified very close to another delay point, probably observations from the old and the true new concept will be used to create the representation of the new concept. This can generate a series of new erroneous drift identifications. These series of erroneous drift identifications can severely degrade the forecasting accuracy of the proposed method. A way to overcome this limitation is to add

a drift confirmation mechanism to the proposed method. This confirmation mechanism could implement a retrospective concept drift test and in parallel to the online processing it could try to confirm if a detected change was a legitimate drift or a false alarms and correcting the representation of the current concept.

A second limitation of the method is that it does not implement a mechanism to deal with gradual concept drifts. Since gradual drifts take more time instants to happen, the proposed method may present high drift detection delay to identify these concept drifts. A simpler way to overcome this limitation is to include a mechanism to monitor the continuous increasing of the distances between features vectors and fires an early detection of the drift, without waiting for the gradual change finishes. The drift confirmation method should posteriorly confirm if the gradual drift actually happened or it was a false alarm.

A third important limitation of the method is the computational cost to update the values of the features extracted from the time series when the window slides to incorporate a new time series observation arriving from the data stream. Some of the features used in this work need to be recomputed from scratch for all the observations in the current sliding window. So, the larger the window, the more costly is the computation of the time series features. In the case of working on high-frequency data streams, the method may present some computation delay and require a hardware with high processing power.

6.2 Future Work

There are several ways to go further with this research, besides overcoming the limitations of the proposed method. Following, there are some interesting points to be investigated:

- **Feature-based and residual-based drift detection.** Although the residuals of a fitted model are not so effective in detecting changes, it can be a good source of information for improving concept drift detection of a feature-based method. An interesting investigation would be the combination of these two sources of information to improve concept drift detection.

- **Distance metrics.** The distance metric used to compute the dissimilarity among the feature vectors has a high influence in the concept drift detection accuracy of the method. If a higher space of features used to characterize time series is used, perhaps some distance metrics may make the method less sensitive to drifts. So, an investigation about other distance metrics, as well as some divergence metrics arising from information theory, such the *Kullback-Leibler* (KULLBACK; LEIBLER, 1951) and *Jensen-Shannon* (LIN, 1991) divergence metrics, and its effects on drift detections would be an interesting future investigation.

- **Multivariate time series.** The present study was focused on univariate time series. However, many time series are better analyzed not as just one source of time-dependent observations, but as components of some vector-valued sources, in which there is also an interdependence between the different component series (BROCKWELL; DAVIS, 2002). The covariation of time series which present similar time-based patterns is a source of information that may improve the time series analysis (Du Preez; WITT, 2003). In the financial context multivariate time series are very common. In this context, a stock price time series is always related to other stocks on the same market segment, and with national and international financial indices. This source of extra information could be used to improve the drift detection in time series. In particular, the multivariate analysis could be useful in both early drift detection and in drift confirmation.

- **Deep-Learning.** Recently, the attention of the machine learning and pattern recognition communities has been devoted to applying different methods to hierarchically learn useful features from a large amount of data (BENGIO; COURVILLE; VINCENT, 2013). The main goal of these approaches is to model complex real-world data by extracting robust features that capture the relevant information (HINTON; OSINDERO; TEH, 2006). The extraction and recognition of the patterns occur through a deep nonlinear network topology, in which the layers of feature representations can be stacked to create deep networks capable of modeling complex structures in the data (Le Roux; BENGIO, 2008). The deep learning approach has been successfully applied to tasks such as classification (LEE et al., 2009), speech recognition (ZHANG; WU, 2013) and dimensionality reduction (HINTON; SALAKHUTDINOV, 2006). The mainstream deep machine learning approaches include convolutional neural networks (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), deep belief networks (HINTON; OSINDERO; TEH, 2006) and stacked auto-encoders (VINCENT et al., 2008). Perhaps a promising future research would be the use of deep learning to characterize time series concepts and improve the concept drift detection.

REFERENCES

ABDUAL-SALAM, M. E.; ABDUL-KADER, H. M.; ABDEL-WAHED, W. F. Comparative study between differential evolution and particle swarm optimization algorithms in training of feed-forward neural network for stock price prediction. In: IEEE. *7th International Conference on Informatics and Systems*. Cairo, 2010. p. 1–8. Citado na página 45.

AGGARWAL, C. C. *Data streams: models and algorithms*. New York: Springer Science & Business Media, 2007. v. 31. Citado 3 vezes nas páginas 46, 47, and 49.

AGHABOZORGI, S.; SHIRKHORSHIDI, A. S.; WAH, T. Y. Time-series clustering – A decade review. *Information Systems*, v. 53, p. 16–38, 2015. Citado 2 vezes nas páginas 24 and 71.

ALIPPI, C.; BORACCHI, G.; ROVERI, M. Change detection tests using the ICI rule. In: IEEE. *International Joint Conference on Neural Networks*. Barcelona, 2010. p. 1–7. Citado na página 78.

