Udacity 2018: Capstone Project

Forecasting time series with machine learning

with applications to currency exchange rates

Author: Oscar Javier Hernandez

September 3, 2018

1 Definition

1.1 Project Overview

A set of data that is indexed by time is known as a time series. They appear in many different fields, such as statistics, physics, finance, economics, biology, or even business [1]. Because of their wide applicability, it is important to generate accurate forecasts of time series data. These forecasts are generated using specific mathematical models or algorithms which are trained on a subset of the past values of a given time series. For the purpose of simplifying future discussions, we will adopt the following notation for a time series, denoted $X(t)$ or X_t, as

$$\{X(t); t = 0, 1, ...\}. \tag{1}$$

Where t denotes the time-index of the series. One of the simplest models for a time series is the ARIMA (Auto regressive integrated moving average) model. This model is denoted as ARIMA(p, d, q), and assumes that the time series X_t has the form

$$X_t = \mu + \epsilon_t + \sum_{i=1}^{p} \phi_i L^i \left[(1 - L)^d\right] X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}, \tag{2}$$

where $\{\phi_i | i = 1, ..., p\}$, $\{\theta_i | i = 1, ..., q\}$ are model parameters and L is the lag operator defined as $LX_t = X_{t-1}$. The parameter p, is known as the auto-regressive (AR) order, d is the differencing order, and q is the moving average parameter (MA).

The term ϵ_t denotes the error terms, assumed to be independent, identically distributed random variables sampled from a zero-mean, normal distribution. The value μ denotes the average of this model. ARIMA models can be applied to make forecasts of stationary time series (defined as a time series whose mean, variance and auto correlation does not change over time), or to a time series that can be transformed into a stationary time series. However, there are other state-of-the-art machine learning methods that can be used to model time series methods. Which will the main goal of this project.

One important type of financial time series is the exchange rate between different currencies (Fig. 1). An exchange rate, is the rate at which one currency will be exchanged for another. There are many factors that can influence this rate, such as balance of payments, interest rate levels, inflation levels and other economical factors which are beyond the scope of this project [3].

1.2 Project Statement

The main objective of this project will be to use classical and more recent machine learning techiques to make forecasts of different time series with the goal of applying the best methods to predict currency exchange rates. The simplest model that we will use is the ARIMA model, defined in the previous section as the baseline model, along with a linear regression model. The ARIMA model has been shown to be adequate in estimating the exchange rates of certain currencies Ref. [2]. We will then use different neural networks architectures such as the feed forward and recurrent networks, as in Ref. [5, 4, 6], to make predictions of time series and use our baseline models and root mean square differences to quantify and compare the performance of our different architectures.

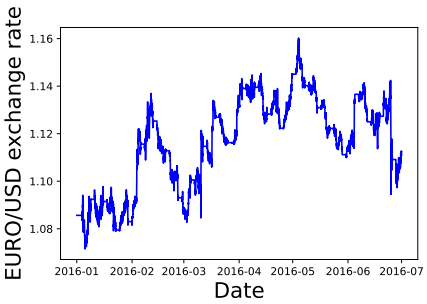


Figure 1: The exchange rate from EUR to USD from Jan 2016 to Jul 2016.

1.3 Metrics

There are several metrics that we can use to evaluate the predictions of our models Ref. [1], however, for our project we will focus on three commonly used metrics. We will first define some terminology, the forecast error, e_t, is given by

$$e_t = X_t - F_t, \tag{3}$$

where X_t is the value of the time series at time step t, and F_t is the forecasted value at the same time step.

The three metrics that we will use for our project are

1. The mean square error (MSE)
   - $\text{MSE} = \frac{1}{N}\sum_{t=1}^{N} e_t^2$
2. The root mean squared error (RMSE)
   - $\text{RMSE} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} e_t^2}$
3. The Akaike information criterion (AIC)
   - $\text{AIC} = 2k - \text{Ln}(\hat{L})$

   where k is the number of parameters in the model and $\hat{L}$ is the maximum value of the likelyhood function for the model.

