UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Prediction of time series using different types of forecasting methods enhanced with a meta-learning approach

Trabajo de integración curricular presentado como requisito para la obtención del título de
Ingeniero en Tecnologías de la Información

Autores:

Betancourt Benavides Hugo Ernesto
Brito González Carlos Andrés

Tutor:

Ph.D Fonseca Delago Rigoberto Salomón

Urcuquí, febrero 2020

Urcuquí, 27 de febrero de 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2020-00004-AD

En la ciudad de San Miguel de Urcuquí, Provincia de Imbabura, a los 27 días del mes de febrero de 2020, a las 14:00 horas, en el Aula CHA-02 de la Universidad de Investigación de Tecnología Experimental Yachay y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Miembro No Tutor	Dr. INFANTE QUIRPA, SABA RAFAEL , Ph.D.
Tutor	Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.

Se presenta el(la) señor(ita) estudiante **BRITO GONZALEZ, CARLOS ANDRES**, con cédula de identidad No. **0603542556**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, con el objeto de rendir la sustentación de su trabajo de titulación denominado: **Prediction of time series using different types of forecasting methods enhanced with a metal-learning approach**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN.**

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Dr. INFANTE QUIRPA, SABA RAFAEL , Ph.D.	10,0
Tutor	Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	10,0

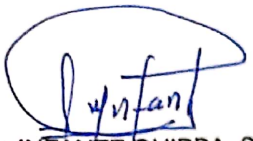
Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

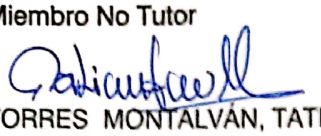
BRITO GONZALEZ, CARLOS ANDRES
Estudiante

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Presidente Tribunal de Defensa

Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.
Tutor



Dr. INFANTE QUIRPA, SABA RAFAEL , Ph.D.
Miembro No Tutor



TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

Urcuquí, 27 de febrero de 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN
ACTA DE DEFENSA No. UITEY-ITE-2020-00005-AD

En la ciudad de San Miguel de Urcuquí, Provincia de Imbabura, a los 27 días del mes de febrero de 2020, a las 14:00 horas, en el Aula CHA-02 de la Universidad de Investigación de Tecnología Experimental Yachay y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Miembro No Tutor	Dr. INFANTE QUIRPA, SABA RAFAEL , Ph.D.
Tutor	Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.

Se presenta el(la) señor(ita) estudiante **BETANCOURT BENAVIDES, HUGO ERNESTO**, con cédula de identidad No. **1104723869**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **TECNOLOGÍAS DE LA INFORMACIÓN**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-43-No.496-2014**, con el objeto de rendir la sustentación de su trabajo de titulación denominado: **Prediction of time series using different types of forecasting methods enhanced with a metal-learning approach**, previa a la obtención del título de **INGENIERO/A EN TECNOLOGÍAS DE LA INFORMACIÓN**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Miembro Tribunal De Defensa	Dr. INFANTE QUIRPA, SABA RAFAEL , Ph.D.	10,0
Tutor	Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.	10,0
Presidente Tribunal De Defensa	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	10,0

Lo que da un promedio de: **10 (Diez punto Cero)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

BETANCOURT BENAVIDES, HUGO ERNESTO
Estudiante

Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Presidente Tribunal de Defensa

Dr. FONSECA DELGADO, RIGOBERTO SALOMON , Ph.D.
Tutor

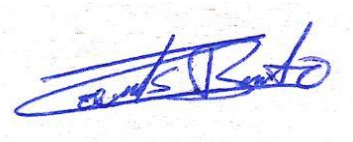
Dr. INFANTE QUIRPA, SABA RAFAEL , Ph.D.
Miembro No Tutor

TORRES MONTALVÁN, TATIANA BEATRIZ
Secretario Ad-hoc

AUTORÍA

Yo, **CARLOS ANDRÉS BRITO GONZÁLEZ**, con cédula de identidad 0603542556, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autora (a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, febrero 2020.



_____

Carlos Andrés Brito González
CI: 0603542556

AUTORÍA

Yo, **HUGO ERNESTO BETANCOURT BENAVIDES**, con cédula de identidad 1104723869, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autora (a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, febrero 2020.

_____

Hugo Ernesto Betancourt Benavides

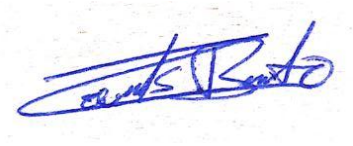
CI: 1104723869

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **CARLOS ANDRÉS BRITO GONZÁLEZ,** con cédula de identidad 0603542556, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior

Urcuquí, febrero 2020.



_____

Carlos Andrés Brito González
CI: 0603542556

AUTORIZACIÓN DE PUBLICACIÓN

Yo, **HUGO ERNESTO BETANCOURT BENAVIDES** con cédula de identidad 1104723869, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior

Urcuquí, febrero 2020.

Hugo Ernesto Betancourt Benavides
CI: 1104723869

Dedicatoria

Dedico este trabajo a mis padres que con su amor incondicional y con la ayuda de Dios han sabido guiarme y apoyarme a través de esta etapa completamente nueva en mi vida. De igual manera dedico este trabajo a mi hermano Juan Sebastian Betancourt, por ser siempre un gran amigo, hermano e hijo que siempre da apoyo y felicidad a la familia, cuando he tenido que ausentarme.

Dedico esta tesis igual a todos mis amigos, a aquellos nuevos amigos que conocí en Yachay, que fueron parte de mi grupo de trabajo y aquellos con los que conviví en el mítico departamento G3-4 y sus recurrentes visitas, de igual manera a mis amigos de Loja que pese a la distancia siempre siento que puedo contar con ellos. Debido a que la memoria es frágil no quiero poner nombres, pero sin lugar a dudas sin contar con tan buenos amigos es difícil mantener la motivación.

Dedico este trabajo a los excelentes profesores de Yachay que tuve. Sin duda este sueño llamado Yachay no fuera posible sin la predisposición de nuestros profes que han ayudado no solo a nuestra formación académica sino también a expandir nuestros sueños. Aprovecho por agradecer a mi tutor Rigoberto Fonseca que nos ha ayudado a introducirnos en un campo tan genial como lo es la inteligencia artificial y a mi compañero de tesis Carlos Brito que sin lugar a dudas somos un gran equipo de trabajo y sobre todo grandes amigos.

Finalmente dedico este trabajo a Dios que me ha guiado hasta este punto de mi vida, y ha puesto excelentes personas en mi camino al fin de cuentas dedico este trabajo a cada persona de la que pude aprender o convivir, porque no somos más que el promedio de las personas que están a nuestro alrededor.

Hugo Ernesto Betancourt Benavides

Dedico este trabajo a mi padre quien sin su apoyo incondicional este logro no sería posible. De la misma manera quisiera agradecer la ayuda que siempre me ha brindado mi madre.

Dedico también esta tesis a mis amigos quienes conocí en Yachay, quienes siempre me han incentivado a ser mejor. Se que ellos siempre serán una segunda familia para mí.

Carlos Andrés Brito González

Resumen

La selección de un modelo de predicción para una serie temporal con un comportamiento irregular generalmente requiere un humano experto; en muchas ocasiones no es posible tener este experto debido al tiempo o costo. Una máquina inteligente puede seleccionar el modelo de predicción analizando el rendimiento de los modelos, esta estrategia se conoce como meta-aprendizaje, en este trabajo se utilizó redes neuronales del tipo Feed-Forward y árboles de decisión para realizar modelos de meta-aprendizaje. Para realizar esta estrategia es necesario la selección de modelos basados en diferentes principios, de esta manera se provee variabilidad a la lista de métodos. Este trabajo propone el uso de predictores basados en redes neuronales como lo son FFNN y LSTM, además, de la inclusión de ARIMA un método estadístico clásico utilizado en la literatura. Es importante mantener en la lista siempre métodos con un desempeño alto que provean información relevante al aprendizaje. Esto fue posible con el uso de algoritmos genéticos que mejoran modelos de redes neuronales. Los resultados obtenidos demuestran que el uso de un enfoque de meta-aprendizaje, permite reducir el costo computacional sin disminutir el desempeño de la predicción.

Palabras Clave:

Meta-aprendizaje, Predicción de series de tiempo, Algoritmos genéticos.

Abstract

The selection of a prediction model for a time series with irregular behaviour generally requires the help of a human expert; in many cases it is not possible to have this expert due to time or cost He/She represents. An intelligent machine can select the prediction model by analyzing the performance of different models, this strategy is called meta-learning, this work used Feed-Forward neural networks and decision trees to make the meta-learning models. To make this strategy possible it is necessary to select models based on different principles, this way we can provide variability to the list of methods. This work proposes the used of predictors based on neural network such as FFNN and LSTM, also, the inclusion of ARIMA a classical statistical method used in literature. It is important to maintain methods on the list with high performance, which provide information relevant to the learning process. This was possible with the used of Genetic Algorithms that improve neural network models. The results that were obtained show that the use of a meta-learning strategy allows the reduction of computational costs without reducing the performance of the prediction.

Key Words:

Meta-learning, Time series prediction, Genetic algorithms

Contents

List of Figures

List of Tables

1 Introduction

There exists a great interest in the industry and academy to improve the accuracy of time series predictions. These are important in a variety of fields such as: Medicine, Finances, Science due to the fact that the data that is recollected from these fields show patterns that we can quantify (seasonality, trend, variability) [6]. The ability to predict time series values requires selecting the best prediction model to perform the forecast, which is a challenging problem due to the complexity of the data and the number of available models [7]. This work studies different forecasting methods, and strategies for model selection with the aim of improving the accuracy of the prediction.

According to Fu (2011) [8], a time series is a set of chronologically recorded observations of one variable or multiple variables called uni-variable or multi-variable respectively. Furthermore, time series data is in general continuous and substantial in its size. This means that the amount of data should be taken in certain intervals and that the size of the data, in this case, it must be enough so that a prediction model may be used to predict the next values.

In literature, there are various models that may be used to produce a forecast. For instance, traditional statistical methods such as Exponential Smoothing and Auto-regressive Integrated Moving Averages (ARIMA) [9] are statistically powerful. Another approach for forecasting comes from the computer science field of artificial intelligence. There exist different types of Artificial Neural Networks (ANNs) such as Long short term memory networks, Feed-Forward Neural Network and more complex networks like Convolutional neural networks. ANNs methods based on the imitation of biological neural systems [10], have advantages over other forecasting models such as statistically based methods. First, ANNs only use a linear amount of parameters where as other methods use an exponential amount to achieve the same results[11]. Second, they have the capability to be flexible while mapping continuous nonlinear functions with a certain accuracy[12]. Although ANNs have many advantages for forecasting some networks still suffer from a local minimum issue, a slow learning rate and a problem to find the correct hyper-parameters of the system.

The issues previously mentioned could be supplemented by using an optimization algorithm such as the Genetic Algorithm (GA)[13, 14, 15]. This hybrid technique consists of different types of AI methods enhanced with a GA to find their optimal hyper parameters in an efficient way. Thus the proper use of the GA is desirable, for this we consider certain hyper parameters that will vary with

each different type of ANN. These hyper parameters could be the number of input neurons, number of neurons in the hidden layer, number of hidden layers, activation function, etc [1]. After applying the GA, we have an enhanced structure for the ANN (from the hyper-parameters that we consider), with this approach we can compare different ANNs avoiding errors due to a set of bad hyper-parameters. This part is crucial because if we are not sure that the ANN is working with an optimal structure we can not compare different ANNs since its accuracy depends on the structure of the network. Therefore if we do not use a hyper-parameter optimization process, for the same types of ANN we will have different accuracies.

There are many methods of predicting time series, such as statistical or artificial intelligence methods, both have their different advantages and disadvantages. Furthermore, depending on the characteristics of the time series the models will perform better or worse. For this reason, there are several studies based on which method will work better than others depending on the characteristics of the time series of interest. Then, we can train meta-learning models, mentioned in the following paragraphs, to learn which forecasters perform better depending on the time series.

As was mention on [4] and [16], meta-learning is an approach that allows us to use the characteristics of the time series to choose the best forecasters or combination of forecasters from a pool (group of forecasters). Meta-learning studies how machines can improve by using experience efficiently [16]. This field is especially important because there is not a learning method that is always better than another [17] in all data fields, this was proven in [18]. Meta-learning can be used to determine when a method could have the best performance from a set of methods by looking at the characteristics of the data first.

1.1 Problem statement

There is a great interest to improve the performance of time series forecasting; this means that the forecasters must preform well in situations where the data is complex. The problem is how to select an appropriate predictor for a time series with certain characteristics. Furthermore, the issue is also what parameters should these predictors use.

1.2 Hypothesis

- The Meta-learning methods, once trained, will reduce the computational cost, since it is not necessary to train all pool methods, only the model selected by the meta-learning approach. This is done without trading precision for computational cost.

1.3 Objectives

To prove or disprove the previous hypothesis we have established the following general objective.

General Objective

- Develop a meta-learning method of time series prediction with different types of forecasting models in order to chose the appropriate predictor for the time series based on the characteristics of the data.

Specific Objectives The following specific objectives will help us accomplish the general objective.

I Analyze methods for time series prediction that has different principles. In this way, we give variability to the pool.

II Improve the basic models chosen for the pool to generate a new pool of more robust methods. This would be the pool for the meta-learning methods.

III Analyze different features for time series, which can allow the use of meta-learning approach.

IV Develop of a meta-learning method applied on the pool using features from time series, to select a suitable predictor.

1.4 Contribution

This work proposes a meta-learning method which uses a decision tree or a Feed Forward Neural Netowrks (FFNN), one or the other. The input that they receive is a set of certain characteristics of a time series, such as Chaos or Kurtosis, and the result of which enhanced forecaster was better for that series. The enhanced forecaster are chosen from a pool or group of forecasters. Finally, we obtained a decision tree and a FFNN that will be able to determine what forecaster is better for a time series with certain characteristics.

The rest of this work continues with section 2, which describes the basic concepts that are required for this work. Related work that has been done to solve the problem is stated in section 3. The solution that we proposed is presented and explained in section 4. Section 5 is dedicated to the presentation and analysis of the results. Finally, sections 6 shows our conclusions future work.

2 Theoretical framework

This section explains basic concepts that are necessary to understand this work. We start by defining a time series and the characteristics that will be used for the meta-learning model. Second, the prediction models used will be explained. Third, methods to improve the performance of these prediction models. Fourth, the concept of meta-learning and the models that we used will be explained. Finally, the metrics used to quantify and analyze our results will be presented.

2.1 Time series

As defined by Brockwell & Davis (2016)[19], *a time series is a set of observations, each one being recorded at a specific time*. From this definition we can divide time series into: univariate, that handles only one variable, and multivariate, that handles two or more variables. Furthermore, we can see that the univariate time series, such as those in the NN3 competition (Appendix B) [1], have only one variable in contrast with gas furnace data from [9], that have more variables. This work focuses on univariate time series forecasting due to simplicity.

2.1.1 Univariate time series

A univariate time series according to [20] is the simplest form of temporal data. A univariate time series can be represented as an ordered set of n real-valued variables:

$$x = x_1, x_2, .., x_{n-1}, x_n \tag{1}$$

A time series can be described using a set of different characteristics such as seasonality, trending, noisy, non-linear, chaos, etc.

2.1.2 1-step ahead vs n-step ahead

As its name mentions 1-step ahead prediction is a method that is used when we have $x_1, x_2, \ldots, x_n$ and then we only want to predict the x_{n+1} value [21]. Where, n is the number of observations of time series. This method is also called Open loop forecasting because it does not use the previous result of a forecast to predict the next value instead, use the real value [22].

[1] The dataset is available at: http://www.neural-forecasting-competition.com/NN3/

N-steps ahead prediction is a method that when we have $x_1, x_2, \ldots, x_n$ observations, we then want to predict the next x_{n+t} values, t is the number of predictions in the future that are of interest [21]. This method is called Closed loop forecasting because it uses a previous forecasting value in order to predict the necessary steps ahead [22]. For example if $t = 2$ then we would require x_{n+1} step in order to predict x_{n+2}. To solve this, first predict x_{n+1} and then predict the following value, x_{n+2}.

From the definitions we can see that the 1-step ahead prediction will be more precise than the n-steps ahead. Because n-steps uses previous predicted values for the next prediction causing the errors of these predictions to accumulate. While 1-step ahead only takes into account the error of a single prediction [23]. Due to this reason, in this work we focus on the 1-step ahead prediction.

2.1.3 Time Series Characteristics or Features

The characteristics of the time series are very diverse as mentioned in different studies [4, 24, 20, 25]. The choice of characteristics is crucial to correctly classify the time series depending on the method that will be used to forecast. For this reason, we decided to use the results obtained in Wang et al (2005) [20], which mentions the most informative and representative characteristics to summarize the time series structure. We have also used as a base Lemke & Gabrys (2010) [4], which uses a simpler method to find the trend and introduces useful features coupled to the NN3 data set, seen in Appendix B. The characteristics used in this work are listed below:

2.1.3.1 Trend : A trend pattern exists when there is a long-term change in the mean level. To estimate the trend and other characteristics is important to modify the time series. For instance, as is mentioned in [20] using the Box–Cox transformation, to modify the time series, allows us to stabilize the variance, to make the seasonal effect additive and to make the data normally distributed. Also, a more simple method is mention in [4], which presents detrended time series using a polynomial regression of up to order three.

For practical reasons, in this work, a detrended time series using a polynomial regression of up to order three will be used. The measure of how much of the variability of the series can be accounted with the trend curve is obtained by equation 2 [4].

$$\frac{\sigma_X}{\sigma_{X'}} \tag{2}$$

Where σ_X is the standard deviation of the series, and $\sigma_{X'}$ is the standard deviation of the detrended series.

2.1.3.2 Autocorrelation and partial autocorrelation : Autocorrelation and partial autocorrelation give indications on stationarity and seasonality of the time series [4]. The implementation of these measures was done with "tsfresh" library [2], according to equation 3 for autocorrelation, and equation 4 for partial autocorrelation.

$$\frac{1}{(n-k)\,\sigma^2}\sum_{t=1}^{n-k}(x_t-\mu)\,(x_{t+k}-\mu) \tag{3}$$

where n is the length of the time series x_i, σ^2 its variance, μ its mean and k denotes the lag.

$$\frac{Covariance\left(x_t, x_{t-k}|x_{t-1},...x_{t-k+1}\right)}{\sqrt{\sigma\left(x_t|x_{t-1},...x_{t-k+1}\right)\sigma\left(x_{t-k}|x_{t-1},...x_{t-k+1}\right)}} \tag{4}$$

where $x_t = f(x_{t-1},...,x_{t-k+1})$ and $x_{t-k} = f(x_{t-1},...,x_{t-k+1})$, being AR($k-1$) models (see Appendix D) that can be fitted by ordinary least squares regression (OLS), more information about the implementation can be obtain in "tsfresh" library.