ALIPPI, C.; BORACCHI, G.; ROVERI, M. A just-in-time adaptive classification system based on the intersection of confidence intervals rule. *Neural Networks*, Elsevier, v. 24, n. 8, p. 791–800, 2011. Citado 5 vezes nas páginas 52, 55, 67, 77, and 78.

ALIPPI, C.; BORACCHI, G.; ROVERI, M. Ensembles of change-point methods to estimate the change point in residual sequences. *Soft Computing*, Springer, v. 17, n. 11, p. 1971–1981, 2013. Citado 2 vezes nas páginas 61 and 66.

ALIPPI, C.; BORACCHI, G.; ROVERI, M. Just-in-time classifiers for recurrent concepts. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 24, n. 4, p. 620–634, 2013. Citado 5 vezes nas páginas 24, 50, 55, 67, and 94.

ALVAREZ, F. M. et al. Energy time series forecasting based on pattern sequence similarity. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 23, n. 8, p. 1230–1243, 2011. Citado na página 20.

ANGELINI, E.; TOLLO, G. di; ROLI, A. A neural network approach for credit risk evaluation. *The quarterly review of economics and finance*, Elsevier, v. 48, n. 4, p. 733–755, 2008. Citado na página 21.

ATSALAKIS, G. S.; VALAVANIS, K. P. Surveying stock market forecasting techniques–Part II: Soft computing methods. *Expert Systems with Applications*, Elsevier, v. 36, n. 3, p. 5932–5941, 2009. Citado 2 vezes nas páginas 21 and 42.

AURET, L.; ALDRICH, C. Change point detection in time series data with random forests. *Control Engineering Practice*, Elsevier, v. 18, n. 8, p. 990–1002, 2010. Citado 2 vezes nas páginas 60 and 66.

BACH, F. R. Active learning for misspecified generalized linear models. In: *Neural Information Processing Systems*. [S.l.: s.n.], 2006. v. 19. Citado na página 53.

BAENA-GARCIA, M. et al. Early drift detection method. In: *Fourth international workshop on knowledge discovery from data streams*. Berlin: [s.n.], 2006. v. 6, p. 77–86. Citado na página 22.

BAI, J.; NG, S. Tests for skewness, kurtosis, and normality for time series data. *Journal of Business & Economic Statistics*, Taylor & Francis, v. 23, n. 1, p. 49–60, 2005. Citado na página 74.

BALLINGS, M. et al. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, v. 42, n. 20, p. 7046–7056, 2015. Citado na página 45.

BAO, Y. et al. A Comparative Study of Multi-step-ahead Prediction for Crude Oil Price with Support Vector Regression. In: IEEE. *Fourth International Joint Conference on Computational Sciences and Optimization*. Kunming and Lijiang, China, 2011. p. 598–602. Citado 2 vezes nas páginas 33 and 44.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, v. 35, n. 8, p. 1798–1828, 2013. Citado na página 137.

BERGMEIR, C.; BENÍTEZ, J. M. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, Elsevier, v. 191, p. 192–213, 2012. Citado 2 vezes nas páginas 37 and 46.

BLYTHE, D. A. et al. Feature extraction for change-point detection using stationary subspace analysis. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 23, n. 4, p. 631–643, 2012. Citado na página 64.

BOEING, G. Visual Analysis of Nonlinear Dynamical Systems: Chaos, Fractals, Self-Similarity and the Limits of Prediction. *Systems*, Multidisciplinary Digital Publishing Institute, v. 4, n. 4, p. 37, 2016. Citado na página 35.

BORACCHI, G.; ROVERI, M. Exploiting self-similarity for change detection. In: IEEE. *International Joint Conference on Neural Networks*. Beijing, 2014. p. 3339–3346. Citado 5 vezes nas páginas 24, 25, 66, 67, and 90.

BOX, G. E.; JENKINS, G. M. *Time series analysis: forecasting and control, revised ed.* [S.l.]: Holden-Day, 1976. Citado 2 vezes nas páginas 37 and 42.

BRASILEIRO, R. C. et al. Automatic method for stock trading combining technical analysis and the Artificial Bee Colony Algorithm. In: IEEE. *IEEE Congress on Evolutionary Computation*. Cancun, 2013. p. 1810–1817. Citado na página 45.

BRITO, R. F. de; OLIVEIRA, A. L. Sliding window-based analysis of multiple foreign exchange trading systems by using soft computing techniques. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2014. p. 4251–4258. Citado na página 20.

BROCKWELL, P. J.; DAVIS, R. A. *Introdution to Time Series and Forecasting*. 2nd. ed. [S.l.]: Springer, 2002. Citado 4 vezes nas páginas 33, 37, 38, and 137.

BRODERSEN, K. H. et al. Inferring causal impact using Bayesian structural time-series models. *The Annals of Applied Statistics*, Institute of Mathematical Statistics, v. 9, n. 1, p. 247–274, 2015. Citado 2 vezes nas páginas 61 and 66.

BRZEZINSKI, D.; STEFANOWSKI, J. Combining block-based and online methods in learning ensembles from concept drifting data streams. *Information Sciences*, Elsevier, v. 265, p. 50–67, 2014. Citado na página 22.

