Name: Rohith Kumar Poshala
rposhala@buffalo.edu

Project 2: Time Series Analysis and Prediction

For this project, **Apple Inc. stock data** of last 20 years (July 18th, 2000 to July 18th, 2020) has been picked. And this project focusses on forecasting this time series data with the help of both classical time series forecasting methods and Deep learning time series forecasting methods.

Apple Inc. stock data has been picked and the dataset has been loaded.

	Date	Open	High	Low	Close	Adj Close	Volume
0	2000-07-19	3.941964	4.058036	3.696429	3.763393	3.257959	114468200
1	2000-07-20	3.928571	4.075893	3.866071	3.937500	3.408683	116393200
2	2000-07-21	3.882813	3.973214	3.781250	3.825893	3.312065	49058800
3	2000-07-24	3.754464	3.776786	3.392857	3.477679	3.010617	103042800
4	2000-07-25	3.593750	3.616071	3.504464	3.575893	3.095641	52901800

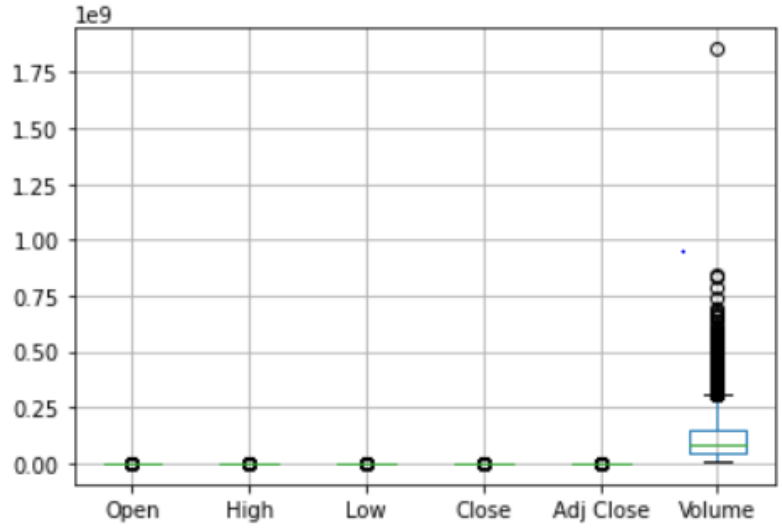
This dataset has 5031 entries with 7 columns.

Below are few statistics about the dataset:

	Open	High	Low	Close	Adj Close	Volume
count	5031.000000	5031.000000	5031.000000	5031.000000	5031.000000	5.031000e+03
mean	67.773052	68.462225	67.097275	67.807873	63.818878	1.131938e+08
std	75.908342	76.728740	75.209210	76.020822	74.919350	9.911286e+07
min	0.927857	0.942143	0.908571	0.937143	0.811282	9.835000e+06
25%	6.243571	6.312500	6.151785	6.252857	5.413082	4.487465e+07
50%	37.111427	37.450001	36.607143	37.040001	32.065430	8.461610e+07
75%	108.584999	109.350003	107.354999	108.525002	100.943378	1.505732e+08
max	395.959991	399.820007	385.959991	390.899994	390.899994	1.855410e+09

We can observe that all columns except volume are in similar range.

Dataset has been visualized to gain insights regarding the data.

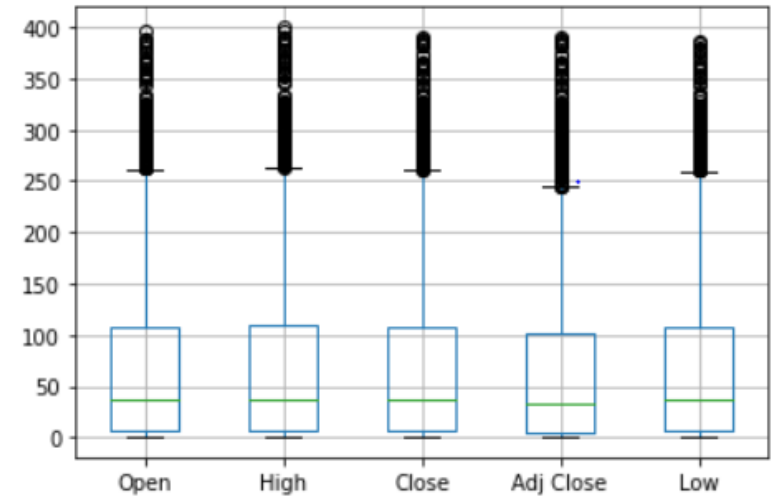


We can see that the values of the volume column is clearly dominating the above plot, leaving us with no specific insights.

As specified above, we can see 'volume' column dominating other columns leaving us with no insights about data.

