Project Proposal:
Exploring the Application of Neural Network Models For ForecastingTemporal Data

Michael Northrup
Han Ambrose
Alok Jadhav

Abstract

There are many different models that can be used to forecast temporal data. The purpose of this project is to survey different models to explore which performs best on a selected time series dataset. In this paper, we will explore both traditional time series model, ARIMA, and deep learning models, LSTM and LSTM variants.

Introduction

The conventional standard time series data analysis technique, ARIMA or Seasonal ARIMA, is based on linear regressions for model fittings and moving average for prediction purposes. This type of model performs reasonably well for short-term forecasts, such as the next month, next year, next 3 months or next lag, but the performance deteriorates for long term predictions.

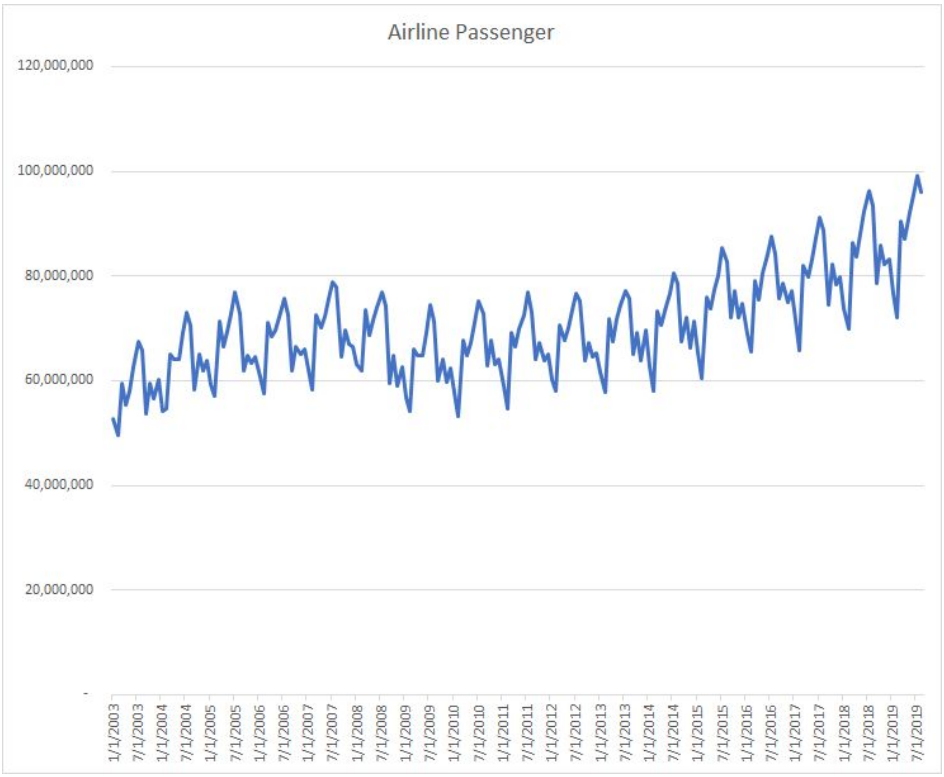
On the other hand, deep learning techniques use AI-based approaches are more data-driven than model-driven. Several variants of Long Short Term Memory (LSTM) architecture for recurrent neural networks has been explored since its inception in 1995. In recent years, these networks have gained popularity and become state-of-the-art to solve machine learning problems from speech recognition, image captioning to time series prediction.

Besides the architectures and forecasting techniques being performed, the type of time series data along with its underlying behaviors are also dominant factors affecting the performance and accuracy of the analysis. Some datasets may behave differently than other datasets due to seasonality, volatility, economic shocks, political influences or unexpected events that could affect the prediction.

