Time Series Forecast with Neural Networks

Zijian Wang (*40011864*)   Matheus Nogueira (*40220168*)

Abstract

This project aims to implement and compare various methods for Time Series Forecasting, since the classical ones, such as ARIMA and GARCH until Recurrence Neural Networks and Prophet. The goal is to evaluate if the later, and more complex methods, necessarily result in better forecasting, with respect to the RMSE metric or if simpler methods are precise enough to be chosen for either stationary and non-stationary financial series.

1. Introduction

Implementing precise models for Time Series Forecasting is crucial to various areas of knowledge, being fundamental to Financial Analysis of Stock Returns and Currency, which are types of time series chosen to be worked with. For that purpose, one could say that is a difficult task to chose between the huge amount of different methods, since each of them have its own advantages and disadvantages. For instance, simple linear models such as ARIMA are easy to understand and implement, but may be too simple and inflexible to model Stock Returns. On the other hand, MLPs and Recurrence Networks may present themselves with precise forecasts but with a complex theory behind and possible training difficulties may arise - over fitting is a classic example.

With being said, this project's goal is to implement different time series methods for 3 financial time series, all them with a stationary and non-stationary form, so that one can compare the quality of each model prediction.

2. Methodology & Experimental Results

Three financial time series were chosen to be used in this project. All of them have weekly frequency with time period from 2010 to 2019 and can be found on `finance.yahoo.com/`. One of the series is the stock price of American Airlines while the two others are currency rates of American Dollars with Canadian Dollars and Brazilian Real:

- American Airlines Group Inc. (AAL)
- USD/CAD (CAD=X)
- USD/BRL (BRL=X)

Once the series are chosen, the first step of the methodology was to develop a Time Series Analysis procedure, in order to gain information about each of them. With this analysis the goal was to gather information regarding seasonality, stationarity, trend, missing values and the number of differentiation needed to obtain a stationary behaviour. The results of this analysis is expressed on the table below.

	Trend	Statio	NumDiff	Season
ALL	Yes	No	1	No
USD/CAD	Yes	No	1	No
USD/BRL	Yes	No	1	No

In order to determine stationarity two methods were used: first, a visual analysis of each series. The American Airlines series, as shown in the image below, is clearly non stationary, for example, since it shows a clear trend over time and one can suspect that it's mean and variance are also not constant over time *(weak stationary, see [3])*. Furthermore, a *Augmented Dickey Fuller Test [3]* was also done so that the non-stationary could be statistically shown.

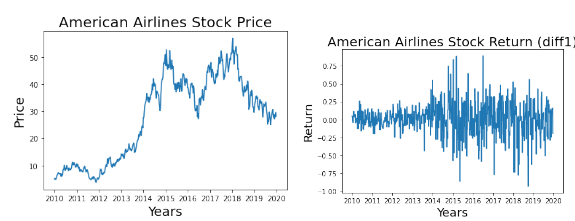

Figure 1. American Airlines Original Stock Price Series (left) and 1 time differentiated Return Series (right)

Even though the series have different behaviour over time, with respect to the analysis of interest all of them have identical results. With respect to the *NumDiff*, all of the series needed to be discretely differentiated one time to gain stationary behaviour and pass the *ADF Test*.

Once the analysis is complete, the next step was to implement each of the following methods for time series forecast:

1. ARIMA
2. ARIMA + GARCH

3. Random Forest Regressor
4. Support Vector Machine Regressor
5. Multi Layer Perceptron Neural Networks
6. Recurrence Neural Networks
7. Facebook Prophet Model

Each of the methods will be briefly explained in the following paragraphs, however the results are clustered together in the end of this section in the tables 2 and 2.

ARIMA and GARCH

The first methods to be implemented are classic linear models for time series forecasting, the Auto Regressive Integrated Moving Average Model and the Generalized Autoregressive Conditional Heteroskedasticity.

The first in constructed with 2 simpler models, the AR (Auto Regressive) and MA (Moving Average), which equations are shown below. This model claims that the present value of the series is linear dependent of the past values until a lag-p in the past as well as in the linear combination of the errors until a lag-q past value. In order to determine the number of past values to use in the model, one can use the Autocorrelation and Partial Autocorrelation Functions, as describer in [3] and shown below. The *Integrated* part of the name suggests the number of differentiation needed to gain stationary behaviour, since this is a requisite for the model itself.

$$y_t = \phi_0 + \sum_{i=1}^{p} \phi_i r y_{t-i} + \alpha_t + \sum_{i=1}^{q} \theta_i \alpha_{t-i} \qquad (1)$$

where y_t is the value of the time series in time step t and α_t is a white noise series.

