Time series analysis for price recommendation in the telecommunications market

Artificial Intelligence Applications - Prof: Sónia Gouveia (sonia.gouveia@ua.pt) - Academic Year 2022/23

André Reis Fernandes
Number: 97977
MSc. Computacional Engineering
Department of Electronics,
Telecommunications and Informatics
andre.fernandes16@ua.pt
Workload : 50%

Gonçalo Jorge Loureiro de Freitas
Number: 98012
MSc. Computacional Engineering
Department of Electronics,
Telecommunications and Informatics
goncalojfreitas@ua.pt
Workload : 50%

***Abstract*—This article focuses on forecasting prices of a given product in a company by applying different models that are prepared to work with time series. In this project we've applied RNN, univariate and multivariate LSTM, GRU, ARIMA, SARIMA and Facebook Prophet. After analyzing all the results, we've came to a conclusion that the best model was the ARIMA(3, 1, 0) or it's equivalent, SARIMA model [(3, 1, 0),(0, 0, 0, 0),'n'], which results gotten are respectively: MAPE=0.04, MSE=12.164 and MAE=0.831; MAPE=0.04, MSE=12.174 and MAE=0.833. We've conclude that the objective of this paper was accomplished with success.**

***Index Terms*—Time Series, RNN, LSTM, ARMA, Hyperparameters, Loss Function, MAPE, MSE, Univariate, Multivariate**

I. Introduction

Time series forecasting is a technique for predicting future events by analyzing past trends, checking for patterns of time decomposition, such as trends, seasonal and cyclical patterns and regularity. It involves building models through historical analysis and using them to make observations and drive future strategic decision-making, with the assumption that future trends will hold similar to historical trends, [1]. The goal of time series forecasting is to predict a future value or classification at a particular point in time.
In order to achieve this goal, it is important to know that time series data can be broken down into 3 main components, trend, seasonality, and random noise. Trend is a long-term increase or decrease in the data with the possibility of this also being stationary. Seasonality is represented by regular patterns that occur at specific time intervals, such as daily, weekly, or annually, while random noise is unpredictable fluctuations in the data that cannot be explained by the other components, [2]. The time series and its components change with time, through the index, and, while trend and seasonality are considered deterministic components, random noise is random.

Some of the most famous time series models are, Time series regression (most common), Exponential smoothing, Auto-regressive integrated moving average (ARIMA) models and Neural Networks.

Time series models are applied in a wide range of domains, like predicting the price of a certain product or predicting weather conditions in a certain region. In situations where a decision needs to be taken, where the uncertainty about the future is involved, these time series models have been pointed to be among the most efficient methods of forecasting, [3].

In this project we are going to study the price of the *huawei Y9S 128GB*, in a company called *Ripley*, between the 22nd of June of 2020 and the 21nd of April of 2021 and try to predict the price between the 22rd of April of 2021 and the 9th of November of 2021, while comparing with the actual prices. In this project we are going to study a total of 11 machine learning models prepared to work with time series, such as RNN's, LTSMS, and others.

II. Data Visualization & Pre-processing

The data-set, provided by the teacher responsible for this module, it's divided in three different folders, *time series* 1, 2 and 3, each one consisting in 9 companies and the price of a certain product in a specified period of time. For this problem we are only going to use the data in the folder *time series* 1 and, as we want to study of the *huawei Y9S 128GB*, we need to set the $product\_group\_id$ to 958. All the code can be easily changed to study a different product, company or a different *time series* folder. You can find all the code and the data-set in our github repository, [4].

In Fig. 1, we can see the price of the *huawei Y9S 128GB* per company per time stamp.

In this figure, and more clearly in Fig. 2, that represents the price of the product per time stamp for the *Ripley* company, we can see that there are missing values in the data. The number of missing values per company is represented in Table I.

As we can see in Table I, only the company *movistar* has none missing values and company we are going to study, *Ripley*, has a total of 145 missing values.

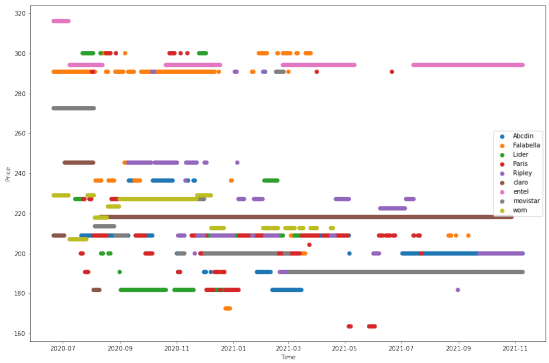

Fig. 1: Price of the product per company per time stamp

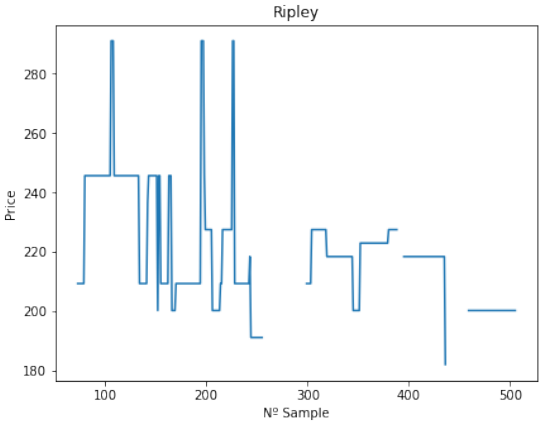

Fig. 2: Price sample of time for the company *Ripley*

TABLE I: Number of missing values per company

Companies	Nº of missing values
Abcdin	68
Falabella	208
Lider	260
Paris	196
Ripley	145
claro	12
entel	198
movistar	0
wom	281

There are many methods to handle missing data but we decide to use the interpolation technique, which works by assuming a relationship within a range of data point, and using the time method, that estimates missing values by focusing more on nearby points than far away points, [5]. We also define the parameter of *limit_direction* as *'both'* so that consecutive NaNs will be filled in both directions, [6].

After the interpolation we got the following graph, represented in Fig. 3, for the *Ripley* company, where we can see that there are no more missing values. For the other companies the graphs of the product price per sample of time, after the interpolation, as well as a box-plot of the data, can be found in the Appendix, section A.

From the box-plot, also known as a box-and-whisker plot, we can see the median of each company in the data, represented by the orange horizontal line, as well as the spread of the data given by the length of the box, that represents the interquartile range (IQR), giving an indication of the spread of the middle 50% of the data; the skewness of the data, i.e., if the median line is not in the center of the box, it indicates skewness and finally, the presence of outliers, that are data points that fall outside the whiskers, which can indicate unusual or unexpected values in the dataset.

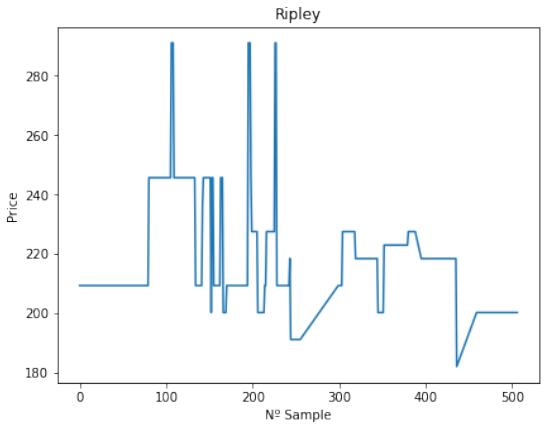

Fig. 3: Price sample of time for the company *Ripley* after using the interpolation

Now that we have handle all missing values for all companies we can start applying the models to predict the price of the product in a given time, for the company *Ripley*.