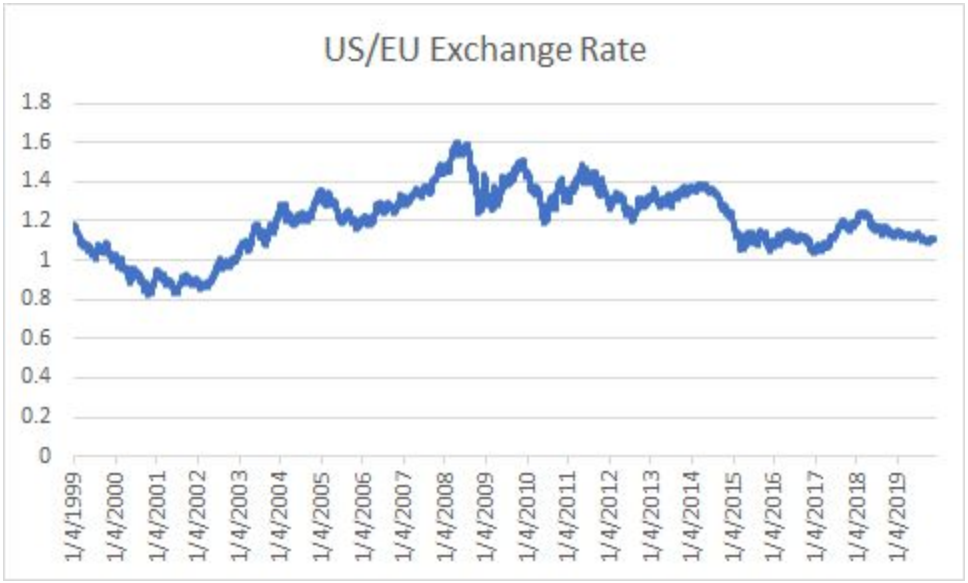
This project will use two datasets. One is airline passengers and the other is the exchange rate between USD and EUR. The reason we are doing this is to explore how volatility affects forecasting abilities. The airline passenger and exchange rate dataset have very different nature and learnable underlying patterns. Airline passenger data is heavily influenced by vacation periods and tends to be seasonal whereas the exchange rate is mainly driven by economic and political environment. However, we chose two very stable currencies, USD and Euro as they are the least volatile currency pair for predictability purposes.

Data

The airline passenger data was downloaded from the Bureau of Transportation Statistics. The data was collected monthly for both domestic and international passengers from all US and foreign carriers from October 2002 to August 2019. There are 203 observations in this dataset. There is seasonality shown in the dataset corresponds to the vacation periods.



The second dataset is US/EURO Foreign Exchange rate. The data was collected from Federal Reserve Bank website daily from January 4th 2019 to 11/29/2019 excluding non-business days and holidays.



Architecture and Results

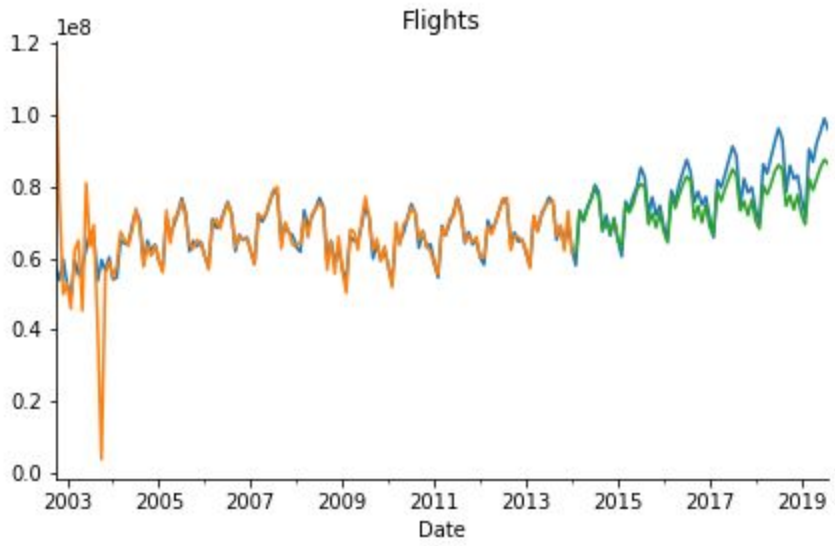
The following four models will be compared in this project.

I. **Autoregressive Integrated Moving Average (ARIMA):**

The ARIMA model is fundamentally a linear regression model accommodated to track linear tendencies in stationary time series data.

