STAT 2270 Final project Fall 2020

Manuel Alejandro Garcia Acosta

November 13, 2020

1 Introduction

Forecasting the evolution of exchange rates is important for people, companies and governments as exchange rates influence several macroeconomic variables such as volume of imported and exported goods, governmental budget (for countries with oil reserves), and remittances. It also affects decisions made in economic and foreign politics.

Theofilatos et al., state that 'traditional statistical methods, used by economists in the past years, seem to fail to capture the discontinuities, the nonlinearities and the high complexity of financial time series'. (Theofilatos, 2012) The nuances of dealing with series such as those of exchange rates provide a challenge. In this regard, 'Complex machine learning techniques like Artificial Neural Networks, Support Vector Machines (SVM), and Random Forests provide enough learning capacity and are more likely to capture the complex non-linear models which are dominant in the financial markets'. (Theofilatos, 2012)

This paper applies several machine learning techniques to the problem of predicting the one day ahead movement direction (positive/negative) of the EUR/USD exchange rate. Techniques used were K-nearest neighbor (KNN), Naïve Bayes classifier, Support Vector Machines, Random Forests, and Neural Networks. Models were tuned with 'naive strategy' and 'moving average' methodologies.

Plakandaras et al. worked on forecasting models for monthly and daily spot prices for five exchange rates: EUR/USD, USD/JPY, AUD/NOK, NZD/BRL, ZAR/PHP.[1] This paper utilized, among other methods, Artificial Neural Networks (ANN) and Support Vector Machines for forecasting and a modified version of the Multivariate Adaptive Regression Splines (MARS) for variable selection (Plakandaras, 2015).

My project aims to investigate the performance of machine learning techniques in forecasting the USD/MXN exchange rate. With data available the Bank of Mexico (BANXICO) I

[1] JPY (Japanese Yen), AUD (Australian dollar), NOK (Norwegian krone) and ZAR (South African rand), PHP (Philippine peso)

trained ANN, Random Forest and ARIMA models to compare their forecasting accuracy using the mean absolute percentage error (MAPE).

2 Methodology

In the following section, I provide a brief summary of ANNs and the model I used to complete this project.

2.1 Neural Networks

Artificial Neural Networks are methods based on mathematical models of the brain. In the human brain, cells known as neurons process information through electrical signals. An individual neuron receives information (input) from another neuron through its dendrites, processes it and then send its response (output) to another neuron using its axon/synapses.

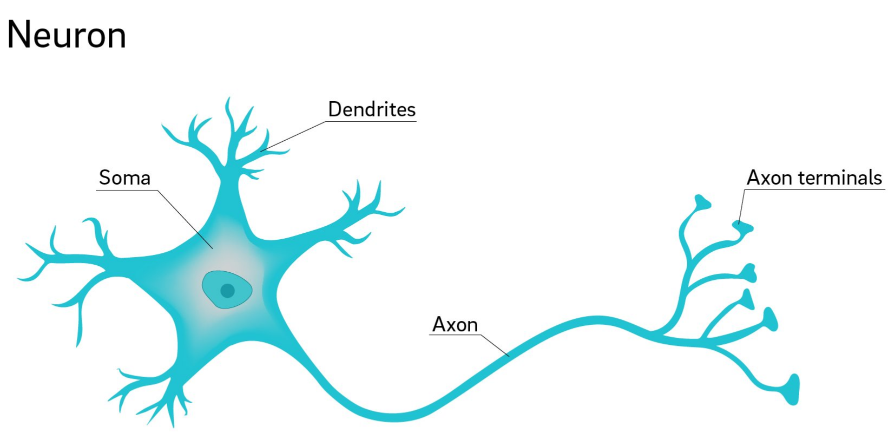

Figure 1: A neuron.

The network formed by all the neurons in our brain makes for a system that allow us to think and process information. The first mathematical model of a neuron was developed by Warren McCulloch and Walter Pitts (McCulloch & Pitts, 1943). In their paper, they described a mathematical model for a single neuron. As in the biological case, this neuron takes inputs, processes the information and returns an output.

An ANN is composed of several interconnected elements grouped in layers. The features (or inputs) are in the first layer, and the response(s) are in the last layer. The artificial neurons in the ANN are interconnected and each connection between two elements has a *weight* attached to it. The ANN "learns" by adjusting these weights to minimize a function.