Autocorrelation and partial autocorrelation was calculated for lags 1 and 2 [4]. Also, a lag of 12 was also used to take advantage of the fact that the NN3 data set was obtained in monthly intervals.

2.1.3.3 Non-linearity : Non-linearity is an important time series characteristic to determine the selection of a forecasting method, nonlinear time series are used to model complex dynamics not adequately represented by linear models, an example of this is the 'sunspot' data sets and 'lynx' data set that have identical non-linearity structure [20, 26]. As was mention on [20], there exist different approaches to test the non-linearity such as non-parametric kernel test and a Neural Network test (NN test). Different studies that compare both methods show better reliability in NN test [27]. Specifically, the test used in this research to measure the time series data non-linearity was the Teräsvirta's NN test [28].

2.1.3.4 Skewness : The skewness of distribution of a dataset is an indication of how much the data "leans" or is skewed to one side or another[29]. This is an indicator for which tail of the data is

[2]The library is available at: https://tsfresh.readthedocs.io/en/latest/

bigger than the other. For example for the normal distribution the skewness is zero, indicating that the data is not skewed to one side or another.

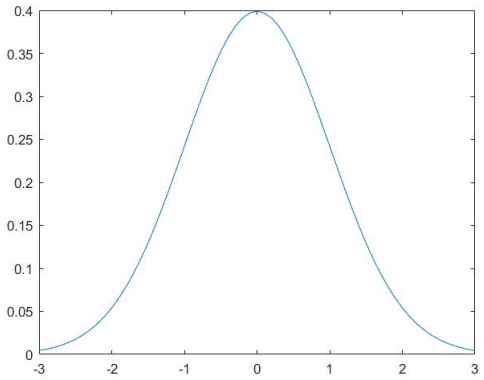


Figure 1: Normal Distribution example with skewness zero

If the skewness is not zero then this indicates that the distribution is skewed to one side or another. If the value is negative then it is skewed to the left, if it is positive then it is skewed to the right. [29]

The equation used in this work is the one proposed in [24]:

$$S = \frac{1}{n\sigma^3}\sum_{t=1}^{n}(x_t - \mu)^3 \tag{5}$$

where μ is the mean, σ is the standard deviation and n is the number of data points:

2.1.3.5 Kurtosis : The Kurtosis, in contrast with skewness, is a statistical measure that is used to describe the distribution as mentioned in [30]. In general, if the value is positive then it indicates a peaking distribution, if it is negative then it is indicative of a flat distribution [29]. The equation that was used is the one proposed in [24].

$$K = \frac{1}{n\sigma^4}\sum_{t=1}^{n}(x_t - \mu)^4 - 3 \tag{6}$$

2.1.3.6 Self-similarity : This characteristic is the long-range dependency that the data may have, showing that parts of the data may be similar to other parts of the same data. To determine the value of this characteristic we used the Hurst Parameter (H) and we used the method described in [24]. To estimate this parameter Wang (2006) [24] uses a Fractional Autoregressive Moving Average

(FARIMA). Then, depending on the d parameter used, H is calculated by :

$$H = d + 0.5 \tag{7}$$

2.1.3.7 Chaos : The measure of chaos is used in non linear dynamic systems to determine the dependence on the initial values that the system shows [24]. The way this characteristic is quantified is by using the Lyapunov Exponent (LE), which measures how much nearby trajectories separate from each other [31]. We used the Hilborn algorithm mentioned by Wang (2006) [24] in order to obtain this characteristic.

2.1.3.8 Periodicity : The periodicity is the measure of how often a certain pattern in the time series is repeated in the data [32]. This characteristic is very important in determining the seasonality of the time series. For this reason we will be using the algorithm that was proposed by Wang (2006) [24]. The algorithm starts by doing a regression spline with 3 knots, then an autocorrelation function is applied with the lags up to 1/3 the length of the data. This function then allows us to look for peaks in the values finally obtaining the value of the periodicity.

All these characteristics extracted from a time series are analyzed in this work to distinguish time series from one another.

2.2 Prediction models

Prediction models are those that use computational and mathematical methods to forecast a value or an occurrence [33]. There are a variety of prediction models that are based on different principles that can be used depending on the situation, such as: Artificial Intelligence models, Statistical models, Hybrid models, etc. The importance of studying these models is because each model has different advantages depending on the data used. In this section we will review the models that are mentioned in the literature.

2.2.1 Autoregressive Integrated Moving Average (ARIMA)

ARIMA is a linear modeling technique that can be represented as ARIMA(p, d, q) [34]. Where p is the number of autoregressive terms, q is the number of lagged forecast errors and d is the number of non-seasonal differences needed for stationarity [35]. The predicted value at a given time t is x_t and

it is expressed as:

$$x_t = \mu + a_1 x_{t-1} + a_2 x_{t-2} + \cdots + a_p x_{t-p} + n_t + b_1 n_{t-1} + \cdots + b_q n_{t-q} \tag{8}$$

where $b_1 n_{t-1} + \cdots + b_q n_{t-q}$ is the MA terms (lagged errors), $a_1 x_{t-1} + a_2 x_{t-2} + \cdots + a_p x_{t-p}$ is the AR terms (lagged values) and μ is a constant. d is used to convert the series into a stationary series, as seen below [34] [36].

This technique can be summarized by taking into account its (p, q, d) values:

1. Differencing (d): This step is to remove any trend that the data has and is generally done by substracting past values, let y be the new time series and x the original:

Table 1: ARIMA (d) Differencing example [3]

d value	**Differenced**
0	$y_t = x_t$
1	$y_t = (x_t - x_{t-1})$
2	$y_t = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$

   This step must be undone after to obtain the original time series.

2. AR and MA p, q: For these values we would usually have to do autocorrelation factor (ACF) and partial autocorrelation factor (PACF) plots. This is done to identify the possible values of p and q. In equation 8 we can see that the p values correspond to the x (values of the series) and the q values to the n (errors).

3. Result: From the previous steps we would now obtain our (p, q, d) values that will be used to produce the equation for the forecast, that is shown in equation 8.

Expanding on the steps previously explained, the selection of these parameters is also usually done by using methods such as correlation analysis for p, q or Box Jenkins for the model coefficients [34]. However in this work, we used an auto ARIMA function that will find the optimal values for p,q and d. This was decided because ARIMA can have many combinations of parameters of p, q and d. Also, by automatizing the ARIMA prediction it allows us to compare the the ANN methods with the "best" ARIMA.

The auto ARIMA method used comes from the pmdarima library [3]. This method searches for the best p, q and d values and chooses the model with the least Akaike Information Criterion (AIC) [37] or Bayesian Information Criterion (BIC) [38] depending on which criterion was chosen. In this case, we use the AIC criteria, which is used to obtain an estimate of how well econometric models will perform on the data of interest. Also, it gives an estimate on how much information will be lost when using a certain model. For differencing terms it uses tests for seasonality and stationarity for seasonal models [39].

2.2.2 Feed-forward neural network

A Feed-Forward Neural Network (FFNN) is a straightforward type of neural network where the information moves only in one direction from the input nodes, through the hidden nodes, and to the output nodes. There are no cycles or loops in the network. FFNN is composed of three type of layers, which are input layer, hidden layers and the output layer (see Figure 2). According to Haikyn [40] the hidden layers funcion is to find a relation between the external input and the expected output, the existence of hidden layers let the network to extract higher-order statistics.

FFNNs have the following hyper-parameters:

- **Activation function :** The activation function is used in each neuron to determine if it will be activated and use the input that it has received.
- **Number of Hidden layers :** This number refer to how many layers there exists between the input and output layer. These layers determine the structure of the network.
- **Number of Hidden layer's neurons :** This parameter indicates how many neurons are in each hidden layer.
- **Input neurons :** The input neurons indicate the amount of data that will enter the Network.
- **Learning rate :** This parameter is a value in the interval $[0, 1]$ that indicates how much information will be learned each time a new value enters the cell. In other words, if the learning rate is low then the cell will ignore more data than what it learns, otherwise it will learn more than what it ignores.

[3] The library is available at: https://www.alkaline-ml.com/pmdarima/

- **Gradient Descent Optimizer :** The gradient descent optimizer is used for the training phase of the FFNN so that it can recalculate its weights.

The layers distribution can be observed in Figure 2.

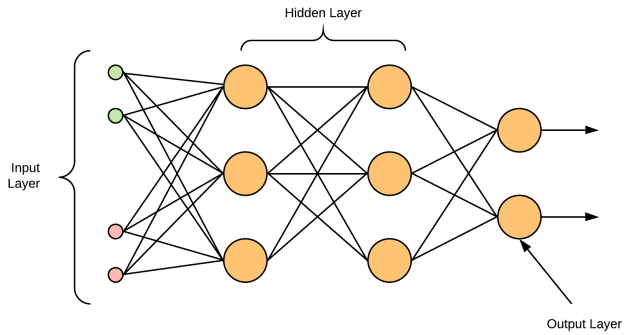


Figure 2: Structure of a Feed-forward neural network based on [1]

The learning process of this neural network consists in to use a group of n values of the time series, which correspond to its input layer. And an output value that corresponds to the prediction of the $n + 1$ value. The real value of the prediction is used to correct the synaptic weights of the neural network, using the backpropagation algorithm presented in the Appendix A. The n values, are taken with a sliding windows approach, which is later explained .

2.2.3 Long Short Term Memory (LSTM)

LSTM is a type of Recurrent Neural Network (RNN) that uses memory blocks instead of neurons [41]. LSTM was introduced by Hochreiter and Schmidhuber (1997) as a solution to the long-range dependency that the Backpropagation training algorithm has. This is done in the memory blocks through three gates: input, output and forget, which allows the network to learn information for long periods of time and to eliminate unnecessary information.

As mentioned above the memory block of the LSTM is the most important part of this RNN. Figure 3 shows the three gates together in order to produce: a forecasted value and the memory of the cell. In the forget layer, that receives C_{t-1} is in charge of determining what information should be ignored from the previous cell. The input layer, decides what information should be saved to the cell state (or memory). Finally, the output layer provides us with the next cell state and/or the

forecasted value [41]. Refer to Appendix C for a more detailed explanation of how LSTM works and its equations. The variables of the LSTM cell are defined in Table 2.

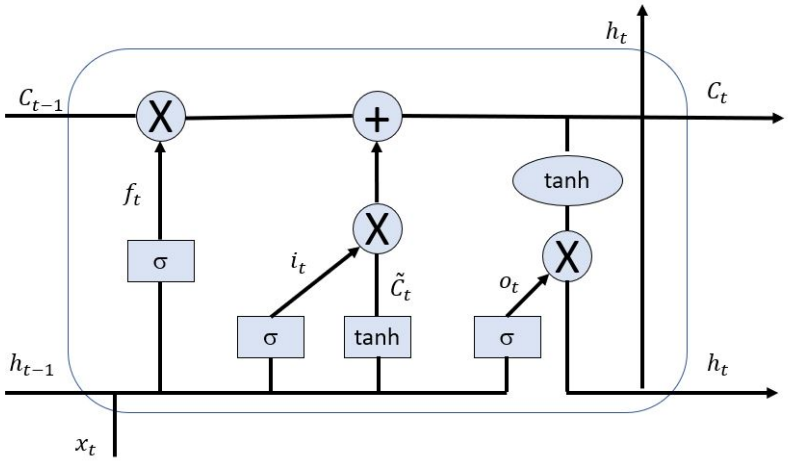


Figure 3: LSTM cell [2]

Table 2: Definition of LSTM Variables

LSTM variables			
Variable	**Definition**	**Variable**	**Definition**
σ	sigmoid layer	$tanh$	hyperbolic tangent layer
h_{t-1}	output of the previous LSTM layer	C_{t-1}	cell state of the previous LSTM layer
x_t	current input	C_t	cell state
$\tilde{C}_t$	new candidate values vector for the cell state	h_t	current output
f_t	what the cell will forget		

Description of the operators:

$\otimes$: is the scaling of information

$\oplus$: is the addition of information

As well as the FFNN, the LSTM networks, has hyper-parameters which are the following.

- **Activation function :** In this case the activation function is used in the gates that the LSTM has, to determine if the new values will or will not be learned. It is also used to to determine the output of the cell.
- **Number of Hidden layers :** The number of hidden layers corresponds to the number of layers of cells that are in the middle of the input and output layers. They also determine, in part, the structure of the LSTM network.
- **Number of Hidden layer's cells :** This is the number of cells that will be in each hidden layer.
- **Input cells :** The number of input cells is the number of data entries that the LSTM network will have, this is the starting point of the network.
- **Learning rate :** This parameter has the same use as the FFNN, that is, how a cell treats the data it receives. If the value is closer to 1 then it learns at a fast rate, if it is closer to 0 it learns slow rate.
- **Gradient Descent Optimizer :** The gradient descent optimizer is used for the training phase of the LSTM in order to improve its performance.

2.3 Adjusting hyperparameters of predictors

These methods are necessary since it is not possible to test and adjust each parameter of the methods mentioned in the previous section in a realistic time frame. For instance, the number of input neurons, layers, which optimizer or activation function. From this, we consider the following improvement methods from which we may obtain an improved version of the forecaster.

2.3.1 Genetic Algorithms

Genetic algorithms (GA) are a heuristic optimization that generates a nearly optimal solution [42]. Introduced by J. Holland in the 1970's and inspired by the biological evolution of living beings. Genetic algorithms abstract the problem space as a population of individuals, and try to explore the fittest individual by producing generations iterative. The objective of GA is to evolve generations of initial individuals iteratively into a population of better individuals in each iteration. Each of these of

the final population has the features to find a solution of the problem. The procedure starts from an initial random population which tries to contain all the genes of interest for the test[43].

During each generation, three basic genetic operations are applied to each individual; the operations are: selection of the fitness individual, crossover and mutation on genes [43]. The diagram of this process is showed in Figure 4:

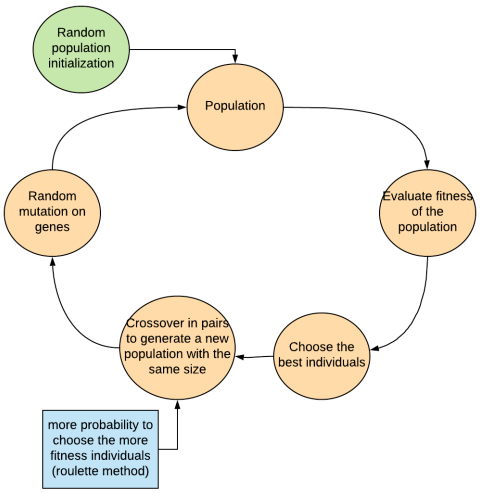


Figure 4: Steps of a genetic algorithm

2.3.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a heuristic method of optimization is similar to the GA instead of having a selection process each individual is influenced by each other and each individual usually always survives until the end [44]. This method was first proposed by Kennedy [45] a social psychologist that was modelled by observing flocks of animals. The objective of this method is to allow a group of individuals to move bit by bit to find an optimal solution.

This method works by first creating a group of individuals at random positions and that have a certain velocity. Then for the first iteration the fitness function is evaluated and the position and velocity are adjusted accordingly. They are adjusted by taking into account the best result from the group and the best result that they obtained individually; these values are used with the velocity of the individual to calculate the new position and velocity [45].

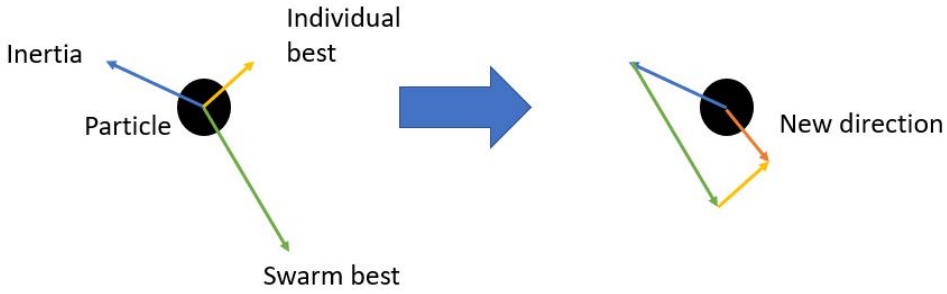


Figure 5: Example of estimating the new velocity in PSO

2.3.3 Summary of Methods to improve the performance of forecasters

In summary, in this section we presented some methods to improve the performance of forecasters, such as: GA and PSO. Each method can be used in specific circumstances but in this case we will be using the GA. The reason for choosing GAs is the excellent results they demonstrate when working with Neural Networks as mentioned in [43], where Lingaraj shows many successful combinations of neural networks with genetic algorithms. For example, a proposal that uses GAs to fit the parameters of a model to optimize pumping locations and schedules, they combined the GA with an NNs to model the complex response functions [46]. Or Zhao that use the NN and GA for functional testing [47].

2.4 Meta-learning

There exist different interpretations of the term meta-learning in the literature [4]. For example, is mention in [48] that the objective of meta-learning is to "understand how learning itself can become flexible according to the domain or task understudy." For us, the interpretation that is more in line with the tasks under study is in [49], which mentions that "the aim of the area of meta-learning is to facilitate the task of selecting, adapting or composing machine learning, or data mining solutions". This definition is quite similar to the one mentioned in [4, 50], which mentions that meta-learning "is referred to as the process of linking knowledge on the performance of so-called base-learners to the characteristics of the problem".

Taking into account the aforementioned, the approach that we give to meta-learning is to facilitate and optimize the selection of a good prediction method based on the characteristics of the time series. For this, we use the performance data of the base-learners, which in our specific case are the optimized

forecasters, FFNNs with GA, LSTM with GA, and ARIMA with auto-ARIMA approach. In this context, we are going to review a set of classification methods for the meta-learning level.