BRZEZINSKI, D.; STEFANOWSKI, J. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 25, n. 1, p. 81–94, 2014. Citado na página 22.

CAVALCANTE, R. C. et al. Computational Intelligence and Financial Markets: A Survey and Future Directions. *Expert Systems with Applications*, Elsevier, v. 55, p. 194–211, 2016. Citado 2 vezes nas páginas 20 and 32.

CAVALCANTE, R. C.; OLIVEIRA, A. L. An autonomous trader agent for the stock market based on online sequential extreme learning machine ensemble. In: IEEE. *International Joint Conference on Neural Networks*. [S.l.], 2014. p. 1424–1431. Citado 2 vezes nas páginas 20 and 45.

CAVALCANTE, R. C.; OLIVEIRA, A. L. An approach to handle concept drift in financial time series based on Extreme Learning Machines and explicit Drift Detection. In: IEEE. *International Joint Conference on Neural Networks*. Killarney, 2015. p. 1–8. Citado 3 vezes nas páginas 24, 62, and 66.

CHAO, H.; LI-LI, H.; TING-TING, H. Financial time series forecasting based on wavelet kernel support vector machine. In: IEEE. *Eighth International Conference on Natural Computation*. Chongqing, China, 2012. p. 79–83. Citado na página 44.

CHEN, J. SVM application of financial time series forecasting using empirical technical indicators. In: IEEE. *International Conference on Information, Networking and Automation*. Kunming, China, 2010. v. 1, p. 77–81. Citado 2 vezes nas páginas 33 and 44.

CHEN, S.-M.; HWANG, J.-R. Temperature prediction using fuzzy time series. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, IEEE, v. 30, n. 2, p. 263–275, 2000. Citado na página 33.

CHENG, C.-H.; WEI, L.-Y. A novel time-series model based on empirical mode decomposition for forecasting TAIEX. *Economic Modelling*, Elsevier, v. 36, p. 136–141, 2014. Citado na página 42.

CHORDIA, T. et al. High-Frequency trading. *Journal of Financial Markets*, v. 16, n. 4, p. 637–645, 2013. Citado na página 52.

COHEN, L. et al. Real-time data mining of non-stationary data streams from sensor networks. *Information Fusion*, Elsevier, v. 9, n. 3, p. 344–353, 2008. Citado na página 22.

COWPERTWAIT, P. S.; METCALFE, A. V. *Introductory time series with R*. [S.l.]: Springer, 2009. Citado 12 vezes nas páginas 20, 32, 33, 34, 36, 37, 38, 39, 41, 72, 73, and 89.

CRYER, J. D.; CHAN, K.-S. *Time series analysis: with applications in R*. 2nd. ed. [S.l.]: Springer, 2008. Citado 2 vezes nas páginas 36 and 73.

DAI, W.; WU, J.-Y.; LU, C.-J. Combining nonlinear independent component analysis and neural network for the prediction of Asian stock market indexes. *Expert Systems with Applications*, Elsevier, v. 39, n. 4, p. 4444–4452, 2012. Citado na página 43.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, JMLR. org, v. 7, p. 1–30, 2006. Citado 3 vezes nas páginas 96, 118, and 120.

DHAR, S.; MUKHERJEE, T.; GHOSHAL, A. K. Performance evaluation of neural network approach in financial prediction: Evidence from Indian market. In: IEEE. *International Conference on Communication and Computational Intelligence*. Coimbatore, India, 2010. p. 597–602. Citado na página 44.

DITZLER, G. et al. Learning in nonstationary environments: A survey. *Computational Intelligence Magazine, IEEE*, IEEE, v. 10, n. 4, p. 12–25, 2015. Citado 7 vezes nas páginas 21, 23, 32, 49, 50, 51, and 59.

DOGANIS, P. et al. Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. *Journal of Food Engineering*, Elsevier, v. 75, n. 2, p. 196–204, 2006. Citado na página 33.

Du Preez, J.; WITT, S. F. Univariate versus multivariate time series forecasting: an application to international tourism demand. *International Journal of Forecasting*, Elsevier, v. 19, n. 3, p. 435–451, 2003. Citado na página 137.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification*. [S.l.]: John Wiley & Sons, 2012. Citado na página 46.

D'URSO, P. et al. Clustering of financial time series. *Physica A: Statistical Mechanics and its Applications*, Elsevier, v. 392, n. 9, p. 2114–2129, 2013. Citado na página 33.

EVANS, C.; PAPPAS, K.; XHAFA, F. Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation. *Mathematical and Computer Modelling*, Elsevier, v. 58, n. 5, p. 1249–1266, 2013. Citado na página 45.

FDEZ-RIVEROLA, F. et al. Applying lazy learning algorithms to tackle concept drift in spam filtering. *Expert Systems with Applications*, Elsevier, v. 33, n. 1, p. 36–48, 2007. Citado 5 vezes nas páginas 21, 22, 47, 51, and 52.

FERN, A.; GIVAN, R. Online ensemble learning: An empirical study. *Machine Learning*, Springer, v. 53, n. 1-2, p. 71–109, 2003. Citado na página 52.