2.2 Single-layered ANNs

'The simplest networks contain no hidden layers and are equivalent to linear regressions. The figure below shows the neural network version of a linear regression with four predictors. The coefficients attached to these predictors are called "weights". The forecasts are obtained by a linear combination of the inputs. The weights are selected in the neural network framework using a "learning algorithm" that minimizes a "cost function" such as the MSE' (Hyndman & Athanasopoulos, 2018).

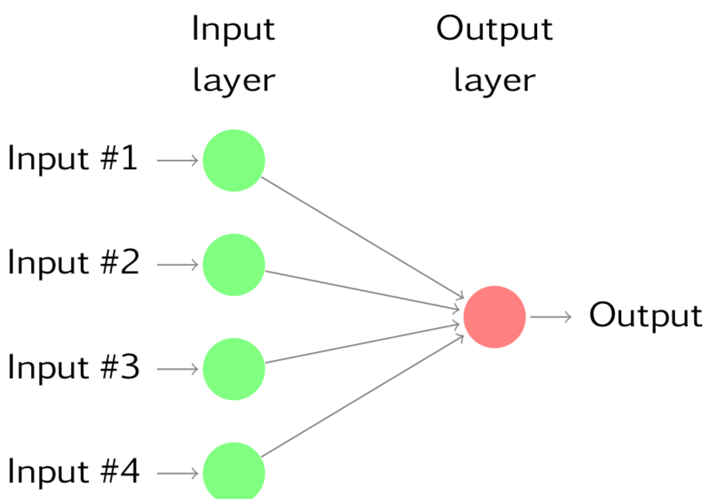

Figure 2: Neural network with no hidden layers (Hyndman & Athanasopoulos, 2018).

The neural network above receives 4 inputs and returns only one output. For an observation $x_i \in \mathbb{R}^4$ We can write this as

$$\hat{y}_i = b_0 + \sum_{k=1}^{4} w_k x_{ik}, \tag{1}$$

where w_k are the weights, $x_i = (x_{i1}, x_{i2}, x_{i3}, x_{i4})$ and b_0 is a term called bias (the equivalent of an intercept in a linear model). Now, for time series and this project in particular *feed-forward* neural networks are used. In such networks each layer of nodes receives inputs from previous layers and the original inputs are past entries of the series. This is called neural network autoregression. For example, in figure 2 we are trying to predict the y_t entry of the series using the past 4 entries in the series $y_{t-1}, y_{t-2}, y_{t-3}.y_{t-4}$. We can rewrite model (1) as

$$\hat{y}_t = b_0 + \sum_{k=1}^{4} w_k y_{t-k}\,. \tag{2}$$

The terms y_{t-k} are commonly known as "lagged entries".

2.3 Multi-layered ANNs

From (1) we can create a slightly more complicated model after adding more layers between the input and the output. We can add one of more separate layers which will be called "hidden layers". These new layers will allow the neural network to work with non-linear functions. An example of such neural networks is in the next figure.

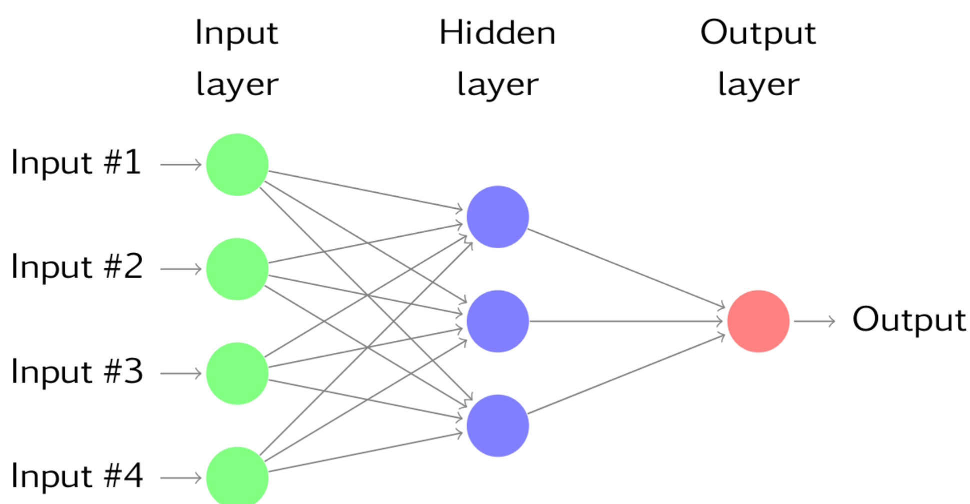

Figure 3: Neural network with one hidden layer (Hyndman & Athanasopoulos, 2018).