2.4.1 Decision Trees

A Decision Tree (DT) is one of the most used machine learning algorithms in classification and for meta learning proposes due to the induction of decision tree is deterministic. An example of its usage for meta-learning approaches can be found on [51, 4]. A DT creates question nodes using the labeled data set as input; each node separates the classes in the best way possible, the nodes that perform a clear separation between classes are preferable. If there is no way to create a node such that there is a clear distinction between classes, then it is created by separating the classes the best way possible. Figure 6 shows the structure of the DTs which is the following [52]:

- Root node: Represents the entire population.
- Decision node: When a sub-node is divided into sub-nodes.
- Leaf: Nodes that do not split.
- Branch: A full sub-section of the entire tree.
- Parent: Node divided into sub-nodes is the parent.
- Child node: The sub-nodes of a Parent.

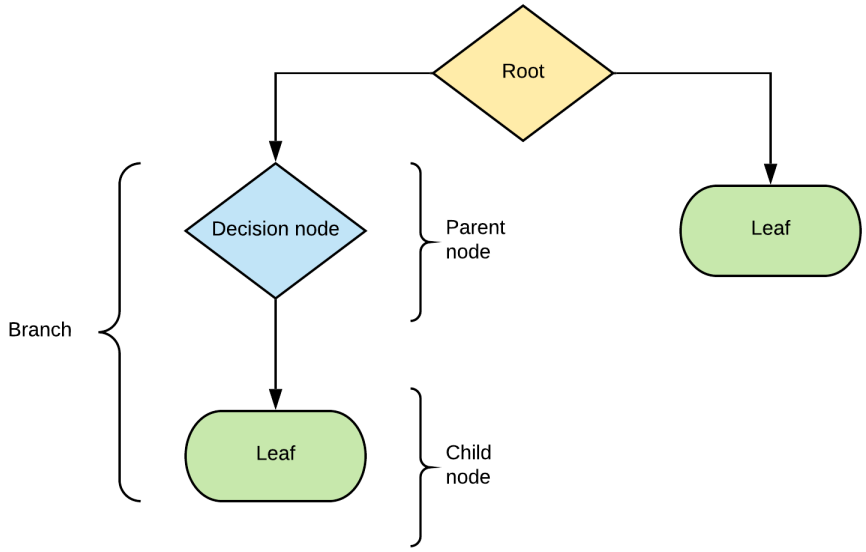


Figure 6: Decision tree structure

Decision Tree Algorithm Pseudocode : The following process that allows us to form a decision tree and its constrains was based on [52].

1. Use the best attributes of the data-set as the root of the tree.
2. Split the training set into subsets. The subsets must be split according to the previously selected attributes.
3. Recursively repeat previous steps.
4. Finally to increase the classification accuracy of the tree is important to identify and remove the irrelevance branches (possible outliers).

It is also important to set constraints on some parameters used to define a tree, and control over-fitting:

- Minimum samples for a node split.
- Minimum samples for a terminal node.
- Maximum depth of tree.
- Maximum features to consider for split.

Note: The implementation was done using the DecisionTreeClassifier class, from the sklearn.tree library. The parameters used were the default library parameters, using Gini as the default function to measure the quality of a split. For more information consult the website of Sklearn[4].

2.4.2 Feed-forward neural network applied in a meta-learning approach

A Feed-forward neural networks (FFNNs) was previously explained as a method to forecasting. However, due to the flexibility of the neural networks, there are several studies in which neural networks of the Feed-forward type are used as an approach to meta-learning. [4, 16, 53]. To use FFNNs as a meta-learning method, we use a network with one hidden layer and 30 hidden nodes, taking into account the results obtained by Lemke and Gabrys [4]. This work uses the characteristics of time series as the input layer, and as an output layer the combination of different methods, in our case the output will be the method already optimized by the genetic algorithm or by the automatic focus on the ARIMA case.

2.5 Metrics

An error in forecasting is the measure of the distance between the forecast value and the observed value. These can be classified into 3 sections: Scale dependant, Percentage and Scaled. We will define y as the observed value, $\hat{y}$ as the forecast value n, as the total number of data points.

2.5.1 Scale dependant

These errors depend on the scale that the values of the time series has. This means that it cannot be compared across time series that use a different scale. The most common are: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE).

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

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2}$$

[4]The library is available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

2.5.2 Percentage

Percentage errors are unit free and are usually used to compare the performance of forecasters. The most common is: Mean Absolute Percentage Error (MAPE); but this metric punishes negative values more than positive values, this impulsed [54] to propose the symmetric Mean Absolute Percentage Error (sMAPE), unfortunately this metric also has a risk that if the forecast value is close to the observed value then it would involve a division by a number close to zero.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}|\frac{y_i - \hat{y}_i}{y_i}| \tag{3}$$

$$sMAPE = \frac{100}{n}\sum_{i=1}^{n}|\frac{y_i - \hat{y}_i}{(|y_i| + |\hat{y}_i|)/2}| \tag{4}$$

2.5.3 Scaled

In [55] it was proposed to use the Mean Absolute Scaled Error (MASE) instead of Percentage errors when comparing various time series with different units. The reason behind this is because it involves values on the scale of the data both in the numerator and denominator. For the seasonal MASE m is the seasonal period.

$$MASE(NonSeasonal) = \frac{y - \hat{y}}{\frac{1}{n}\sum_{i=2}^{n}|y_i - y_{i-1}|} \tag{5}$$

$$MASE(Seasonal) = \frac{y - \hat{y}}{\frac{1}{n-m}\sum_{i=m+1}^{n}|y_i - y_{i-m}|} \tag{6}$$

For the experiments, we use the MSE to calculate the forecasting accuracy. We use MSE to select the model with the best accuracy. Although the model was trained with MSE, the predictions can be evaluated with the different existing metrics.

2.5.4 Validation of the experiments

As it is mentioned in [56] K-fold cross-validation (CV) is one of the most used standard procedures for model evaluation in classification and regression. However, in time series prediction exists a lot

of controversies to apply this validation method. For example, Opsomer et al. mention that standard cross-validation underestimates bandwidths in estimator regression based on a kernel. If autocorrelation of the error is high, the method tends to overfit [57]. The main problem of applying a K-fold cross-validation is mentioned in [58] which mentions that the serial correlation in the data, along with a possible change in the seasonal characteristics of the series make the use of this method problematic; what is more, [58] mentions the problem of being using future data to predict the past. As we have seen, CV has a lot of problems for been applied on time series. However, [56] explains how to apply a CV method, this works only if the method is applied to purely autoregressive forecasting methods. This approach is interesting but it can not be used in our proposal that searches for a meta-learning approach using different forecasting methods that cannot be assured purely autoregressive forecasting methods. Therefore, K-fold cross-validation (CV) is unviable for our propose. Another method that works to evaluate and train models for time series prediction is a Rolling Window Analysis that uses a sliding window approach [59].

It is important to explain that this method will not be used to evaluate the predictions of the different methods for the reasons stated above. This is not the same as using it to evaluate the functioning of the decision tree and neural network in meta-learning.

2.5.5 Sliding Window Training

Sliding Window for training is a technique that consists on selecting a portion of n-size from the full data that will be used as input for the algorithm. The first portion will be initialized on the first element from the data until the element n; the next portion will be constituted from the second element until the element $n + 1$ and so on until reaching the last element from the full training data [60]. This approach will help to cover all elements from the training data. An example of this is shown on Figure 7.

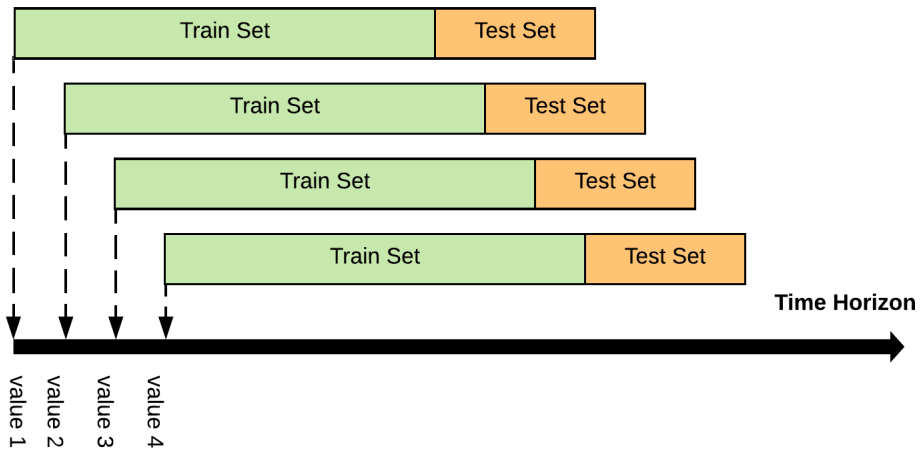


Figure 7: Example of Sliding Window Training

2.6 Summary of Theoretical Framework

In this section we have explained important concepts that will be necessary to understand the method that will be described in section 4.

We started by defining what is a time series and many of the characteristics or features that can be used to analyze them. Next, we described the prediction models that are: FFNN, LSTM and ARIMA; which will be used in following sections. Afterwards, we explained that through certain algorithms such as GA or PSO we can obtain better or enhanced predictors. Continuing, we defined meta-learning and described the models (Decision Trees and FFNN) that will be used. Finally, we explained how we can evaluate the performance of these models; explaining the metrics and the sliding window training that was used.

3 Related Work

For time series forecasting there have been many different models that try to find the best forecast for any time series. In this case, the best model is the one that has the least error. Early, simple algorithms, such as models based on statistical or neural networks were once considered good forecasters. Nevertheless, forecasting models have become more sophisticated as time has passed. Forecasters now may include more than one model in order to obtain a better forecast, known as Hybrid models. These models have been proven to provide better forecasting results [61]. We have focused mainly on these approaches that were useful to develop our proposed method.

From Figure 8 we can see the articles that have been reviewed and classified by using the types of approaches that were used. There are two distinctions for the forecasters, those based on: statistical or computer science methods.

Computer science methods such as [1] show that models, for example: FFNN trained using the Backpropagation learning algorithm (Appendix A) are outperformed by hybrid models, namely, models that use more than one approach in order to improve the forecast accuracy. In Wang et al (2015) [1], the FFNN was combined with a genetic algorithm that was used to find the optimal initial conditions of the network, thus providing an improved forecast in comparison to other models. Furthermore, statistical methods also show similar results [34]; in this case, an ARIMA model was chosen to be used in conjuntion with an ANN, both models analyzed different parts of the data and the results were later combined. Which resulted in a model that was more precise than only the ARIMA or ANN models.

Makridakis (2018) [61] presents a study from the M4 forecasting competition that provided us with an overview of the forecasting models that performed well. This study concluded that the models that had the best performance were those that were a combination of statistical and machine learning models. These Hybrid models would outperform others by a margin of about 10% .

From the reasons that were previously mentioned we have decided to focus on Hybrid models in order to obtain the best possible results. Specifically, we have decided to use a genetic algorithm (GA) with machine learning and statistical models. The reason is that the GA would be able to find the best starting point for the ML and statistical approaches, thus improving their performance. The problem with this model is that the GA requires a considerable amount of resources in order to operate. To

solve this we propose that, first we obtain the characteristics of the time series that will be used for the proposed method and the information of what improved forecaster is better for each time series. Then by training a meta-learning model we can use the data that was obtained by running the GA a single time; thus allowing future decisions to be made instantly, another benefit is that we can clearly see how the forecaster was chosen.

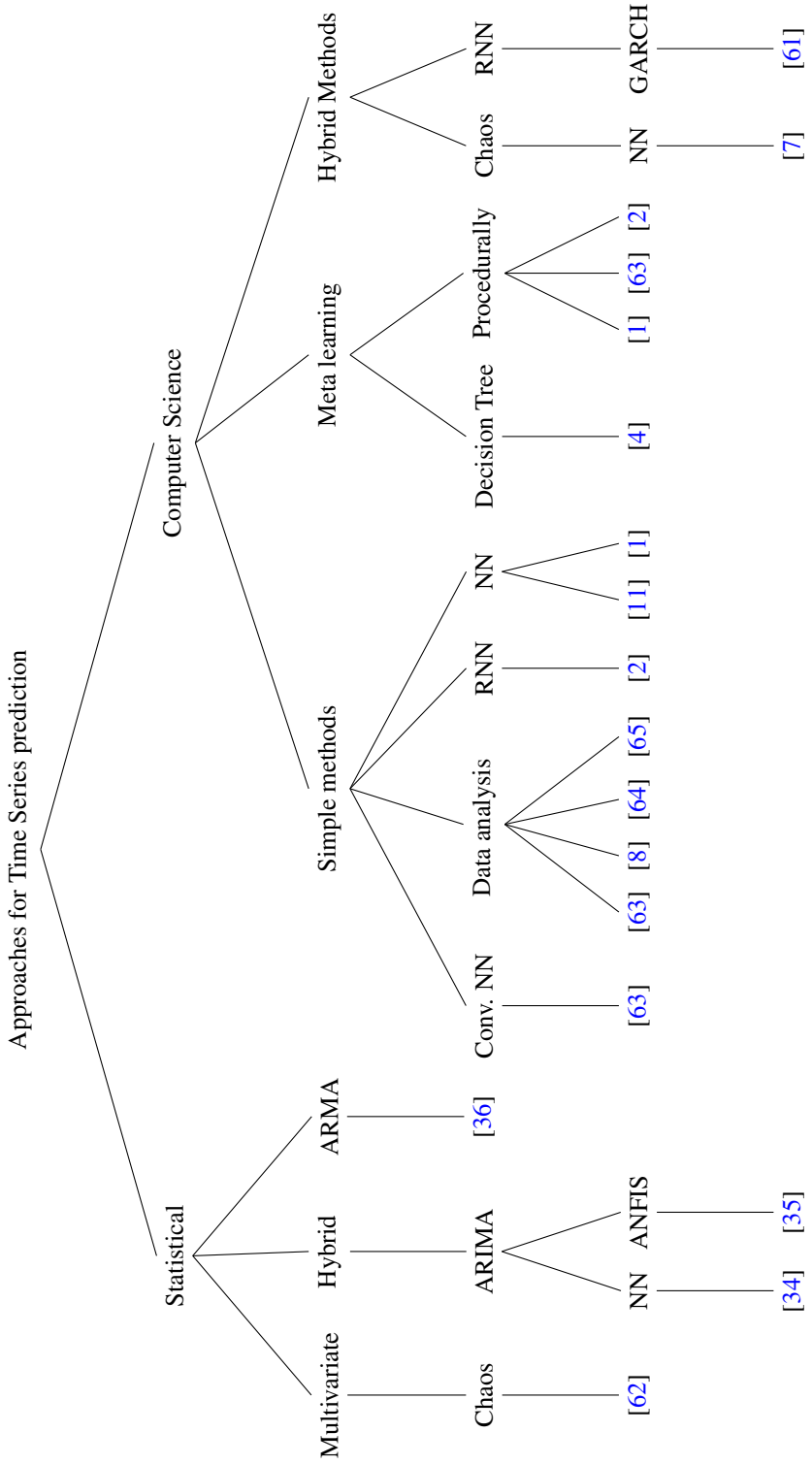


Figure 8: Related Works

4 Study and Proposed Method

4.1 Overview

This section presents the proposed method in order to achieve the objectives of this work. First, we represent the entire process using block diagrams in Figure 9; which consist on the selection and tuning of the models that will be used for the forecasting process; then a brief description of each step is given.

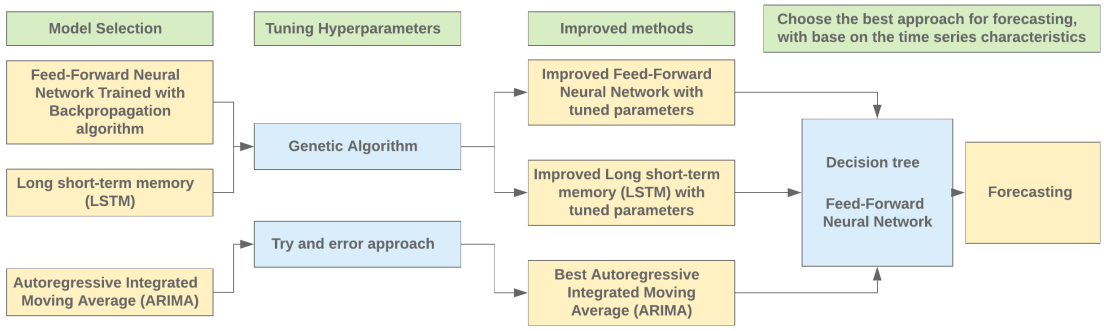


Figure 9: Proposed Method

I **Model Selection (Basic forecasters) :** In this step, we select different forecasting methods. The main idea is to select methods in which their mathematical base is different. For this purpose. we selected Feed-Forward Neural Network, Long-Short Term Memory, and ARIMA. These models were chosen due to their high utilization and performance in the literature.[1][4][7][2]

II **Tuning Hyper-parameters:** In the second step, we focused on choosing an optimal structure for the neural network methods, using a Genetic algorithm. To find the best parameters for the ARIMA model we use a search in a predetermined range. This step is to avoid wrong predictions or just predictions with random precision. This step was based on literature about the improvement of NNs.[43][1][42][14]

III **Meta-Learning:** In the final step, we obtain different features of the time series and select the most representative ones with a multi-variable analysis. These features constitute the input of a DT and an FFNN, which are trained to choose which method is the best depending on the

characteristics of the series. Finally, the decision tree allow us to analyze which methods or which approaches are the best for each type of feature, with a reverse engineering step [52, 51, 4].

In the next section, each step of the proposed method is presented more descriptively. Also, the details of its implementation are presented.

4.2 Studying basic forecasters

This section details the experiment to which the basic predictors were subjected. It is important to emphasize that all predictions are made to predict a single value of the future, using n values of the step or 1-step ahead as mentioned in the theoretical framework. We use the reduced data set from the NN3 competition.

4.2.1 Feed-Forward Neural Network

The FFNN that was implemented as the basic FFNN uses the standard Back Propagation training algorithm presented in Appendix A. The structure and parameters that were used for this experiment are presented in Table 3 that were obtained from previous studies.

Table 3: Parameters of the basic FFNN following the values suggested in [4]

Parameter	**Values of [4]**
Number of input neurons	12
Number of hidden layers	1
Number of neurons in hidden layers	30
Optimizer	Adam
Activation Function	Sigmoid

The structure of this neural network is the same as was presented before. That is, an input layer, hidden layers and the output as shown in Figure 2. In this case, we will consider the input layer as the consecutive months to the desired target prediction, this interval is known as the window.