In these three cases, the smaller the value of the RMSE, MSE, and AIC then the better the overall model.

2 Analysis

2.1 Data Exploration

For our project, we focused on three time series data sets. **i**. The international airline passenger data set, containing the total number of airline passengers in thousands from Jan 1949 until Dec 1960; **ii**. The sunspot data set, showing the monthly number of sunspots from 1705-1989; **iii**. The EUR to USD exchange rate from Jan 2016- July 2016. Datasets **i**. and **ii**. both consist of two columns, one for the time steps t and the valued being measured X_t.

Month (t)	Int. airline passengers (X_t)
1949-01	112
1949-02	118
1949-03	132
⋮	⋮

Table 1: The format of datasets **i**. and **ii**.

The third data set, **iii**. contains several columns related to the exchange rate market. The columns are Time, the Opening rate, the maximum value of the rate , the lowest value , the closing exchange rate value, and the total volume of the exchange market.

Time	Open	High	Low	Close	Volume
2010-01-01 00:00	1.43283	1.43293	1.43224	1.43293	608600007.1
2010-01-01 00:15	1.43285	1.43295	1.43229	1.43275	535600003.2
2010-01-01 00:30	1.43280	1.43303	1.43239	1.43281	436299999.2
⋮	⋮	⋮	⋮	⋮	⋮

Table 2: A sample of the format of dataset **iii**.

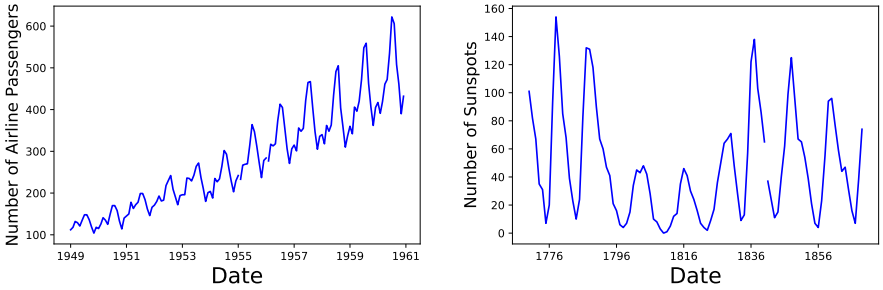
Here we list some of the statistics of the dataset.

	Data set **i.**	Data set **ii.**	Data set **iii.**
count	144	309	245441
mean	280.298611	49.752104	1.268375
std	119.966317	40.452595	0.112992
min	104.000000	0.0000000	1.035580
25%	180.000000	16.000000	1.135360
50%	265.500000	40.000000	1.302350
75%	360.500000	69.800000	1.356560
max	622.000000	190.200000	1.493240

Table 3: Descriptive statistics of the data sets **i**, **ii**. and **iii**.

2.2 Exploratory Visualization

Here we produce visualizations of the three data sets.



(a) The number of airline passengers from 1949-1961.

(b) The number of observed sunspots from 1705-1989.

Figure 2: A figure with two subfigures

The airline data set in Fig. 2a shows two interesting patterns, the general increasing number of airline passengers over time, along with a seasonal yearly pattern. The sunspot data set in Fig. 2b shows a cyclical pattern that repeats about every 11-years.

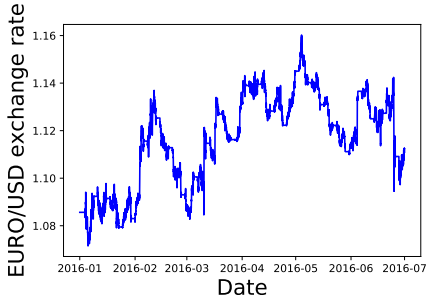


Figure 3: The exchange rate from EUR to USD from Jan 2016 to Jul 2016.

In Fig. 3 we plot the Euro to USD closing price from Jan 2016 to July 2016. There does not seem to be any apparent seasonal or cyclical patterns in this data sample.