FERREIRA, J. A.; LOSCHI, R. H.; COSTA, M. A. Detecting changes in time series: A product partition model with across-cluster correlation. *Signal Processing*, Elsevier, v. 96, p. 212–227, 2014. Citado na página 23.

FOKIANOS, K.; GOMBAY, E.; HUSSEIN, A. Retrospective change detection for binary time series models. *Journal of Statistical Planning and Inference*, Elsevier, v. 145, p. 102–112, 2014. Citado 2 vezes nas páginas 61 and 66.

FRIEDMAN, M. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, JSTOR, v. 11, n. 1, p. 86–92, 1940. Citado na página 94.

GABER, M. M.; ZASLAVSKY, A.; KRISHNASWAMY, S. A survey of classification methods in data streams. In: _____. *Data Streams: Models and Algorithms*. Boston: Springer, 2007. p. 39–59. Citado na página 48.

GAMA, J. *Knowledge discovery from data streams*. [S.l.]: CRC Press, 2010. Citado 2 vezes nas páginas 46 and 47.

GAMA, J. A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, Springer, v. 1, n. 1, p. 45–55, 2012. Citado 5 vezes nas páginas 20, 23, 32, 46, and 47.

GAMA, J. et al. Learning with drift detection. In: *Advances in Artificial Intelligence–SBIA 2004*. Berlin: Springer, 2004. p. 286–295. Citado 7 vezes nas páginas 21, 22, 33, 48, 52, 53, and 81.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. Issues in evaluation of stream learning algorithms. In: ACM. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. Paris, 2009. p. 329–338. Citado 2 vezes nas páginas 46 and 47.

GAMA, J. et al. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, ACM, v. 46, n. 4, p. 44, 2014. Citado 6 vezes nas páginas 22, 26, 48, 49, 51, and 79.

GARCIA, R. C. et al. A GARCH forecasting model to predict day-ahead electricity prices. *IEEE Transactions on Power Systems*, IEEE, v. 20, n. 2, p. 867–874, 2005. Citado na página 41.

GOMBAY, E.; SERBAN, D. Monitoring parameter change in time series models. *Journal of Multivariate Analysis*, Elsevier, v. 100, n. 4, p. 715–725, 2009. Citado 3 vezes nas páginas 55, 63, and 66.

GONCALVES, P. M.; BARROS, R. S. M. d. RCD: A recurring concept drift framework. *Pattern Recognition Letters*, Elsevier, v. 34, n. 9, p. 1018–1025, 2013. Citado 3 vezes nas páginas 50, 52, and 56.

GONCALVES, P. M. et al. A comparative study on concept drift detectors. *Expert Systems with Applications*, Elsevier, v. 41, n. 18, p. 8144–8156, 2014. Citado 2 vezes nas páginas 21 and 53.

GRANÉ, A.; VEIGA, H. Wavelet-based detection of outliers in financial time series. *Computational Statistics & Data Analysis*, Elsevier, v. 54, n. 11, p. 2580–2593, 2010. Citado na página 43.

GREENHOUSE, S. W.; GEISSER, S. On methods in the analysis of profile data. *Psychometrika*, Springer, v. 24, n. 2, p. 95–112, 1959. Citado na página 120.

GU, S.; TAN, Y.; HE, X. Recentness biased learning for time series forecasting. *Information Sciences*, Elsevier, v. 237, p. 29–38, 2013. Citado 4 vezes nas páginas 52, 59, 66, and 81.

GUAJARDO, J. A.; WEBER, R.; MIRANDA, J. A model updating strategy for predicting time series with seasonal patterns. *Applied Soft Computing*, Elsevier, v. 10, n. 1, p. 276–283, 2010. Citado 3 vezes nas páginas 23, 59, and 66.

GUO-QIANG, X. The optimization of share price prediction model based on support vector machine. In: IEEE. *International Conference on Control, Automation and Systems Engineering*. Singapore, 2011. p. 1–4. Citado na página 44.

GUZELLA, T. S.; CAMINHAS, W. M. A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, Elsevier, v. 36, n. 7, p. 10206–10222, 2009. Citado na página 21.

HAMZACEBI, C. Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences*, Elsevier, v. 178, n. 23, p. 4550–4559, 2008. Citado na página 20.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, MIT Press, v. 18, n. 7, p. 1527–1554, 2006. Citado na página 137.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *Science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006. Citado na página 137.

HU, Y. et al. Stock trading rule discovery with an evolutionary trend following model. *Expert Systems with Applications*, Elsevier, v. 42, n. 1, p. 212–222, 2015. Citado na página 23.

HUANG, C.-F. A hybrid stock selection model using genetic algorithms and support vector regression. *Applied Soft Computing*, Elsevier, v. 12, n. 2, p. 807–818, 2012. Citado na página 45.

HUANG, G.-B. et al. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, IEEE, v. 42, n. 2, p. 513–529, 2012. Citado na página 86.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. In: IEEE. *International Joint Conference on Neural Networks*. Budapest, 2004. v. 2, p. 985–990. Citado na página 81.