4.2.2 Long Short-Term memory

The experiment used the standard LSTM cell that was explained previously in the theoretical framework. For the development of the network with LSTM cells, we use the structure that can be observed in Table 4 that were taken from the literature.

Table 4: Parameters of basic LSTM following the values suggested in [5]

Parameter	**Values of [5]**
Number of LSTM cell per layers	6
Number of layers	2
Optimizer for fully connected layer	Adam
Activation Function	Sigmoid

The structure of this network begins with a layer of LSTM cells that are always connected to the next LSTM cell and the upper layer. While LSTM cells share their ”memory” and their output with the following LSTM cells, with the cells in the upper layer LSTM cells, simply share their output. The structure ends with a fully connected layer that takes the output values of the last layer of LSTM cells as its input layer and uses the number of values to be predicted as its output layer. This structure can be seen in Figure 10.

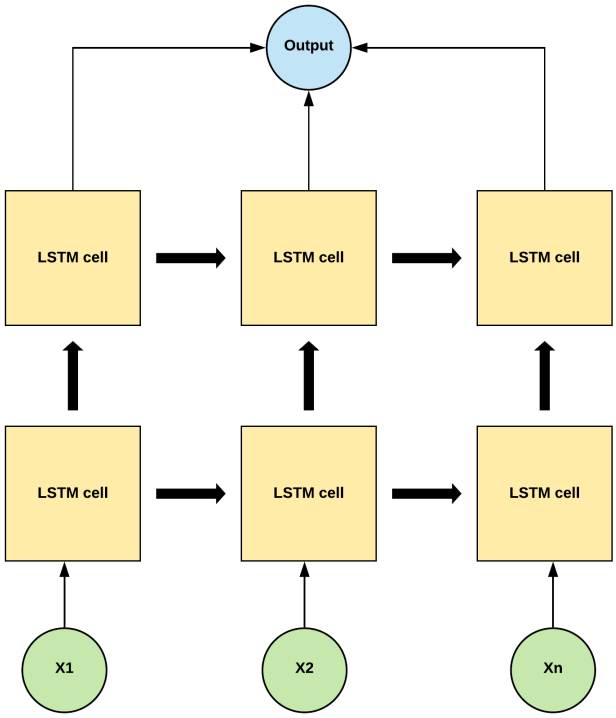


Figure 10: LSTM network structure

4.2.3 Autoregressive Integrated Moving Average

The standard ARIMA as mentioned before was implemented for the experiment. The parameters that were tested are shown in Table 5

Table 5: Parameters of basic ARIMA

Parameter	**Try and error**
p	0
q	1
d	0

This was a simple test since ARIMA is a statistical method that only requires the p, q and d values; it does not require a certain structure as the previous methods.

4.3 Tuning Hyper-parameters of basic forecasters

We continue using the aforementioned NN3 data set, and 1-step ahead prediction. This time we can not use a static number of values for training, validation, and testing set because the number of input neurons changes according to the genetic algorithm, then the number of shifts in the sliding window change. So we use "early_stopper [5]" Keras function, which stops the training when the learning with the testing set has stopped improving. This to avoid over-fitting.

4.3.1 Genetic Algorithm

To reach a correct hyper-parameters selection and taking into account the large search space that hyper-parameters of neural network methods can take, the use of a genetic algorithm that improves the search process takes place. To implement the genetic algorithm explained in the theoretical framework, the pool of parameters in Tables 6 and 7 were defined.

Table 6: Parameter pool for FFNN with GA

Parameter	**Possible Value**
Hidden layer neurons (1)	4,6,8,12,24
Number of hidden layers (2)	1,2,3
Optimizer (3)	Adam, Stochastic gradient descent
Activation Function (4)	Relu, Elu, Tanh, Sigmoid
Input neurons (5)	3,4,6,8,12

Table 7: Parameter pool for LSTM with GA

Parameter	**Possible Value**
Number of cells for layer	10,20,30,40,50
Number of hidden layers	1,2,3
Optimizer	Adam, Stochastic gradient descent
Activation Function	Relu, Elu, Tanh, Sigmoid
Input neurons	3,4,6,8,12

[5]More information can be found at: https://keras.io/callbacks/

The possible values for input neurons were defined taking advantage of the fact that the NN3 data-set (as explained in Appendix B) is taken monthly. This means that we can make the learning process using the data quarterly, every four months, semiannually, every eight months or annually. Due to the small number of samples in the NN3 data-set and the computational cost, it was set as a limit 12 months ago. In addition to this, when tests were conducted using 15 and 18 as the window size, the genetic algorithm never converged to these values.

The number of neurons in the hidden layer for simplicity is the same for each Layer. For example, if the genetic algorithm choose 12 as the number of neurons in the hidden layer, and 6 as the number of neurons in the input layer with 3 hidden layers, the structure of the neural network will be 6 12 12 12 and the output layer. The possible values for the number of neurons in the hidden layer are multiples of the input neurons.

Activation Function and Optimizer were chosen from the literature, due to the different advantages of each one.

For example using FFNN, when the genetic algorithm is initialized a population is made using the values from table 6. Each individual is assigned values from the parameters of the table, this is considered as the chromosome of an individual; It may look like:

Table 8: Example Chromosome for FFNN

Parameter (Gene)	**Value**
Hidden layer neurons	4
Number of hidden layers	1
Optimizer	Adam
Activation Function	Relu
Input neurons	3

Next, the population goes through crossing over to obtain a new individual; this was done by using the best individuals from each generation. Crossing over was done by choosing half the genes of one individual and half from the other. Furthermore, if a mutation was triggered then a random gene (parameter) of the chromosome is then randomly selected again from the parameter pool. This is the process each generation goes through.

4.3.2 Automatic Approach

The automatic approach is used in the ARIMA model since we can test all the possible combinations and with the AIC (Akaike Information Criterion) we can see what parameters are the ”best” for the time series. For this, we used the python library mentioned before, with a range of parameters that can be seen in Table 9.

Table 9: Parameter intervals for the automatic approach

Parameter	**Interval**
p	[0,10]
q	[0,10]
d	[0,10]

4.4 Meta-Learning

As was mentioned before using an automatic approach or a genetic algorithm allows a correct choice of parameters. However, genetic algorithms and automatic approaches depending on the search space have a great computational cost. In order to reduce this computational cost and the training time, a meta-learning approach is proposed.

The goal of using meta-learning at this stage is that instead of training all the different individual methods with their corresponding optimizer (GA with LSTM, GA with FFNN, auto-Arima), it is directly selected which is the best method based on the characteristics of the time series. In this way, we reduce the time needed for a good prediction.

4.4.1 Characteristics extraction

The characteristics of Table 10 were extracted individually for each time series using the library tsfresh[6] that implements the formulas previously mentioned in the theoretical framework, and implementations based and Hyndman [20] and Lemke [4] articles. The procedure for the extraction of characteristics, as well as the necessary transformations (detrend), were followed strictly according to what was mentioned in the theoretical framework.

[6]The library is available at: https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html

Table 10: Time series characteristics (before selection)

Characteristic	Implementation
Non-linearity	Hyndmann implementation
Skewness	Tsfresh
Autocorrelation	Tsfresh
Partial autocorrelation	Tsfresh
Trend	Hyndmann implementation
Kurtosis	Tsfresh
Self-similarity	Hyndmann implementation
Periodicity	Hyndmann implementation
Chaos	Hyndmann implementation

4.4.2 Characteristics selection

The characteristics selection was done by applying the statistical methods mentioned in section E. First, PCA was done in order to replace the characteristics with a smaller amount of Principal Components, these should represent at least the 80% of the data. Second, Biplot was used in order to identify the characteristics that could be used for the decision tree; these were chosen as was described in section E. The result of each method would then be used for the Decision Tree and the better Tree would then be chosen.

4.4.3 Decision Tree

Once the results of the enhanced predictors for each of the 11 time series of NN3 (reduced version) have been obtained and the characteristics of these series have been extracted. A table with the following columns have been created: Non-linearity, Skewness, Autocorrelation, Partial autocorrelation, Trend, Kurtosis, Self-similarity, Periodicity, Chaos, Mse of LSTM with GA, Mse of FFNN with GA, Mse of auto-ARIMA. A more simple example of this table reduced to only two characteristics can be seen in Table 11).

Table 11: Example Table of features and MSE (with two features and the performance of three models with three time series)

Series	Non-linearity	Skewness	Mse of LSTM	Mse of FFNN	Mse of auto-ARIMA
Example 1	0	1	0.09	0.08	0.06
Example 2	1	0	0.06	0.06	0.05
Example 3	1	1	0.05	0.06	0.09

This example will help us because the idea is to distinguish 3 possible classes (LSTM with GA, FFNN with GA, auto-ARIMA). Each individual (each time series) belong to the class to which its MSE is smaller, and the features of the individual are the characteristics extracted. For instance, continuing with the example table we saw before, a table with the classes to which each individual belongs, as well as its characteristics would be the following Table 12.

Table 12: Example table for training a decision tree

Series	Features of series		Class (Best method for forecasting)
	Non-linearity	Skewness	
Example 1	0	1	ARIMA
Example 2	1	0	ARIMA
Example 3	1	1	LSTM

Finally, with this information obtained and making an adjustment of hyper-parameters with a trial and error method and with cross-validation as was explained in the theoretical framework, the decision tree is obtained. The tree allow us to determine which method is better depending on the characteristics of the series. In addition to this, an advantage of the decision tree is that it allows us to analyze which characteristics are the most relevant to make the decision, analyze in-depth this result is interesting as is detailed in the results section.

Note: The data in the example tables are not real, and are only intended to better explain the method.

4.4.4 Feed-Forward Neural Network

The FFNN similarly as the Decision Tree will use the results of the enhanced predictors and the results of the characteristics extraction of the previous section. As seen in table 11.

From table 11 we used a FFNN that has the structure as seen in Figure 11 that was based on the structure proposed by Lemke [4]. The input layer receives the characteristics of each time series that is then passed to the hidden layer which has 30 neurons. Finally the output layer has 3 neurons, which, depending on what method is better will be activated.

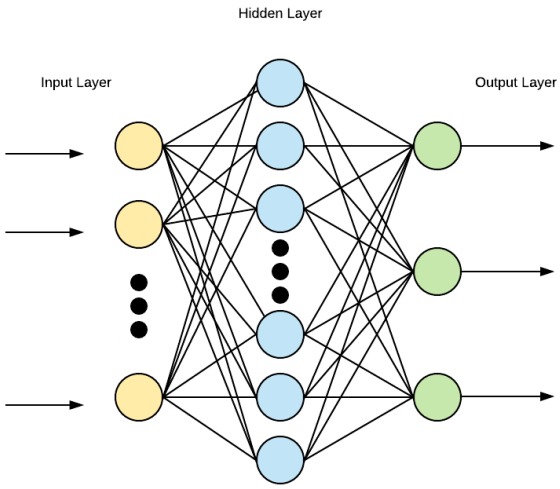


Figure 11: FFNN Meta Learning structure

5 Results and Discussion

5.1 Testing the implementations

First of all, we test all the methods with a simple $sine$ series to discard that the proposed methods are defective due to a problem during the implementation. With the following results.

Table 13: $sine$ series forecasting results.

Methods	ARIMA	FFNN	LSTM
MSE	0.00000001	0.003982	0.00475
CPU time	3 seg	2 seg	4 seg

For this test we used the $sine$ function evaluated at values from 1 to 144. In this case, the function does not have noise components. From this generated data 120 values was used to train the forecasters, 10 for validation and 14 for testing. The ARIMA forecaster, a pure statistical method, performs better than neural networks, this is because the training process of NNs have been limited to 200 epochs.

5.2 Forecasting with basic forecasters

In this section, we use the NN3 reduced data set, which corresponds to 11 representative time series, as was detailed in the methodology. We use 1-step ahead approach to make the predictions, testing different network structures that can be observed in Tables 6, 7 and using the ARIMA parameters of Table 5. The data was used as follows: 10 values for validation, 14 for testing and the rest is used to train the forecasters. We repeat the experiment 10 times, averaging the results of CPU time and the mean square error. Furthermore, the data used was first normalized to now use an interval of 0.1 to 0.9; this was done because the activation functions reach their maximum and minimum values asymptopically [66].

Table 14: Results using Basic Forecasters

	FFNN		LSTM		ARIMA	
Series	**MSE**	**CPU time**	**MSE**	**CPU time**	**MSE**	**CPU time**
Serie 1	0.0172	6.3293 seg	0.02	40.780 seg	0.0185	0.006 seg
Serie 2	0.0176	10.7288 seg	0.0327	119.160 seg	0.0314	0.006 seg
Serie 3	0.0081	17.2733 seg	0.0303	312.475 seg	0.0518	0.007 seg
Serie 4	0.0069	14.4994 seg	0.0228	226.585 seg	0.02569	0.007 seg
Serie 5	0.0123	6.6668 seg	0.0139	41.368 seg	0.0044	0.008 seg
Serie 6	0.0083	7.4566 seg	0.0087	38.6 seg	0.0147	0.005 seg
Serie 7	0.0312	6.3831 seg	0.0356	43.575 seg	0.0176	0.008 seg
Serie 8	0.0192	6.6092 seg	0.0176	38.322 seg	0.0197	0.006 seg
Serie 9	0.0096	7.6636 seg	0.0117	41.770 seg	0.0020	0.007 seg
Serie 10	0.0021	6.9016 seg	0.0018	43.0062seg	0.0043	0.007 seg
Serie 11	0.0321	6.7045 seg	0.0327	39.2413 seg	0.0355	0.010 seg
Average :	0.0150 $\pm$0.009	8.84 seg $\pm 3.7 seg$	0.0207 $\pm$0.0111	89.5 seg $\pm 94 seg$	0.0205 $\pm$0.0150	0.007 seg $\pm 0.001 seg$

As we can see in the results of Table 14, the network structure used by the literature in general shows a good performance. However, we can observe how there are series in which the performance is worse, taking into account that the experiment was replicated 10 times, the random initialization of the synaptic weights loses a lot of force when we want to explain this result and the hypothesis of that the structure of the network interfere in each case (each of the different series) take more sense.

Specifically, if we take into account when using neural networks of the Feed-Forward type, we have an average mean square error of 0.0150, and we have examples like Series 7 and Series 11 in which there is an MSE of 0.0312 and 0.0321 respectively, which indicates a much worse performance for these series, another example occurs in LSTM networks, in which Series 2, Series 7 and Series 11 have MSE above the average. One point that is important to consider is that the complexity of the series also affects the MSE of the prediction, for example in Series 7 and series 11 the performance is below average for all methods. Nevertheless, it is not the only reason since there are examples like the Series 4 that show that FFNN type networks have better results, so it is also important to note how there are methods that are better for a specific series on the NN3 reduced Data set, or at least they seem to be better using the structure of literature.

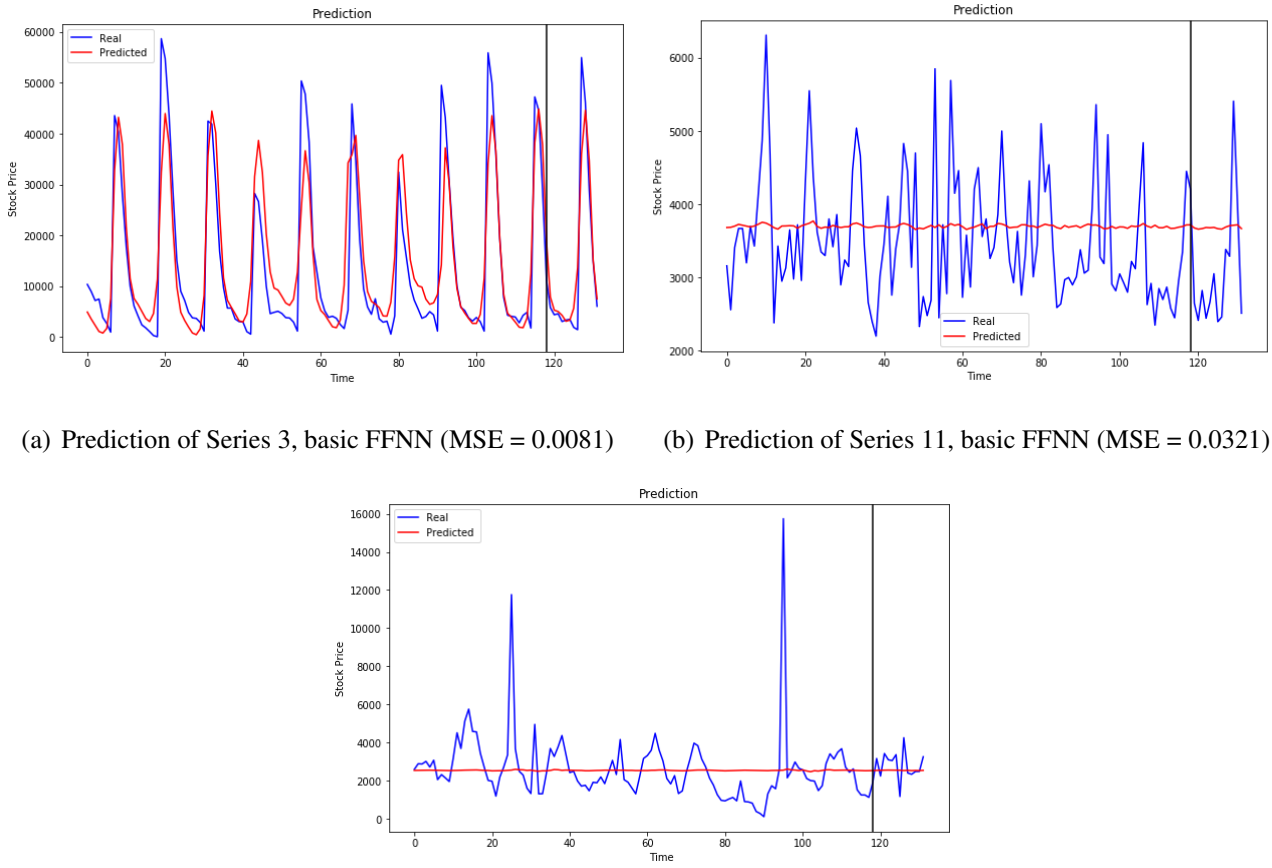
In addition, it is interesting to show how the method that works best on average is FFNN. This could be since more complex methods such as LSTM are more sensitive to the choice of initial

parameters, an interesting topic of study, but that escapes the objectives of this study. However, FFNN is not always better than the rest of the methods, for example in Series 11, 8 and 6 this could be since their parameters are not sufficiently adjusted, or because LSTM and ARIMA work better for these series. It is for this reason that it is necessary to test the methods with optimized parameters for each series specifically, which is explained in the next section with the use of Genetic algorithms.