2.3 Algorithms and Techniques

For this project, we will use the ARIMA model as a benchmark model as described in Sec. 1.1 and implemented as in Sec. 3.2. ARIMA models have been used to predict foreign exchange rates [2] and are a classical method for time series forecasts Ref. [1]. For these reasons, we use the ARIMA model as a benchmark.

Neural networks for time series models have been explored in the literature [1, 4, 5] and have shown good predictive abilities. This motivates us to use

explore neural networks for time series modeling. The two neural networks that we will use is the feed-forward neural network with 4 nodes along with a long-short-term memory network with 4 LSTM units. Fig. 4, Fig. 5

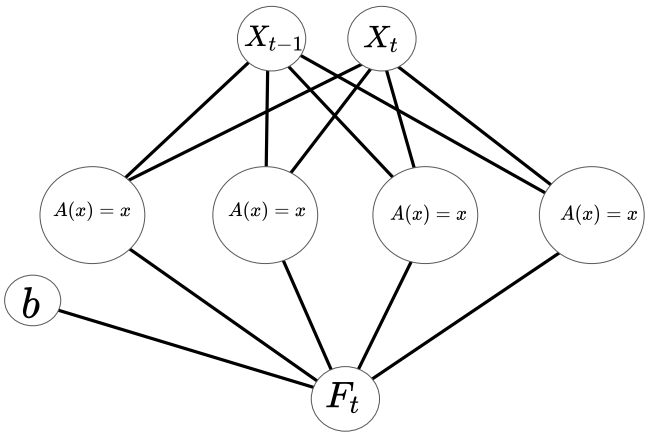


Figure 4: The chosen feed-forward neural network architecture.

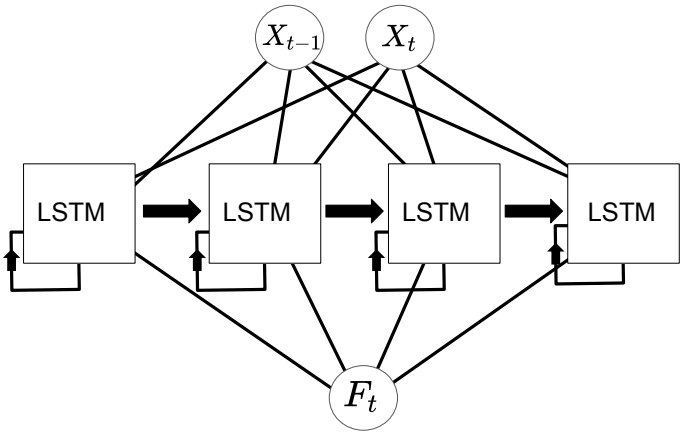
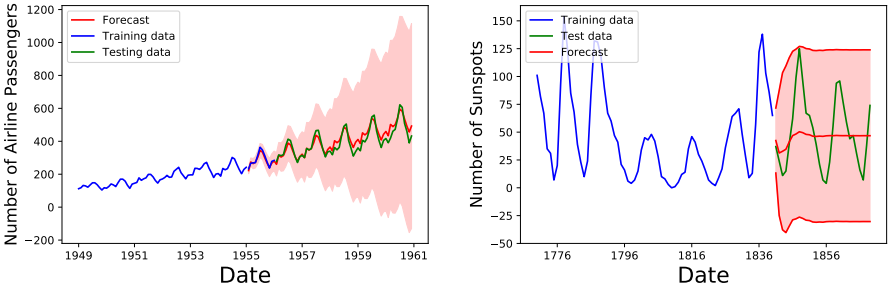


Figure 5: The chosen LSTM neural network architecture.

In Sec. 3.3, we will discus how these architectures were implemented in python.

2.4 Benchmark

The ARIMA model and seasonal ARIMA model were fit with optimized grid-search parameters, as described in Sec. 3.2, with optimized parameters given in Table 5. In Fig. 6a, Fig. 6b and Fig. 7 the curve in blue is the training data used for fitting, the green line in the testing data and the red lines show the average value of the ARIMA forecast while the red shaded area in the 95% confidence region of the prediction.