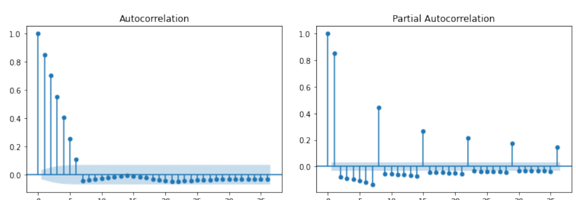

Figure 2. ACF and PACF American Airlines Stationary Series

The Garch model is used to describe the volatility (the variance) of the time series and it can integrate the ARIMA model by modeling the α_t white noise (error). The equation that defines a GARCH(m,s) is as follows:

$$\alpha_t = \sigma_t \epsilon_t, \sigma_t^2 = \alpha_0 + \sum_{i=1}^{m} \alpha_i \alpha_{t-i}^2 + \sum_{j=1}^{s} \beta_j \sigma_{t-j}^2 \qquad (2)$$

Random Forests

As known, a Random Forest is an ensemble method implemented with a collection of Decision Trees [1]. Although known better for classification, Decision Trees, and Random Forest, can be used for regression problems as well.

For a regression problem, each Decision Tree is constructed in a similar way that it would be in a classification problem. Given an tabular input shaped as a *moving window*, each feature is used to determine the best split based on the target value y_{t+1}. Once the best splits are determined and the tree is constructed, given a new input, the value of the prediction will be the average of the leaf values this new input reaches.

With respect to the moving window approach, as well as the other input format needed for each model, see the subsection **Input Formats** in the end of the model's introduction.

Support Vector Machines

SVMs for regression are implemented in a analogous way to classification SVMs. The regressor tries to fit the best line *hyperplane* within a predefined threshold error value *decision boundaries*. Instead of using the hyperplane to separate different classes, the *SVM Regressor* uses it to fit the behaviour of the time series.

Multi Layer Perceptrons

The MLP Networks for regression and classification are implemented and trained in the exact same way. The only difference is that the output layer, instead of being composed of n nodes representing n classes, it is made of a single node, which represents the predicted value y_{t+1} of the series. With that value in hand, the network uses back propagation and gradient descent to find the best weight for its connections, as it is done with a classification network. The input layer is made of the l past values of the series and, for this project, $l = 12$.

Recurrent Neural Networks

A recurrent neural network (RNN) is a type of artificial neural network commonly used in speech recognition and natural language processing. Recurrent neural networks recognize data's sequential characteristics and use patterns to predict the next likely scenario. RNNs are distinct from other types of artificial neural networks because they use feedback loops to process a sequence of data that that informs the final output. These feedback loops allow information to persist.

The RNN processes the sequence of vectors one by one.

While processing, it passes the previous hidden state to the next step of the sequence. The hidden states act as the neural network's memory, which holds information on previous data the network has seen before. Then the input and previous hidden state are combined to form a vector. The vector then goes through the activation which decides what to do with the data, and the output is the new hidden state.

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture. LSTMs were created as the solution to vanishing gradient problem due to short-term memory. In this project, we used a Vanilla LSTM, which is an LSTM model that has a single hidden layer of LSTM units, and an output layer used to get one prediction value yt+1. The ni is the number of the time steps for one LSTM cell/unit, and nf is the is the number of features which is 1 since we are working with uni-variate time series. Choosing a right value of ni would avoid you from over fitting with short amount of data and getting explosive predict value. u is the unit number of LSTM and e is the epochs. The bigger the u and e are, and the more data are used to train and fit model. To demonstrate the model, a batch which contains the value of the end of the training data series which has the same length with test series are used to predict the first value in the test series. Then the prediction is added into the end of current batch, and the first value was removed. Keep doing this until the batch is full of predictions. Then get the error by comparing the test series and the prediction series.

Prophet

The Prophet is a Facebook Open Source project for time series predictions based on an additive model that tries to sum together 3 different functions that tries to model trend $g(t)$, seasonality $s(t)$ and holiday/weekend effects $h(t)$. See [2].

