MINOR PROJECT (Report)

A Comparison of ARIMA and LSTM in Time Series Forecasting

Name : Shashank Sinha

Roll no. : 1810002

Subject : PHL8902

Supervisor : Dr. Anurag Sahay

Dept. : Int. M.Sc. Physics

❖Time Series Forecasting :

Time series forecasting is the process of analyzing time series data using statistics and modeling to make predictions and inform strategic decision-making.

❖Models Used :

1) ***ARIMA model*** :
   - *Auto Regression Integrated Moving Average* (ARIMA) model.
   - ARIMA is a combination of 2 traditional algorithms : *AR (Auto Regression)* & *MA (Moving Average)*.
   - In name, “Integrated” states that, we have to first convert non-stationary dataset to stationary dataset if required.
   - AR model as the name suggests, is a regression model, i.e. predicts next value based on some feature values. And “Auto” regression means, it uses the variable’s own past values to predict future values.
   - MA model uses past error to predict next value, in an attempt to minimise the error in next prediction.
   - AR model is based on *PACF (Partial Auto Correlation Function)*, i.e. only considers direct correlation with past values.
   - MA model is based on *ACF (Auto Correlation Function)*, i.e. indirectly many past values impact it’s current prediction.

2) ***LSTM model*** :

- *Long Short Term Memory* (usually called LSTMs).
- LSTM is a special kind of *RNN (Recurrent Neural Network)*. In RNN, same weights & activation function are used in each iteration to predict next future value.
- In RNN :

$$a_1 = A[(W_{xa})*(x_1) + (W_{aa})*(a_0) + b_a]$$
$$y_1 = A'[(W_{ay})*(a_1) + b_y]$$

.....

$$a_n = A[(W_{xa})*(x_n) + (W_{aa})*(a_{n-1}) + b_a]$$
$$y_n = A'[(W_{ay})*(a_n) + b_y]$$

- In calculation of a_n, gradient corresponding to a_0 will be n-times product of (W_{aa}). So if :
  $W_{aa} > 1$: *Exploding Gradient problem* will emerge. The resulting gradient will be extremely large, & will negatively effect predictions. This problem is solvable by implementing some limitations on each gradient.
  $W_{aa} < 1$: *Vanishing Gradient problem* will emerge. The resulting gradient will be almost 0, i.e. no effect of old outputs on new predictions. This is also a negative effect, and solving this is a relatively difficult task.
- For solving Vanishing Gradient problem, a different Neural network unit comes in use, i.e. *Gated Recurrent Unit (GRU)*. In this unit, a potential value for prediction is calculated & a sigmoid update-function is there to decide whether value of a_{i-1} is assigned to a_i , or the calculated potential value. There is also a *Reset*

Gate, which decides how much is (a_{i-1}) contributing to the potential value. Following are the equations :

➔ Reset Gate :

$R_{gate} = sigma[(W_r)*(x_i) + (W'_r)*(a_{i-1}) + b_3]$

➔ Potential value :

$a_{potential\_i} = \tanh[(W_{aa})*(a_{i-1})*(R_{gate}) + (W_{xa})*(x_i) + b_1]$

➔ Sigmoid update-function :

$u = sigma[(W'_a)*(a_{i-1}) + (W'_x)*(x_i) + b_2]$

➔ Decider equation :

$a_i = (u)*(a_{potential\_i}) + (1-u)*(a_{i-1})$

- In LSTM, the algorithm goes one-step beyond. As there is a Reset Gate in GRU, there are 3-gates in LSTM :- (equation of all gates are similar to that of reset gate, i.e. all are sigma function)
  i.) *Input Gate(in)* -- tells what will be the contribution of potential value (c_{t_cap}) in prediction(c_t) .
  ii.) *Forget Gate(f)* – tells what will be the contribution of past value (c_{t-1}) in prediction(c_t) .
  iii.) *Output Gate(q_{out})* – calculates activation function for next layer of LSTM.
  ➔ Prediction :
  $c_t = (in)*(c_{t\_cap}) + (f)*(c_{t-1})$
- LSTM is more complex algorithm than RNN & GRU. As they also store more memory for future predictions. They store – 3 gates, many weights, past value(c_{t-1}), and past activation function(a_{t-1}).