III. Models Used

A. RNN

In traditional neural networks, all the inputs and outputs are independent of each other. However, while working with time series, whose objective is to make predictions of, in this specific case, future prices, the previous prices are required and hence there is a need to remember them, [7]. Thus, Recurrent Neural Network(RNN) came into existence. A RNN is a special type of artificial neural network (ANN) that is adapted to work for time series data or data that involves sequences. RNNs have the concept of "memory" that helps them store the states or information of previous inputs to generate the next output of the sequence, i.e, in machine learning terms, the output from the previous step are fed as input to the current step, [8], just like represented in Fig. 4. Due to this, the main and most important feature of RNN is the hidden state, which remembers some information about a sequence.

More about RNN's can be found in [10]. As for the code we developed for this model, this was based in [11].

B. LSTM

The problem with Recurrent Neural Networks is that although they have a short-term memory to retain previous

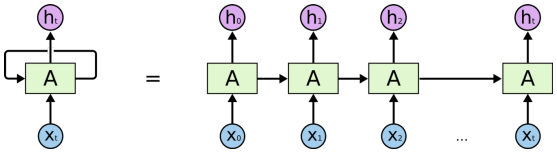

Fig. 4: Recurrent Neural Network diagram where A represents the hidden layer/state, [9]

information in the current neuron, this ability decreases very quickly for longer sequences. To fight this problem, the LSTM models were introduced to be able to retain past information even longer, handling the vanishing gradient problem faced by RNN, [12], as they are known to hold information for a long time by default. The vanishing gradient problem is an issue that can occur when training deep neural networks. It is caused by the gradients of the error function becoming increasingly small as they are backpropagated through the layers of the network. This can eventually lead to the gradients becoming so small that they are effectively zero, and the network is unable to learn any further [13].

In general, a LSTM structure is comprised of a cell and three gates, the Input, Output and Forget gates. The cell retains values over arbitrary time intervals, and the three gates are responsible for controlling the flow of information into and out of the cell. In the Forget Gate, is decided which current and previous information is kept and which is thrown out. This includes the hidden status from the previous run and the current status. These values are passed into a *sigmoid* function, which can only output values between 0 and 1, 0 meaning that all previous information is forgotten and 1 that all previous information is kept. As for the Input Gate, this is where it is decided how valuable the current input is to solve the task. Finally, in the Output Gate, the output of the LSTM model is calculated in the Hidden State. To do this, the *sigmoid* function decides what information can come through the output gate and then the cell state is multiplied after it is activated with the *tanh* functio, [14]. The standard LSTM Architecture is represented in Fig. 5.

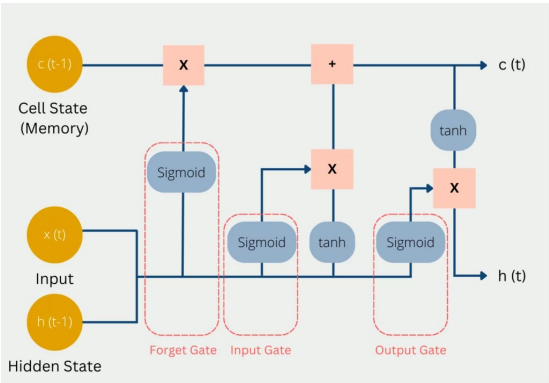

Fig. 5: The standard LSTM architecture [14]

A more deep explanation about LSTM's can be found in [15].

It is important to mention that there are several types of LSTMs based on the number and types of input variables used in the model but, in this project, we only used Univariate and Multivariate LSTMs.

The main difference between this two types of LSTMs is the number of input variables used in the model. In a univariate LSTM, only a single variable is used as input, while in a multivariate LSTM multiple variables are used as input.

Univariate LSTMs are typically used for time series forecasting when the goal is to predict the future value of a single variable based on its historical values. They are simpler to design, train and interpret than multivariate LSTMs, and are useful when the focus is on understanding the relationship between a single variable and its future values.

In this project we've used five different types of univariate LSTM.

- LSTM Vanilla: A vanilla LSTM has only a single hidden layer of LSTM units which the output layer is used to make a prediction.
- Stacked LSTM: The stacked LSTM has multiple hidden LSTM layers that are stacked one on top of another.
- Bidirectional LSTM: Bidirectional LSTM consists in learning the input sequence, both forward and backwards, and concatenate both interpretations. This is beneficial for sequence prediction problems.
- CNN LSTM: Convolutional Neural Network (CNN) is a Neural Network developed for working with two-dimensional image data. CNN can be used in a hybrid model with an LSTM where CNN is used to interpret sub-sequences of inputs that together are provided as a sequence to a LSTM model to interpret.
- ConvLSTM: ConvLSTM is related to the previous one, the CNN LSTM. This one was developed for reading two-dimensional spatial-temporal data, but can be adapted for use with univariate time series forecasting.

On the other hand, multivariate LSTMs are used when multiple variables are believed to affect the outcome or when the goal is to understand the relationship between multiple variables. These models can be more complex than univariate models, but they allow for a more in-depth analysis of the data and can provide more accurate predictions. In this subject, makes all the sense to study multivariate LSTMs because, as each company wants to sell the most, they need to keep track with the price of the product in others companies, which means that the price of a product in a specific company, depends on the state of the market.

One multivariate LSTM model is the Multiple Input Series. This model, as opposed to the univariate LSTM, receives multiple inputs (data). This multiples inputs are, in our case, the price of the same product, but in multiple companies and with this, we can use this model to predict the price of the product in one specific company, while taking in consideration the price in the other companies.

The code to implement all the LSTMs models, used in this

project, was based on the implementation by Jason Brownlee, which can be found in [16].

C. GRU

Gated Recurrent Unit (GRU) also aims to solve the vanishing gradient problem which comes with a standard RNN and, because of this, can also be considered as a variation on the LSTM. To solve this vanishing gradient problem, GRU uses an update and a reset gate. Basically, these are two vectors which decide what information should be passed to the output, [17]. This two vectors can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction. The standard GRU architecture is represented in Fig.6.

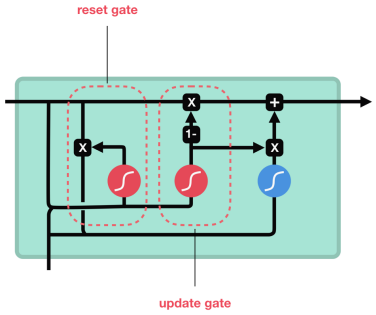

Fig. 6: The standard GRU architecture, [18]

To understand more about GRU we recommend the following [19] and it is important to mention that to implement this model we took inspiration from [20].

D. ARIMA

The ARIMA (Auto Regressive Moving Average) model is a very common time series-forecasting model. It is a more sophisticated extension of the simpler ARMA (Auto Regressive Moving Average) model, that is composed by two components: Auto Regressive (AR), and the Moving Average (MA).

Auto Regressive (AR) regression model is built on top of the auto-correlation concept, where the dependent variable depends on the past values of itself. This model got a parameter p, called the lag order which indicates the number of prior lag observations we include in the model.

The Moving Average (MA) model attempts to reduce the noise in our time series data by performing some sort of aggregation operation to your past observations in terms of residual errors, from the aggregation function. In this model we have another parameter, q, that is identical to p, but instead of identifying the time window (p) to the time series data itself, q specifies the time window for the moving average's residual error.[21]

ARMA models only work with stationary time series, which are series with statistical properties, such as mean and variance, that do not change over time. Unfortunately, majority of real world time series are not stationary, and thus they must often be transformed in order to make them stationary. The process of transformation is referred to as integration. Due to this, the ARIMA model has one more component than the ARMA model, this being the Integrated part (I).[22]

This part of the ARIMA model attempts to convert the non-stationary nature of the time-series data to something a little bit more stationary, by performing prediction on the difference between any two pair of observation rather than directly on the data itself. We can perform the operations of differentiation at levels of many times, depending on the hyper-parameter d that we set when training the ARIMA model.