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{3}$$

The Trend model, $g(t)$, can be implemented in two different ways, according to [2]. First, there is a model that assumes a linear growth over time without a saturation point. Secondly, there is a non linear growth with a carrying capacity as saturation point. The equations that define these models are, respectively:

$$g(t) = (k + a(t)^T \delta)t + (m + a(t)^T \gamma) \tag{4}$$

$$g(t) = \frac{C}{1 + exp(-k(t-m))} \tag{5}$$

where k is the growth rate, δ has the rate adjustment, m is the offset parameter, γ is a variable to make the function continuous and C is the carrying capacity.

The seasonality model is, basically, a Fourier Series that captures the periodic behaviour of the seasonal model.

$$s(t) = \sum_{n=1}^{N} \left(\alpha_n cos\left(\frac{2\pi nt}{P}\right) + \beta_n sin\left(\frac{2\pi nt}{P}\right)\right) \tag{6}$$

At last, the holidays/weekend model is implemented with a auxiliary table that list all the holidays and events. This table is used to add an indicator functions that tells if a time step t occurs during a holiday i and assign a parameter for each holiday which is the corresponding change in the forecast [2]. Note that the holiday model is not of large importance for this project since the frequency of the time series is weekly and the table presented above was not implemented.

Input Formats

The models implemented in this project needed two types of input formats, that is, two different ways to reshape the time series in order to implement the algorithms.

First, for *ARIMA*, *GARCH*, *RNN* and *Prophet*, the series were passed as input as the were originally, a pandas series, which is, basically, a list with the values of the series at each time step. Sometimes, instead of a pandas series, a pandas dataframe was needed so that the model had the data of each observation as index.

For *Random Forests*, *SVM* and *MLP* the series were reshaped into a moving window table. That means a original series with the format $[y_0, y_1, y_2, ..., y_t]$ was reshaped into the following table:

Past Steps				Y_{t}	Y_{t+1}
y_0	y_1	y_2	y_3	y_4	y_5
y_1	y_2	y_3	y_4	y_5	y_6
...	...	...	...	...	...
y_{t-5}	y_4	y_{t-3}	y_{t-2}	y_{t-1}	y_t

Each row is a window of the time series and the last column will be used as "target" for the supervised learning algorithms. Note that, from row i to row $i+1$ thee window is shifted one unit to the right. The Y_{t+1} column is used as target while the others are passed as features/inputs.

RMSE Tables

American Series Training RMSE		
Method	Statio	Non-statio
ARIMA	0.0880	X
GARCH	0.0880	X
RF	0.0487	0.0013
SVM	0.0609	0.0725
MLP	0.0577	0.0102
RNN		
Prophet	0.1253	0.0431

American Series Forecasting RMSE		
Method	Statio	Non-statio
ARIMA	0.1534	X
GARCH	0.1534	X
RF	0.0539	0.0099
SVM	0.0730	0.0965
MLP	0.0648	0.0050
RNN		
Prophet	0.0920	0.0758

3. Conclusions

COMPLETE!!!!

References

[1] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2

[2] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. 3

[3] Ruey S Tsay. *Analysis of financial time series*, volume 543. John wiley & sons, 2005. 1, 2

A. Helpful Instructions

This project contains a *README.md* file with **all** the instructions needed to understand the structure of the project's directories and files.

In order to see the same images for the remaining series, refer to *img directory* in the main directory of this project.

In order to see the all the jupyter notebooks implemented for this project, refer to *code directory* in the main directory of this project.

The *ref directory* has some of the bibliography used during the project and *data directory* has the time series in *csv files*.

B. Images and Results of remaining Time Series

This section, although optional for the reader, gives, first, the RMSE Tables for the two remaining time series and the images of all training and forecasts each series.

USD CAD Series Training RMSE		
Method	Statio	Non-statio
ARIMA	0.0001	X
GARCH	0.0001	X
RF	0.0475	0.0016
SVM	0.0604	0.0421
MLP	0.0548	0.0088
RNN		
Prophet	0.1205	0.0440

USD CAD Series Forecasting RMSE		
Method	Statio	Non-statio
ARIMA	0.0011	X
GARCH	0.0011	X
RF	0.0547	0.0079
SVM	0.0586	0.0463
MLP	0.0538	0.0043
RNN		
Prophet	0.1004	0.1302