(a) SARIMA results (b) ARIMA results

Figure 6: A figure with two subfigures

In Fiq. 6a, we see that the SARIMA model was able to correctly find the overall trend and as time increases, the forecast uncertainty also increases as would be expected. Fig. 6b which used a ARIMA model, did not seem to to detect the correct trend and produces an uncertainty band that does not change over time. However, the height of the band does overlap with the training data.

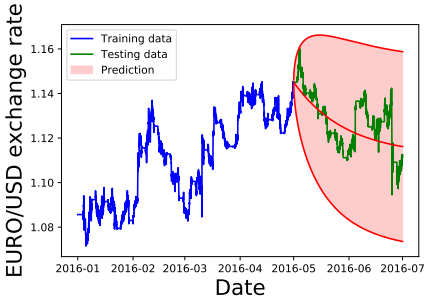


Figure 7: The forecasted exchange rate from EUR to USD from Jan 2016 to Jul 2016.

In Fig. 7 above, we see that the ARIMA model did not do a very good job at reproducing the data and it predicts a decreasing trend over time. The uncertainty band is very large in this case and encompasses a wide scale. In Table 4, we provide the MSE and RMSE values of the optimized ARIMA and SARIMA models for the above data sets.

Data Set	Model	MSE	RMSE
i.	SARIMA	1878.63101438	43.3431772529
ii.	ARIMA	902.466806466	30.0410853077
iii.	ARIMA	8.67079188384e-05	0.00931170869596

Table 4: The RMS and RMSE values of the benchmark models.

3 Methodology

3.1 Data Preprocessing

The forecasting models that we used in this work only require the values (t, X_t), since the data sets that we used do not contain any missing values, our data sets did not require any cleanup. However, it was necessary to scale the data, from $X_t \to \tilde{X}_t$ for the neural network data so that it was in the range $[0, 1]$. This transformation was accomplished using the following scaling function

$$S(X_t) = \frac{X_t - X_{\min}}{X_{\max} - X_{\min}}, \tag{4}$$

where $X_{\max}, X_{\min}$ are the maximal and minimum values of the data sets, respectively. This is implemented in python as the following code snippet.

```
def scale_array(y_vec):

    ymax = np.max(y_vec)
    ymin = np.min(y_vec)

    y_vec_scaled = np.zeros((len(y_vec),1))

    for k in range(0,len(y_vec)):
        y_vec_scaled[k][0] = (y_vec[k][0]-ymin)/(ymax-ymin)

    return y_vec_scaled,ymin,ymax
```

To invert the value from the scaling function, we use,

$$S^{-1}(\tilde{X}_t) = X_{\min} + \tilde{X}_t \cdot (X_{\max} - X_{\min})\,. \tag{5}$$

The inversion of the transformation was carried out after the model made its predictions so that the predictions would be in the original range of the data set. This is implemented as follows,

```
def invert_scaling(y_vec_scaled,ymin,ymax):

    y_vec = np.zeros(y_vec_scaled.shape)

    for k in range(0,len(y_vec_scaled)):
        y_vec[k] = ymin+(ymax-ymin)*y_vec_scaled[k]
    return y_vec
```

3.2 ARIMA Model: Implementation

We used the ARIMA and the Seasonal ARIMA models from the python package `statsmodels` (Ref. [10]). The ARIMA(p,d,q) consists of the variables (p,d,q), which are the non-seasonal AR order, differencing, and MA order, respectively. In order to fit this model to data, we use grid search from $(p,d,q) = (0,0,0) \rightarrow (p_{\text{Max}}, d_{\text{Max}}, q_{\text{Max}})$. To fit the ARIMA model we chose to either minimize the AIC, or the root mean square (RMS) value. The following code generates the grid that we will use to search the model space for the best fitting ARIMA model,

```
p = range(0,pMax+1)
d = range(0,dMax+1)
q = range(0,qMax+1)

# This creates all combinations of p,q,d
pdq = list(itertools.product(p, d, q))
```

For each item in the grid generated with the previous code snippet, we fit ARIMA(p,d,q) model using the following code.

```
for param in pdq:
     arma_mod =
         statsmodels.tsa.arima_model.ARIMA(train_data,order=param).fit()
```

Once the model is fit, the AIC or RMS value is computed and the set of parameters (p,d,q) that generated the best score is taken to be the optimal model.