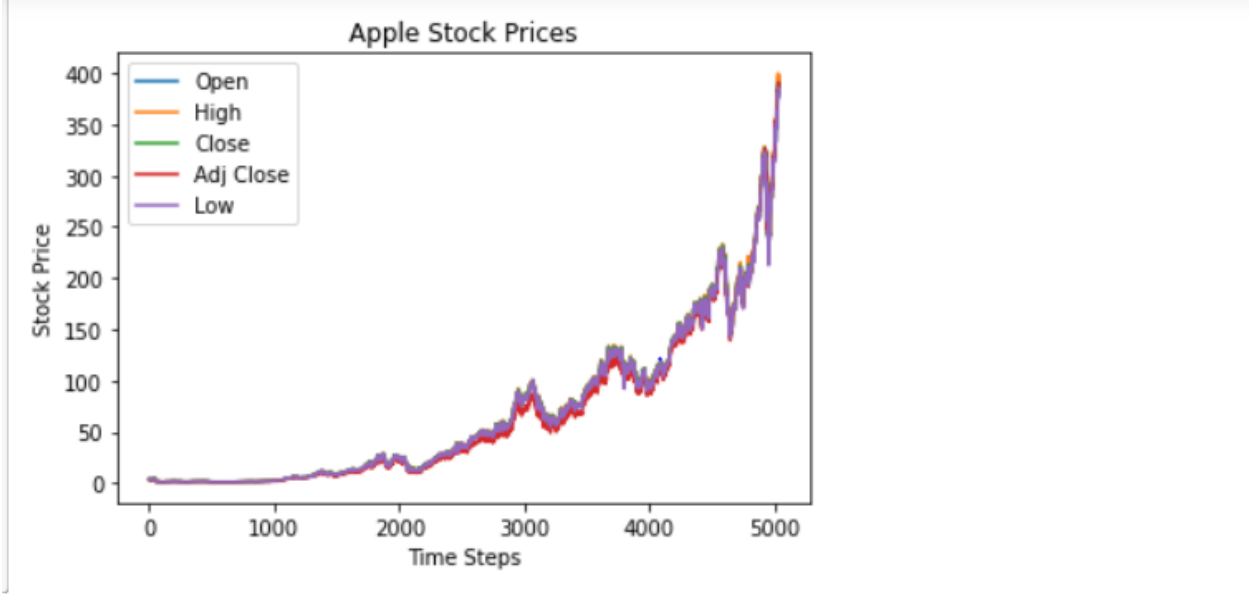
So, volume column has been excluded to visualize data more clearly.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe27784a160>
```



We can observe that values of all columns are very much close to each other. This implies that the values for each column of a single entry varies in very shorter range.

Stock data across all time steps (in the last 20 years) for each column separately excluding 'Volume' column are plotted.



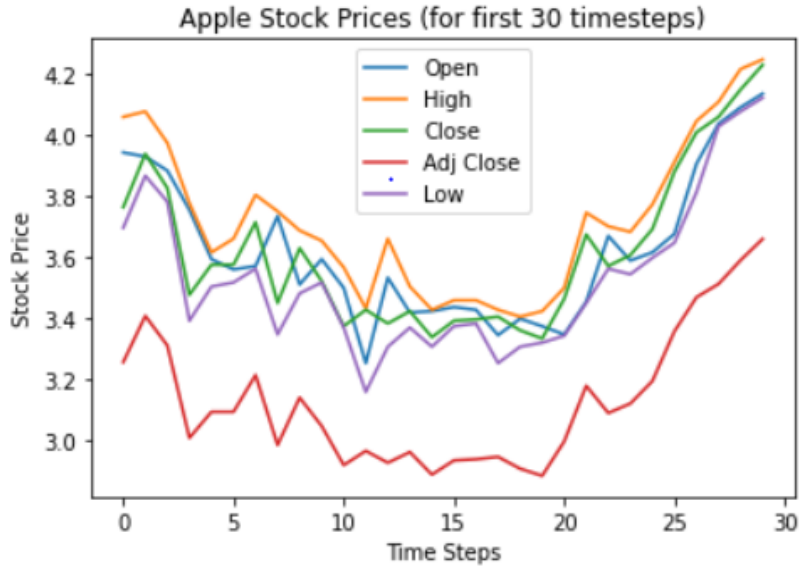
We can observe that columns ('Open', 'close', 'high', 'low', 'Adj Close') are very much aligned with each other through out entire timeline.

For better visualization, taking a specific time frame.

```
Text(0.5, 1.0, 'Apple Stock Prices (for last 30 timesteps)')
```

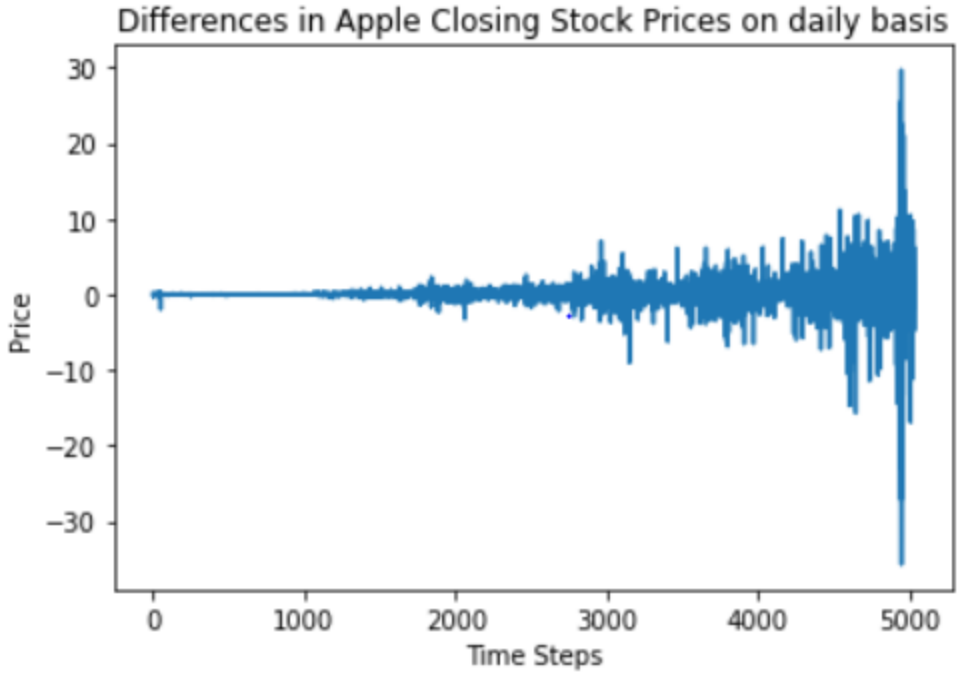
And for first 30 time-steps:

```
Text(0.5, 1.0, 'Apple Stock Prices (for first 30 timesteps)')
```



From the above two plots, we can observe that for last 30 time steps (last few months), the adjusted close value is almost same as close value but for first 30 time steps, the adjusted close value can be seen very much below low value. And another obvious insight is open and close is always between Open and Close.

Visualizing the Apple stock data and its differences in stock prices on daily basis to understand the trend, stationarity and other few properties.



Looking at the above graph 'Differences in Apple Closing Stock Prices on daily basis', we can observe that the difference of closing stock price from before day got fluctuated largely between time steps from around 4500 to 5000.

Similar insight can be observed from the above 'Apple Closing Stock Prices' graph, there was a sudden dip and huge rise in the closing stock price from around 4500 to 5000.

Pre-processing the dataset for training: We can see below that there are no null values in the dataset. Dataset is already clean.