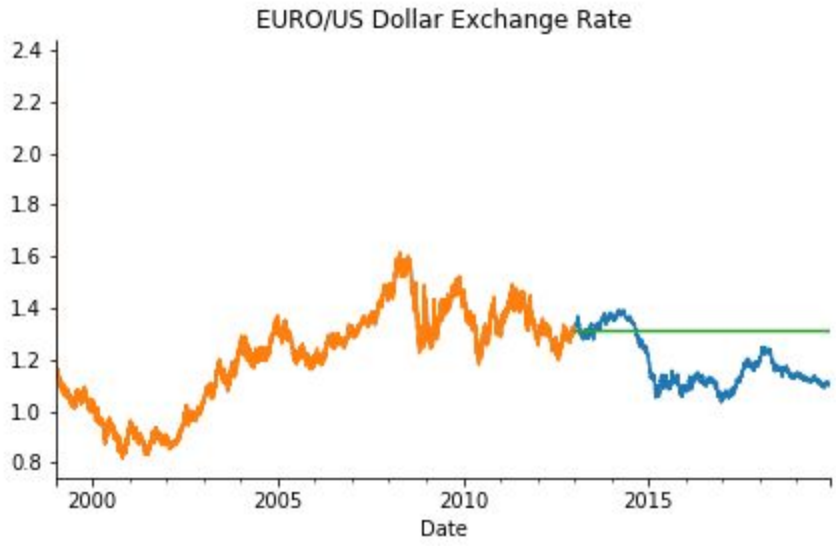
Results of vanilla ARIMA:

1. Flight Data



This model has a training error of 4.3% and a testing error of around 4.7%. Also, the RSME for this model is 7720380 for training and 4907007 for testing. The model hyper-parameters of the ARIMA model were selected itterively using the auto_arima function, a stepwise selection method.

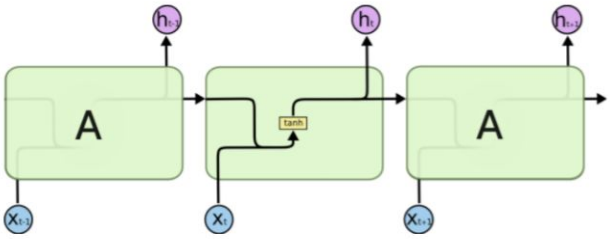
2. Forex Data



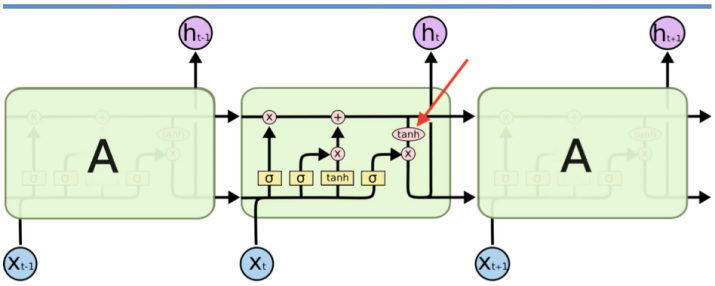
It was clear right away that this model was not good. While the training error was only 0.497% (Highly overfit) the testing error was 12.39%. These findings were not a surprise given the nature of ARIMA models and the nature of currency data.

II. Long Short Term Memory (LSTM):

LSTM is a special type of RNN. LSTMs were developed to deal with the problem of 'vanishing' and 'exploding gradient'. A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell. LSTM has the ability to perform well with spatiotemporal data. There are several variants of LSTMs as well: GRU, Residual Connections, Attention-based LSTM.



Simple RNN



Standard LSTM cell

General Methodology and Implementation:

a. The data was normalized using MinMaxscaler() function of sklearn module.
b. The dataset was divided into two where 67% is used for training and 33% are for testing.
c. Optimizer, learning rate, epochs and batched sized were tweaked to yield the best fit model
d. Baseline, training set and test set results are graphed for comparison. The data is then plotted, showing the original dataset in blue, the predictions for the training dataset in orange, and the predictions on the unseen test dataset in green.