For this model, the implementation developed is based on [23] and, for a more deep overview about ARIMA models we recommend [24].

E. SARIMA framework

SARIMA stands for Seasonal-ARIMA and it includes seasonality contribution to the forecast. Seasonality refers to the recurring patterns or trends in data that occur at specific times of the year and can have a significant impact on the accuracy of the predictions, and ARIMA fails to encapsulate that information implicitly. In the SARIMA model the Autoregressive (AR), Integrated (I), and Moving Average (MA) parts of ARIMA model remain and we add a part of Seasonality, which adds robustness to the SARIMA model. With this, this model is composed by 4 new parameters, as shown in Fig. 7, using the uppercase notation for the seasonal parts of the model, and lowercase notation for the non-seasonal parts of the model.

$$SARIMA \underbrace{(p,d,q)}_{non-seasonal} \underbrace{(P,D,Q)_m}_{seasonal}$$

Fig. 7: SARIMA parameters [25]

In this figure, m is the number of time steps for a single seasonal period. As for (P, D, M) this represents the Seasonal auto-regressive, difference and moving average order, respectively, [26]. In addition to this 7 parameters, SARIMA has one more optional parameter, the trend, t, which is responsible for controlling the deterministic trend polynomial. This parameter can be any of the following values $\{'n','c','t','ct'\}$ [27].

For this model, our recommendation for a more deep understanding is [28]. If you are interested, we developed the code taking inspiration from [29].

F. Facebook's Prophet

When a forecasting model doesn't run as planned, we want to be able to tune the parameters of the method with regards to the specific problem at hand. Tuning these methods requires a thorough understanding of how the underlying time series models work, so it can be difficult to a typical analyst to know how to adjust these or it may take a lot of time to do so. This is where Prophet package comes handy as it provides intuitive parameters which are easy to tune, making someone who

lacks expertise in forecasting models able to make meaningful predictions for a variety of problems [30].

At its core is the sum of three functions of time plus an error term: growth $g(t)$, seasonality $s(t)$, holidays $h(t)$, and error ϵ_t. The growth function has three main options: Linear, Logistic Growth or Flat (no growth over time). The seasonality function is simply a Fourier Series as a function of time and Prophet will automatically detect an optimal number of terms in the series, also known as the Fourier order. We can also choose between additive and multiplicative seasonality. Finally, the holiday function allows Facebook Prophet to adjust forecasting when a holiday or major event may change the forecast like Christmas, New Years, Black Friday, etc., [31]. As for the error term, ϵ_t, this represents any idiosyncratic changes which are not accommodated by the model, with ϵ_t being assumed as normally distributed, [32].

For a more deep understanding of this model we recommend the original paper for this model, [32]. Regarding our implementation, we inspired ourselves from [33] and [34].

G. Hyperparameters

For the RNN, LSTMs (Uni and Multivariate) and GRU models we gonna proceed to find the number of lag, i.e., the the number of prior observations we include in each step, that returns the lower mean absolute percentage error (**MAPE**). As for the ARIMA model we gonna proceed to find the 3 hyper-parameters p, d, q, while for the SARIMA model we gonna find the 8 hyper-parameters $(p, d, q); (P, D, Q, m); t$, assuming that the hyper-parameters are parameters that returns the lowest **MAPE**. Finally, for the Facebook's Prophet this model has 4 parameters to be tuned, $changepoint\_prior\_scale$, $seasonality\_prior\_scale$, $holidays\_prior\_scale$ and $seasonality\_mode$. As it wasn't refer in which country the prices of the product where taken we decide to not tune the $holidays\_prior\_scale$ parameter setting it to the default value (10.0) which basically applies no regularization, [35]. Therefore, we are only going to find which values for the other 3 parameters returns the lowest **MAPE**.

Although, for the ARIMA model, we are going to analyze all combinations of the (p, d, q) parameters to find which are the hyper-parameters, there are other ways to do so that are less computationally expensive. One example is the autocorrelation analysis, which can help detect patterns and check for randomness. For this, it's normally used the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) to figure out the order of AR(p) and MA(q), [36], and after only vary the value d. This graphics are represent in Fig. 26 and 27, respectively, in the Appendix B. Another well known way to find the hyper-parameters of an ARIMA model is the Hyndman-Khandakar algorithm which models the ARIMA model in a automatic form. More about this algorithm can be found in [37].

IV. Results

Although it's more common to use a 80:20 ratio we have decide to use, for every model, a 60:40 split in order to test our model against a higher variation of values in the test data. This split means that 60% of the data is for model training and 40% for the testing part. The graphical representation of this split is illustrated in Fig. 8.

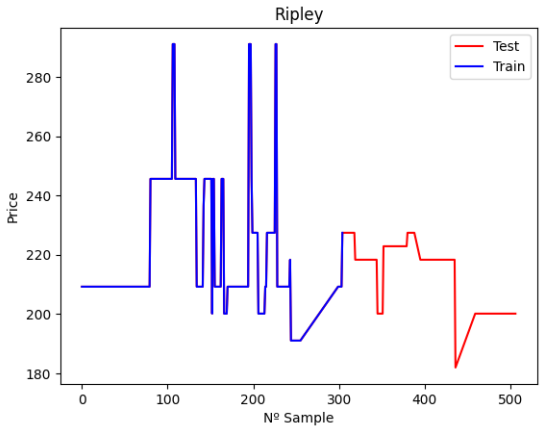

Fig. 8: 60:40 Train:Test split ratio on our data

For the first 10 models the lag, p, was set as $p \in [3, 4, ..., 20]$. As referred in III-G, the lag is the only parameter to tune for the first 8 models while the ARIMA and SARIMA models have 3 and 7 , respectively, with the lag being only one of them. Then, for both models we vary p as $[3, 4, ..., 20]$, d as $[0, 1]$ and q as $[0, 1, 2]$ and, for SARIMA, we also vary P as $[0, 1, 2]$, D as $[0, 1]$, Q as $[0, 1, 2]$, m as $[0]$ and t as $\{'n', 'c', 't', 'ct'\}$.

Finally, for Facebook's Prophet we vary $'changepoint\_prior\_scale'$ as $[0.001, 0.01, 0.1, 0.5]$, $'seasonality\_prior\_scale'$ as $[0.01, 0.1, 1.0, 10.0]$ and $'seasonality\_mode'$ as $['additive', 'multiplicative']$. For this model we use cross validation to evaluate all parameters, defining $initial =' 100\ days'$, $period =' 1\ days'$, $horizon =' 20\ days'$. [35]

To evaluate each model we have calculated the corresponding mean absolute percentage error (**MAPE**), mean squared error (**MSE**) and the mean Absolute Error (**MAE**), that can be calculated, respectively, as:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - P_i}{A_i}\right|$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(A_i - P_i\right)^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|A_i - P_i\right|$$

where A is the actual value, P the predicted value and n the sample size.

The results obtain for this metrics for each model is represented in Table II, with the green and red highlights showing the best and second best parameter, respectively, for each metric. As for the worst results for this metrics, this are highlighted as yellow, for the worst, and as blue, for the second worst. In this table is also represented the corresponding hyper-parameter(s) for each model.