The neural network in figure 3 possesses one hidden layer. 'This is known as a multi-layer feed-forward network, where each layer of nodes receives inputs from the previous layers. The outputs of the nodes in one layer are inputs to the next layer. The inputs to each node are combined using a weighted linear combination. The result is then modified by a nonlinear function before being output' (Hyndman & Athanasopoulos, 2018).

The output in multi-layer neural networks becomes more complex, and this gives them the ability to work with non-linear trends. In the neural network from 3 the output of the j-th hidden node/neuron can be written as

$$z_{ij} = b_j + \sum_{k=1}^{4} w_{kj} x_{ik}, \quad j \in \{1,2,3\} \tag{3}$$

In an autoregression setting, model (3) turns into

$$z_{tj} = b_j + \sum_{k=1}^{4} w_{kj} y_{t-k} \tag{4}$$

After getting the outputs from the nodes in the hidden layers, a function known as the "activation function", $\phi(z)$, will be applied to z_{ij}. The activation function used in my project was a sigmoid function

$$\phi(z) \frac{1}{1+e^{-z}}. \tag{5}$$

For the last layer of 3, a linear combination of $\phi(z_{ij})$ will be taken to produce the final output. The use of the activation functions has the objective of making the neural network robust to outliers. For the ANN 3 the final output will look like

$$\hat{y}_i = a + \sum_{j=1}^{3} h_j \phi(z_{ij}). \tag{6}$$

Finally, in the autoregressive model, the output will look like

$$\hat{y}_t = a + \sum_{j=1}^{3} \left[h_j \phi \left(b_j + \sum_{k=1}^{4} w_{kj} y_{t-k} \right) \right]. \tag{7}$$

2.4 Data

The data series employed was the USD/MXN FIX exchange rate from January 2017 to January 2020, there are 260 observations per year. The FIX is determined by BANXICO based on an average of wholesale exchange market prices for operations settled on the second following banking business day and are obtained from exchange transaction platforms and other electronic media with representation in the foreign exchange market. Every bank business day starting at 12:00 hours the FIX is disclosed, one bank business day after the determination date it is published in the Official Gazette of the Federation (DOF) and the day after publication in the DOF it is used to settle dollar obligations payable in Mexico.

2.5 Procedure

In my study, I used the USD/MXN dataset as follows; I used the data from January 2017 to December 2019 as the *training* set and the observations for January 2020 as the *test* set. The training dataset has 782 observations, and the test dataset has 23 observations.

For the ANN model I used the nnetar() function from the **forecast** package in R. This function fits a neural network with a given number of lagged entries p and one hidden layer with a given number ($size$) of nodes. Each neural network is run 20 times with random starting points to select the weights of the nodes. Theofilatos et al. used 10 lagged observations while Plakandaras et al. used up to 30 lagged entries. Due to computational limitations, I trained neural networks with $p \in \{1, \ldots, 20\}$ lagged entries and size $m \in \{1, \ldots, 2p\}$.

For Random Forest I ran 15 models using from 1 to 15 lagged entries as features and compared their performance, the $mtry$ parameter was not tuned in my simulations.

Performance of the models was measures using the Mean Absolute Percentage Error (MAPE). The percentage errors are defined as

$$p_t = \frac{100e_t}{y_t},$$

where $e_t = y_t - \hat{y}_t$. The MAPE is then defined as

$$MAPE = \frac{1}{n}\sum_{t=1}^{n} |p_t|. \tag{8}$$

Percentage errors are commonly used to compare performance between models and data sets. This was the metric used in (Theofilatos, 2012).

3 Results

To compare a classic method with machine learning techniques, I adjusted an ARIMA(2,1,0) using the auto.arima() function from the **forecast** package. I did this because I did not want to transform the data in any way since the input of the trees and networks is the series without manipulating it. Thus, an overview can be given from the simplest application of these methods with the same measure. As a result, the ARIMA forecast obtained a $MAPE$ error of 0.3819605, which is higher than the others. I conclude that this technique was weaker in a simple application, it would be necessary to analyze and transform the series in detail to achieve a better forecast.