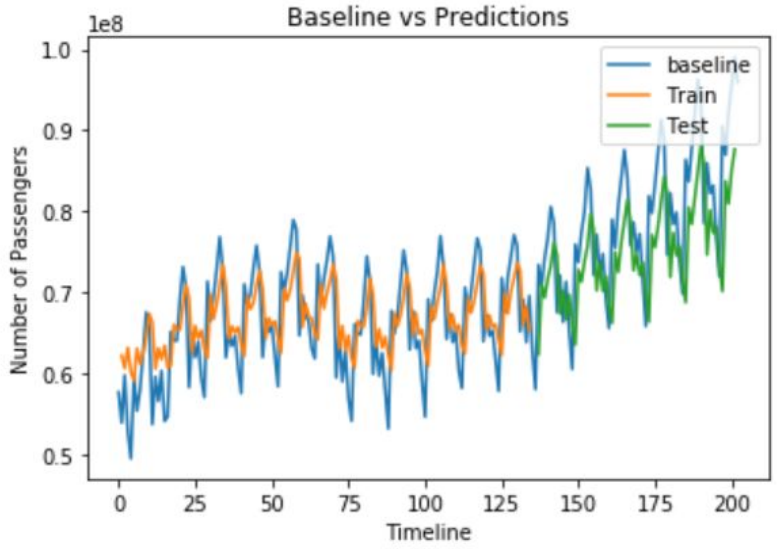
Results of vanilla LSTM:

```
loss='mean_squared_error', optimizer='Adam'
```

1. Flight Data:

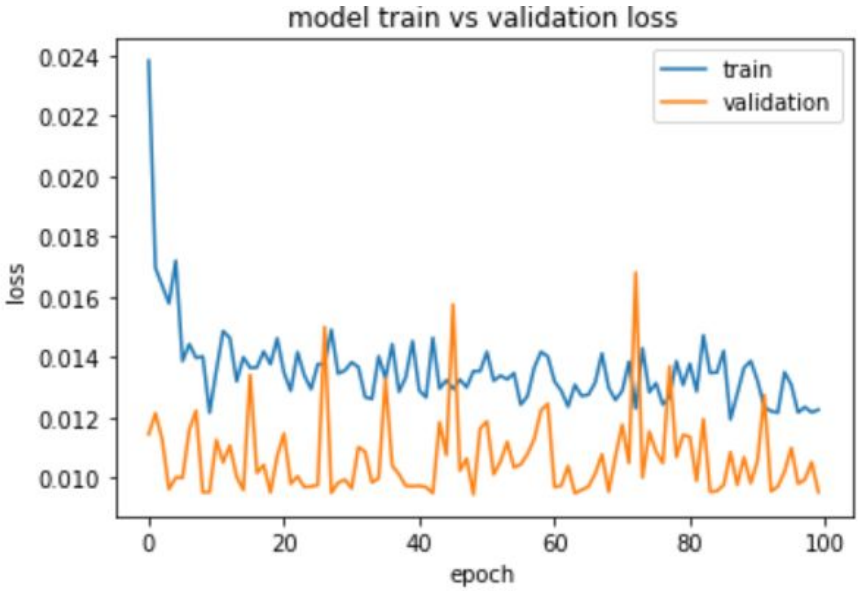
```
model.compile(loss='mean_squared_error', optimizer=Adam(lr= 5e-3),
metrics=['mean_absolute_error'])
history = model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2,validation_split
=0.33)
```

```
Train Score: 5315751.88 RMSE
Test Score: 7566629.93 RMSE
```



The root-mean-squared error (RMSE) is used to measure of the differences between predicted values and actual values. As shown on the graph, the model have pretty high error. Training error is about 5 millions passengers out of on average roughly 70 millions passengers. The pattern is learned but not closely align with baseline.