TABLE II: Results of each model with hyper-parameters with respective Mean absolute percentage error (**MAPE**), Mean squared error (**MSE**) and Mean absolute error (**MAE**)

Model	Best Parameters	**MAPE**	**MSE**	**MAE**
RNN	3	0.007	15.237	1.513
LSTM Vanilla	3	0.005	11.894	1.053
Stacked LSTM	4	0.006	12.418	1.146
Bidirectional LSTM	4	0.006	12.658	1.183
CNN LSTM	6	0.012	22.335	2.398
ConvLSTM	3	0.007	17.482	1.346
LSTM Multivariate	5	0.027	73.5054	5.609
GRU	19	0.007	13.813	1.443
ARIMA	(3,1,0)	0.004	12.164	0.831
SARIMA	[(3,1,0); (0,0,0,0); 'n']	0.004	12.174	0.833
Facebook's Prophet	(0.1, 1.0, 'additive')	0.078	542.149	17.403

As we can see in Table II, the two best models, i.e, the ones with the lowest **MAPE**, are SARIMA and ARIMA, with ARIMA being considered the best as it has a lower **MSE**. Although it wasn't the the model with the best **MAPE** it is possible to see that LSTM Vanilla was the model with the lowest **MSE** what leads us to conclude that this model would also be a good approach to forecast this time series. As for the worst models we see that the LSTM Multivariate and the Facebook's Prophet were the ones with the higher **MAPE** and **MSE**, being Facebook's Prophet the worst model in this project.

If we focus now on the hyper-parameters of the ARIMA and SARIMA models we see that the non-seasonality part is the same and the seasonality part of the SARIMA is 0 in every parameter. This means that this SARIMA is actually equal to the ARIMA model. Now, if we pay attention to the d parameter we can conclude that both this models are equal to a ARMA(3,0) model after 1-order differences series. As this model doesn't include a MA component it can be assumed as an AR(3) model, with the following equation:

$$Y_t = c + \sum_{q=1}^{3}(\phi_q \times Y_{t-q}) + \epsilon_t \Leftrightarrow \quad (1)$$

$$\Leftrightarrow Y_t = c + \phi_1 \times Y_{t-1} + \phi_2 \times Y_{t-2} + \phi_3 \times Y_{t-3} + \epsilon_t \quad (2)$$

Where Y_t is the price of the product at time t, c is a constant term, ϕ_1, ϕ_2, and ϕ_3 are the autoregressive coefficients, and ϵ_t is white noise error term.

As the SARIMA and ARIMA models are equivalent, we are going to ignore the SARIMA model, only analyzing the results of the ARIMA.

It's also a good forecasting practice to analyze the graphics that represent the loss function and the prediction of each model. For the ARIMA model the graphic that represents the prediction versus actual value is represented in Fig. 9. As for the LSTM Vanilla model this graphic and the loss function one can be found in Fig. 10 and 11, respectively. For the Facebook's Prophet model, the prediction versus actual values graphic is represent in Fig.16, and for the other models, the prediction versus actual values and loss function graphics can be found in the Appendix, section C and D, respectively.

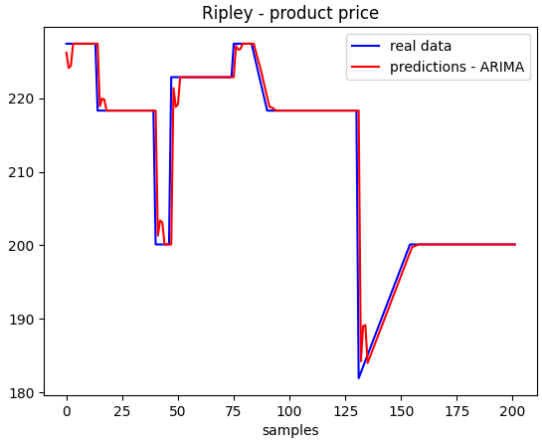

Fig. 9: Prediction graphic of ARIMA.

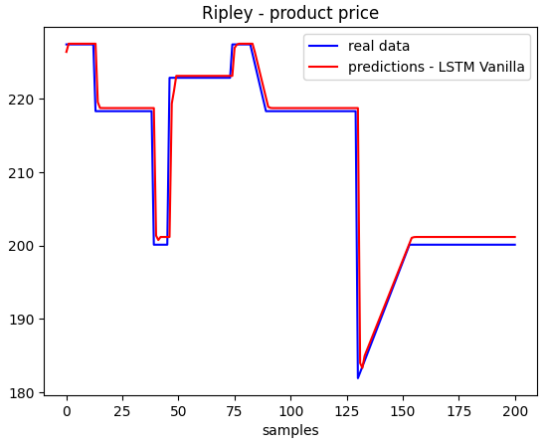

Fig. 10: Prediction graphic of the LSTM Vanilla.

For the prediction against the actual value graphics, represented in section C of the Appendix, it is possible to verify that in the vast majority the predict values are very close to the real/actual values, although, there are cases that presents a considerable gap between the two values even with a similar behavior. There is also some cases where the price has high variation and some of the models struggle to predict this.

As for the graphics that represent the loss function, section D of the Appendix, it's important to see that all the models converge. For this to happened we use one of the *keras callbacks*, the *EarlyStopping* and we define it to stop the compilation of the model when the loss of the validation (test) data hasn't improve, i.e., lowered, in 10 consecutive epochs.

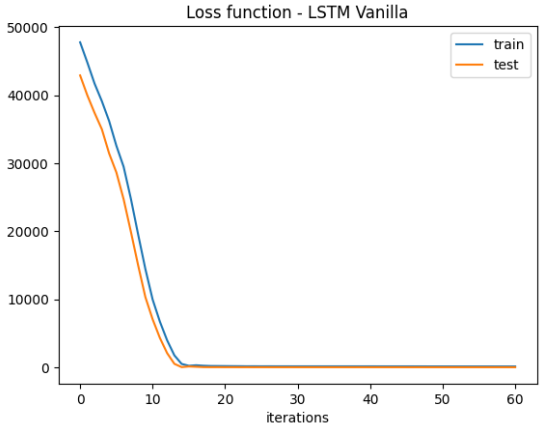

Fig. 11: Loss graphic of the LSTM Vanilla.

A. *Facebook's Prophet*

Although Facebook's Prophet was the worst model having a higher **MAPE** and **MSE**, it is interesting to show the graphics that were obtained, to show how powerful and helpful this model can be in the study of time series.

We can apply cross-validation to see how the **MAPE** varies depending on the days, as it is illustrated in Figure 12, being possible with this to notice that **MAPE** tends to increase as the number of days gets higher.

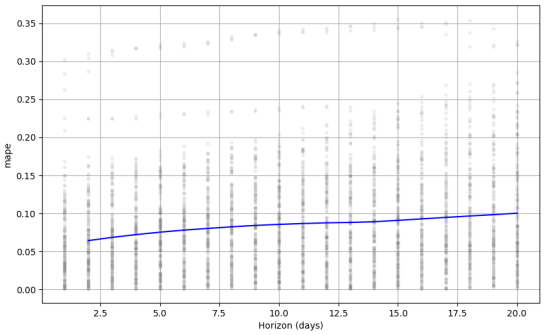

Fig. 12: **MAPE** behavior in function to the number of days.

In Figure 13 it is represented the price variation in function to the time and there are plotted the actual price, the model prediction, the uncertainty interval and the train and test curves, where we can see that the test set has a lot of values outside the uncertainty interval, indicating a bad performance by the model.

One interesting functionality of Prophet is that the model is capable of finding the points where the growth rate changes and plot them, as shown by the red vertical lines in Figure 14.