Finally, Regarding the execution times, it can be observed that in general, they are low, there are series with higher execution times due to the complexity of these. ARIMA is the method with the shortest execution time which makes sense, due to the simplicity of the calculations that must be done as explained in the theoretical framework.

5.2.1 Example predictions of basic forecasters

Results using the basic FFNN



(a) Prediction of Series 3, basic FFNN (MSE = 0.0081)

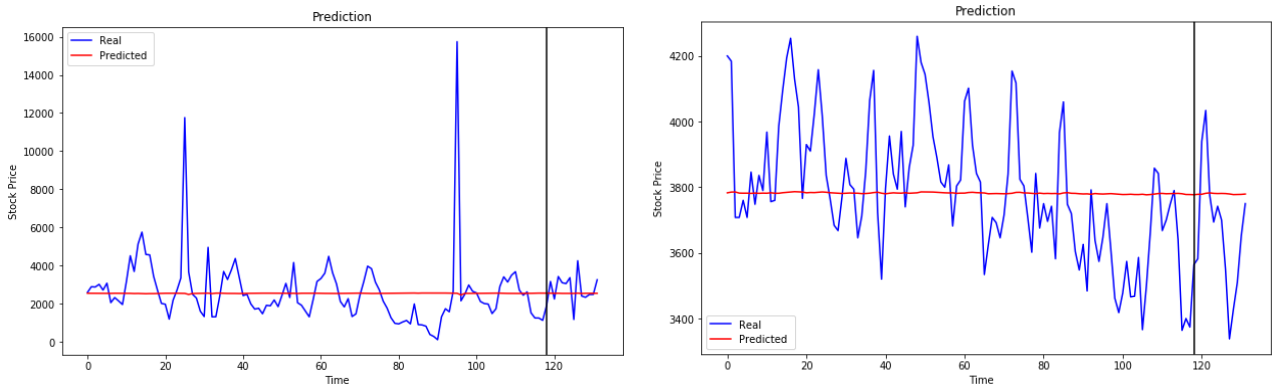
(b) Prediction of Series 11, basic FFNN (MSE = 0.0321)

(c) Prediction of Series 10, basic FFNN (MSE = 0.0021)

Figure 12: Predictions using basic FFNN

In Figure 12 using the basic FFNN, we have on one hand in 12(a) a prediction with a low MSE and in 12(b) the worst prediction (the highest MSE). As we can see the prediction in 12(a) is quite attached to the real one, the blue line represents the real value of the time series and the red line the prediction that is made with the 1-step ahead method and using the sliding window method (mentioned in the theoretical framework). The black vertical line separates the training zone, that is, the data with which the algorithm was trained from the test data, this data has never been used and represents a completely new prediction.

On the other hand, although the prediction error seems to be low for series 10, once we get its graph as we can see in Figure 12(c), the prediction is bad since the prediction tends to the average. This result is unacceptable in some domains when we make predictions and make decisions based on this result, this is why it is necessary to avoid this type of event. More about this event of finding the average as the prediction is detailed in the Appendix F.

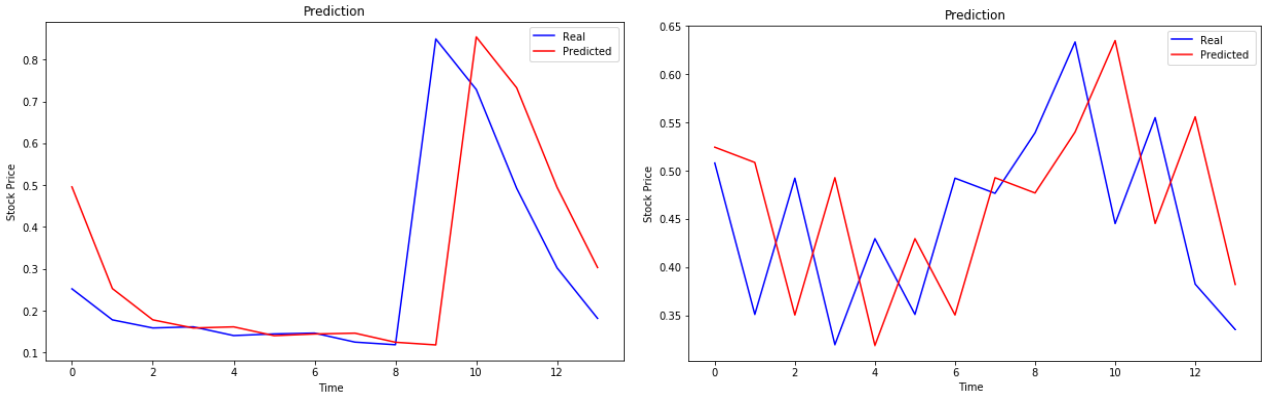


(a) Prediction of Series 10, using basic LSTM

(b) Prediction of Series 7, using basic LSTM

Figure 13: Predictions using basic LSTM

Another example can be seen in Figure 13 since the same occurs with the best prediction of LSTM and with the worst prediction, more of this event can also be observed in the Appendix F.



(a) Prediction of Series 10, using basic ARIMA

(b) Prediction of Series 7, using basic ARIMA

Figure 14: Predictions using basic ARIMA

ARIMA does not show results in which converges to its mean. This problem was typical in our experiments with neural networks. However, due to the sensitivity of its initial parameters, and that we are the same parameters for all predictions, we have delayed predictions as can be seen in the Figure 14.

5.3 Optimized Forecasters

In this section, the results of the optimized predictors are presented, using the methodology previously explained. The genetic algorithm was run only once, however using 5 generations with 10 individuals each, the parameters tend to stabilize. The chromosome structure and the crossing over process are described in section 4.3.1. Also, the pool of values that were used to initialize the population can be seen in tables 6, 7.

5.3.1 Results using FFNN with GA

Table 15: Comparison of results using FFNN and GA

	basic FFNN		FFNN GA		Hyper-parameters obtained				
Series	MSE	CPU time	MSE	CPU time	nb_neurons	i_neurons	nb_layers	Activation	Optimizer
Series 1	0.0172	6.3293 seg	0.0045	196 seg	24	12	1	Hyperbolic Tangent	Adam
Series 2	0.0176	10.7288 seg	0.0041	380 seg	24	12	2	Rectified Linear Unit	Adam
Series 3	0.0081	17.2733 seg	0.0071	1,184 seg	8	12	1	Hyperbolic Tangent	Adam
Series 4	0.0069	14.4994 seg	0.0054	1,103 seg	12	12	3	Exponential Linear Unit	Adam
Series 5	0.0123	6.6668 seg	0.0039	1,404 seg	12	4	3	Hyperbolic Tangent	SGD
Series 6	0.0083	7.4566 seg	0.0064	2,172 seg	8	6	1	Exponential Linear Unit	Adam
Series 7	0.0312	6.3831 seg	0.0122	2,464 seg	24	4	2	Exponential Linear Unit	Adam
Series 8	0.0192	6.6092 seg	0.0112	2,874 seg	4	12	2	Hyperbolic Tangent	SGD
Series 9	0.0096	7.6636 seg	0.0021	3,638 seg	12	6	1	Hyperbolic Tangent	Adam
Series 10	0.0021	6.9016 seg	0.0014	4,602 seg	6	12	2	Rectified Linear Unit	SGD
Series 11	0.0321	6.7045 seg	0.0108	5,590 seg	6	12	1	Exponential Linear Unit	Adam
Average	0.015	8.84 seg	0.006	2,327.91 seg					
Std Dev	$\pm$0.009	$\pm$3.7 seg	$\pm$0.004	$\pm$1728.35 seg					

As can be seen in table 15, there is a considerable improvement when using genetic algorithms with neural networks of the Feed-forward type. The CPU time increases dramatically when we implement the genetic algorithm, this is completely acceptable since we significantly improve the prediction, reducing the mean square error. To show the significance in the improvement we will use a hypothesis test, to do this we must first test the normality (Using the Wilk Shapiro test) of the MSE of the predictors. The results of this test may be seen in table 16.

Table 16: Table with the results of the Normality test

	LSTM	**LSTM with GA**	**FFNN**	**FFNN with GA**	**ARIMA**	**Auto ARIMA**
p-value	0.208605	0.861689	0.208605	0.324777	0.517246	0.358076
Significance level (α)	0.05	0.05	0.05	0.05	0.05	0.05

From table 16 we can see that the p-value that we obtain from each test is greater than the α so then the null hypothesis is accepted and the data is normally distributed.

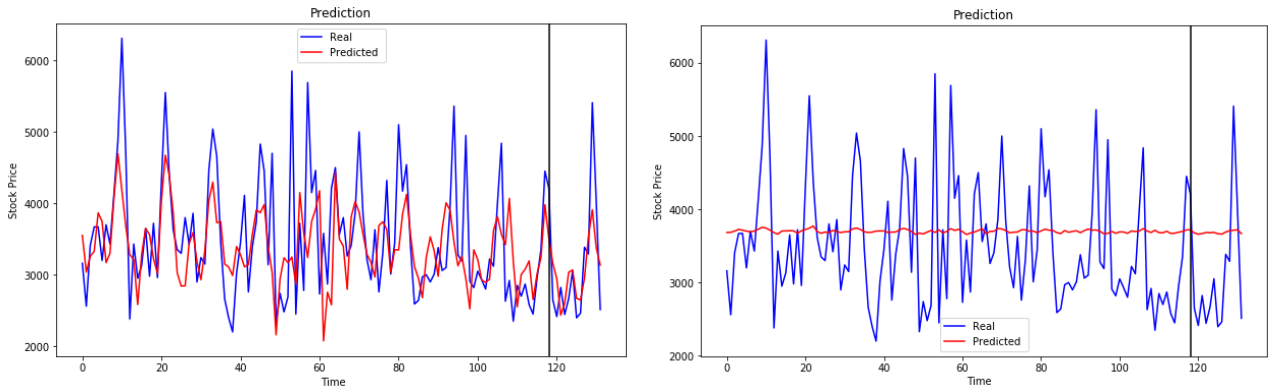
The next step is to use an ANOVA test to find if the MSE obtained from the enhanced predictors represent an improvement to the simple predictors.

Table 17: Results of the hypothesis test on the MSE of the predictors used

Hypothesis test						
	LSTM vs LSTM GA		**FFNN vs FFNN GA**		**ARIMA vs Auto ARIMA**	
p-value	0.053777		0.011831		0.071665	
Significance level (α)	0.1		0.1		0.1	
	LSTM	**LSTM with GA**	**FFNN**	**FFNN GA**	**ARIMA**	**Auto ARIMA**
Standard Deviation	0.0097	0.0044	0.015	0.0037	0.015	0.0077
Mean	0.015	0.0084	0.0097	0.0063	0.0205	0.0108

From table 17, we can see that the p-values for all the comparisons are less than the significance level of 0.1, then we reject the hypothesis. For this reason we can conclude that the means are different. The only issue is that the standard deviations should also have the same values, however since we have a variety of time series this is very difficult. A solution may be to remove certain time series that could be considered outliers, although that would leave us with even less time series to test. This solution could be applied if we had a greater amount of time series to test.

Figure 15: Predictions using FFNN in Series 11

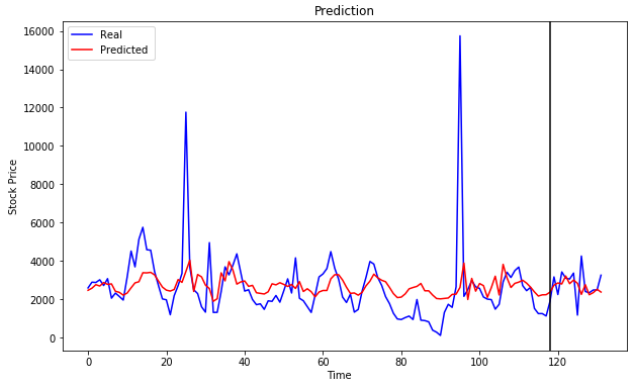


(a) Prediction of Series 11, using FFNN with GA

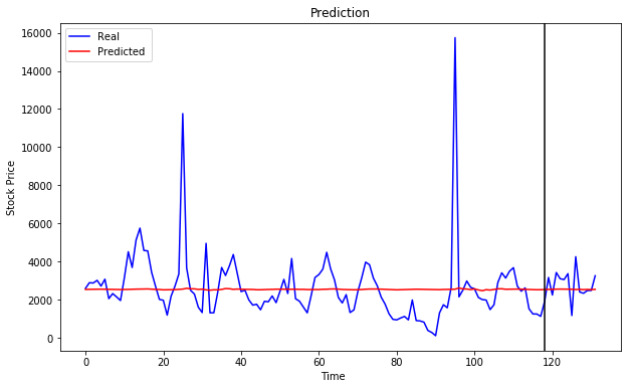
(b) Prediction of Series 11, using FFNN

As can be seen in the Figure 15, the use of genetic algorithms dramatically improved the performance of FFNN. For example, the Figure 16(b) shows how the series 11 that was the worst prediction,

not only decrease its MSE, but also the graph shows a huge improvement in 16(a).



(c) Prediction using FFNN with GA



(d) Prediction using FFNN

Figure 16: Predictions using FFNN in Series 10

Another example of the improvement granted by the use of genetic algorithms can be seen in the Figure 16. Where Series 10 that was previously discussed since its low MSE could confuse us thinking that the prediction is correct when in fact it tends to the mean (Figure 16(c)), now it has a much better prediction as can be seen in the Figure 16(d).

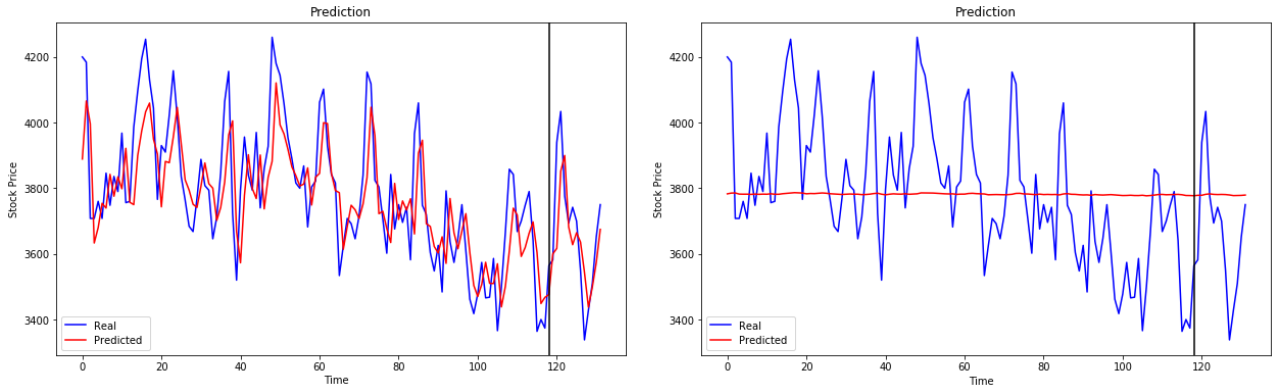
5.3.2 LSTM with GA

Table 18: Comparison of results using LSTM with GA

	basic LSTM		LSTM GA		Hyper-parameters obtained				
Series	**MSE**	**CPU time**	**MSE**	**CPU time**	**nb_neurons**	**i_neurons**	**nb_layers**	**Activation**	**Optimizer**
Series 1	0.02	40.780 seg	0.0076	2,737 seg	20	12	3	Exponential Linear Unit	Adam
Series 2	0.0327	119.160 seg	0.0159	3,637 seg	30	4	2	Hyperbolic Tangent	Adam
Series 3	0.0303	312.475 seg	0.0073	8,256 seg	20	12	3	Rectified Linear Unit	Adam
Series 4	0.0228	226.585 seg	0.012	9,235 seg	10	8	2	Rectified Linear Unit	Adam
Series 5	0.0139	41.368 seg	0.0049	7,349 seg	30	8	1	Hyperbolic Tangent	Adam
Series 6	0.0087	38.6 seg	0.0083	8,080 seg	40	12	1	Rectified Linear Unit	SGD
Series 7	0.0356	43.575 seg	0.0115	14,386 seg	10	12	1	Exponential Linear Unit	Adam
Series 8	0.0176	38.322 seg	0.0103	11,424 seg	50	6	1	Rectified Linear Unit	Adam
Series 9	0.0117	41.770 seg	0.002	17,206 seg	40	4	2	Hyperbolic Tangent	Adam
Series 10	0.0018	43.0062 seg	0.0014	20,384 seg	30	6	1	Hyperbolic Tangent	SGD
Series 11	0.0327	39.2413 seg	0.0251	25,323 seg	30	4	2	Sigmoid	Adam
Average	0.0207	89.5 seg	0.0207	11,637.9 seg					
Std Dev	$\pm$0.0111	$\pm$94 seg	$\pm$0.006	$\pm$7027.68 seg					

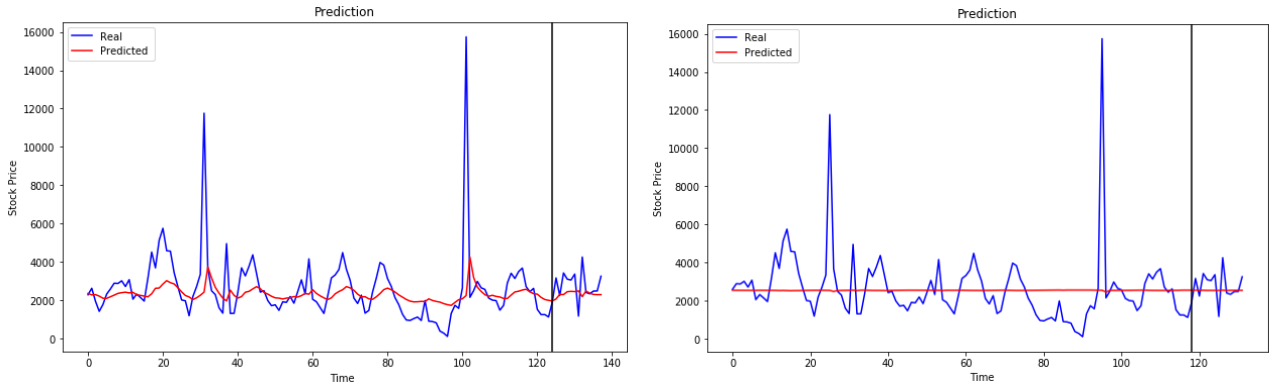
We can see that in table 18 that there is a significant improvement to the performance when a GA is used with the LSTM. Nevertheless, the improvement came with a drastic increment of the CPU time which can be acceptable because of the improved prediction. The difference is significant because of the tests shown previously.