The simplest application of an ARIMA model proved to not fit properly this variable, that could be because the exchange rate of the dollar against Mexican peso is impacted by various economic, political, social, and psychological factors, which makes it quite difficult to forecast. Also, in recent years there have been many changes in this rate's level that cannot been captured by this model.

From all the trained neural networks, the one with the minimum MAPE used 5 lagged entries and had an architecture of 9 nodes in the hidden layer. The MAPE in this case was $MAPE_{ANN} = 0.3135339$.

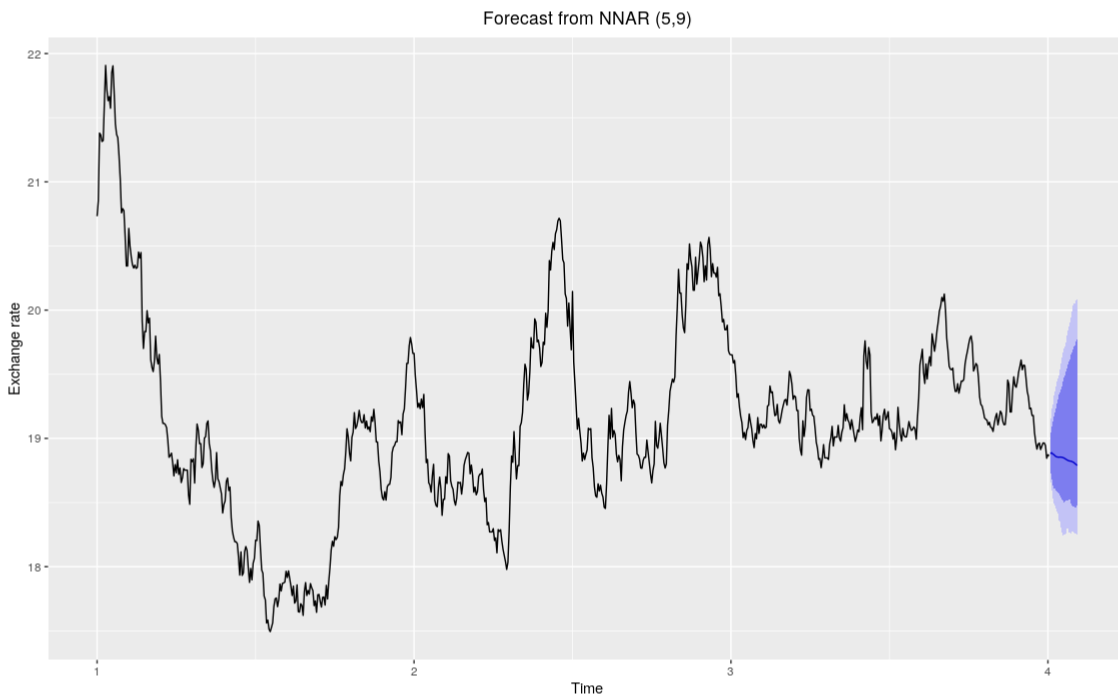

Figure 4: Forecast from neural network autoregression.

The 'out of the bag' random forest (with 5 lagged entries) performed surprisingly (or not) well. The MAPE for Random Forest with 5 lagged entries was $MAPE_{OOB} = 0.3506094$. After tuning p, the number of lagged entries, I got that the best p was 12 with corresponding error $MAPE_{RF} = 0.3030751$. The number of variables randomly sampled as candidates at each split, or $mtry$, was left as $p/3$. Further tuning of $mtry$ is expected to improve the performance of the random forest.

As an end note, I think is worth mentioning that the random forest procedure required less computation time and, even while only tuning the number of lagged entries, it was able to outperform the neural network procedure.