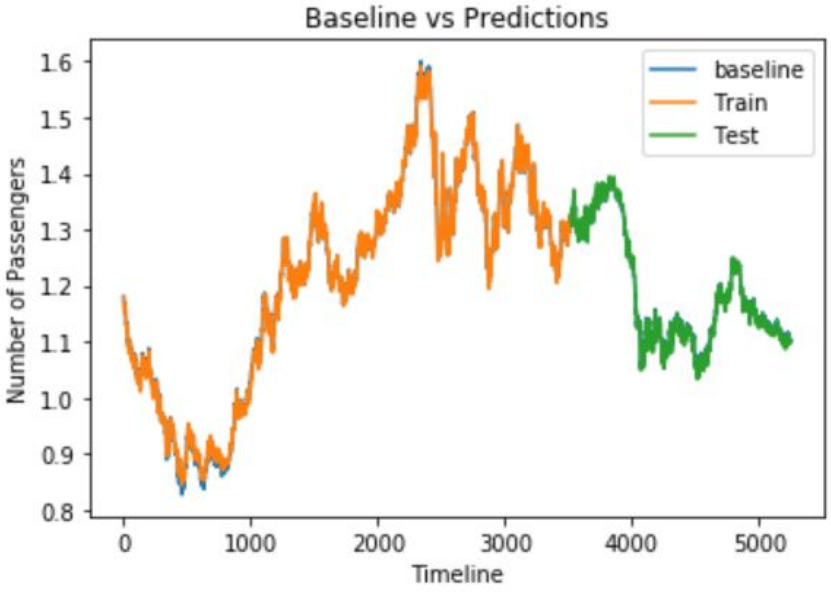
Different learning rate were applied to the loss seems to converge quickly. Even when low learning rate,5e-3, and larger iterations are applied, the model stops learning after a certain point. The graph below shows that training loss is larger than validation loss. This could be that the training set had ‘hard’ cases to learn whereas the validation set had mostly ‘easy’ cases to predict. This could be due to splitting techniques where training set is more volatile then validation set as we see training data includes a big drop in traveling in 2009. Overall, there are only 203 observations and we only use 67% of that for training, so there is not much data to train. This could be improved by having a larger dataset.



2. Forex data:

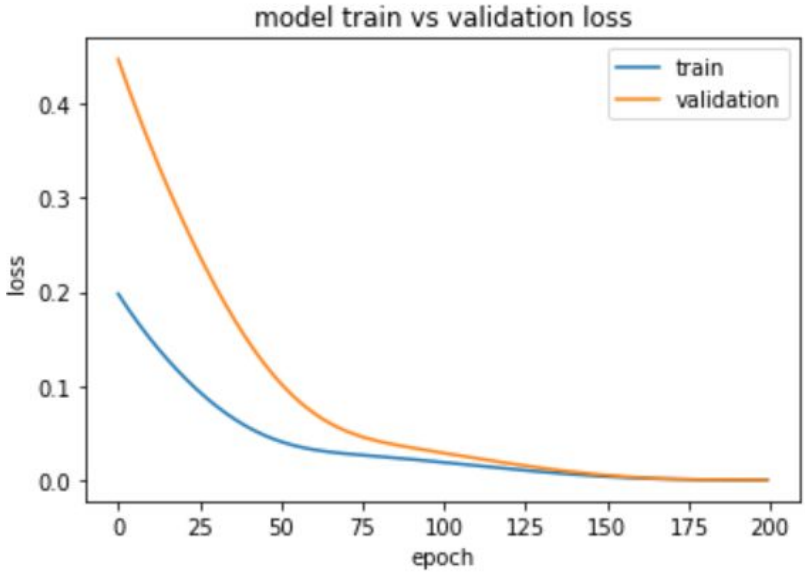
```
model.compile(loss='mean_squared_error', optimizer=Adam(lr= 5e-5),
metrics=['mean_absolute_error'])
history = model.fit(trainX, trainY, epochs=200, batch_size=32, verbose=2,validation_split
=0.33)
```

```
Train Score: 0.00872 RMSE
Test Score: 0.00661 RMSE
```



As shown on the graph, the model have low error. Training error is about 0.009 and testing error is about 0.007. The training and testing data almost emerged into the baseline, which shows good fitting of the model.

Unlike flight data, the training and validation loss shows a good learning rate as it is steadily decreasing; training and validation loss also converge and meet.



Conclusion and Comparison between 2 forecasts: Forex data is being forecasted with more accuracy and less error.

III. Dual Stage Attention-based -RNN (DA-RNN):

RNN models are powerful to exhibit quite sophisticated dynamic temporal structure for sequential data. RNN models come in many forms, one of which is the Long-Short Term Memory (LSTM) model. The 'Attention Mechanism' performs feature selection at each stage. The model can keep only the most useful information at each temporal stage.