Figure 17: Predictions using LSTM in Series 7



(a) Prediction using LSTM with GA (b) Prediction using LSTM

Figure 18: Predictions using LSTM in Series 10



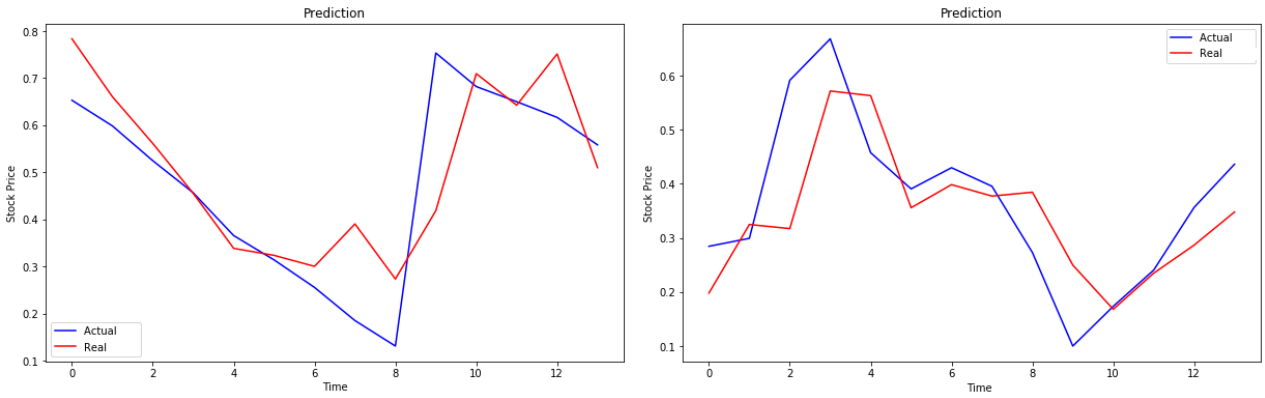
(a) Prediction using LSTM with GA (b) Prediction using LSTM

The result of using GAs is even more significant when applied to a more complex method such as LSTM networks. Since, as we can see in the Figure 18 corresponding to the series 10, that is, the best prediction of the basic LSTM and in the Figure 17 corresponding to the series 7 the worst prediction of the basic LSTM, the improvement is not only substantial in terms of the MSE as explained before, but also the resulting graph clearly shows a better prediction. This explains the importance of the correct choice of hyperparameters, and that more complex methods such as an LSTM network are very sensitive to this choice, since without the use of GA the best and worst prediction simply tended to the average of the series.

5.3.3 ARIMA with automatic approach

Table 19: Comparison of results using auto ARIMA

Auto ARIMA							
	ARIMA basic		Auto ARIMA		Hyper-parameters		
Series	**MSE**	**CPU time**	**MSE**	**CPU time**	**p**	**q**	**d**
Series 1	0.0185	0.006 seg	0.0051	31.167 seg	4	1	6
Series 2	0.0315	0.006 seg	0.0158	29.588 seg	2	0	8
Series 3	0.0518	0.007 seg	0.0258	28.280 seg	4	0	6
Series 4	0.0257	0.007 seg	0.0141	29.832 seg	10	0	0
Series 5	0.0044	0.008 seg	0.004	30.123 seg	4	1	0
Series 6	0.0148	0.005 seg	0.0063	26.860 seg	9	0	0
Series 7	0.0176	0.008 seg	0.0109	30.398 seg	2	1	3
Series 8	0.0198	0.006 seg	0.0157	27.152 seg	0	0	5
Series 9	0.0020	0.007 seg	0.0019	29.956 seg	2	1	2
Series 10	0.0043	0.007seg	0.0019	27.240 seg	1	0	0
Series 11	0.0355	0.010 seg	0.0176	30.906 seg	9	1	1
Average	0.0205	0.007 seg	0.0108	29.2274 seg			
Std Dev	±0.0150	±0.001 seg	±0.007	±1.56 seg			



(c) Prediction of Series 2, using auto-ARIMA

(d) Prediction of Series 10, using ARIMA

Figure 19: Predictions using ARIMA

In the case of the ARIMA, being an autoregressive model and being predicting only 1-step ahead, it can be observed in Figure 19 that the predicted values are always close to the real value and there

are no predictions in which it begins to converge to the average. The problem of delay seems to be solved as shown in Figure 19. This is possible thanks to a correct selection of their coefficients using an automatic approach because as we know this model is very sensitive to the precision with which its coefficients are determined. The graphs in the Figure 19 show the prediction made in the testing set.

5.4 Meta-Learning approach

5.4.1 Characteristics Extraction

Now we extracted the characteristics that were mentioned in section 4.4.1 the results can be seen in appendix G. From these values we proceeded to use the PCA and biplot statistical methods to help us to decide what characteristics could better represent the time series.

We started by doing the PCA method, where, we found that if we used 3 Principal Components (PC) then we could represent 81% of the data this can be seen in Table 20, where the cumulative proportion column shows the percentage of representation that is obtained when taking the set of principal components, the proportion column shows the representation that each component has individually. Furthermore, we can see in Table 21 the relation each PC has with each characteristic. This method was done in the Infostat[7] software for statistical analysis.

[7]Software available at: https://www.infostat.com.ar/

Table 20: Percentage of representation, in the characteristics extracted from the series.

Lambda	**Proportion**	**Cumulative Proportion**
1	0.42	0.42
2	0.27	0.69
3	0.12	0.81
4	0.07	0.88
5	0.06	0.94
6	0.03	0.97
7	0.02	0.99
8	0.01	1.00
9	3.60E-03	1.00
10	1.30E-03	1.00
11	3.60E-03	1.00
12	0.00	1.00
13	0.00	1.00
14	0.00	1.00
15	0.00	1.00

Then we also made a biplot from the data and obtained three figures (this depends with the Principal Components that were used). The figure that was the best fit for our need was the following:

Table 21: Correlation of the characteristics using three Principal Components

Variables	**PC 1**	**PC 2**	**PC 3**
Trend	-0.02	0.92	0.12
sweekness	-0.082	-0.16	0.47
autocorrelation_1	0.62	0.72	0.26
autocorrelation_2	-0.16	0.93	-0.16
autocorrelation_12	0.87	-0.08	0.2
partial_autocorrelation_1	0.62	0.72	0.26
partial_autocorrelation_2	0.7	0.38	-0.38
partial_autocorrelation_12	0.64	-0.57	0.28
kurtosis	0.77	-0.17	0.42
energy	0.8	0.09	-0.3
variance	0.8	-0.31	0.34
Non-linearity	-0.16	-0.31	0.58
Self-similarity	0.78	0.47	0.29
Periodicity	0.45	-0.27	-0.38
Chaos	-0.68	0.56	0.35

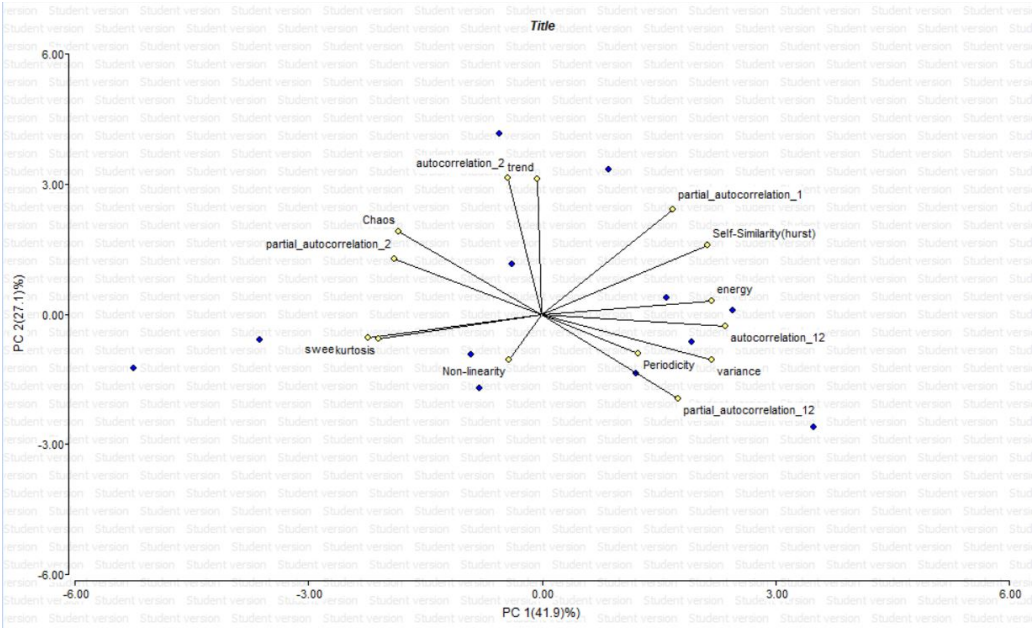


Figure 20: Biplot obtained, using the first two principal components as axes

From this biplot we can see that there are certain groups of lines that have a small angle between each other. The criteria by how we chose the characteristics is further explained in Appendix E. For example, we can see that Kurtosis and Skewness are almost on top of each other, thus they are very correlated and we can use only one or the other. Following, Trend and Kurtosis are almost perpendicular so then they are independent of each other; for this reason they can be used to difference the time series. The same logic was used for the rest of the characteristics. Finally, after applying this method we decided to use the characteristics obtained by the biplot, since they were the ones that obtained the best results. This characteristics can be seen in appendix G.

5.4.2 Decision Tree for selecting predictor

Following the method proposed in section 4, and using the characteristics extracted based on Biplot (tables 31, 32) we were able to train a Decision Tree. We use 10 Series for training and was validated with 1, making a Leave-one-out cross-validation (LOOCV), with all possible combinations. Figure 21 shows the tree that was obtained. The accuracy that was obtained from the decision tree is 0.55 .

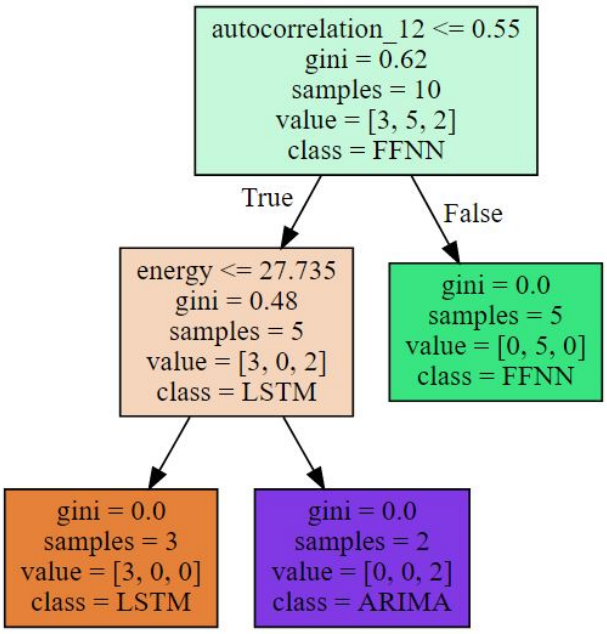


Figure 21: Resulting decision tree using the selected characteristics based on Biplot

5.4.3 Feed Forward Neural Network for selecting predictor

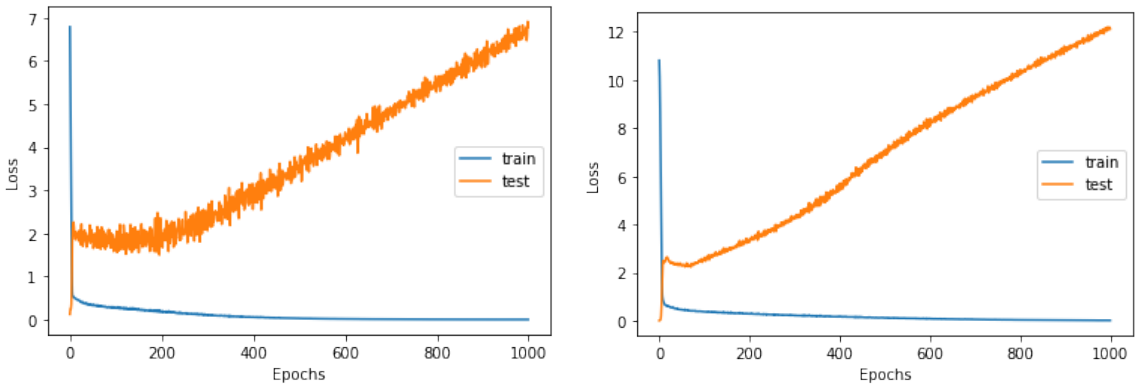
Using an FFNN to perform the meta-learning, we begin generating 11 different folds, to use a Leave-one-out cross-validation (LOOCV) approach to be able to measure the performance of this method. For this: we take the 11 series we have with their respective characteristics, remove one and use it to evaluate the method while the remaining 10 are used for training. The results obtained can be seen in the Table 22. It is important to mention that in the case of the test set it can only be either 100% or 0% because it is only tested in 1 series. If the FFNN return the best predictor for the series, then the accuracy is 100%, otherwise is 0%. It is not possible to take more series for testing, because the data is small, for example as we saw before there are only two series that are chosen by ARIMA.

Table 22: Results of Meta-learning with FFNN for selecting predictor

Folds	**Accuracy of training set**	**Accuracy of testing set**
1	100%	100%
2	100%	100%
3	100%	100%
4	100%	100%
5	100%	0%
6	100%	0%
7	100%	0%
8	100%	100%
9	100%	0%
10	100%	100%
11	100%	100%
Average:	**100%**	**55%**

As we can see the results are bad, clearly the reason for this is due to over-fitting. This can be observed by visualizing that the accuracy with the training data always reaches 100 percent, a clear indicator of over-fitting. Although this result is bad, it is quite expected due to the small number of time series available. Below in Figure 22 we have a graph of the "loss" (how far it is from the expected value) through epochs during training. We can observe how the value begins to decrease but as the network continues learning this begins to rise, making the classification worse, this is another proof of the existence of over-adjustment.

Figure 22: Examples of over-fitting during the training



(a) (b)

A positive result is that because 100 percent accuracy is always found with the training data, it can be seen that the network is possible to identify which models are better from the characteristics. If we had not even achieved that the training set has high accuracy, we would be in a case in which it is not possible to determine which method is better from the characteristics.

Now, assuming that the problem is the over-fitting on the model for the series it receives, we use data augmentation to generate a validation set in which the class of the test and validation sets are the same. When the prediction starts to get worse for that series we stop the training. The results of this experiment can be seen in the Table 23.

Table 23: Results of Meta-learning with FFNN avoiding over-fitting

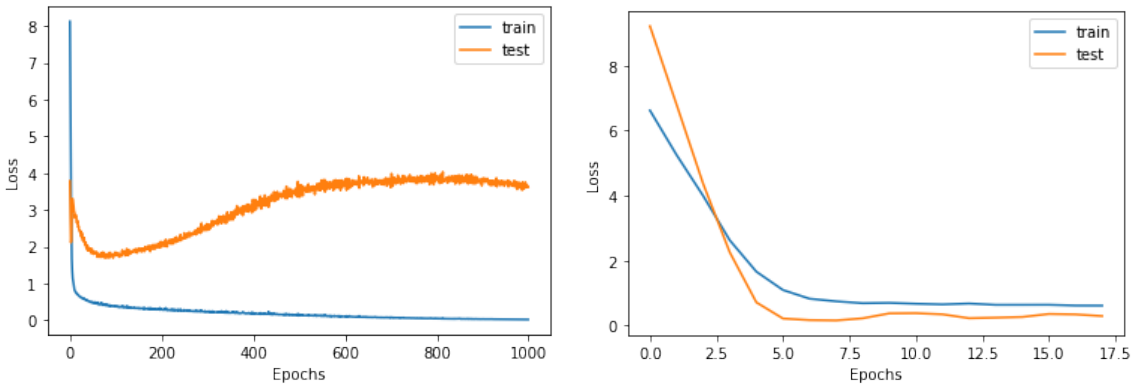
Folds	**Accuracy of training set**	**Accuracy of testing set**
1	50%	100%
2	80%	100%
3	80%	100%
4	80%	100%
5	80%	100%
6	90%	0%
7	90%	0%
8	80%	100%
9	80%	0%
10	80%	100%
11	80%	100%
Average:	**79%**	**73%**

These results are very interesting the accuracy is significantly improved in the testing set, which is quite encouraging and indicates that the over-fitting existed, as was deduced from the first experiment. The decrease in accuracy in the training set was expected by avoiding over-adjustment, but the drastic decrease indicates that more information is needed in the network to improve the accuracy without beginning to over-adjust.

The Figure 23 also shows improvement concerning over-training, we can see how now the value of loss does not begin to increase. It is important to mention that to achieve this correctly we use "early stopping" and "ModelCheckpoint", the implementation of these callbacks can be found at keras documentation [8].

[8]The implementation of early stopping and ModelCheckpoint can be found at: https://github.com/keras-team/keras/blob/master/keras/callbacks/callbacks.py#L633

Figure 23: Comparison of loss through epochs, stopping and not stopping training

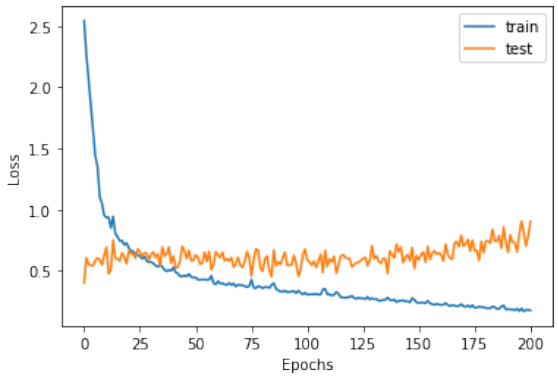


(a) Without stop the training

(b) Stopping the training

Unfortunately, the small data set does not allow us to use more series to make this process more robust (only 11 series). Although a significant improvement was obtained by stopping the training, there are graphs of the ”loss” through the epochs, such as Figure 24 that shows that the network still does not converge and this may be due to lack of information, this could be solved having more series. Despite all this, the result is successful, since 72% shows us that it is possible to use an FFNN to choose the prediction model using the characteristics of the time series. Due to a small data set and that the NN3 set has series of a specific field, we cannot extrapolate this result to all types of time series. But we can conclude that for the 11 time series of NN3 if we successfully stop making the learning before over-fitting, we can get a neural network that determines which of the methods works best for your prediction.