❖ Stationarity & Seasonality :

- A Plot is considered stationary, if it fulfills 3 statistical properties :
  1. Constant mean
  2. Constant variance
  3. No seasonality
- *Seasonality* : A repeating trend/pattern over time.
- *Check for Stationarity :*
  1. Visual inspection
  2. Global vs Local check
  3. ADF test (Augmented Dickey Fuller test) – statistical test.
- *Convert Non-stationary plot to Stationary plot* :
  1. Differencing. [Y(t) = Y(t) - Y(t-1)]
  2. Log operation. (to smooth exponential curves)
  3. Seasonal differencing. [Y(t) = Y(t) - Y(t-N)]
  4. Gauss filtering. (to smooth sharp edges in curves)

❖ Autoregressive Integrated Moving Average (ARIMA) -> Code description

- *arima model function* : The function below creates an ARIMA model rolling forecast for a given input time series. The various steps in the function include:

1. log transforming data : for smoothing exponential curves, if required, i.e. if curves have an exponential behaviour.
2. creating train/test splits
3. creating an ARIMA model for the train set : using the functions of statsmodels library.
4. forecasting the first value in the test set, followed by adding that value to the training set and remodeling, forecasting the next value in the test series, adding that second value to the train set, and so on.
5. inverse transforming the data : only those data, which were log transformed before.
6. creating plots and generating error metrics : Creating plots using matplotlib.pyplot module. And, calculating RMSE (Root Mean Squared Error) in each case using sklearn library inbuilt function, to evaluate how well the algorithms have performed on the testing data.

❖ Long short term memory (LSTM) -> Code description

The LSTM calculations use three different functions for creating a dataset that can be modeled. Each function is described in more detail below.

1. ***Creating the dataset*** :
   This function is used to create the datasets required for training and testing LSTM neural nets. It accepts a time series, the number of previous periods the user would like to model, the train / test split fractions, and whether to perform differencing or log transforms on the data to make it stationary. It will also scale all data between 0 and 1 for input into the LSTM.

2. ***Inverse transformations*** :
   This inverse transform function simply reverses any transformations performed when generating the dataset. Inversing the transformations allows for the model predictions to be based on the same scale as the original dataset for more intuitive interpretation of results. Both the model creation and inverse transformation functions are automatically called within the LSTM function below.

3. ***LSTM modelling*** :
   Calling the LSTM model only requires a time series dataset, the number of desired look-back periods, the train/test split, whether to log transform or difference the data, and parameters for training such as number of nodes and epochs. Within the function, it creates the train and test datasets—both features and targets—and then trains an LSTM model, followed by forecasting the out-of-sample data. The predictions from the model, as

well as the actual target values are then inverse transformed using the function above, and plots are generated along with error metrics.

❖ DATA COLLECTION :

- For this project I have used seven datasets which are following :