HYNDMAN, R. J.; BOOTH, H.; YASMEEN, F. Coherent mortality forecasting: the product-ratio method with functional time series models. *Demography*, Springer, v. 50, n. 1, p. 261–283, 2013. Citado na página 33.

HYNDMAN, R. J.; KOEHLER, A. B. Another look at measures of forecast accuracy. *International journal of forecasting*, Elsevier, v. 22, n. 4, p. 679–688, 2006. Citado na página 126.

KANTZ, H.; SCHREIBER, T. *Nonlinear time series analysis*. [S.l.]: Cambridge university press, 2004. v. 7. Citado na página 35.

KAO, L.-J. et al. Integration of nonlinear independent component analysis and support vector regression for stock price forecasting. *Neurocomputing*, Elsevier, v. 99, p. 534–542, 2013. Citado na página 43.

KAYAL, A. A Neural Networks filtering mechanism for foreign exchange trading signals. In: IEEE. *International Conference on Intelligent Computing and Intelligent Systems*. Xiamen, China, 2010. p. 159–167. Citado na página 44.

KILLICK, R.; ECKLEY, I. changepoint: An R package for changepoint analysis. *Journal of Statistical Software*, v. 58, n. 3, p. 1–19, 2014. Citado 3 vezes nas páginas 23, 64, and 66.

KIRCH, C.; KAMGAING, J. T. On the use of estimating functions in monitoring time series for change points. *Journal of Statistical Planning and Inference*, Elsevier, v. 161, p. 25–49, 2015. Citado 2 vezes nas páginas 62 and 66.

KOLTER, J. Z.; MALOOF, M. A. Dynamic weighted majority: An ensemble method for drifting concepts. *The Journal of Machine Learning Research*, JMLR. org, v. 8, p. 2755–2790, 2007. Citado 5 vezes nas páginas 21, 22, 32, 51, and 124.

KOURENTZES, N.; BARROW, D. K.; CRONE, S. F. Neural network ensemble operators for time series forecasting. *Expert Systems with Applications*, Elsevier, v. 41, n. 9, p. 4235–4244, 2014. Citado na página 44.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 137.

KUGIUMTZIS, D. Evaluation of surrogate and bootstrap tests for nonlinearity in time series. *Studies in Nonlinear Dynamics & Econometrics*, v. 12, n. 1, p. 1–24, 2008. Citado 2 vezes nas páginas 74 and 75.

KUGIUMTZIS, D.; TSIMPIRIS, A. Measures of analysis of time series (MATS): A MATLAB toolkit for computation of multiple measures on time series data bases. *Journal of Statistical Software*, v. 33, p. 1–30, 2010. Citado 2 vezes nas páginas 74 and 75.

KULLBACK, S.; LEIBLER, R. A. On information and sufficiency. *The annals of mathematical statistics*, JSTOR, v. 22, n. 1, p. 79–86, 1951. Citado na página 136.

KUMAR, D.; MURUGAN, S. Performance analysis of Indian stock market index using neural network time series model. In: *International Conference on Pattern Recognition, Informatics and Mobile Engineering*. Salem, India: [s.n.], 2013. p. 72–78. Citado 3 vezes nas páginas 20, 32, and 42.

KUNCHEVA, L. I.; FAITHFULL, W. J. PCA feature extraction for change detection in multidimensional unlabeled data. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 25, n. 1, p. 69–80, 2014. Citado 3 vezes nas páginas 29, 54, and 76.

Le Roux, N.; BENGIO, Y. Representational power of restricted Boltzmann machines and deep belief networks. *Neural computation*, MIT Press, v. 20, n. 6, p. 1631–1649, 2008. Citado na página 137.

LEE, H. et al. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: ACM. *26th Annual International Conference on Machine Learning*. Montreal, 2009. p. 609–616. Citado na página 137.

LEE, M.-C. Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, Elsevier, v. 36, n. 8, p. 10896–10904, 2009. Citado 2 vezes nas páginas 32 and 42.

LEMKE, C.; GABRYS, B. Meta-learning for time series forecasting and forecast combination. *Neurocomputing*, Elsevier, v. 73, n. 10, p. 2006–2016, 2010. Citado na página 24.

LEPAGE, Y. A combination of Wilcoxon's and Ansari-Bradley's statistics. *Biometrika*, Biometrika Trust, v. 58, n. 1, p. 213–217, 1971. Citado 2 vezes nas páginas 61 and 86.

LIANG, N.-Y. et al. A fast and accurate online sequential learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 17, n. 6, p. 1411–1423, 2006. Citado na página 81.

LIANG, X. et al. Improving option price forecasts with neural networks and support vector regressions. *Neurocomputing*, Elsevier, v. 72, n. 13, p. 3055–3065, 2009. Citado 2 vezes nas páginas 42 and 44.

LIN, J. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information theory*, IEEE, v. 37, n. 1, p. 145–151, 1991. Citado na página 136.

LIU, F.; WANG, J. Fluctuation prediction of stock market index by Legendre neural network with random time strength function. *Neurocomputing*, Elsevier, v. 83, p. 12–21, 2012. Citado na página 44.