Figure 24: Training that does not finds patterns



5.5 Comparison of the different results obtained using a meta-learning approach

Table 24: Results of meta-learning methods and optimized models

Best Model	**Meta-learning with FFNN**		**Meta-learning with Decision Tree**		**LSTM GA**	**FFNN GA**	**Auto ARIMA**
FFNN	FFNN	0.0045	LSTM	0.0076	0.0076	0.0045	0.0051
FFNN	FFNN	0.0041	FFNN	0.0041	0.0159	0.0041	0.0158
FFNN	FFNN	0.0071	LSTM	0.0073	0.0073	0.0071	0.0258
FFNN	FFNN	0.0054	FFNN	0.0054	0.0120	0.0054	0.0141
FFNN	FFNN	0.0039	FFNN	0.0039	0.0049	0.0039	0.0040
ARIMA	FFNN	0.0064	LSTM	0.0083	0.0083	0.0064	0.0063
ARIMA	FFNN	0.0122	FFNN	0.0122	0.0115	0.0122	0.0109
LSTM	LSTM	0.0103	LSTM	0.0103	0.0103	0.0112	0.0157
LSTM	FFNN	0.0021	FFNN	0.0021	0.0020	0.0021	0.0019
LSTM	LSTM	0.0014	LSTM	0.0014	0.0014	0.0014	0.0019
FFNN	FFNN	0.0108	FFNN	0.0108	0.0109	0.0108	0.0176
	Average MSE	0.0062		0.0067	0.0084	0.0063	0.0108
	Std Dev	$\pm$0.003		$\pm$0.003	$\pm$0.006	$\pm$0.004	$\pm$0.007

To compare the meta-learning methods we used the MSE of the model that was determined to be the best from each meta-learning method. This also allows us to compare the difference between using or not using a meta-learning method. These results can be observed in the Table 24, with the following analysis:

The MSE obtained by using the meta-learning method with FFNN is on average better than its counterpart, this means that its prediction is better. In addition to this, the meta-learning method with the decision tree has better MSE than the ARIMA with its automatic approach, and that the optimized LSTM networks with GA. Therefore, we can conclude that the use of meta-learning is a success already which decreases our error in prediction, thanks to the previously acquired knowledge of the previous methods.

When we performed a statistical analysis of the results of the different meta-learning methods and optimized models; first, we obtained that they all follow a normal distribution with a significance level of 0.05. Therefore we proceed to perform an ANOVA test. Then, by using null hypothesis that their

means are equal, with a result of p-value = 0.139689. With this p-value, we cannot reject the fact that their means are equal, so we cannot guarantee that meta-learning is better than each individual model unless we take a significance level of 0.15.

Nevertheless, this result is not bad, on the contrary we can say that the predictions of the meta-learning approach are not worse than the use of FFNN with or LSTM with GA. This is a success considering the fact that the total training time is reduced since we simply have to use the recommended model and no longer have to train other models. Instead of training the three different models (LSTM with GA, Auto ARIMA, FFNN with GA), we would only have to train the model chosen by the meta-learning approach.

By averaging the MSE of the different models we obtain a value of 0.0085, and the average MSE of meta-learning approaches is 0.00645, due to there are only 3 prediction models, and two different meta-learning approaches a test of statistical significance is not possible. But this result is quite interesting because if we compare the use of a meta-learning approach, against not using it we can see a considerable difference.

6 Conclusions and Future Work

- We can conclude that the proposed method was successful because we can effectively reduce the CPU time required by the predictors. Furthermore, in some cases it gave better results than using traditional methods. This was done by extracting characteristics from the time series and then by selecting, using statistical methods, the ones that could be used to classify the time series. These would then be used to train the meta-learning methods to obtain a method to reduce the time needed to find the best predictor.

- Time series forecasting can be done by using methods such as statistical model and artificial intelligence. Although these methods are good they can be improved by using a optimization method such as genetic algorithms. From the results we can see that that due to improper architectures the FFNN and LSTM could not learn properly and as such they only converge to the mean value.

- We can see that improved methods using the genetic algorithm are better than the traditional methods, with a significance level of 0.10. However, the computational cost is often too high to train large data sets. Due to this limitation the use of Decision Trees and FFNNs as meta-learning approaches that reduce training time is important.

- We can also conclude that the statistical methods (PCA and biplot) that were used to reduce the number of characteristics used helped the decision tree. This was because the method had a tendency to be over fitted if we used all the time series characteristics.

6.1 Future Work

- During the research, we found that the hyper-parameters reached by the genetic algorithm also fit the error function used. This result opens a new research field to try new error functions that can improve the forecasting. A precursor of this research can be found in Appendix H.

- An important improvement to consider in our future work, is include the suggested scaled metric MASE, and dynamic time warping (DTW) which is useful to measure the similarity

between temporal sequences that may vary in speed. This will also help prevent predictions that converge to the average from having a low error.

- It is recommended to use random forests instead of decision trees. Although the possibility of analyzing what characteristics you use for classification is lost, the performance would increase and this analysis of characteristics can be done with Biplot. To do this the use of more time series is needed.

- In a future work, the use of a GPU is recommended because the training time that was needed for each series, normal and enhanced, was high. This could help tremendously and allow more time series to be analyzed in a reasonable time.

- The research could be expanded with the use of time series that have more data, such as the WTI petroleum price. Longer time series would make cross validation possible while training the predictors; also, we could extend the genetic pool since the time series is longer. This may improve the performance of the predictors.

References

[1] L. Wang, Y. Zeng, and T. Chen, "Back propagation neural network with adaptive differential evolution algorithm for time series forecasting," Expert Systems with Applications, vol. 42, no. 2, pp. 855 – 863, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417414004941

[2] A. Sagheer and M. Kotb, "Time series forecasting of petroleum production using deep lstm recurrent networks," Neurocomputing, vol. 323, pp. 203 – 213, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218311639

[3] M. Mishra, "Unboxing arima models," Jun 2018. [Online]. Available: https://towardsdatascience.com/unboxing-arima-models-1dc09d2746f8

[4] C. Lemke and B. Gabrys, "Meta-learning for time series forecasting and forecast combination," Neurocomputing, vol. 73, pp. 2006–2016, Jun 2010.

[5] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," in PloS one, 2018.

[6] G. Tiao, "Time series: Arima methods," in International Encyclopedia of the Social Behavioral Sciences, N. J. Smelser and P. B. Baltes, Eds. Oxford: Pergamon, 2001, pp. 15 704 – 15 709. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B0080430767005209

[7] R. Vadlamani, P. Dadabada, and K. Deb, "Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms," Swarm and Evolutionary Computation, vol. 36, May 2017.

[8] T.-C. Fu, "A review on time series data mining," Eng. Appl. of AI, vol. 24, pp. 164–181, 2011.

[9] G. E. P. Box, G. C. Reinsel, and G. M. Jenkins, Time series analysis : forecasting and control. Englewood Cliffs, NJ: Prentice-Hall, 1994.

[10] D. E. Rumelhart and J. L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. MIT Press, 1986.

[11] V. Chandrasekara and C. Tilakaratne, "Forecasting exchange rates using artificial neural networks," Sri Lankan Journal of Applied Statistics, vol. 10, pp. 187–201, Jan 2009.

[12] G. Cybenko, "Approximation by superpositions of a sigmoidal function," Mathematics of Control, Signals and Systems, vol. 2, no. 4, pp. 303–314, Dec 1989. [Online]. Available: https://doi.org/10.1007/BF02551274

[13] S. Bornholdt and D. Graudenz, "General asymmetric neural networks and structure design by genetic algorithms," Neural Networks, vol. 5, no. 2, pp. 327 – 334, 1992. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608005800309

[14] E. Vonk, L. Jain, L. Veelenturf, and R. Johnson, "Automatic general of a neural network architecture using evolutionary computation," in Proceedings of the International Conference and Workshops ETD2000. United States: IEEE, 5 1995.

[15] M. Saemi, M. Ahmadi, and A. Y. Varjani, "Design of neural networks using genetic algorithm for the permeability estimation of the reservoir," Journal of Petroleum Science and Engineering, vol. 59, no. 1, pp. 97 – 105, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0920410507000472

[16] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," Artificial Intelligence Review, vol. 18, no. 2, pp. 77–95, Jun 2002. [Online]. Available: https://doi.org/10.1023/A:1019956318069

[17] C. Schaffer, "Cross-validation, stacking and bi-level stacking: Meta-methods for classification learning," in Selecting Models from Data, P. Cheeseman and R. W. Oldford, Eds. New York, NY: Springer New York, 1994, pp. 51–59.

[18] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," Neural Computation, vol. 8, no. 7, pp. 1341–1390, 1996. [Online]. Available: https://doi.org/10.1162/neco.1996.8.7.1341

[19] P. Brockwell and R. Davis, Introduction to Time Series and Forecasting, ser. Springer Texts in Statistics. Springer International Publishing, 2016.

[20] X. Wang, K. Smith-Miles, and R. Hyndman, "Characteristic-based clustering for time series data," Data Min. Knowl. Discov., vol. 13, pp. 335–364, Sept 2006.

[21] V. Landassuri, C. Bustillo-Hernández, J. Hernández, and L. Sanchez Fernandez, "Single-step-ahead and multi-step-ahead prediction with evolutionary artificial neural networks," Nov 2013, pp. 65–72.

[22] I. The MathWorks, "Multistep neural network prediction," 2019. [Online]. Available: https://www.mathworks.com/help/deeplearning/ug/multistep-neural-network-prediction.html

[23] R. Hyndman and G. Athanasopoulos, Forecasting: principles and practice. OTexts, 2018.

[24] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," Data Mining and Knowledge Discovery, vol. 13, no. 3, pp. 335–364, Nov 2006. [Online]. Available: https://doi.org/10.1007/s10618-005-0039-x

[25] T. S. Talagala, R. J. Hyndman, and G. Athanasopoulos, "Meta-learning how to forecast time series," Tech. Rep., 2018.

[26] J. Harvill and B. Ray, "Testing for nonlinearity in a vector time series," Feb 1970.

[27] T. Lee, "Neural network test and nonparametric kernel test for neglected nonlinearity in regression models," Studies in Nonlinear Dynamics and Econometrics, vol. 4, pp. 169–182, Dec 2000.

[28] T. Terasvirta, "Power properties of linearity tests for time series," Studies in Nonlinear Dynamics Econometrics, vol. 1, pp. 3–10, Feb 2007.

[29] M. K. Cain, Z. Zhang, and K.-H. Yuan, "Univariate and multivariate skewness and kurtosis for measuring nonnormality: Prevalence, influence and estimation," Behavior Research Methods, vol. 49, no. 5, pp. 1716–1735, Oct 2017. [Online]. Available: https://doi.org/10.3758/s13428-016-0814-1

[30] L. T. Decarlo, "On the meaning and use of kurtosis," Psychological Methods, pp. 292–307, 1997.

[31] K. Guan, "Important notes on lyapunov exponents," Jan 2014.

[32] OECD, "Periodicity - imf," https://stats.oecd.org/glossary/detail.asp?ID=2041, last accessed: 2019-09-26.

[33] I. MathWorks, "Predictive analytics 3 things you need to know," https://www.mathworks.com/discovery/predictive-analytics.html?s_tid=srchtitle, last accessed: 2019-09-23.

[34] C. N. Babu and B. E. Reddy, "A moving-average filter based hybrid arima–ann model for forecasting time series data," Applied Soft Computing, vol. 23, pp. 27 – 38, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494614002555

[35] S. Barak and S. S. Sadegh, "Forecasting energy consumption using ensemble arima–anfis hybrid algorithm," International Journal of Electrical Power Energy Systems, vol. 82, pp. 92 – 104, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061516303702

[36] M. Valipour, M. E. Banihabib, and S. M. R. Behbahani, "Comparison of the arma, arima, and the autoregressive artificial neural network models in forecasting the monthly inflow of dez dam reservoir," Journal of Hydrology, vol. 476, pp. 433 – 441, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S002216941200981X

[37] H. Akaike, "A new look at the statistical model identification," IEEE Transactions on Automatic Control, vol. 19, no. 6, pp. 716–723, Dec 1974.

[38] G. Schwarz, "Estimating the dimension of a model," The Annals of Statistics, vol. 6, no. 2, pp. 461–464, 1978. [Online]. Available: http://www.jstor.org/stable/2958889

[39] T. G. Smith, "pmdarima: Arima estimators for python," https://www.alkaline-ml.com/pmdarima/about.html#about, last accessed: 2019-04-10.

[40] S. Haykin, Neural Networks: A Comprehensive Foundation. Prentice Hall, 1999.

[41] H. Y. Kim and C. H. Won, "Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models," Expert Systems with Applications, vol. 103, pp. 25 – 37, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418301416

[42] Q. Zhang, J. Sun, G. Xiao, and E. Tsang, "Evolutionary algorithms refining a heuristic: A hybrid method for shared-path protections in wdm networks under srlg constraints," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 37, no. 1, pp. 51–61, Feb 2007.

[43] H. Lingaraj, "A study on genetic algorithm and its applications," International Journal of Computer Sciences and Engineering, vol. 4, pp. 139–143, Oct 2016.

[44] J. Kennedy, Particle Swarm Optimization. Boston, MA: Springer US, 2010, pp. 760–766. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_630

[45] J. L. Fernández-Martínez, "A brief historical review of particle swarm optimization (pso)," Journal of Bioinformatics and Intelligent Control, vol. 1, pp. 3–16, May 2012.

[46] A. Dastanpour, "Effect of genetic algorithm on artificial neural network for intrusion detection system," INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING, vol. 04, pp. 10–18, Oct 2016.

[47] R. Zhao and S. Lv, "Neural-network based test cases generation using genetic algorithm," in 13th Pacific Rim International Symposium on Dependable Computing (PRDC 2007), Dec 2007, pp. 97–100.

[48] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," Artificial Intelligence Review, vol. 18, Sept 2001.

[49] K. Grabczewski, Meta-Learning in Decision Tree Induction, Jan 2014, vol. 498.

[50] R. Prudêncio and T. Ludermir, "Meta-learning approaches to selecting time series models," Neurocomputing, vol. 61, pp. 121–137, Oct 2004.

[51] Y. Peng, P. Flach, P. Brazdil, and C. Soares, "Decision tree-based characterization for meta-learning," M. Bohanec, B. Kasek, N. Lavrac, and D. Mladenic, Eds. University of Helsinki, 8 2002, pp. 111 – 122, conference Proceedings/Title of Journal: ECML/PKDD'02 workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning.

[52] R. S. Brid, "Decision trees: A simple way to visualize a decision," https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb, last accessed: 2019-05-01.

[53] P. B. Brazdil, C. Soares, and J. P. da Costa, "Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results," Machine Learning, vol. 50, no. 3, pp. 251–277, Mar 2003. [Online]. Available: https://doi.org/10.1023/A:1021713901879

[54] J. S. Armstrong, Long-range forecasting : from crystal ball to computer. Wiley New York, 1978.

[55] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," International Journal of Forecasting, vol. 22, no. 4, pp. 679 – 688, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207006000239

[56] C. Bergmeir, R. Hyndman, and B. Koo, "A note on the validity of cross-validation for evaluating autoregressive time series prediction," Computational Statistics Data Analysis, vol. 120, Nov 2017.

[57] J. Opsomer, Y. Wang, and Y. Yang, "Nonparametric regressin with correlated errors," Statist. Sci., vol. 16, no. 2, pp. 134–153, May 2001. [Online]. Available: https://doi.org/10.1214/ss/1009213287

[58] C. Bergmeir and J. Benítez, "On the use of cross-validation for time series predictor evaluation," Information Sciences, vol. 191, p. 192–213, May 2012.

[59] MATHLAB, "Analysis of time-series models," https://se.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html, last accessed: Apr 2019.

[60] L. Mozaffari, A. Mozaffari, and N. L. Azad, "Vehicle speed prediction via a sliding-window time series analysis and an evolutionary least learning machine: A case study on san francisco urban roads," Engineering Science and Technology, an International Journal, vol. 6, Dec 2014.

[61] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The m4 competition: Results, findings, conclusion and way forward," International Journal of Forecasting, vol. 34, no. 4, pp. 802 – 808, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207018300785

[62] C. T. Dhanya and D. Nagesh Kumar, "Multivariate nonlinear ensemble prediction of daily chaotic rainfall with climate inputs," Journal of Hydrology, vol. 403, pp. 292–306, June 2011.

[63] J. Chen, W. Chen, C. Huang, S. Huang, and A. Chen, "Financial time-series data analysis using deep convolutional neural networks," pp. 87–92, Nov 2016.

[64] Y. Kang, R. J. Hyndman, and K. Smith-Miles, "Visualising forecasting algorithm performance using time series instance spaces," International Journal of Forecasting, vol. 33, no. 2, pp. 345 – 358, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207016301030

[65] D. Nibbering, R. Paap, and M. van der Wel, "What do professional forecasters actually predict?" International Journal of Forecasting, vol. 34, no. 2, pp. 288 – 311, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207018300049

[66] A. Jayawardena, Environmental and Hydrological Systems Modelling. CRC Press, 2013.

[67] M. Sazli, "A brief review of feed-forward neural networks," Communications, Faculty Of Science, University of Ankara, vol. 50, pp. 11–17, Jan 2006.

[68] I. Jolliffe and Springer-Verlag, Principal Component Analysis, ser. Springer Series in Statistics. Springer, 2002.

[69] A. C. Rencher and W. F. Christensen, Methods of multivariate analysis. Hoboken, New Jersey: Wiley, 2012.