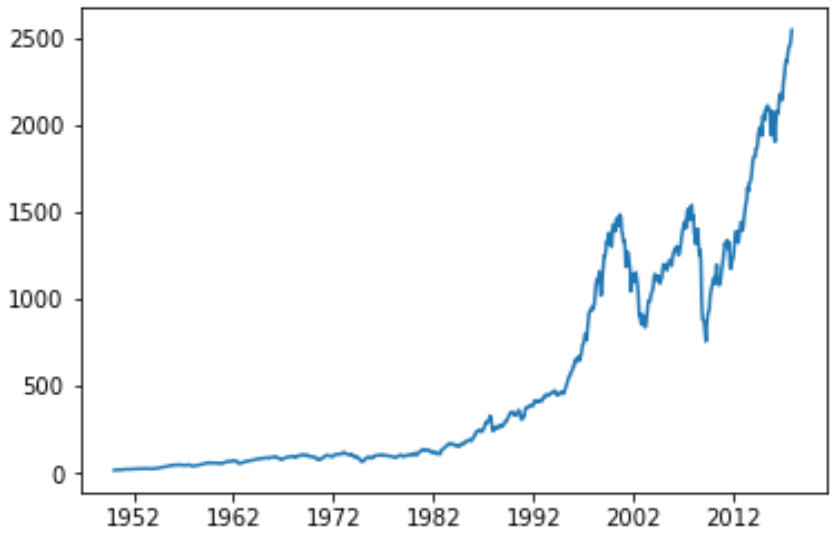
1) S&P 500 historical data (1733 * 6)
   - Modelled the closing index data of day-wise data.
2) "LeBron James" Google Trends (170 * 2)
3) "Coldbrew" Google Trends (170 * 2)
4) "Kentucky Derby" Google Trends (170 * 2)
5) "Gilmore Girls" Google Trends (170 * 2)
6) "Olympics" Google Trends (170 * 2)
7) "Zika Virus" Google Trends (170 * 2)

- *S&P 500 Dataset* :

Date	Open	High	Low	Close	Adj Close	Volume
1950-01-03	16.660000	16.660000	16.660000	16.660000	16.660000	1260000
1950-01-04	16.850000	16.850000	16.850000	16.850000	16.850000	1890000
1950-01-05	16.930000	16.930000	16.930000	16.930000	16.930000	2550000
1950-01-06	16.980000	16.980000	16.980000	16.980000	16.980000	2010000
1950-01-09	17.090000	17.090000	17.080000	17.080000	17.080000	3850000
...	...	...	...	...	...	...
2017-10-09	2551.389893	2551.820068	2541.600098	2544.729980	2544.729980	2483970000
2017-10-10	2549.989990	2555.229980	2544.860107	2550.639893	2550.639893	2960500000
2017-10-11	2550.620117	2555.239990	2547.949951	2555.239990	2555.239990	2976090000
2017-10-12	2552.879883	2555.330078	2548.310059	2550.929932	2550.929932	3151510000
2017-10-13	2555.659912	2557.649902	2552.090088	2553.169922	2553.169922	3149440000

17057 rows × 6 columns

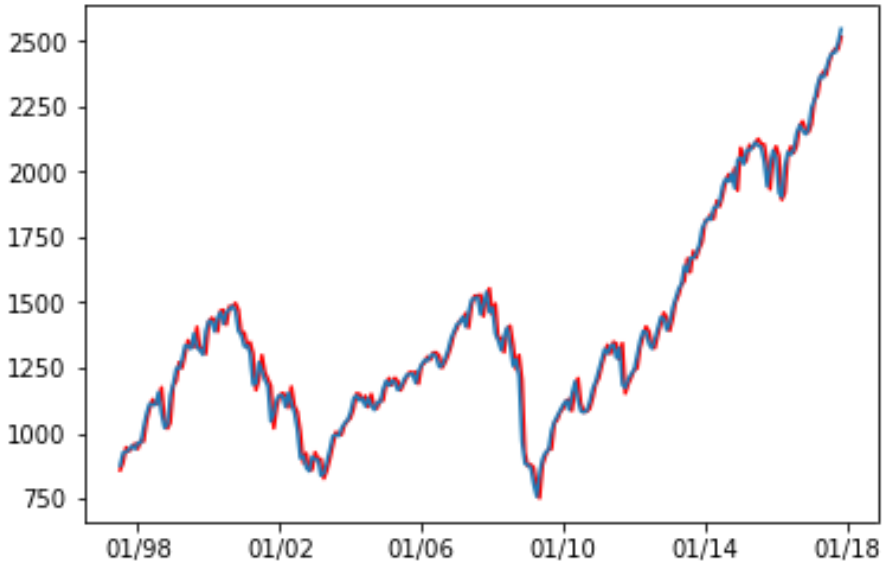
This is one of the dataset, & the code is processed only over the “Close” column of this dataset.