The seasonal ARIMA model that incorporates non-seasonal and seasonal factors as a multiplicative model. Therefore, we can schematically write

$$\text{SARIMA} = \text{ARIMA}(p,d,q) \times \text{S}(P,D,Q,T), \tag{6}$$

where the variables (p,d,q) are the non-seasonal AR order, differencing, and MA order, respectively. The variables (P,D,Q) denote the same variables, except for the seasonal component S. The variable T in the S component denotes the number of time steps for a single period. For example, if the time units are in Months, then T=6 indicates a half-year seasonal pattern. In order to fit the SARIMA(p,q,d,P,D,Q,T) model to data, we use a gridsearch. The following code snippet generates a grid of $(p,d,q) = (0,0,0) \rightarrow (p_{\text{Max}}, d_{\text{Max}}, q_{\text{Max}})$ and $(P,D,Q) = (0,0,0) \rightarrow (p_{\text{Max}}, d_{\text{Max}}, q_{\text{Max}}, T)$, where we take T as fixed.

```
p = range(0,pMax+1)
d = range(0,dMax+1)
q = range(0,qMax+1)
t = [t]

# This creates all combinations of p,q,d
pdq = list(itertools.product(p, d, q))

# This creates all combinations of the seasonal variables
seasonal_pdq = [(x[0], x[1], x[2], x[3]) for x in
    list(itertools.product(p, d, q,t))]
```

For each combination, (p, q, d, P, D, Q, T), we fit the SARIMA model using the code below.

```
  for param in pdq:
     for param_seasonal in seasonal_pdq:
             mod = sm.tsa.statespace.SARIMAX(train_data,
                                             order=param,
                                  seasonal_order=param_seasonal,
                                      enforce_stationarity=False,
                                    enforce_invertibility=False)
             results = mod.fit()
```

Using the above prescription for fitting the ARIMA and SARIMA models, the following table summarizes the results of the gridsearch.

Data Set	Model	Optimal (p, d, q)	Optimal (P, S, Q, T)
i. Airline Passengers	SARIMA	$(1, 1, 1)$	$(1, 0, 0, 12)$
ii. Sunspots	ARIMA	$(5, 0, 4)$	-
iii. EUR to USD rate	ARIMA	(2, 0, 2)	-

Table 5: The optimal fitting parameters of the ARIMA, or SARIMA model.

3.3 Neural Network Model: Implementation

The feed-forward neural network in 4 is implemented in Keras with the following code snippet which also fits the model using the `mean_squared_error` metric and the `adam` optimizer.

```
model = Sequential()
model.add(Dense(4, input_dim=lags, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history=model.fit(X_train, y_train, epochs=number_of_epochs,
    batch_size=2, verbose=1)
```

The following is a summary of this implemented architecture

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 4)                 8
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 5
=================================================================
Total params: 13
Trainable params: 13
Non-trainable params: 0
_________________________________________________________________
```

Using the same optimizer and loss function as the feed-forward neural network the LSTM network is implemented as follows

```
model = Sequential()
model.add(LSTM(4, input_dim=lags))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
history=model.fit(X_train, y_train, epochs=number_of_epochs,
    batch_size=2, verbose=1)
```

Below, we summarize the implemented LSTM network,