This model can also represent the trend and the weekly variation which can be very helpful for a more in deep study on the behave of the price of the product throughout the time and in each day of the week, Figure 15.

Finally, the graphic that best illustrates why this model was the worst one, in Figure 16 is represented the price prediction

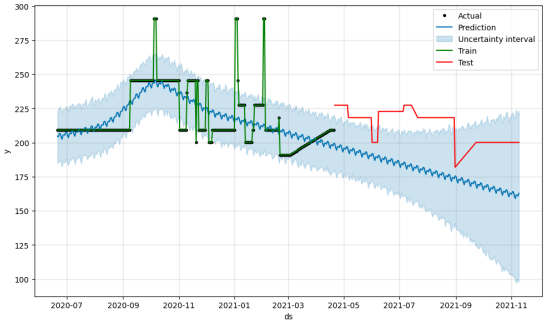

Fig. 13: Price variation in function to time.

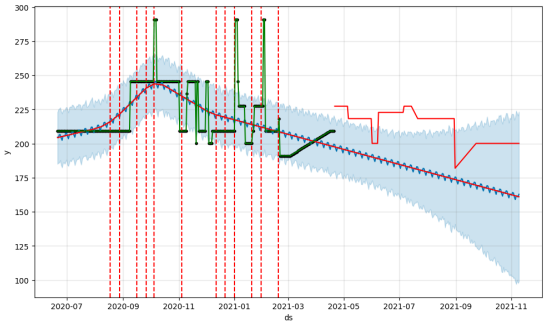

Fig. 14: Price variation in function to time with marked change points.

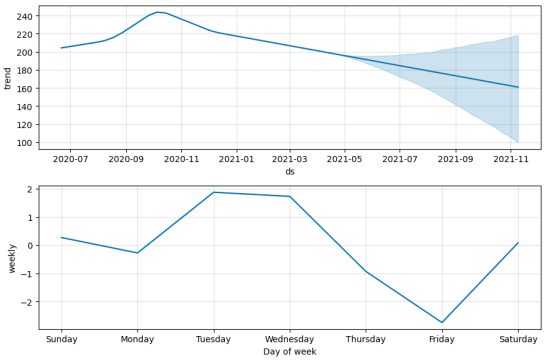

Fig. 15: Trend and Weekly variation in function of time.

against the actual price throughout the months in the test set. As it is possible to see, there is a big difference between the two, where the model predicts a decrease in the product price while, in reality, this has an almost constant behavior, only with some small variations over time. This difference is what translates into a big **MAPE**.

B. *Comparing with other Articles*

In order to understand and be aware if the results obtained were good, it is important to compare our work with other works.

Since we don't know the origin of the data-set, we can't compare our results directly but, we can compare with results

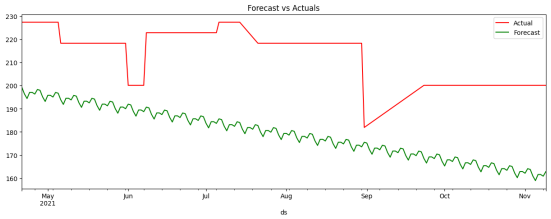

Fig. 16: Prediction graphic of Facebook's Prophet.

from other articles, in the same area of research, and see if the graphs that we've got for the loss and, perhaps, the prediction have a a relatively similar behavior.

As we assume, the loss curve must converge after a certain number of iterations. Ogulcan Ertunc applied LSTM to average monthly temperatures from Florida since 1743[38]. Like us, he prepared his data and divided in training and test and plotted the loss of each one, being possible to verify that the behavior is the same as that of the results that we obtained for the loss curves, that is convergence after a certain number of iterations. The author does not specify which LSTM model he has used.

Soubhik Khankary used LSTM RNN in multivariate time series with the objective of forecasting Open prices for stock of company 'XYZ' which is dependent on multiple other features like 'High', 'Low' and 'Close' price [39]. The author has also divided the data in training and test making a prediction in the end. The results he got by plotting the actual and predicted values have a similarly behavior to the results gotten in this project.

So, after comparing our results with others in the area, we came to the conclusion that our implementations are working as we expected.

V. Conclusion & Future Work

The objective of this project was performing forecast on product pricing for a specific company. For that, it was provided a data-set that consists in three different time series containing information of different products.

To perform the forecasting, we've applied different models such as RNN, Univariate and Multivariate LSTM, GRU, ARIMA, SARIMA and Facebook Prophet. After analysing all the results we came to conclusion that the models that had the best results, i.e, produced the lowest **MAPE**, were SARIMA and ARIMA, which, according to their hyper-parameters, are, in this case, equal models as the seasonality part of SARIMA is all 0 and the non-seasonality parameters are equal to the parameters of the ARIMA model. On the other hand, the models which the results were the worst were the LSTM Multivariate and the Facebook's Prophet, having extremely high **MAPE** when comparing to the other models.

With this, we can say that the objective of this project was accomplished successfully since the models applied worked well and we were able to implemented them to the given data-set, analysing the different results and choosing the best ones.

Since it's always possible to improve the performance of the machine learning models, the future work could focus on a more rigorously hyper-parametization, such as finding the best train:test split ratio or study for a higher range of lag, instead of only values between 3 and 20. For the RNN, LSTM's (Uni and Multivariate) and GRU models we can also try to find the number of NN units that maximizes the quality of the results of each model. Finally, as the bigger and more complete the data-set used is, the more accurate the results and, as the data used in this project contains only 507 values, which many of them are missing values, a future work could focus on studying this project to a more complete and bigger data set.

References

[1] New Tech Forum By Anais Dotis-Georgiou and Anais Dotis-Georgiou. *An introduction to time series forecasting*. July 2021. URL: https://www.infoworld.com/article/3622246/an-introduction-to-time-series-forecasting.html.

[2] Darryl Jarman, Zhi Quan Zhou, and Tsong Chen. "Metamorphic Testing for Adobe Data Analytics Software". In: May 2017, pp. 21–27. DOI: 10.1109/MET.2017.1.

[3] *Time series forecasting: Definition, applications, and examples*. URL: https://www.tableau.com/learn/articles/time-series-forecasting.

[4] André Fernandes & Gonçalo Freitas. "AIA-Project2". In: *GitHub repository* (2023). URL: https://github.com/gjfreitas/AIA-Project2.

[5] Satyam Kumar. *4 techniques to handle missing values in time series data*. Apr. 2022. URL: https://towardsdatascience.com/4-techniques-to-handle-missing-values-in-time-series-data-c3568589b5a8.

[6] *Pandas 1.5.3 documentation - Pandas.dataframe.interpolate*. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.interpolate.html.

[7] Mehreen Saeed. *An introduction to recurrent neural networks and the math that powers them*. Nov. 2022. URL: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/.

[8] Aman Singh. *Recurrent neural networks : Introduction for beginners : Introduction for beginners*. June 2021. URL: https://www.analyticsvidhya.com/blog/2021/06/recurrent-neural-networks-introduction-for-beginners/.

[9] Pranoy Radhakrishnan. *Introduction to recurrent neural network*. Jan. 2018. URL: https://towardsdatascience.com/introduction-to-recurrent-neural-network-27202c3945f3.

[10] Robert DiPietro and Gregory D. Hager. "Deep learning: RNNs and LSTM". English (US). In: *Handbook of Medical Image Computing and Computer Assisted Intervention*. Elsevier, Jan. 2019. Chap. 21, pp. 503–519. DOI: 10.1016/B978-0-12-816176-0.00026-0.