USD BRL Series Training RMSE		
Method	Statio	Non-statio
ARIMA	0.0042	X
GARCH	0.0042	X
RF	0.0357	0.00087
SVM	0.0448	0.0734
MLP	0.0420	0.0066
RNN		
Prophet	0.0895	0.0377

USD BRL Series Forecasting RMSE		
Method	Statio	Non-statio
ARIMA	0.0031	X
GARCH	0.0031	X
RF	0.0687	0.0080
SVM	0.0794	0.1038
MLP	0.0863	0.0063
RNN		
Prophet	0.1605	0.0912

American Airlines
USD/CAD Currency
USD/BRL Currency

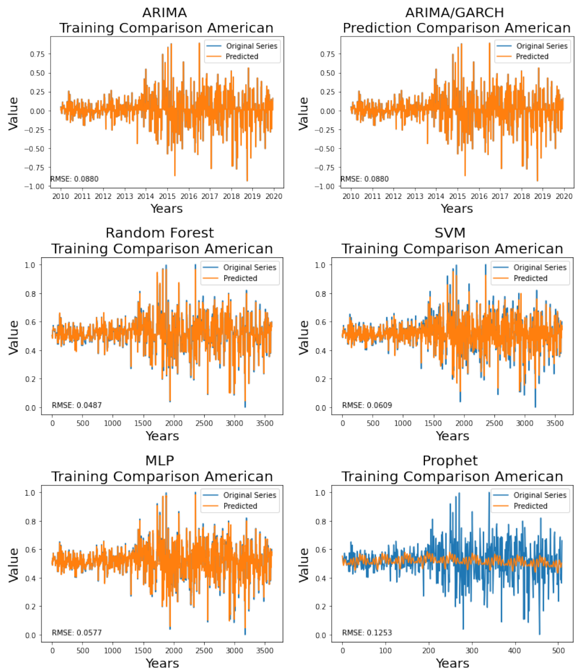

Figure 3. American Stationary Series Training