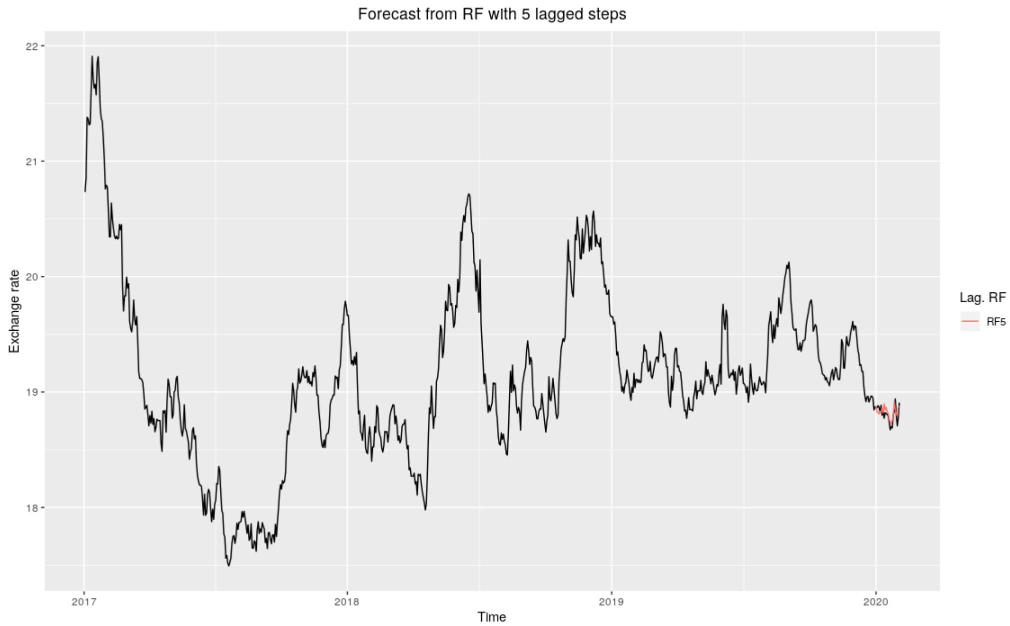

Figure 5: Forecast from random forest with 5 lagged entries.

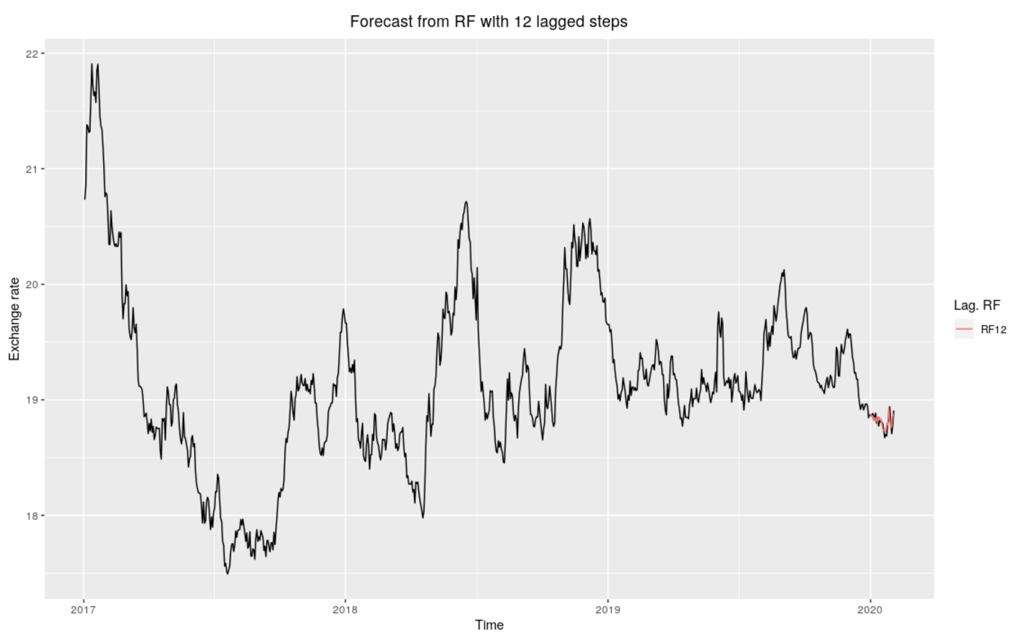

Figure 6: Forecast from random forest with 12 lagged entries.

4 References:

1. K.Theofilatos, S.Likothanassis and A.Karathanasopoulos, 'Modeling and trading the EUR/USD exchange rate using machine learning techniques', Eng., Technol. Appl. Sci. Res. 2(5)(2012)269–272.

2. Plakandaras V., Papadimitriou T. and Gogas P., 2015, 'Forecasting daily and monthly exchange rates with machine learning techniques',Journal of Forecasting, vol. 34 (7), pp. 560-573.

3. Hyndman, R.J., & Athanasopoulos, G. (2018) Forecasting: principles and practice, 2nd edition, OTexts: Melbourne, Australia. OTexts.com/fpp2. Accessed on Nov 5, 2020.

4. Database from BANXICO can be found in `https://www.banxico.org.mx/tipcamb/tipCamMIAction.do?idioma=sp`