[11] Mehreen Saeed. "Understanding Simple Recurrent Neural Networks in Keras". In: *Machine Learning Mastery* (Sept. 2022). DOI: https://machinelearningmastery.com/understanding-simple-recurrent-neural-networks-in-keras/.

[12] *Understanding LSTM networks*. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[13] CLIMB. "20 Long Short-Term Memory Interview Questions and Answers". In: *PROPHET* CLIMB (Aug. 2022). URL: https://climbtheladder.com/long-short-term-memory-interview-questions/.

[14] *Long short-term memory networks (LSTM)- simply explained!* Oct. 2022. URL: https://databasecamp.de/en/ml/lstms.

[15] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. "A Review on the Long Short-Term Memory Model". In: *Artificial Intelligence Review* 53 (Dec. 2020). DOI: 10.1007/s10462-020-09838-1.

[16] Jason Brownlee. "How to Develop LSTM Models for Time Series Forecasting". In: *Machine Learning Mastery* (Nov. 2018). URL: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/.

[17] Simeon Kostadinov. *Understanding GRU networks*. Nov. 2019. URL: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

[18] Michael Phi. *Illustrated guide to LSTM's and GRU's: A step by step explanation*. June 2020. URL: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[19] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[20] Niousha Rasifaghihi. "Predictive Analytics: Time-Series Forecasting with GRU and BiLSTM in TensorFlow". In: *Medium* (Aug. 2020). URL: https://towardsdatascience.com/predictive-analytics-time-series-forecasting-with-gru-and-bilstm-in-tensorflow-87588c852915.

[21] Juan Nathaniel. *Introduction to arima for Time Series forecasting*. Aug. 2021. URL: https://towardsdatascience.com/introduction-to-arima-for-time-series-forecasting-ee0bc285807a.

[22] Lleyton Ariton. *A thorough introduction to Arima models*. Jan. 2021. URL: https://medium.com/analytics-vidhya/a-thorough-introduction-to-arima-models-987a24e9ff71.

[23] Jason Brownlee. "How to Grid Search ARIMA Model Hyperparameters with Python". In: *Machine Learning Mastery* (Jan. 2017). URL: https://machinelearningmastery.com/grid-search-arima-hyperparameters-with-python/.

[24] Tian Jie. *Time series analysis-Arima based models*. Sept. 2021. URL: https://towardsdatascience.com/time-series-analysis-arima-based-models-541de9c7b4db.

[25] Aayush Bajaj. *Arima & Sarima: Real-world time series forecasting*. Nov. 2022. URL: https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide.

[26] Jason Brownlee. *A gentle introduction to sarima for time series forecasting in Python*. Aug. 2019. URL: https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/.

[27] *Statsmodels - SARIMAX*. URL: https://www.statsmodels.org/dev/generated/statsmodels.tsa.statespace.sarimax.SARIMAX.html.

[28] Tony Yiu. *Understanding sarima*. Sept. 2021. URL: https://towardsdatascience.com/understanding-sarima-955fe217bc77.

[29] Jason Brownlee. "How to Grid Search SARIMA Hyperparameters for Time Series Forecasting". In: *Machine Learning Mastery* (Oct. 2018). URL: https://machinelearningmastery.com/how-to-grid-search-sarima-model-hyperparameters-for-time-series-forecasting-in-python/.

[30] Ankit Choudhary. *Time series forecasts using Facebook's prophet*. June 2022. URL: https://www.analyticsvidhya.com/blog/2018/05/generate-accurate-forecasts-facebook-prophet-python-r/.

[31] Mitchell Krieger. *Time series analysis with Facebook Prophet: How it works and how to use it*. Feb. 2022. URL: https://towardsdatascience.com/time-series-analysis-with-facebook-prophet-how-it-works-and-how-to-use-it-f15ecf2c0e3a.

[32] Sean Taylor and Benjamin Letham. *Forecasting at scale*. Sept. 2017. DOI: 10.7287/peerj.preprints.3190v2.

[33] Minkyung's blog. "How To Use FB Prophet for Time-series Forecasting: Vehicle Traffic Volume". In: *Minkyung's blog* (Dec. 2020). URL: https://mkang32.github.io/python/2020/12/15/prophet-intro.html#train.

[34] The PROPHET paper. "Cross validation". In: *PROPHET* (). URL: https://facebook.github.io/prophet/docs/diagnostics.html.

[35] *Diagnostics*. Sept. 2022. URL: https://facebook.github.io/prophet/docs/diagnostics.html.

[36] Leonie Monigatti. *Interpreting ACF and PACF plots for time series forecasting*. Sept. 2022. URL: https://towardsdatascience.com/interpreting-acf-and-pacf-plots-for-time-series-forecasting-af0d6db4061c.

[37] Rob J. Hyndman and Yeasmin Khandakar. "Automatic Time Series Forecasting: The forecast Package for R". In: *Journal of Statistical Software* 27.3 (2008), pp. 1–22. DOI: 10.18637/jss.v027.i03. URL: https://www.jstatsoft.org/index.php/jss/article/view/v027i03.

[38] Ogulcan Ertunc. *3- Time Series Forecasting Using LSTM*. June 2021. URL: https://medium.com/analytics-vidhya/3-time-series-forecasting-using-lstm-e14b93f4ec7c.

[39] Soubhik Khankary. *Multivariate Time Series Forecasting using RNN(LSTM)*. Jan. 2022. URL: https://medium.com/mlearning-ai/multivariate-time-series-forecasting-using-rnn-lstm-8d840f3f9aa7.

APPENDIX

A. *Price per company*

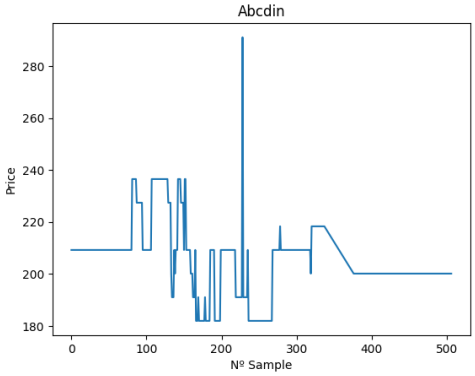

Fig. 17: Price sample of time for the company *Abcdin* after using the interpolation

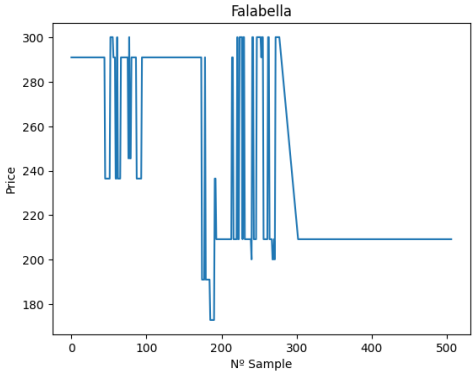

Fig. 18: Price sample of time for the company *Falabella* after using the interpolation

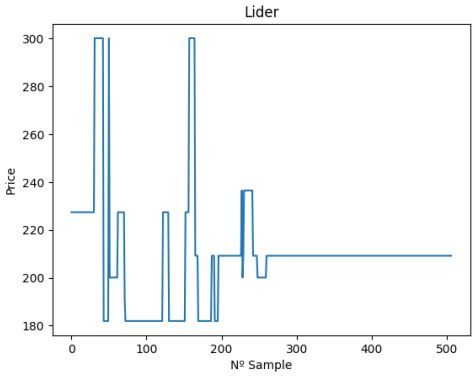

Fig. 19: Price sample of time for the company *Lider* after using the interpolation