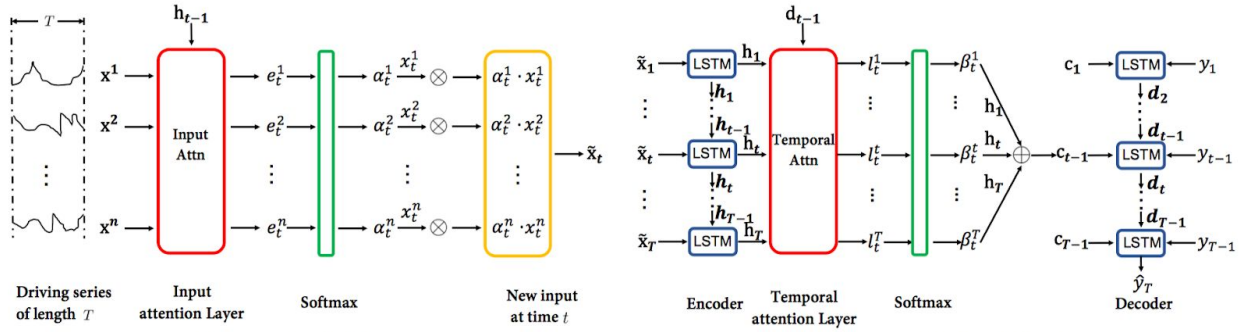
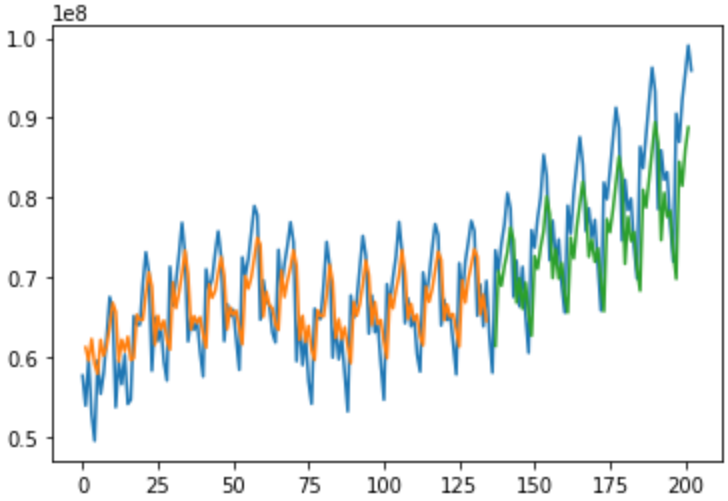


Figure 1: Graphical illustration of the dual-stage attention-based recurrent neural network. (a) The input attention mechanism computes the attention weights α_t^k for multiple driving series $\{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n\}$ conditioned on the previous hidden state $\mathbf{h}_{t-1}$ in the encoder and then feeds the newly computed $\tilde{\mathbf{x}}_t = (\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, \cdots, \alpha_t^n x_t^n)^\top$ into the encoder LSTM unit. (b) The temporal attention system computes the attention weights β_t^t based on the previous decoder hidden state $\mathbf{d}_{t-1}$ and represents the input information as a weighted sum of the encoder hidden states across all the time steps. The generated context vector $\mathbf{c}_t$ is then used as an input to the decoder LSTM unit. The output $\hat{y}_T$ of the last decoder LSTM unit is the predicted result.

Results of DA-RNN:

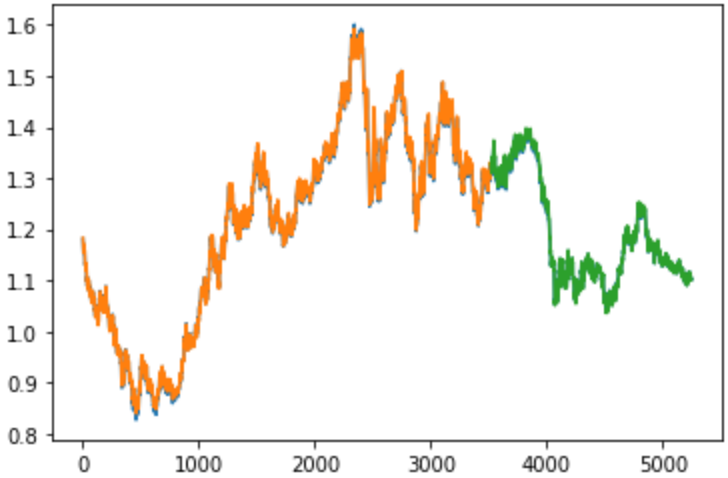
```
loss='mean_squared_error', optimizer='rmsprop'
```