```
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 4)                 96
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 5
=================================================================
Total params: 101
Trainable params: 101
Non-trainable params: 0
_________________________________________________________________
```

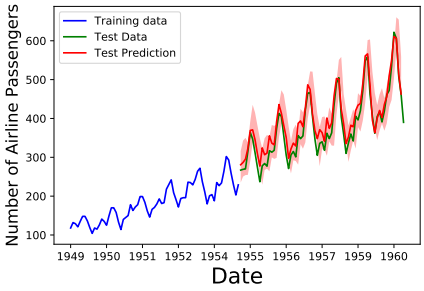
4 Results

4.1 Model Evaluation and Validation

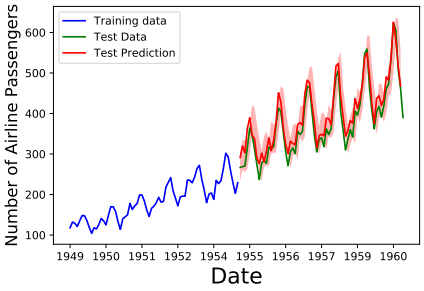
Having implemented the feed forward and LSTM networks in Keras as described in Sec. 3.3 we fit the model and generate the time series predictions $\{F_t\}$. One issue that we had using this model was that the predictions of the neural networks do not produce any uncertainties. To try to gauge the uncertainties of the model, we do the following procedure

1. Train the model using the set $X_{T,\text{Train}}$
2. Using the training data, calculate the predictions $F_{T,\text{Train}}$
3. Compute the mean squared error (RMSE), and the standard deviation σ_{RMS} between the training set $X_{T,\text{Train}}$ and the prediction on the training set F_T.
4. We generate the perturbation, δX_T, assumed to be a Gaussian random variable sampled from the distribution $\delta X_T \in \mathcal{N}(\frac{1}{2}\text{RMS}, \frac{1}{2}\sigma_{\text{RMS}})$
5. We add the perturbation to the test data $X_T + \delta X_T$ and generate predictions on the testing set, $F_{T,\text{Test}}$.
6. We repeat this process many times, until we generate a distribution of Forecasts. This distribution will the estimated uncertainty of the neural network predictions.

Using this procedure, we generated the uncertainty bands in Figs. 8,9,10.

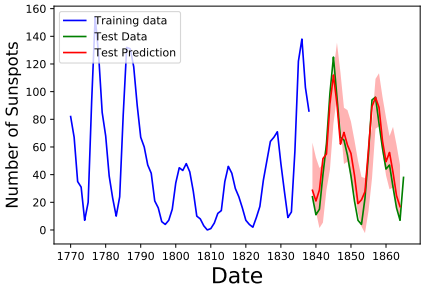


(a) NN results

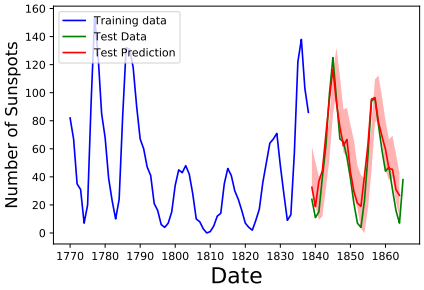


(b) LSTM results

Figure 8: (a) The predictions with uncertainty of the feed forward neural network model. (b) The predictions with uncertainty of the LSTM model.

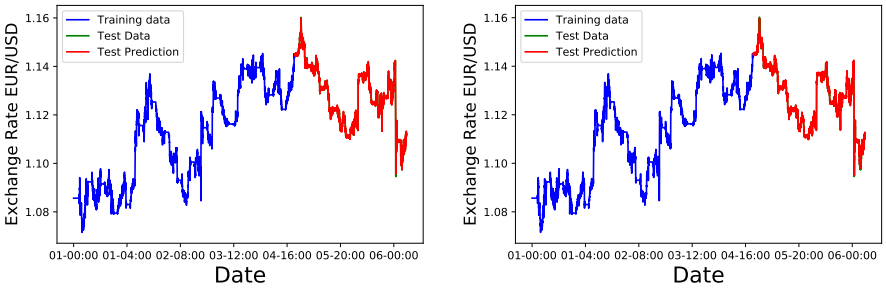


(a) NN results



(b) LSTM results

Figure 9: (a) The predictions with uncertainty of the feed forward neural network model. (b) The predictions with uncertainty of the LSTM model.



(a) NN results

(b) LSTM results

Figure 10: (a) The predictions with uncertainty of the feed forward neural network model. (b) The predictions with uncertainty of the LSTM model.

Data Set	Model	MSE	RMSE
i.	SARIMA	1878.63101438	43.3431772529
i.	FFNN	604.999299329	24.5967335093
i.	LSTM	567.812500312	23.8288165949
ii.	ARIMA	902.466806466	30.0410853077
ii.	FFNN	101.171109089	10.058385014
ii.	LSTM	119.392504984	10.926687741
iii.	ARIMA	8.67079188384e-05	0.00931170869596
iii.	FFNN	7.24275099e-08	0.000269123596
iii.	LSTM	8.45390421e-08	0.000290755984

Table 6: The RMS and RMSE values of the Feed Forward (FFNN) and long-short term memory (LSTM) neural network model results.

4.2 Justification

5 Conclusion

5.1 Free-Form Visualization

5.2 Reflection