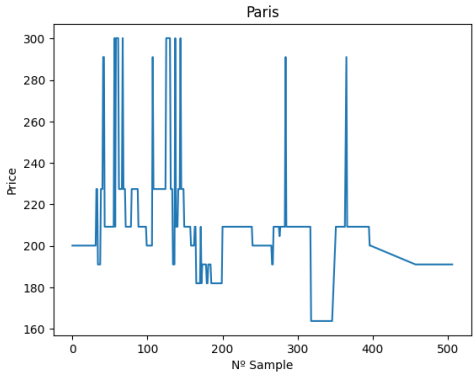

Fig. 20: Price sample of time for the company *Paris* after using the interpolation

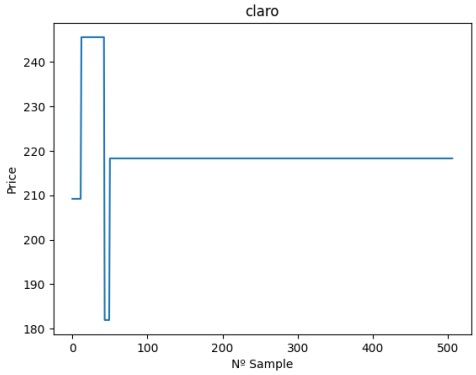

Fig. 21: Price sample of time for the company *Claro* after using the interpolation

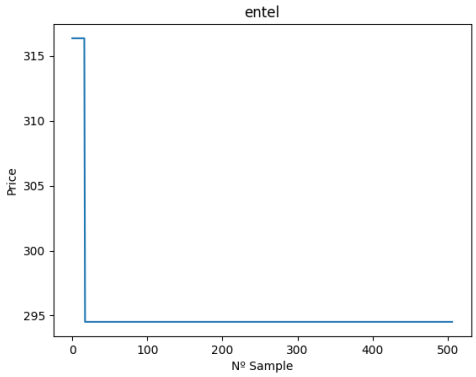

Fig. 22: Price sample of time for the company *Entel* after using the interpolation

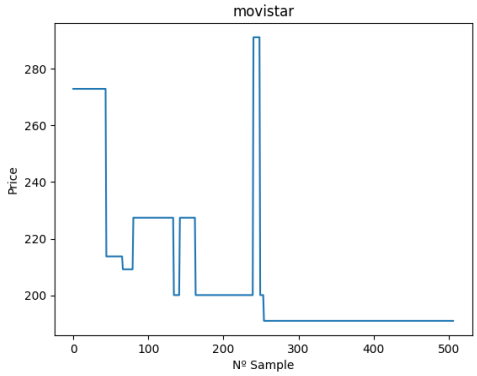

Fig. 23: Price sample of time for the company *Movistar* after using the interpolation

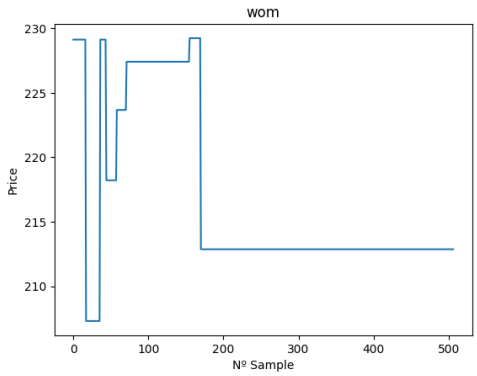

Fig. 24: Price sample of time for the company *Wom* after using the interpolation

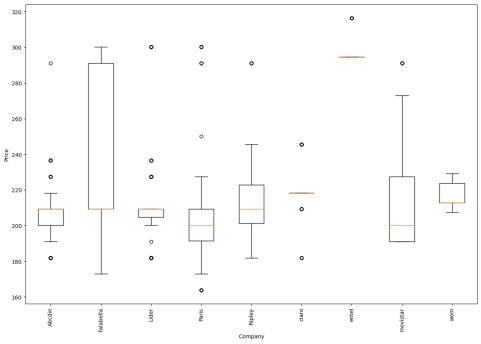

Fig. 25: Box-plot of the data after using the interpolation

B. ACF & PACF

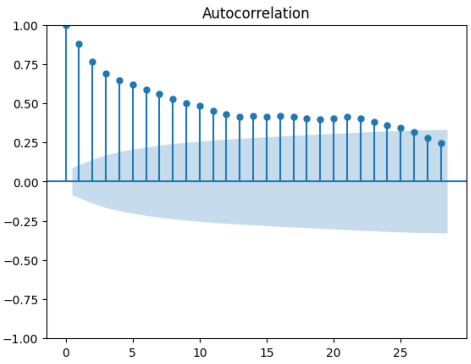

Fig. 26: Autocorrelation Function (ACF) of the values for the product in company *Ripley*

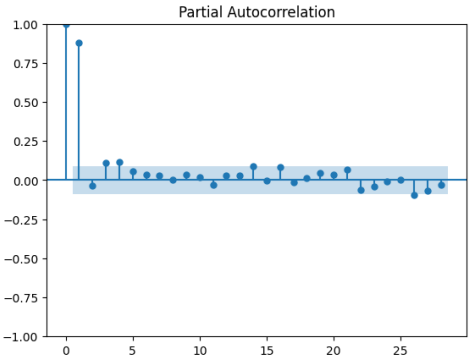

Fig. 27: Partial Autocorrelation Function (ACF) of the values for the product in company *Ripley*

C. Predictions vs Actual Value

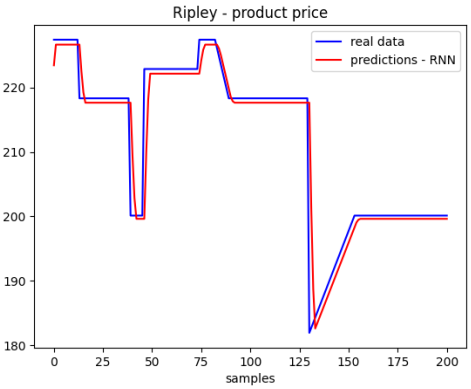

Fig. 28: Prediction graphic of RNN.

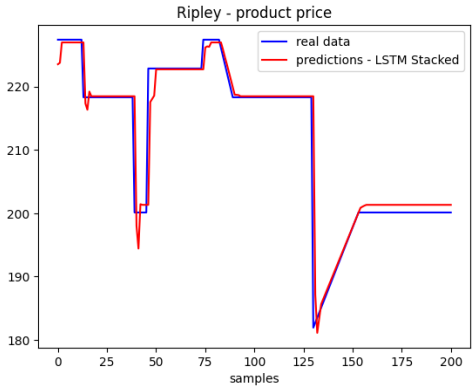

Fig. 29: Prediction graphic of the LSTM Stacked.

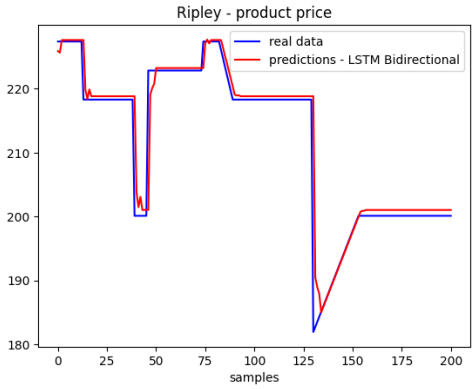

Fig. 30: Prediction graphic of the LSTM Bidirectional.