1. Flight Data:



Here we can see that the Orange and Blue signals till 140 are for training data. It has a training error of 0.0115 = 11.5%. When testing it has around 0.0155 = 15% test error.
Iterations done = 500.

2. Forex data:



For forex data, training error is around: `1.3648e-04 = 0.1%`
Testing error is around: 0.001 = 1%
Iterations were Done = 500.
Orange signal is training data and green is test data; the Blue signal is forecasting of data.
Blue and green signals are almost merged.

Conclusion and Comparison between 2 forecasts: As we can see that forex data is being forecasted with more accuracy and less error. This might be because of the low volatility of the

forex data. Whereas flight data is much more unstable. That's why we are getting less accuracy with its forecasting.

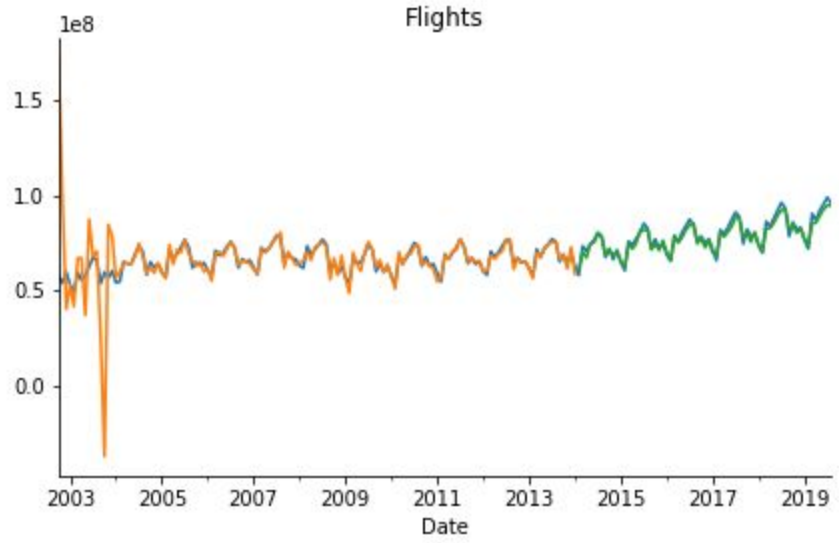
IV. ARIMA-LSTM hybrid:

The Autoregressive Integrated Moving Average (ARIMA) model is one of the traditional statistical models for time series prediction. The model is known to perform decently on linear problems. On the other hand, the Long Short-Term Memory (LSTM) model can capture nonlinear trends in the dataset. So, the two models are consecutively combined to encompass both linear and nonlinear tendencies in the model.

Results of ARIMA-LSTM hybrid:

1. Flight Data

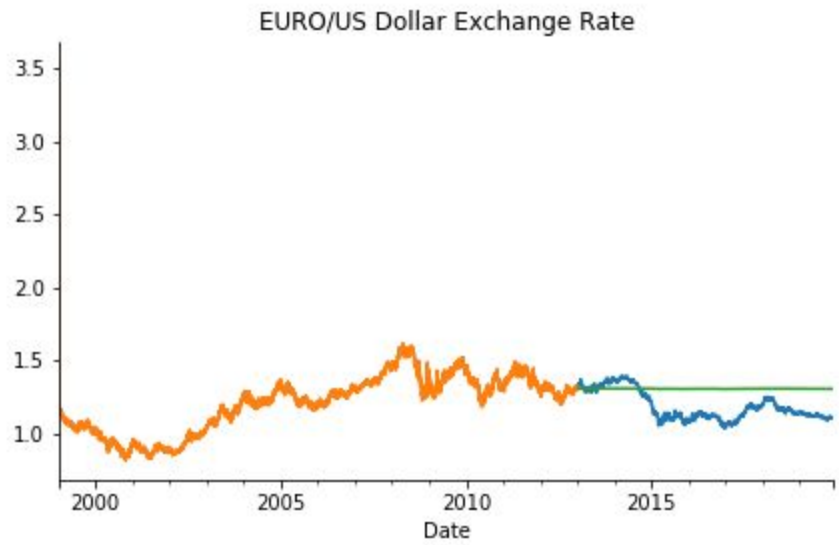
```
loss='mean_squared_error', optimizer='rmsprop'
```