In this project we have taken data for time series data (t, X_t) and we have trained two neural network models, the feed forward neural network and the Long-short term memory network, denoted as G_{FFNN} ,G_{LSTM} respectively; to generate a prediction of the future series given the value of the time series at the previous time step. In other words, we have generated a function G such that

$$G(X_t) = F_{t+1} \tag{7}$$

where F_{t+1} should be similar to X_{t+1} of the actual time series data.

This problem was very interesting to me as it

5.3 Improvement

There are some further improvements that we could have carried out to the implemented models, such as an in-depth exploration of different neural network architectures. Adding more dense nodes or adding more layers would have been interesting to explore. However I ran out of time before I could carry out such a study. In the future I would also like to implement a generative adversarial artificial neural network for times series prediction. Unfortunately, I ran out of the time needed to understand and implement this model, but it seemed quite interesting and powerful. I will try to extend my project in the future to do this.

References

[1] R. Adhikari and R. K. Agrawal, An Introductory Study on Time Series Modeling and Forecasting, 2013, [arXiv:1302.6613] https://arxiv.org/abs/1302.6613.

[2] T. Mong and U. Ngan, Research Journal of Finance and Accounting www.iiste.org ISSN 2222-1697 (Paper) ISSN 2222-2847 (Online) Vol.7, No.12, 2016 http://iiste.org/Journals/index.php/RJFA/article/viewFile/31511/32351.

[3] P. J. Patel, N. J. Patel and A. R. Patel, IJAIEM 3, 3, 2014. http://www.ijaiem.org/volume3issue3/IJAIEM-2014-03-05-013.pdf

[4] B. Oancea, S. Cristian Ciucu, Proceedings of the CKS 2013, [arXiv:1401.1333] https://arxiv.org/abs/1401.1333.

[5] T. D. Chaudhuri and I. Ghosh, Journal of Insurance and Financial Management, Vol. 1, Issue 5, PP. 92-123, 2016, [arXiv:1607.02093] https://arxiv.org/abs/1607.02093.

[6] Pant, N. (2017, September 07). A Guide For Time Series Prediction Using Recurrent Neural Networks (LSTMs). Retrieved from https://blog.statsbot.co/time-series-prediction-using-recurrent-neural-networks-lstms-807fa6ca7f.

[7] D. K. Acatay, (2017, Nov. 21) Part 6: Time Series Prediction with Neural Networks in Python. Retrieved from http://dacatay.com/data-science/part-6-time-series-prediction-neural-networks-python/

[8] Vincent, T. (2018). ARIMA Time Series Data Forecasting and Visualization in Python — DigitalOcean. [online] digitalocean.com. Available at: https://www.digitalocean.com/community/tutorials/a-guide-to-time-series-forecasting-with-arima-in-python-3 [Accessed 29 Jul. 2018].

[9] C. Esteban, S. L. Hyland and G Rätsch, https://github.com/ratschlab/RGAN [arXiv:1706.02633].

[10] S. Skipper and J. Perktold. "Statsmodels: Econometric and statistical modeling with python." Proceedings of the 9th Python in Science Conference. 2010. https://www.statsmodels.org/stable/index.html