LIU, S. et al. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, Elsevier, v. 43, p. 72–83, 2013. Citado 4 vezes nas páginas 23, 63, 65, and 66.

LU, C.-J.; LEE, T.-S.; CHIU, C.-C. Financial time series forecasting using independent component analysis and support vector regression. *Decision Support Systems*, Elsevier, v. 47, n. 2, p. 115–125, 2009. Citado 4 vezes nas páginas 20, 32, 42, and 43.

LUGHOFER, E.; ANGELOV, P. Handling drifts and shifts in on-line data streams with evolving fuzzy systems. *Applied Soft Computing*, Elsevier, v. 11, n. 2, p. 2057–2068, 2011. Citado na página 20.

LUO, F.; WU, J.; YAN, K. A novel nonlinear combination model based on support vector machine for stock market prediction. In: IEEE. *8th World Congress on Intelligent Control and Automation*. Jinan, China, 2010. p. 5048–5053. Citado na página 44.

MAES, S. et al. Credit card fraud detection using Bayesian and neural networks. In: *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*. Havan, Cuba: [s.n.], 2002. p. 261–270. Citado na página 21.

MAHDI, A.; HUSSAIN, A.; AL-JUMEILY, D. Adaptive neural network model using the immune system for financial time series forecasting. In: IEEE. *International Conference on Computational Intelligence, Modelling and Simulation*. Brno, Czech Republic, 2009. p. 104–109. Citado na página 44.

MAJHI, R.; PANDA, G.; SAHOO, G. Efficient prediction of exchange rates with low complexity artificial neural network models. *Expert systems with applications*, Elsevier, v. 36, n. 1, p. 181–189, 2009. Citado na página 44.

MAUCHLY, J. W. Significance test for sphericity of a normal n-variate distribution. *The Annals of Mathematical Statistics*, JSTOR, v. 11, n. 2, p. 204–209, 1940. Citado na página 120.

MINKU, L. L.; WHITE, A. P.; YAO, X. The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 22, n. 5, p. 730–742, 2010. Citado na página 50.

MINKU, L. L.; YAO, X. DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 24, n. 4, p. 619–633, 2012. Citado 4 vezes nas páginas 21, 53, 66, and 88.

MIRZA, B.; LIN, Z.; LIU, N. Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift. *Neurocomputing*, v. 149, Part A, p. 316–329, 2015. Citado na página 51.

MITCHELL, T. M. et al. *Machine learning*. [S.l.]: McGraw-hill, 1997. I–XVII p. Citado na página 46.

MONTGOMERY, D. C. *Design an Analysis of Experiments*. 6th. ed. [S.l.]: John Wiley and Sons, 2004. Citado na página 119.

MOOD, A. M. On the asymptotic efficiency of certain nonparametric two-sample tests. *The Annals of Mathematical Statistics*, JSTOR, v. 25, p. 514–522, 1954. Citado 2 vezes nas páginas 64 and 86.

NASIRI, B.; MEYBODI, M.; EBADZADEH, M. History-Driven Particle Swarm Optimization in dynamic and uncertain environments. *Neurocomputing*, Elsevier, v. 172, p. 356–370, 2016. Citado na página 88.

NAYAK, R. K.; MISHRA, D.; RATH, A. K. A Naive SVM-KNN based stock market trend reversal analysis for Indian benchmark indices. *Applied Soft Computing*, v. 35, p. 670–680, 2015. Citado na página 44.

NEMENYI, P. Distribution-free multiple comparisons. In: INTERNATIONAL BIOMETRIC SOCIETY. *Biometrics*. [S.l.], 1962. v. 18, n. 2, p. 263. Citado na página 96.

NETO, A. et al. Improving financial time series prediction using exogenous series and neural networks committees. In: IEEE. *International Joint Conference on Neural Networks*. Barcelona, 2010. p. 1–8. Citado na página 45.

OH, K. J.; HAN, I. Using change-point detection to support artificial neural networks for interest rates forecasting. *Expert systems with applications*, Elsevier, v. 19, n. 2, p. 105–115, 2000. Citado 4 vezes nas páginas 33, 55, 63, and 66.

OLIVEIRA, A. L.; MEIRA, S. R. Detecting novelties in time series through neural networks forecasting with robust confidence intervals. *Neurocomputing*, Elsevier, v. 70, n. 1, p. 79–92, 2006. Citado na página 20.

OLIVEIRA, F. A. d. et al. The use of artificial neural networks in the analysis and prediction of stock prices. In: IEEE. *International Conference on Systems, Man, and Cybernetics*. Anchorage, USA, 2011. p. 2151–2155. Citado na página 44.

PAGE, E. Continuous inspection schemes. *Biometrika*, JSTOR, v. 41, p. 100–115, 1954. Citado 2 vezes nas páginas 53 and 78.

PALIT, A. K.; POPOVIC, D. *Computational intelligence in time series forecasting: theory and engineering applications (Advances in industrial control)*. [S.l.]: Springer-Verlag, 2006. Citado 6 vezes nas páginas 33, 35, 36, 37, 42, and 43.