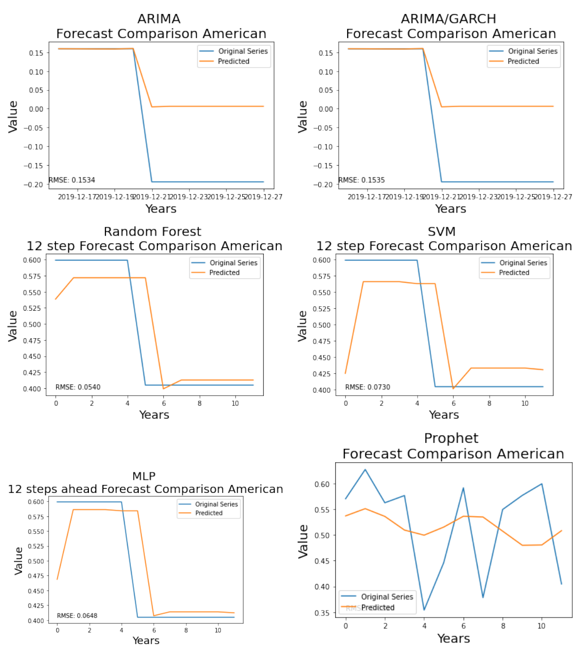

Figure 4. American Stationary Series Forecast

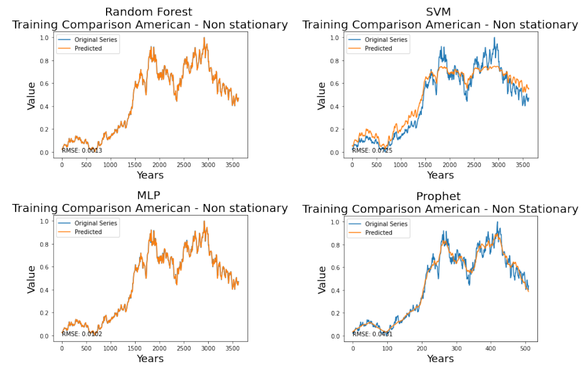

Figure 5. American Original Series Training

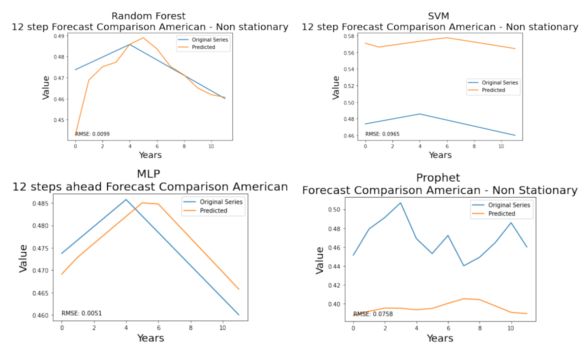

Figure 6. American Original Series Forecast

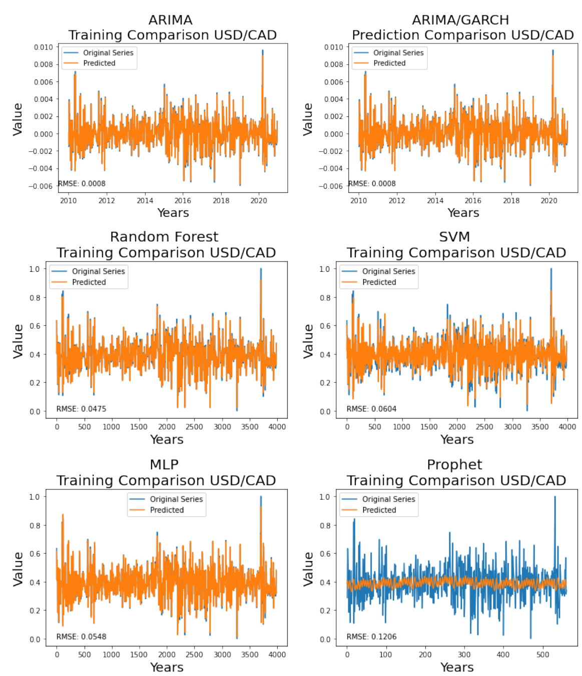

Figure 7. USD/CAD Stationary Series Training

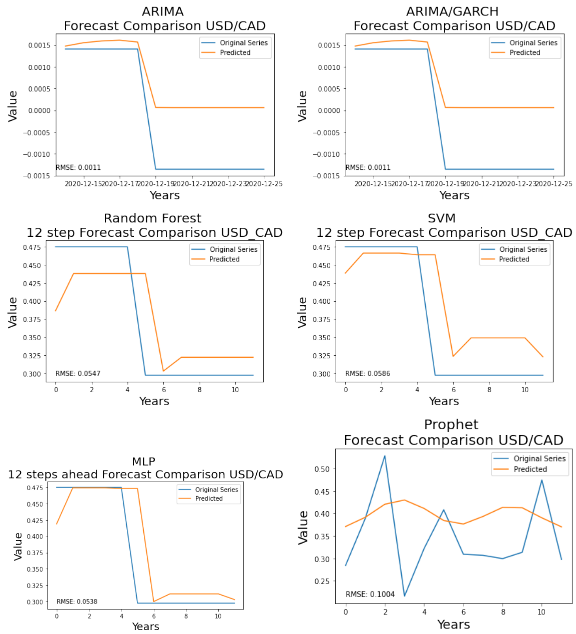

Figure 8. USD/CAD Stationary Series Forecast

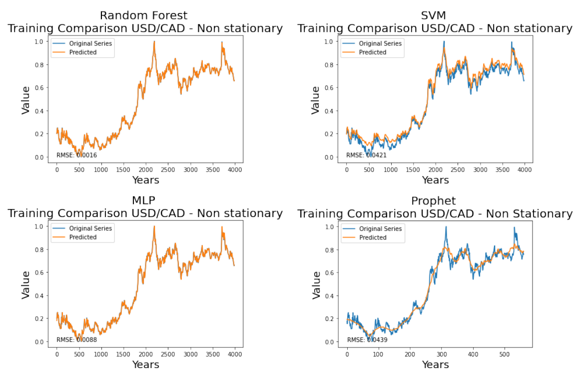