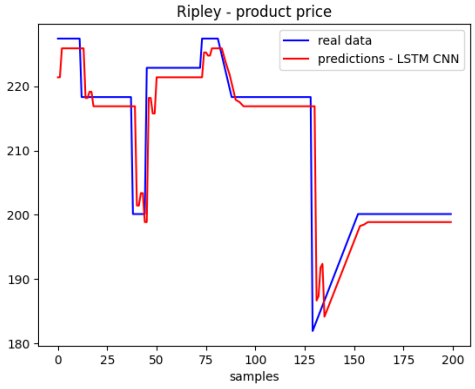

Fig. 31: Prediction graphic of the LSTM CNN.

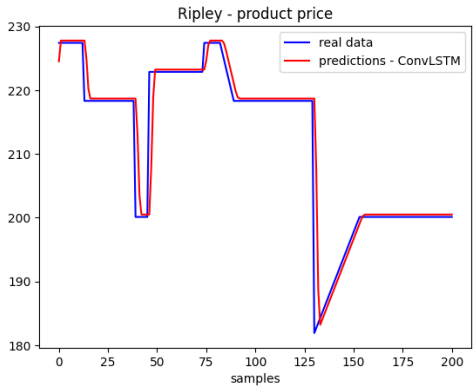

Fig. 32: Prediction graphic of the LSTM Conv.

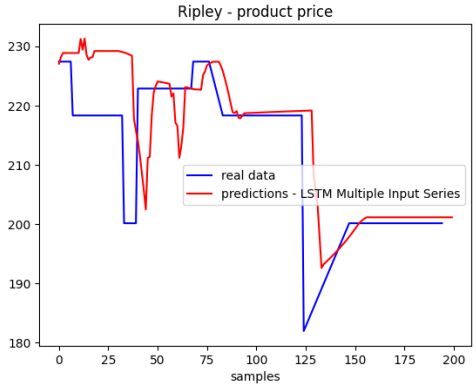

Fig. 33: Prediction graphic of the LSTM Multiple Input Series.

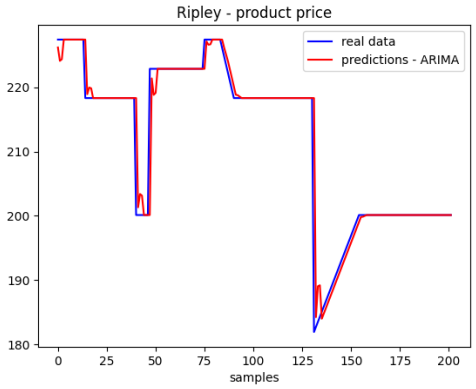

Fig. 35: Prediction graphic of SARIMA.

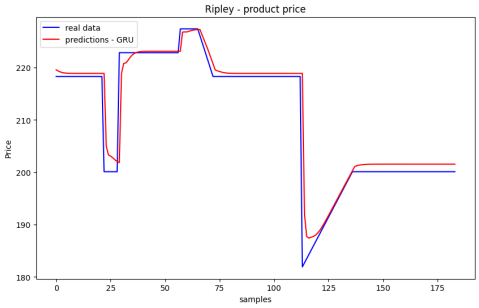

Fig. 34: Prediction graphic of GRU.

D. Loss Functions

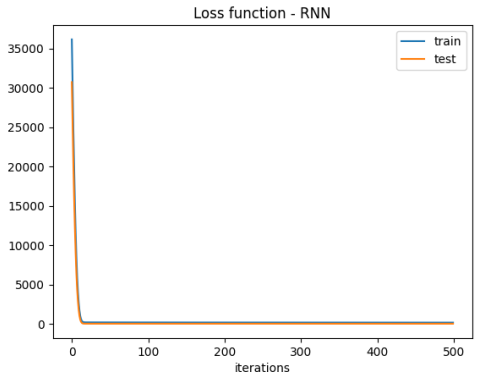

Fig. 36: Loss graphic of the RNN.

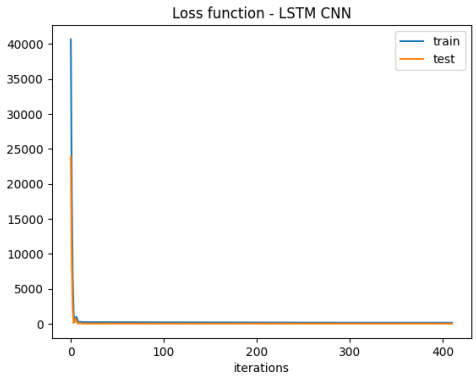

Fig. 39: Loss graphic of the LSTM CNN.

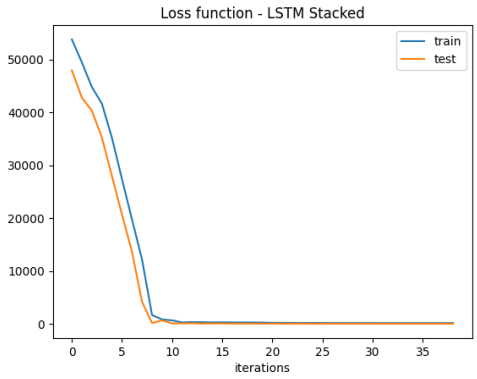

Fig. 37: Loss graphic of the LSTM Stacked.

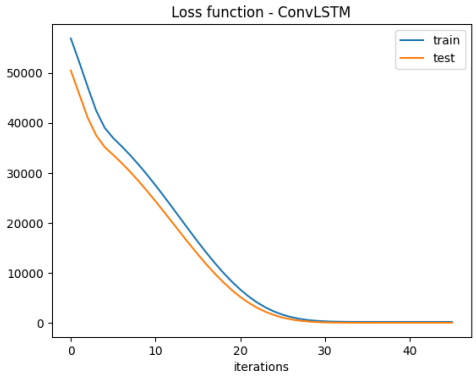

Fig. 40: Loss graphic of the LSTM Conv.

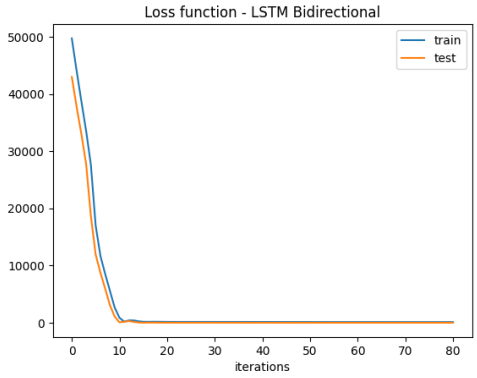

Fig. 38: Loss graphic of the LSTM Bidirectional.

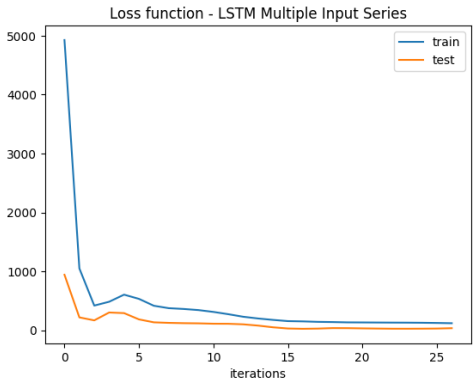

Fig. 41: Loss graphic of the LSTM Multiple Input Series.

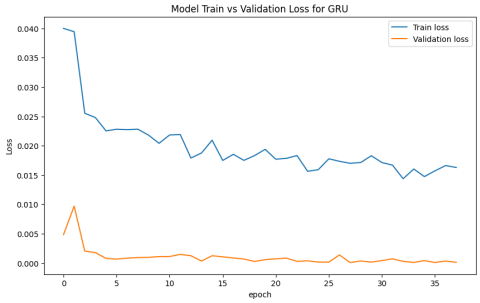

Fig. 42: Loss graphic of the GRU.