```
: data.isnull().sum()

: Date         0
  Open         0
  High         0
  Low          0
  Close        0
  Adj Close    0
  Volume       0
  dtype: int64
```

Splitting the dataset into Training set and Testing set (it is also used as Validation set):

Classical time series forecasting methods:

1. Choosing features and targets in the dataset:

   Target variable in the dataset is 'Close'.

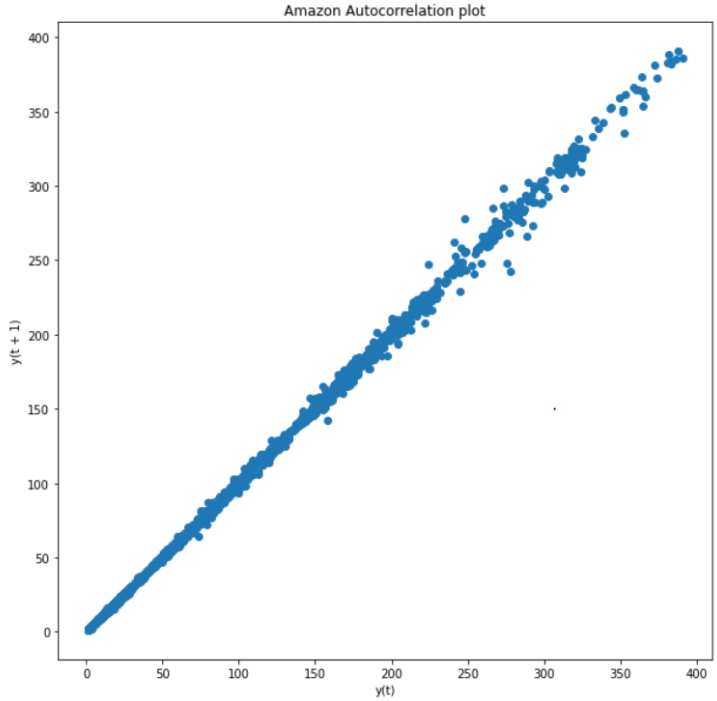
   Feature variable is 'Close'.

   (We are going to predict the next upcoming closing stock value based on the previous closing stock values).

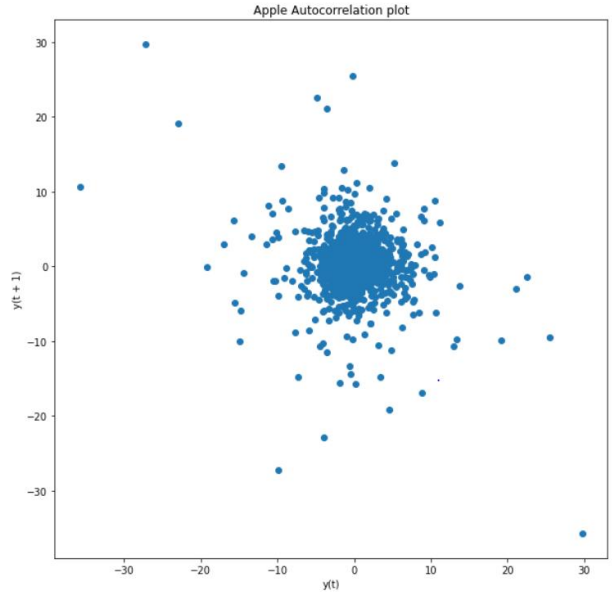
   And the reason for not picking remaining variables is:

   From the above visualizations, it is clear that except volume all other columns are very much correlated.

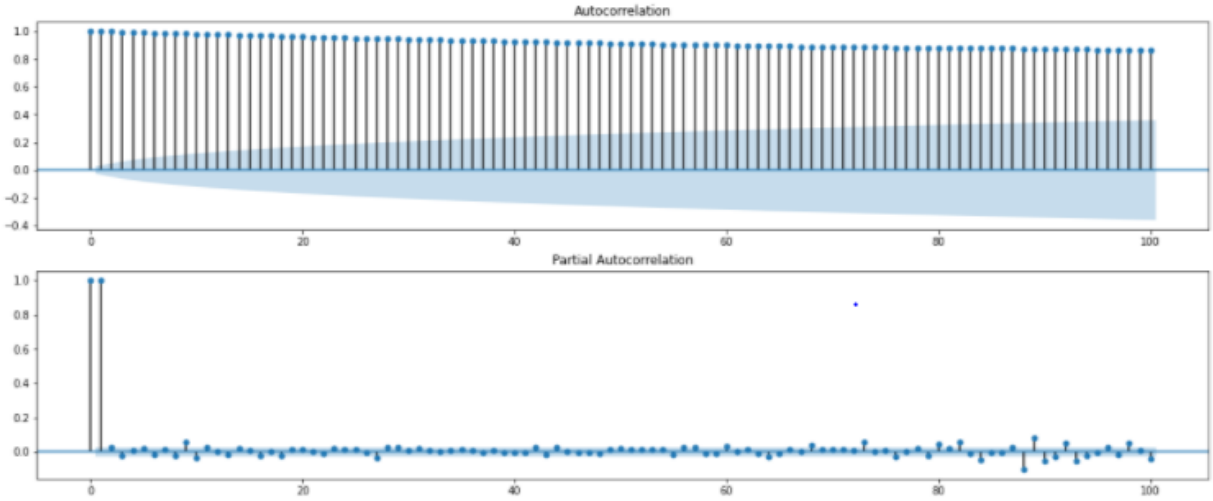
2. Applying statistical algorithms to forecast the values:



We can see that the closing prices are very much correlated with the previous day closing prices.



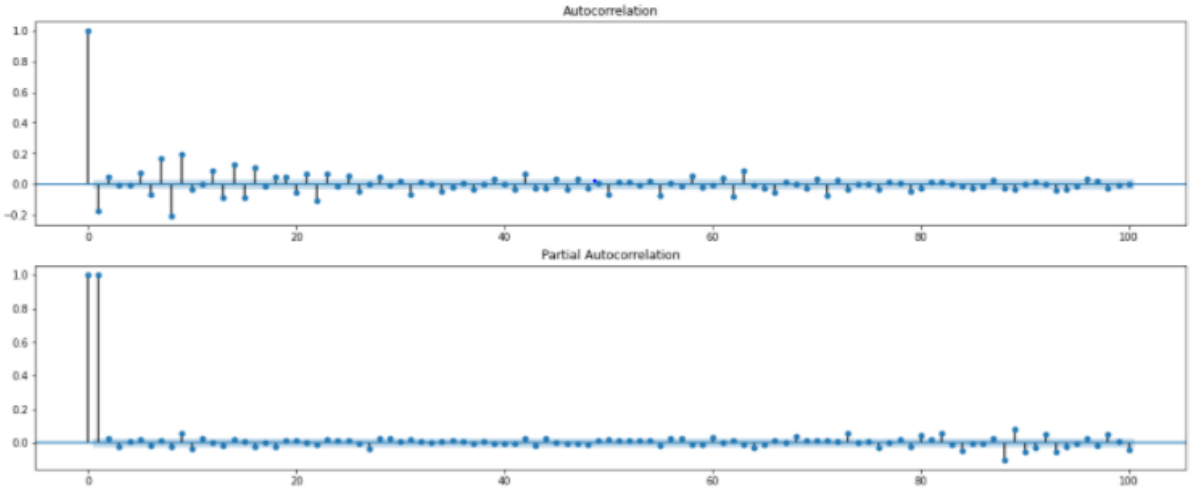
We can observe that most of the difference values are near zero indicating that the closing value varied very little on from a day to next day. The points which are far from the cluster indicate that in very few instances, the closing value hugely dropped or raised compared to previous day.



From the above plots showing Auto- correlation function and Partial Auto-correlation function, we can see that in Autocorrelation plot (it considers direct and indirect effect) even the closing values of stock from a day which is 100 days far from the current day effects the current day closing stock price to greater extent. Whereas in case of Partial Autocorrelation plot (it considers only direct effect) the closing price value of the current day mostly depends on the previous day and negligibly small on any other previous day.