PARTAL, T.; KISI, Ö. Wavelet and neuro-fuzzy conjunction model for precipitation forecasting. *Journal of Hydrology*, Elsevier, v. 342, n. 1, p. 199–212, 2007. Citado na página 33.

PETTITT, A. Some results on estimating a change-point using non-parametric type statistics. *Journal of Statistical Computation and Simulation*, Taylor & Francis, v. 11, n. 3-4, p. 261–272, 1980. Citado na página 63.

PRUDÊNCIO, R. B.; LUDERMIR, T. B. Meta-learning approaches to selecting time series models. *Neurocomputing*, Elsevier, v. 61, p. 121–137, 2004. Citado 5 vezes nas páginas 24, 25, 71, 73, and 74.

PRUDÊNCIO, R. B.; LUDERMIR, T. B.; CARVALHO, F. d. A. de. A modal symbolic classifier for selecting time series models. *Pattern Recognition Letters*, Elsevier, v. 25, n. 8, p. 911–921, 2004. Citado 2 vezes nas páginas 24 and 71.

PULIDO, M.; MELIN, P.; CASTILLO, O. Particle swarm optimization of ensemble neural networks with fuzzy aggregation for time series prediction of the Mexican Stock Exchange. *Information Sciences*, Elsevier, v. 280, p. 188–204, 2014. Citado na página 45.

RAZA, H.; PRASAD, G.; LI, Y. EWMA model based shift-detection methods for detecting covariate shifts in non-stationary environments. *Pattern Recognition*, Elsevier, v. 48, n. 3, p. 659–669, 2015. Citado na página 20.

ROSS, G. J. Parametric and nonparametric sequential change detection in R: The cpm package. *Journal of Statistical Software*, v. 66, p. 1–20, 2013. Citado 5 vezes nas páginas 23, 64, 66, 86, and 94.

ROSS, G. J. et al. Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, Elsevier, v. 33, n. 2, p. 191–198, 2012. Citado 6 vezes nas páginas 22, 33, 52, 53, 78, and 79.

ROSSI, A. L. D.; CARVALHO, A. C.; SOARES, C. Meta-learning for periodic algorithm selection in time-changing data. In: IEEE. *Brazilian Symposium on Neural Networks (SBRN)*. [S.l.], 2012. p. 7–12. Citado 2 vezes nas páginas 25 and 26.

ROY, N.; MCCALLUM, A. Toward optimal active learning through monte carlo estimation of error reduction. In: *International Conference on Machine Learning*. [S.l.: s.n.], 2001. p. 441–448. Citado na página 53.

SCHLIMMER, J. C.; GRANGER, R. H. Beyond Incremental Processing: Tracking Concept Drift. In: *AAAI*. [S.l.: s.n.], 1986. p. 502–507. Citado na página 20.

SCHLIMMER, J. C.; GRANGER, R. H. Incremental learning from noisy data. *Machine learning*, Springer, v. 1, n. 3, p. 317–354, 1986. Citado na página 20.

SFETSOS, A. A novel approach for the forecasting of mean hourly wind speed time series. *Renewable Energy*, Elsevier, v. 27, n. 2, p. 163–174, 2002. Citado na página 33.

SHUMWAY, R. H.; STOFFER, D. S. *Time series analysis and its applications: with R examples*. [S.l.]: Springer Science & Business Media, 2010. Citado 2 vezes nas páginas 36 and 38.

SI, Y.-W.; YIN, J. OBST-based segmentation approach to financial time series. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 26, n. 10, p. 2581–2596, 2013. Citado na página 42.

SINGH, P.; BORAH, B. High-order fuzzy-neuro expert system for time series forecasting. *Knowledge-Based Systems*, Elsevier, v. 46, p. 12–21, 2013. Citado na página 20.

SOARES, S. G.; ARAÚJO, R. An on-line weighted ensemble of regressor models to handle concept drifts. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 37, p. 392–406, 2015. Citado 6 vezes nas páginas 22, 32, 51, 81, 88, and 124.

STREHL, A.; GHOSH, J.; MOONEY, R. Impact of similarity measures on web-page clustering. In: *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*. Austin, USA: [s.n.], 2000. p. 58–64. Citado na página 77.

TAY, F. E.; CAO, L. Application of support vector machines in financial time series forecasting. *Omega*, Elsevier, v. 29, n. 4, p. 309–317, 2001. Citado 2 vezes nas páginas 23 and 42.

TICKNOR, J. L. A Bayesian regularized artificial neural network for stock market forecasting. *Expert Systems with Applications*, Elsevier, v. 40, n. 14, p. 5501–5506, 2013. Citado na página 44.

TRESP, V. Committee machines. In: *Handbook for neural network signal processing*. [S.l.]: CRC Press, 2001. p. 1–18. Citado na página 44.

TSAI, C.-F.; HSIAO, Y.-C. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, Elsevier, v. 50, n. 1, p. 258–269, 2010. Citado na página 43.