This model has a training error of 7.7% and a testing error of 2.3%. Also, the RSME for this model is 14581259 for training and 2156726 for testing. Iterations done = 500. The most notable fact about this model is that the test error is 2.3% vs 4.7% for the ARIMA model on its own. The model was also superior to the LSTM model, with a test RSME of 2156726 vs 7266629.

2. Forex Data

```
loss='mean_squared_error', optimizer='rmsprop'
```



This model has a training error of 5.8% and a testing error of around 12.0%.. Iterations done = 500. While this was not a performant model, it is of note that the test error for this model is 12.0% vs 12.3% for the ARIMA model alone. Both the LSTM and DA-NN models were superior to the LSTM-ARIMA hybrid for this data.

Overall Results and Discussion

Flight Data: This dataset is more volatile and scarce. We had small training and testing datasets: 136 for training and 67 for testing. We believe that it is due to the scarcity of the dataset that it did not have the same levels of accuracy as the Forex predictions did. The ARIMA model performed best on this dataset.

Forex Data: This dataset had better coverage and was less volatile. Training and Testing sets were more robust: 3654 for training and 1801 for testing. Both the vanilla LSTM and the DA-RNN models performed well on this dataset and got less than 1% error on the forecasting. Neither the ARIMA nor the ARIMA-LSTM ensemble models had very good performance for these models.

ARIMA model: The ARIMA model was only effective for the flight data and not the Forex data. This was expected due to the seasonal nature of the flight data and the non-seasonal nature of

the Forex data. The ARIMA did perform decently well for the flight data, but it appears it overestimated the underlying increase in flights overtime.

Hybrid ARIMA and LSTM: Just as the ARIMA model, ARIMA LSTM hybrid model was only effective for the flight data and not the Forex data. ARIMA should be involved only if there are seasonal trends in the dataset. The ensemble of statistical models and neural networks resulted in improved performance over just statistical models, especially for the flight data.

Vanilla LSTM: Vanilla LSTM performed well on Forex dataset but with the Flight data it got less accuracy. We could use different optimizers to improve the performance of LSTM.

DA-RNN: DA-RNN performs decently well on Forex dataset and testing error is less than 1%. It performed similarly on the Flight data compared to vanilla LSTM. While the DA-RNN model did have slightly better performance over the vanilla LSTM, it required a notably larger amount of processing power. This trade off is another variable that should be considered when deciding which model to use for time-series prediction.

Conclusion: Models should be selected based on the type and size of the dataset when dealing with particular problem. Some models might perform well even when we have small dataset. Some models will not perform well even if we add more layers or ensambles of different models together to solve that problem.

Overall, the forex data has a better significant performance than the flight data as we believe LSTM architecture tends to work better with larger dataset.

Related Research

Choi, Hyeong Kyu. "Stock price correlation coefficient prediction with ARIMA-LSTM hybrid model." *arXiv preprint arXiv:1808.01560* (2018).

Fu, Rui, Zuo Zhang, and Li Li. "Using LSTM and GRU neural network methods for traffic flow prediction." *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2016.

Liu, Boyi, et al. "Traffic Flow Combination Forecasting Method Based on Improved LSTM and ARIMA." *arXiv preprint arXiv:1906.10407* (2019).

Qin, Yao, et al. "A dual-stage attention-based recurrent neural network for time series prediction." *arXiv preprint arXiv:1704.02971* (2017).

Siami-Namini, Sima, and Akbar Siami Namin. "Forecasting economics and financial time series: Arima vs. lstm." *arXiv preprint arXiv:1803.06386* (2018).