[70] A. Prasad, "Prediction and analysis of time series data using tensorflow," Nov 2019. [Online]. Available: https://towardsdatascience.com/prediction-and-analysis-of-time-series-data-using-tensorflow-2136ef633018

Appendices

A Back-propagation training algorithm

Back-propagation is the most used training algorithm to train feed-forward neural networks. This method is used basically for update the synaptic weights of the network by back-propagating a gradient vector. The elements of the gradient vector are composed by the derivative of an error measure with respect to a parameter. The most common error measure is the difference between the desired output and the real output. Consequently, Back-propagation is a supervised learning algorithm because it needs a set of data with desirable output values, also called labeled data. [67]. The way to update weights that Back-propagation use, follows the following stages:

First of all, obtaining the error of the feed-forward network during each iteration n with the use of an error measure. The error measure is used for finding the error between the desired output y_i of the neuron i, and the calculated output of that neuron $\hat{y}_i$ using the current weights, as was previously mentioned the most common error measure is the difference between the desired output and the real output.

$$e_i(n) = y_i - \hat{y}_i(n) \tag{1}$$

Where, n represents the iteration number that corresponds to each presented training example from the training set. The error and the obtained output depend on the iteration number due to the learning process.

Secondly, with the use of error function is possible to define the total energy function as follows:

$$E(n) = \frac{1}{2}\sum_{i=1}^{n} e_i^2(n) \tag{2}$$

Finally, according to the learning rate (LR) and with the use of the energy function and activation function, the update of weights and biases occur with:

$$\Delta w_{ji} = \eta e_j(n) f' \left(\sum_{i=0}^{m} w_{ji}(n) y_i(n) \right) y_j(n) \tag{3}$$

Where Δw_{ji} is the value in which the weights must change, η represents the learning rate, and f' is the derivative of the activation function.

B Artificial Neural Network & Computational Intelligence Forecasting Competition (NN3)

NN3 is a Forecasting Competition for Neural Networks and Computational Intelligence that challenges academics to produce the best forecast using any method of computational intelligence. The competition has 2 datasets, one of them is a complete dataset of 111 monthly time series drawn from homogeneous population of empirical business time series. The another dataset is a sub sample of 11 time series from the 111 time series, and is therefore contained in the larger dataset.

C LSTM

To explain the functionality of the LSTM cell it is necessary to define the variables that LSTM uses in its functions. That are:

Table 25: Definition of LSTM variables

LSTM variables			
Variable	**Definition**	**Variable**	**Definition**
σ	sigmoid layer	$tanh$	hyperbolic tangent layer
h_{t-1}	output of the previous LSTM layer	C_{t-1}	previous cell state of LSTM layer
x_t	current input	C_t	cell state
$\tilde{C}_t$	new candidate values vector for the cell state	h_t	current output
f_t	what the cell will forget	W_f, W_i, W_c, W_o	Recurrent Weights
b_f, b_i, b_c, b_o	Biases		

The mathematical functions that describe each part are :

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C} = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{3}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t tanh(C_t) \tag{6}$$

Description of the operators that are used signify:

$\otimes$: is the scaling of information

$\oplus$: is the addition of information

Now we will detail the process that an LSTM cell goes through in order to predict the next value:

1. First, the cell will decide with h_{t-1} and x_t if the new information will be used or discarded. This is done by using a sigmoid layer, if the result is 1 then the information is used, else if the result is 0 then it is discarded though the forget gate. The results is f_t.

2. Second, the cell decides what new information will be stored in the cell state (C_t). This is done in two steps: First a sigmoid layer will decide which values should be updated (input gate), i_t, then a $tanh$ layer will create a candidate vector for the cell state $\tilde{C}$. Second, the values are scaled together to update the cell state.

3. Third, the cell will now be updated using the previous cell state. This is done by multiplying the old state C_{t-1} by f_t, thus forgetting information from the previous state. This is then added by the result of step 2, now the cell has been updated.

4. Fourth, the outputs of the cell will now be calculated by taking into account the cell state. This is done by using a sigmoid layer to determine what is going to be part of the output. Then the state is applied a $tanh$ layer to transform the data to a scale -1 to 1. Finally these two results are multiplied so we get the outputs (output gate) of only the parts that were decided.

D Autoregressive (AR) models

In statistics an AR model (autoregressive model) is a representation of random process. The autoregressive model aim that the output variable depends linearly on its own previous values and on a stochastic term; this is why the model is in the form of a stochastic difference equation.

Moving-average (MA) model and autoregressive model (AR model) are special cases and key components of the more general autoregressive moving average (ARMA) and the autoregressive integrated moving average (ARIMA) explained in the theoretical framework.

The adjustment of an AR model to the data through the use of ordinary least squares, is already implemented in R, its implementation can be seen in its documentation [9].

[9] Implemetation in R is available at: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/ar.ols.html

E Statistical Methods foe dimensionality reduction

These methods will be used to determine which characteristics of the time series will be used for the meta-learning training.

E.1 Principal Component Analysis (PCA)

PCA is used transform the variables of the data set into a set of linearly uncorrelated variables, known as Principal Components (PC), these are are mainly used to reduce the dimension of the data set [68].

Each PC is calculated as a linear combination of the variables in the data with maximum variance. This is done so that they represent most of the variability of the data. After the first PC is calculated, the next is calculated as a linear combination in an orthogonal direction to the mentioned PC [69].

E.2 Biplot

A biplot is a 2D graphical representation of a set of data. Where there two types of points: the first represents the observations (rows) and the second represents the variables (columns) of the data [69].

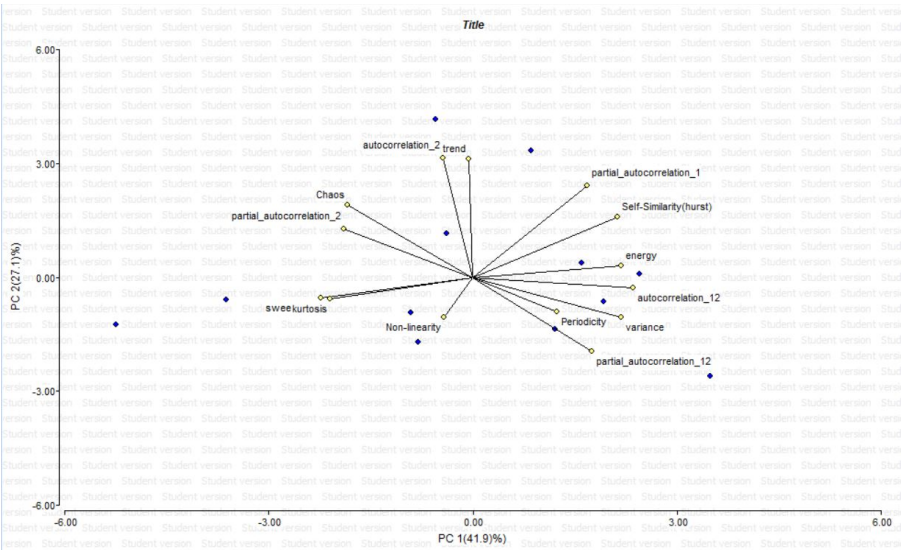


Figure 25: Biplot example

Figure 25 shows an example of the biplot that was applied to the 15 characteristics, obtained from the 11 time series of NN3. Furthermore we can interpret the lines as the variables and the points as

the observations of the data. Depending on the orientation of the lines we can draw the following conclusions:

- If two lines are perpendicular to each other, then they are independent.
- If two lines have a small angle between each other then they are closely correlated.
- If two lines have a 180° angle then they are inversely correlated

F Mean convergence problem in predictors based on neural networks

Forecasters may sometimes result in a forecast that is the mean value throughout a time series. This behaviour could seem strange but it is caused by the objective that the forecasters have, that is: reduce the amount of MSE that the forecast has [70]. This means that in some cases the best forecast is just the mean value of the time series.

G Characteristics Extraction

In tables 26, 27 and 28 we can observe the values obtained from all the characteristics proposed in the theoretical framework.

Table 26: All characteristics obtained from the 11 time Series (1/3)

trend	**sweekness**	**autocorrelation_1**	**autocorrelation_2**	**autocorrelation_12**
1.43	-0.10	0.50	0.60	0.75
1.03	-0.15	0.65	0.34	0.79
1.00	1.33	0.60	0.15	0.83
1.02	-0.72	0.72	0.30	0.85
1.54	-0.15	0.91	0.82	0.58
1.05	0.23	0.30	0.31	0.37
1.19	0.29	0.71	0.41	0.52
1.04	1.68	0.23	0.32	-0.04
2.04	1.39	0.93	0.86	0.40
1.02	4.46	0.27	0.15	0.17
1.04	0.89	0.32	0.10	0.52

Table 27: All characteristics obtained from the 11 time Series (2/3)

partial_autocorrelation_1	**partial_autocorrelation_2**	**partial_autocorrelation_12**	**kurtosis**	**energy**
0.50	0.48	0.46	-0.27	40.02
0.65	-0.14	0.62	-0.99	42.36
0.60	-0.31	0.58	0.38	19.68
0.72	-0.47	0.37	-0.51	43.37
0.91	0.00	-0.14	-0.13	44.94
0.30	0.25	0.12	0.02	35.75
0.71	-0.20	0.21	-0.25	35.53
0.23	0.28	-0.10	5.10	16.17
0.93	0.05	0.00	1.48	19.94
0.27	0.09	0.06	30.42	8.57
0.32	-0.01	0.35	0.34	22.60

Table 28: All characteristics obtained from the 11 time Series (3/3)

variance	Non-linearity	Self-Similarity(hurst)	Periodicity	Chaos
0.02	0.12	0.83	0.01	0.86
0.04	0.80	0.98	0.11	0.58
0.05	0.74	0.97	0.11	0.57
0.04	0.01	0.99	0.11	0.57
0.03	0.01	1.00	0.02	0.77
0.02	0.02	0.75	0.11	0.55
0.03	0.01	0.99	0.27	0.53
0.01	0.26	0.73	0.02	0.80
0.03	0.07	1.00	0.00	0.97
0.01	0.50	0.70	0.00	0.97
0.03	0.08	0.72	0.11	0.57

In the tables 29 and 30 we can see the characteristics that have been chosen after performing the Biplot analysis, the number of features has been reduced from 15 to 10.

Table 29: Characteristics chosen after Biplot analysis (1/2)

trend	sweekness	autocorrelation_12	kurtosis	energy
1.43	-0.10	0.75	-0.27	40.02
1.03	-0.15	0.79	-0.99	42.36
1.00	1.33	0.83	0.38	19.68
1.02	-0.72	0.85	-0.51	43.37
1.54	-0.15	0.58	-0.13	44.94
1.05	0.23	0.37	0.02	35.75
1.19	0.29	0.52	-0.25	35.53
1.04	1.68	-0.04	5.10	16.17
2.04	1.39	0.40	1.48	19.94
1.02	4.46	0.17	30.42	8.57
1.04	0.89	0.52	0.34	22.60

Table 30: Characteristics chosen after Biplot analysis (2/2)

variance	Non-linearity	Self-Similarity(hurst)	Periodicity	Chaos
0.02	0.12	0.83	0.01	0.86
0.04	0.80	0.98	0.11	0.58
0.05	0.74	0.97	0.11	0.57
0.04	0.01	0.99	0.11	0.57
0.03	0.01	1.00	0.02	0.77
0.02	0.02	0.75	0.11	0.55
0.03	0.01	0.99	0.27	0.53
0.01	0.26	0.73	0.02	0.80
0.03	0.07	1.00	0.00	0.97
0.01	0.50	0.70	0.00	0.97
0.03	0.08	0.72	0.11	0.57

Finally, we can see that the tables 31 and 32 present the selected characteristics after the analysis with Biplot. In addition to these, the tables are labeled, this indicates which model was best for each series, in this way the learning process of meta-learning methods can be done.

Table 31: Table used for the meta-learning training (1/2),

Series	Winner	trend	sweekness	autocorrelation_12	kurtosis	energy
Serie 1	**FFNN**	1.43	-0.10	0.75	-0.27	40.02
Serie 2	**FFNN**	1.03	-0.15	0.79	-0.99	42.36
Serie 3	**FFNN**	1.00	1.33	0.83	0.38	19.68
Serie 4	**FFNN**	1.02	-0.72	0.85	-0.51	43.37
Serie 5	**FFNN**	1.54	-0.15	0.58	-0.13	44.94
Serie 6	**ARIMA**	1.05	0.23	0.37	0.02	35.75
Serie 7	**ARIMA**	1.19	0.29	0.52	-0.25	35.53
Serie 8	**LSTM**	1.04	1.68	-0.04	5.10	16.17
Serie 9	**LSTM**	2.04	1.39	0.40	1.48	19.94
Serie 10	**LSTM**	1.02	4.46	0.17	30.42	8.57
Serie 11	**FFNN**	1.04	0.89	0.52	0.34	22.60

Table 32: Table used for the meta-learning training (2/2)

Series	Winner	variance	Non-linearity	Self-Similarity(hurst)	Periodicity	Chaos
Serie 1	**FFNN**	0.02	0.12	0.83	0.01	0.86
Serie 2	**FFNN**	0.04	0.80	0.98	0.11	0.58
Serie 3	**FFNN**	0.05	0.74	0.97	0.11	0.57
Serie 4	**FFNN**	0.04	0.01	0.99	0.11	0.57
Serie 5	**FFNN**	0.03	0.01	1.00	0.02	0.77
Serie 6	**ARIMA**	0.02	0.02	0.75	0.11	0.55
Serie 7	**ARIMA**	0.03	0.01	0.99	0.27	0.53
Serie 8	**LSTM**	0.01	0.26	0.73	0.02	0.80
Serie 9	**LSTM**	0.03	0.07	1.00	0.00	0.97
Serie 10	**LSTM**	0.01	0.50	0.70	0.00	0.97
Serie 11	**FFNN**	0.03	0.08	0.72	0.11	0.57

H Example of adaptability of hyper-parameters

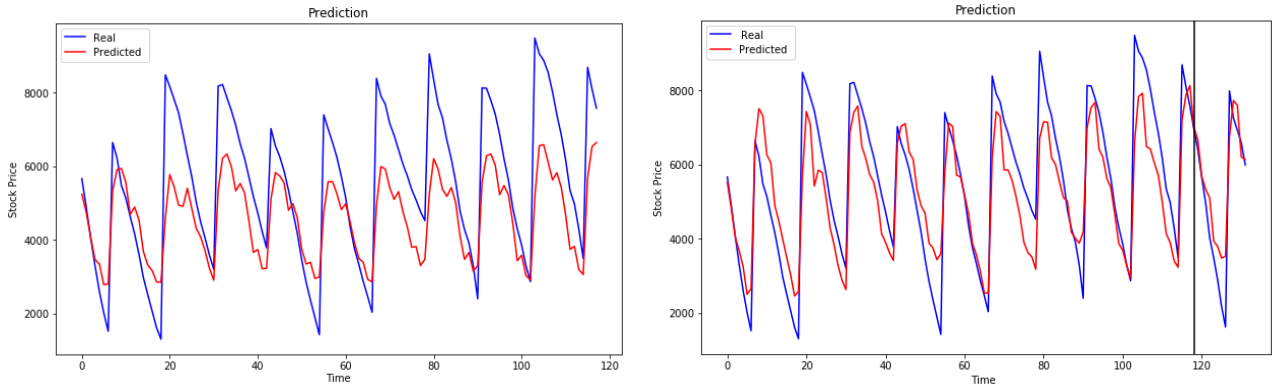
An interesting result we find is that if we use another error function for the input, the genetic algorithm tends to choose different hyper-parameters. This can be seen in the Table 33, where we use a custom error function applied to FFNN with GA. This function benefits the predictions below the real value and penalizes more those that are above the real value. This is done with the following pseudo-code (our implementation in GitHub, Appendix I) :

1. Evaluate if the predicted value is greater than the real value.
2. If the predicted value is greater return $2 \times MSE$.
3. If the predicted value is less or equal than the expected value, return MSE.

Table 33: Parameters found using the proposed function error and MSE

Hyper-Parameters	**Custom Error function**	**MSE**
Input neurons	12	12
Hidden layer	2	2
Hidden layer neurons	12	24
Optimizer	SGD	Adam
Activation function	Relu	Relu

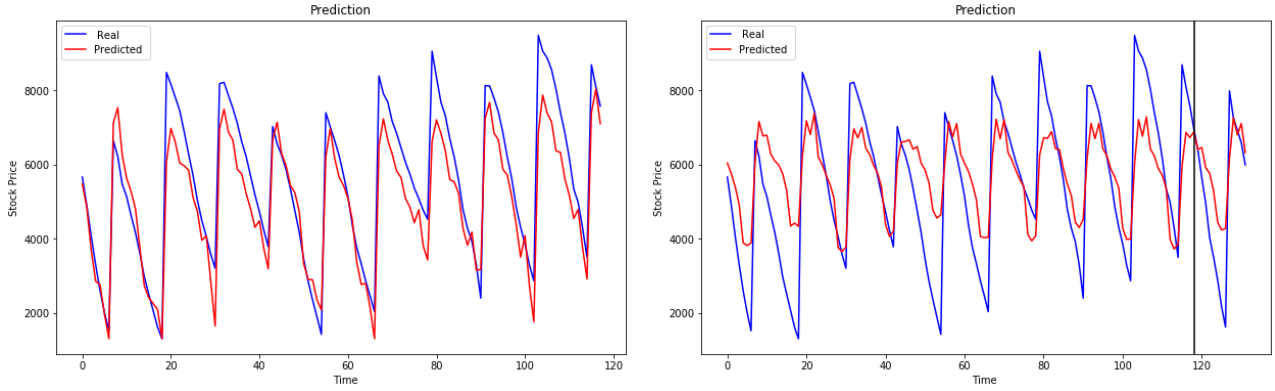
The results of using a different error function not only cause the genetic algorithm to obtain different hyper-parameters, but also a different graph. As we can see in the Figure 26 by penalizing the predictions above the graph the new prediction is always below the real value.



(a) Custom error function (b) MSE

Figure 26: Predictions obtained from FFNN with GA using different error functions

Now, this might seem that everything is because the training is done with a custom error function, however, the most interesting point is obtained when we exchange the hyper-parameters that the genetic algorithm found. In other words, we use the hyper-parameters chosen by the genetic algorithm when it was trained with the custom error function and we use them now to train a network with the MSE error function, the results are shown in the Figure 27.



(a) Custom error function with mse hyper-parameters (b) MSE with custom error function hyper-parameters

Figure 27: Comparing different error functions with hyper-parameters exchanged

The result in the Figure 27(b) demonstrates how even though we now do the training with MSE, the hyper-parameters seem to adapt the prediction below the real value, which again demonstrates the importance of the hyper-parameters selection.

I Algorithm of the Proposed Method

The repository available at: https://github.com/hug0er/Time-Series-Prediction