TSYMBAL, A. et al. Dynamic integration of classifiers for handling concept drift. *Information Fusion*, Elsevier, v. 9, n. 1, p. 56–68, 2008. Citado 2 vezes nas páginas 22 and 51.

VIDYASAGAR, M. *Nonlinear systems analysis*. 2nd. ed. [S.l.]: Society for Industrial and Applied Mathematics (Siam), 2002. v. 42. Citado na página 35.

VINCENT, P. et al. Extracting and composing robust features with denoising autoencoders. In: ACM. *25th international conference on Machine learning*. Helsinki, 2008. p. 1096–1103. Citado na página 137.

WANG, B.; HUANG, H.; WANG, X. A novel text mining approach to financial time series forecasting. *Neurocomputing*, Elsevier, v. 83, p. 136–145, 2012. Citado na página 40.

WANG, J.; WANG, J. Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. *Neurocomputing*, v. 156, p. 68–78, 2015. Citado na página 43.

WANG, J.-Z. et al. Forecasting stock indices with back propagation neural network. *Expert Systems with Applications*, Elsevier, v. 38, n. 11, p. 14346–14355, 2011. Citado 3 vezes nas páginas 20, 32, and 43.

WANG, X.; SMITH, K.; HYNDMAN, R. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, Springer, v. 13, n. 3, p. 335–364, 2006. Citado 5 vezes nas páginas 24, 71, 72, 73, and 74.

WANG, X.; SMITH-MILES, K.; HYNDMAN, R. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomputing*, Elsevier, v. 72, n. 10, p. 2581–2594, 2009. Citado 2 vezes nas páginas 24 and 74.

WEBB, A. R. *Statistical pattern recognition*. [S.l.]: John Wiley & Sons, 2003. Citado na página 71.

WEI, W. W. S. *Time series analysis: univariate and multivariate methods*. 2nd. ed. [S.l.]: Pearson, 2006. Citado na página 33.

WERNER, R. et al. Study of structural break points in global and hemispheric temperature series by piecewise regression. *Advances in Space Research*, Elsevier, v. 56, n. 11, p. 2323–2334, 2015. Citado 3 vezes nas páginas 55, 62, and 66.

WIDMER, G.; KUBAT, M. Effective learning in dynamic environments by explicit context tracking. In: SPRINGER. *Machine learning: ECML-93*. [S.l.], 1993. p. 227–243. Citado na página 20.

WIDMER, G.; KUBAT, M. Learning in the presence of concept drift and hidden contexts. *Machine learning*, Springer, v. 23, n. 1, p. 69–101, 1996. Citado na página 20.

WU, L.; SHAHIDEHPOUR, M. A hybrid model for day-ahead price forecasting. *IEEE Transactions on Power Systems*, IEEE, v. 25, n. 3, p. 1519–1530, 2010. Citado na página 44.

YAMADA, M. et al. Change-Point Detection with Feature Selection in High-Dimensional Time-Series Data. In: *International Joint Conference on Artificial Intelligence*. Beijing: [s.n.], 2013. p. 1827–1833. Citado 2 vezes nas páginas 64 and 66.

YAMANISHI, K.; TAKEUCHI, J.-i. A unifying framework for detecting outliers and change points from non-stationary time series data. In: ACM. *International Conference on Knowledge Discovery and Data Mining*. Edmonton, Canada, 2002. p. 676–681. Citado 2 vezes nas páginas 62 and 66.

YUAN, Y. Forecasting the movement direction of exchange rate with polynomial smooth support vector machine. *Mathematical and Computer Modelling*, Elsevier, v. 57, n. 3, p. 932–944, 2013. Citado na página 44.

ZHANG, G. P.; PATUWO, B. E.; HU, M. Y. A simulation study of artificial neural networks for nonlinear time-series forecasting. *Computers & Operations Research*, Elsevier, v. 28, n. 4, p. 381–396, 2001. Citado na página 89.

ZHANG, T. et al. Adaptive correlation analysis in stream time series with sliding windows. *Computers & Mathematics with Applications*, Elsevier, v. 57, n. 6, p. 937–948, 2009. Citado na página 52.

ZHANG, X.-L.; WU, J. Deep belief networks based voice activity detection. *IEEE Transactions on Audio, Speech, and Language Processing*, IEEE, v. 21, n. 4, p. 697–710, 2013. Citado na página 137.

ZHU, B.; WEI, Y. Carbon price forecasting with a novel hybrid ARIMA and least squares support vector machines methodology. *Omega*, Elsevier, v. 41, n. 3, p. 517–524, 2013. Citado na página 44.

ZLIOBAITE, I. et al. Next challenges for adaptive learning systems. *ACM SIGKDD Explorations Newsletter*, ACM, v. 14, n. 1, p. 48–55, 2012. Citado na página 48.

ZLIOBAITĖ, I.; BUDKA, M.; STAHL, F. Towards cost-sensitive adaptation: when is it worth updating your predictive model? *Neurocomputing*, Elsevier, v. 150, p. 240–249, 2015. Citado 4 vezes nas páginas 22, 23, 52, and 65.