Figure 9. USD/CAD Original Series Training

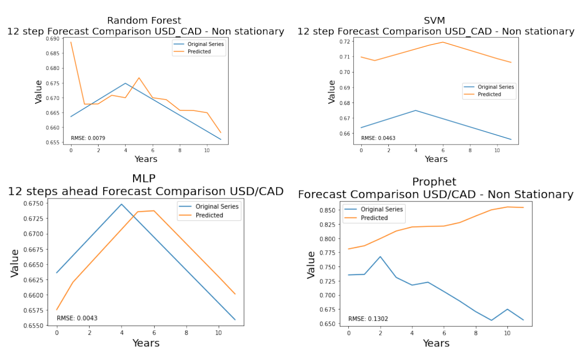

Figure 10. USD/CAD Original Series Forecast

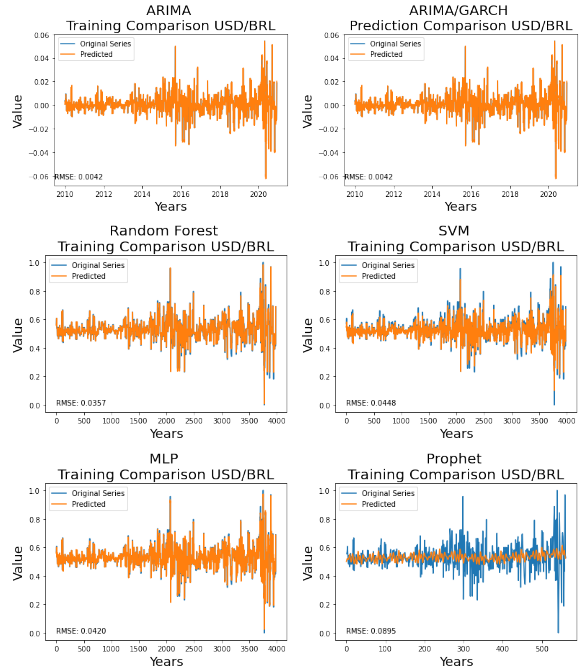

Figure 11. USD/BRL Stationary Series Training

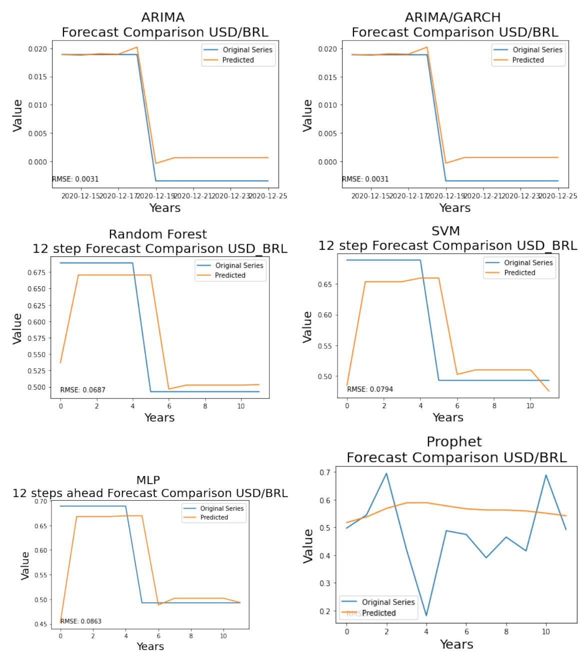

Figure 12. USD/BRL Stationary Series Forecast

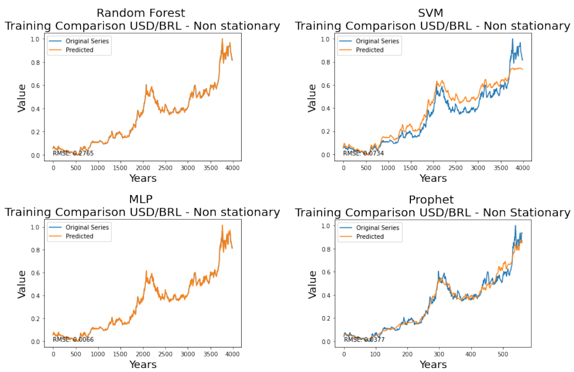

Figure 13. USD/BRL Original Series Training

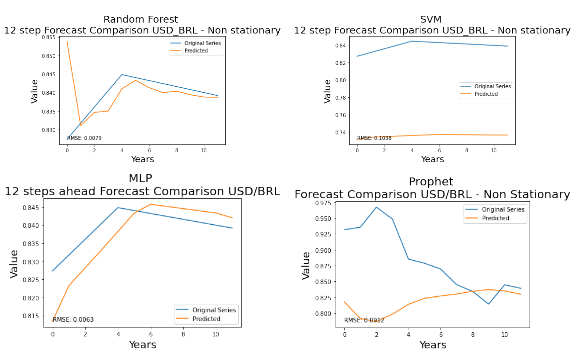

Figure 14. USD/BRL Original Series Forecast