S&P 500 historical closing price data from 1950 through October 2017 (obtained through Yahoo Finance). The data is averaged monthly and plotted in the graph. This trend shows increasing values over time, along with some steep increases and decreases.

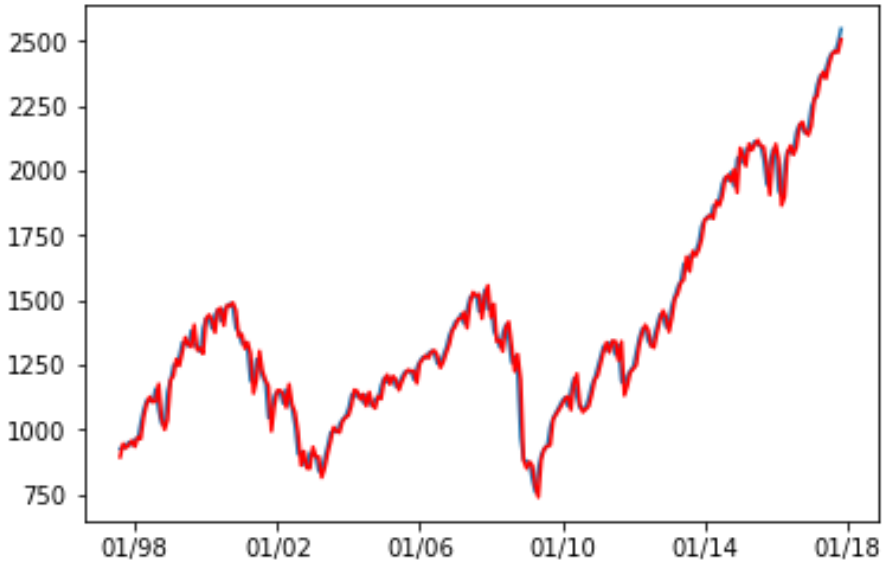
❖ S&P 500 ARIMA model :

- The ARIMA predictions below have an average RMSE value of 45.5 (with the range of actual values being from ~800 to 2500). One differencing period and log transformation was used.



❖ S&P 500 LSTM model :

- The LSTM model below also uses log transformation and differencing, with 5 training epochs to update model weights. It has a very similar average RMSE error of 46.7.



❖ **RESULTS** :

Data Set	ARIMA (test RMSE)	LSTM (test RMSE)	Gaussian ARIMA (RMSE)	Gaussian LSTM (RMSE)
S&P 500	45.495	46.671	24.570	24.751
LeBron James	20.009	27.175	9.816	13.743
Cold Brew	9.157	8.597	3.025	4.543
Kentucky Perby	30.935	30.224	15.232	15.232
Gilmore Girls	12.026	13.885	6.517	7.592
Olympics	17.420	18.506	9.146	11.078
Zika Virus	16.771	16.040	8.543	8.586

- The ARIMA model performed slightly better than the LSTM model, but the improvements were mostly negligible. Gaussing filtering the data improved accuracy for both models.

- Outcomes (for S&P 500 dataset) :
  - ARIMA RMSE (no filter): 45.50
  - LSTM RMSE (no filter): 46.67
  - ARIMA RMSE (filtered): 24.57
  - LSTM RMSE (filtered): 24.75

- From seeing performance in single dataset, we can judge the relative performance of different datasets.

- And from performances in different datasets, we can conclude that the algorithms are reliable, & can perform accordingly.

❖ **CONCLUSION** :

- Overall, the results demonstrate that both ARIMA and LSTM are quality algorithms for forecasting time series data. In general, the ARIMA model provided slightly lower errors, but also can suffer from convergence errors for series with sharp gradients. The LSTM series can train and make predictions on any series—though the accuracy must be evaluated.

- Further, Gaussian filtering of the dataset improved predictions in every case, even when comparing the filtered predictions to the original, un-filtered dataset.