In order to make the dataset stationary, we need to look for at different degrees of differencing starting from 1.

As for many stationary models to perform better, stationarity is the necessary condition.



By shifting by one, we are imposing a degree of differencing as 1. And we can observe that the in the Auto correlation plot the dependencies of current day value on previous days are almost negligible now. Indicating that the data is now stationary.

If this did not work, our next step would be ton check for degree of differencing 2 (by shifting by 2).

As we can see data is made stationary by shifting by 1. parameter d would be 1 if ARIMA is used.

Details about degree of differencing (shifting by d):

If d=0: $y_t = Y_t$

If d=1: $y_t = Y_t - Y_{t-1}$

If d=2: $y_t = (Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$

Implementing ARIMA with lag order: 5, degree of differencing: 1 and order of moving average: 0

ARIMA model is a class of statistical models for analyzing and forecasting time series data

AR = Auto Regressive, A model that uses the dependent relationship

between an observation and some number of lagged observations.

I = Integrated, The use of differencing of raw observations (e.g.

subtracting an observation from an observation at the previous time

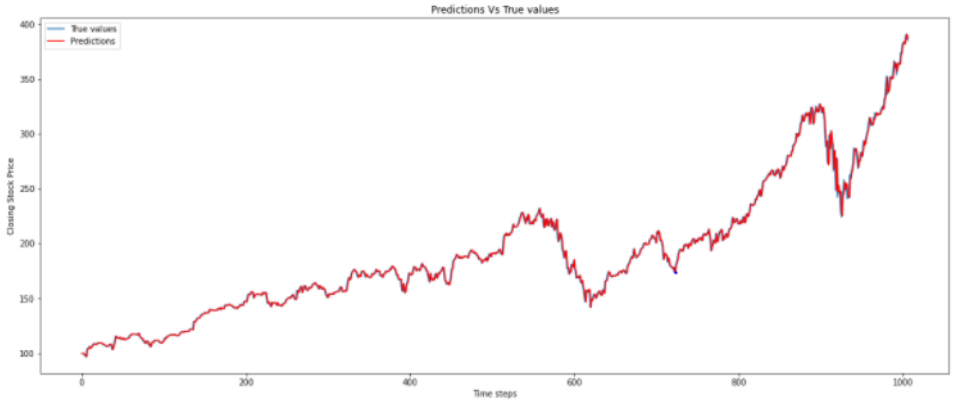
step) in order to make the time series stationary.

MA = Moving Average, A model that uses the dependency between an

observation and a residual error from a moving average model applied

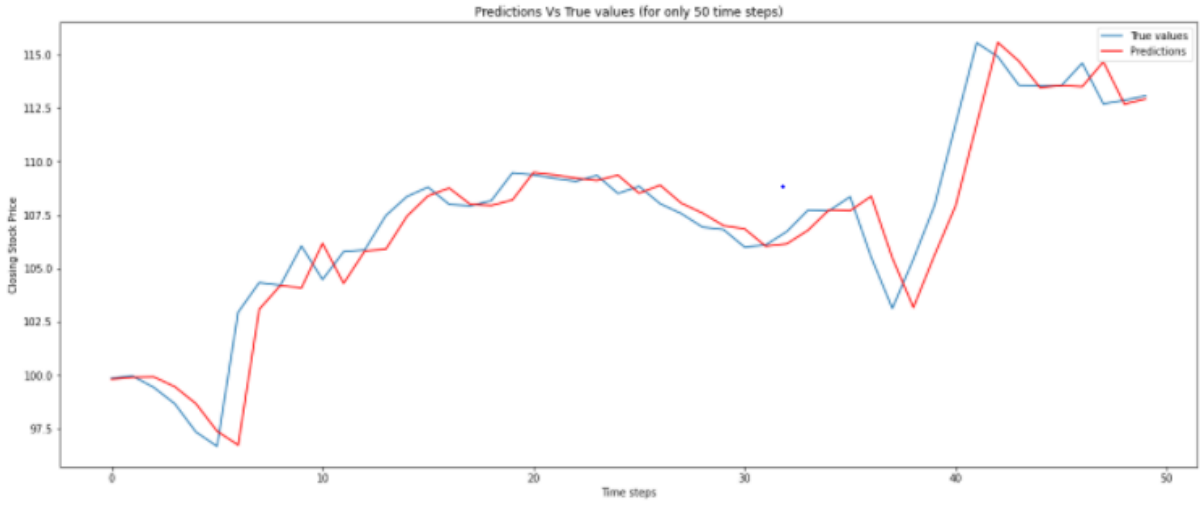
to lagged observations.

lag order and degree of differencing is decided from the insights of the above auto-correlation plot.



It looks like predictions made by ARIMA are exactly matched with true values which is not true.

Plotting for only 50 time-steps shows us the difference between predictions made by ARIMA to the true values.



Result:

```
Testing Mean Squared Error:  17.206066781441177

Testing Mean Absolute Error: 2.44169199947831
```

ARIMA Model Results:

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	5029
Model:	ARIMA(5, 1, 0)	Log Likelihood	-10573.084
Method:	css-mle	S.D. of innovations	1.981
Date:	Wed, 22 Jul 2020	AIC	21160.168
Time:	02:07:10	BIC	21205.828
Sample:	1	HQIC	21176.166

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0762	0.026	2.915	0.004	0.025	0.127
ar.L1.D.y	-0.1699	0.014	-12.077	0.000	-0.197	-0.142
ar.L2.D.y	0.0192	0.014	1.348	0.178	-0.009	0.047
ar.L3.D.y	0.0025	0.014	0.175	0.861	-0.026	0.031
ar.L4.D.y	0.0042	0.014	0.296	0.768	-0.024	0.032
ar.L5.D.y	0.0756	0.014	5.371	0.000	0.048	0.103

Roots

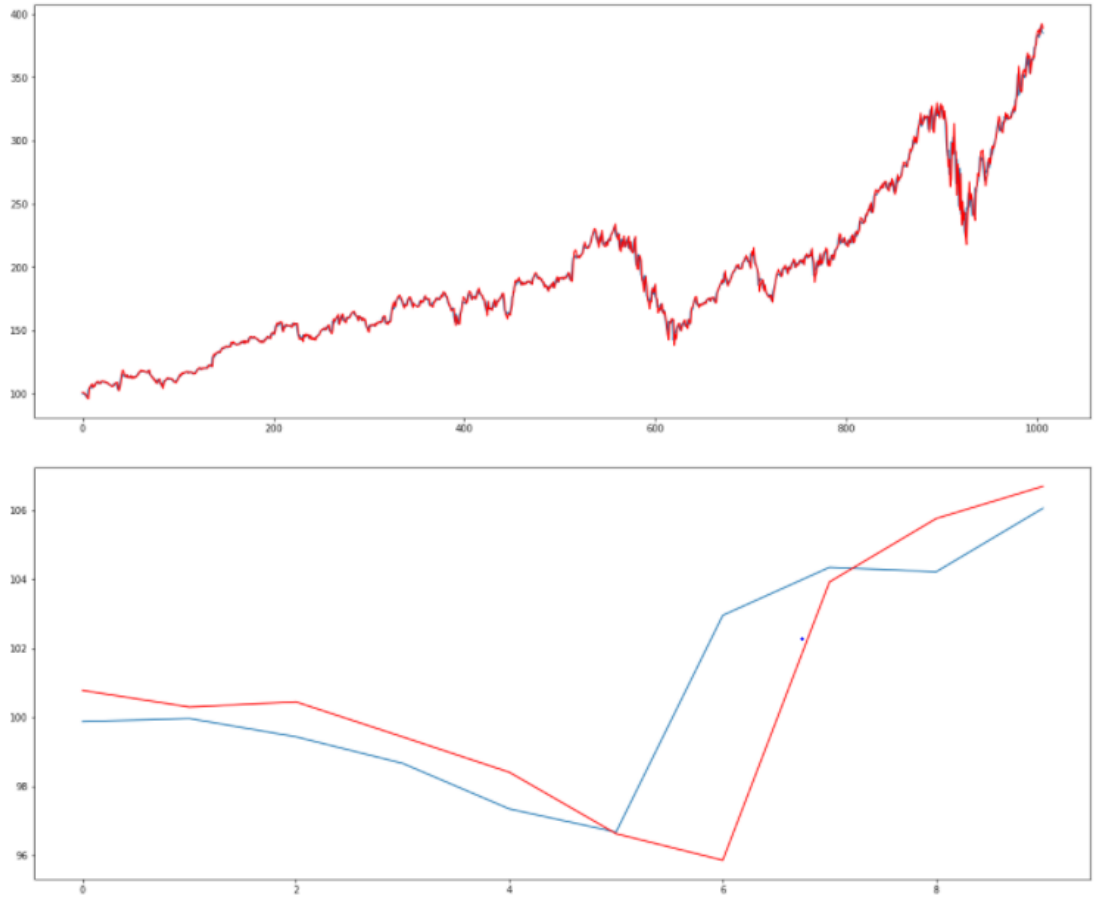
	Real	Imaginary	Modulus	Frequency
AR.1	-1.3422	-0.8727j	1.6010	-0.4082
AR.2	-1.3422	+0.8727j	1.6010	0.4082
AR.3	1.7340	-0.0000j	1.7340	-0.0000
AR.4	0.4473	-1.6658j	1.7248	-0.2082
AR.5	0.4473	+1.6658j	1.7248	0.2082

Implementing statistical model SARIMA with order = (p=2, d=1, q=0) and seasonal order = (P=2, D=1, Q=0, m=1):

Another SARIMA model with order = (p=5, d=1, q=0) and seasonal order = (P=5, D=1, Q=0, m=4) is attached in a separate. ipynb file.

Seasonal Autoregressive Integrated Moving-Average (SARIMA), This method models the next step in the sequence as a linear function of the differenced observations, errors, differenced seasonal observations, and seasonal errors at prior time steps.

It combines the ARIMA model with the ability to perform the same autoregression, differencing, and moving average modeling at the seasonal level.



When plotted for entire time line, it looked like predictions made by ARIMA are exactly matched with true values which is not true.

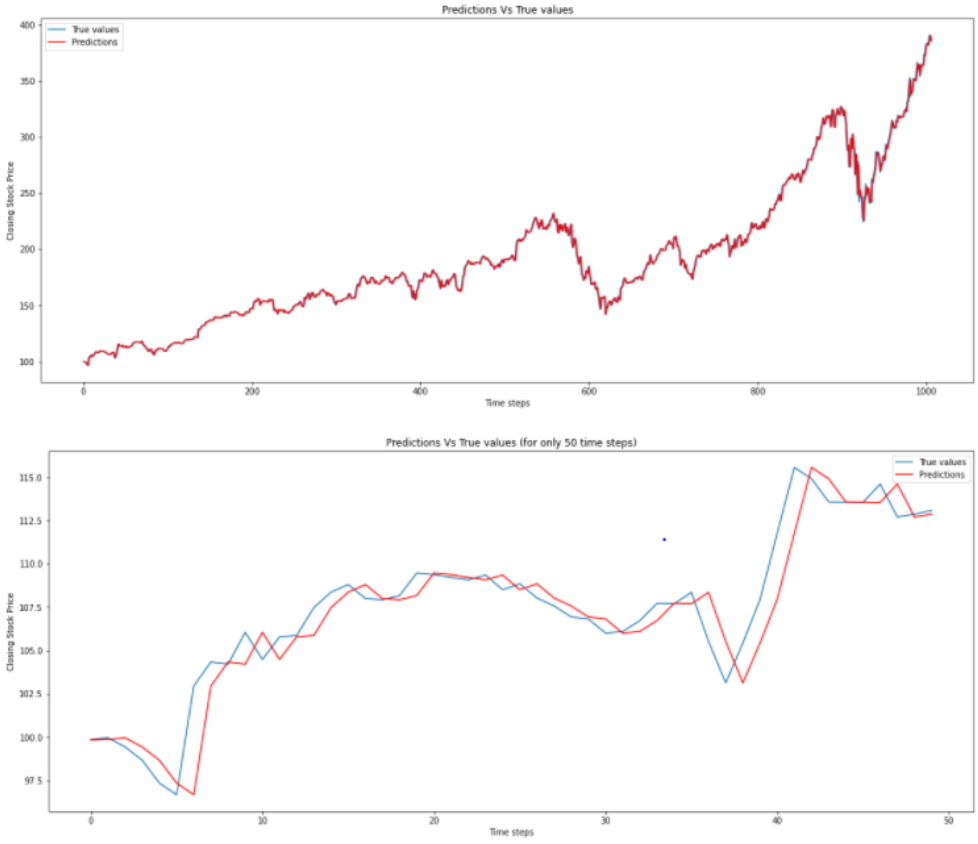
Plotting for only 10 time-steps shows us the difference between predictions made by ARIMA to the true values.

Results:

```
Testing Mean Squared Error: 21.455
Testing Mean Absolute Error: 2.769
```

Implementation of statistical model: Simple Exponential Smoothing:

The Simple Exponential Smoothing (SES) method models the next time step as an exponentially weighted linear function of observations at prior time steps.



When plotted for entire time line, it looked like predictions made by ARIMA are exactly matched with true values which is not true.

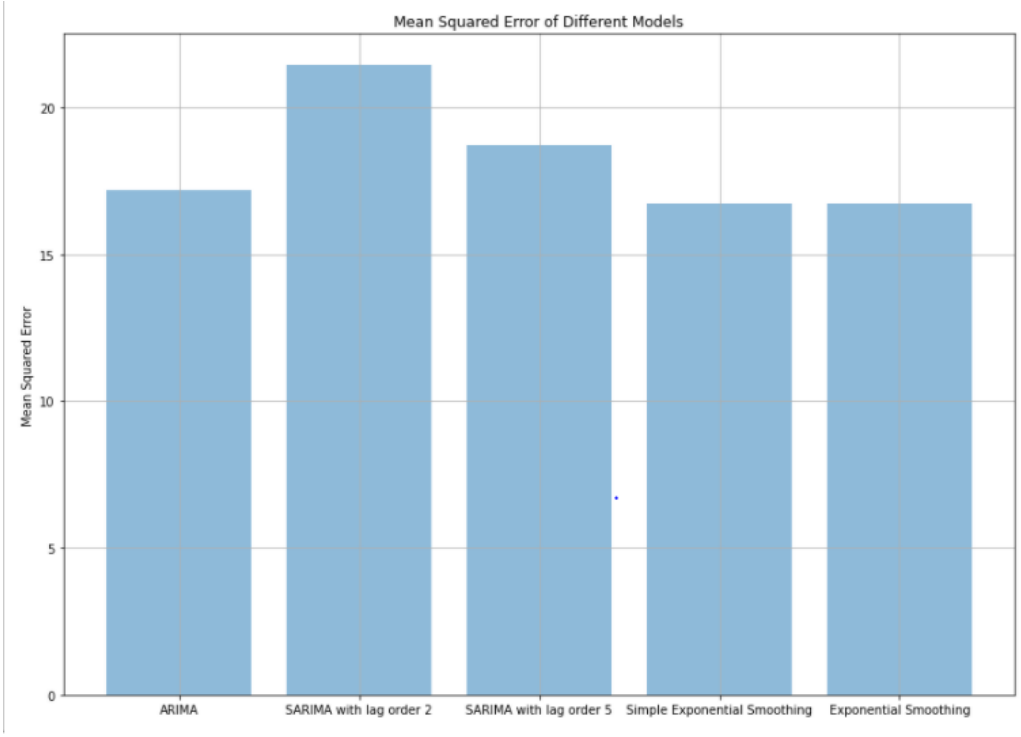
Plotting for only 50 time-steps shows us the difference between predictions made by ARIMA to the true values.

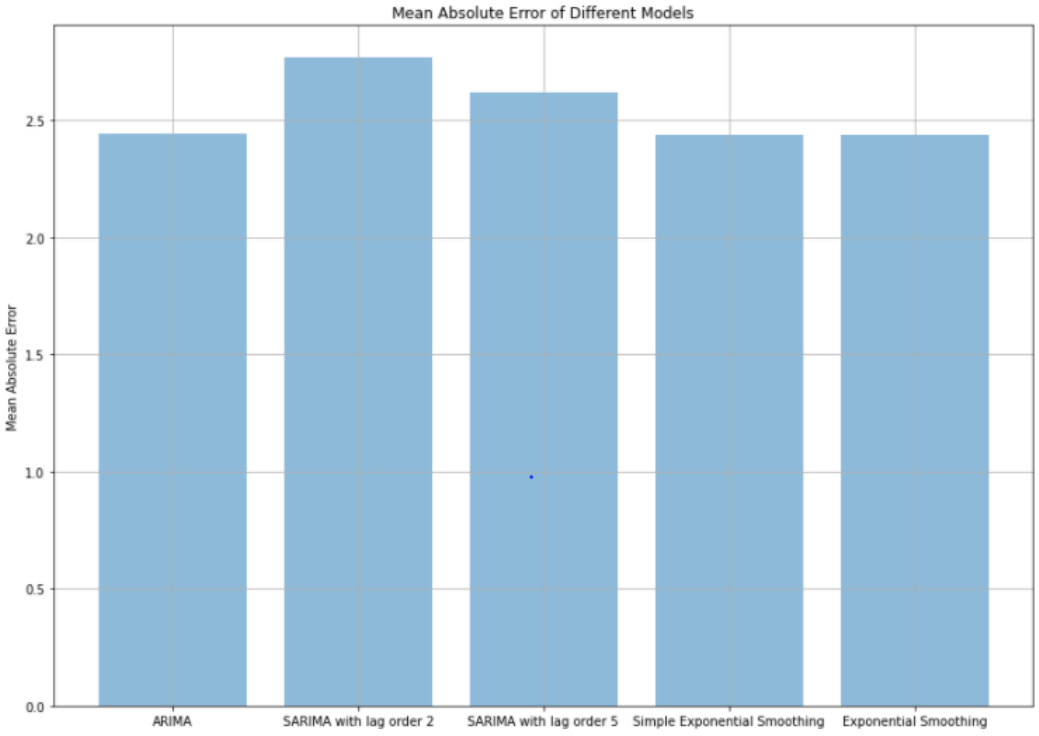
Results:

```
Testing Mean Squared Error:  16.748315489274617

Testing Mean Absolute Error: 2.4358455919559447
```

3. Comparison of the results of different statistical models used above:





Observations: We can see that both Mean Squared Error and Mean Absolute Error of Simple Exponential Smoothing, Exponential Smoothing are same and less than

rest. Even ARIMA is very much closer to the above two statistical models. SARIMA model with lag order of 2, sessional period 1 had higher error than rest followed by SARIMA model with lag order of 5, sessional period 4.

Maybe we may achieve better results if we correctly pick suitable sessional period. Recent irregular variations may have disturbed the seasonality of the data.

Deep learning time series forecasting methods:

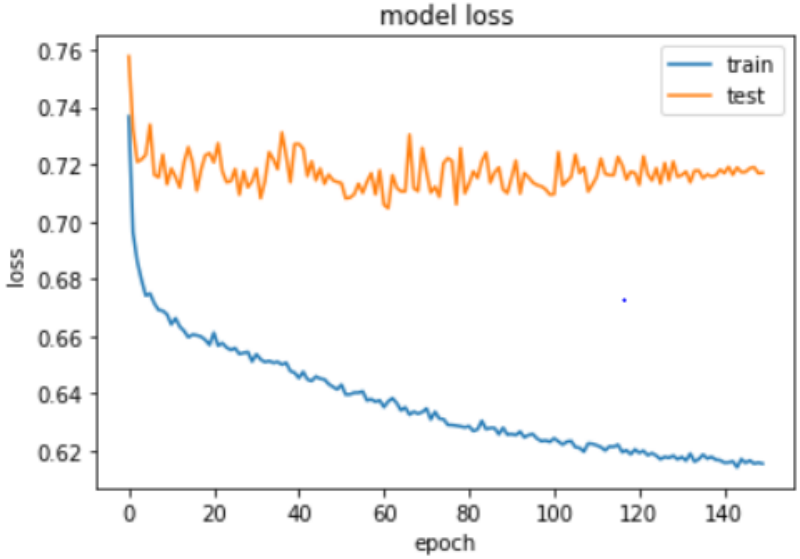
Split of data into train and test was done by defining window size, forecast size and step size. Dataset was built similar to sliding window of above determined window size by moving with the size of each step.

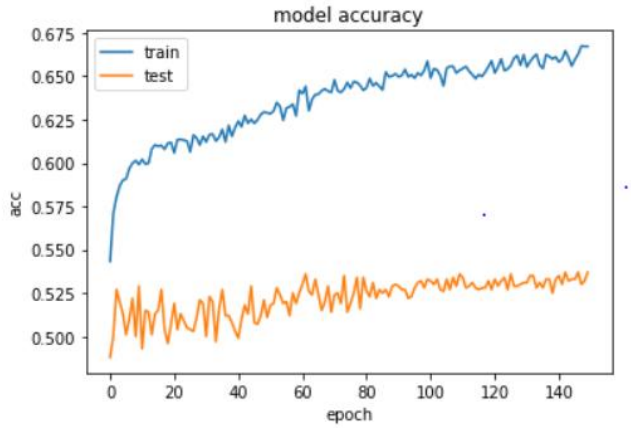
Size of the training and testing data:

```
Shape of X_train:  (3996, 30, 1)
Shape of X_test:  (1000, 30, 1)
Shape of Y_train:  (3996, 2)
Shape of Y_test:  (1000, 2)
```

Applying MLP to predict the increase or decrease of closing stock price value along with Accuracy and Loss graphs:

a) Type 1: Basic MLP Layer with a hidden dense layer of 64 units. Adam Optimizer and SoftMax activation for output.

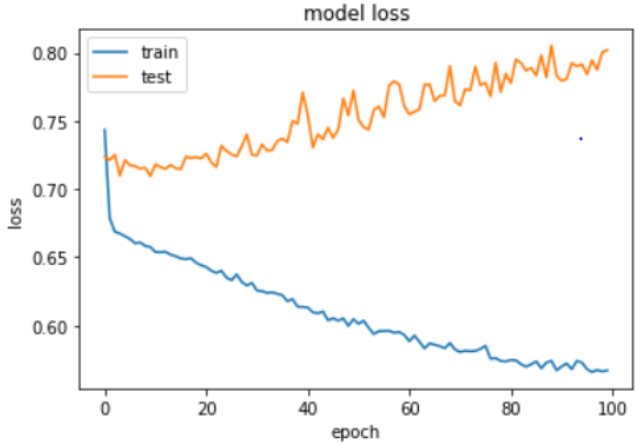


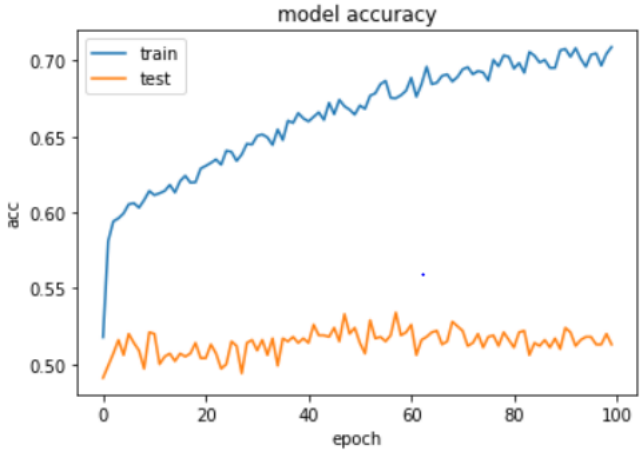


Results:

From the above Model Accuracy graph, we can observe that the train accuracy went up to 67.5 and test accuracy fluctuated around 50 to 53.

b) Type 2: MLP Layer with 2 hidden dense layers. Adam Optimizer and SoftMax activation for output. Number of hidden layers of the MLP are increased to allow the model to perform better by training it little deeper.

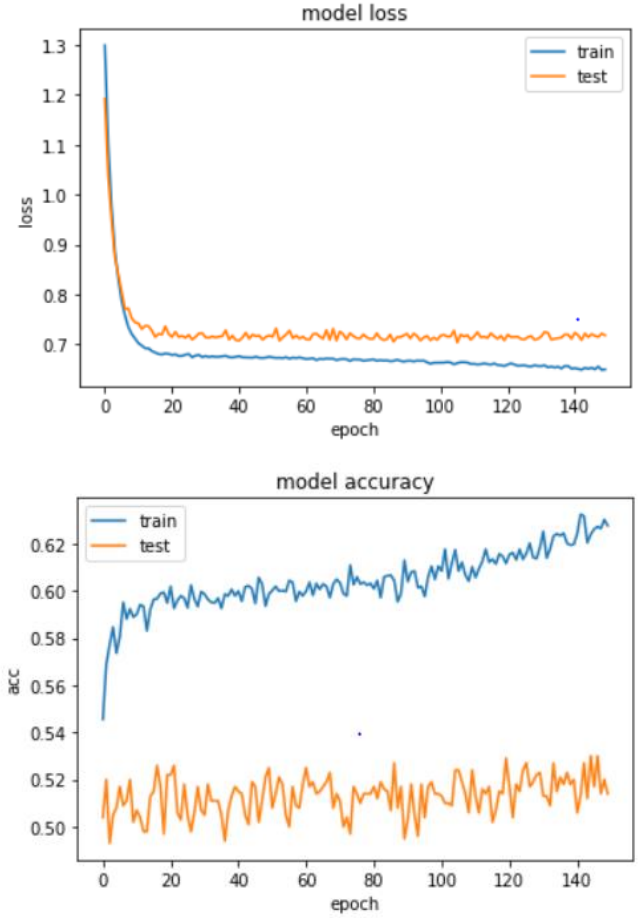




Results:

From the above Model Accuracy graph, we can observe that the train accuracy went up to 70.5 and test accuracy fluctuated around 50 to 52. Training accuracy increased indicating Overfitting.

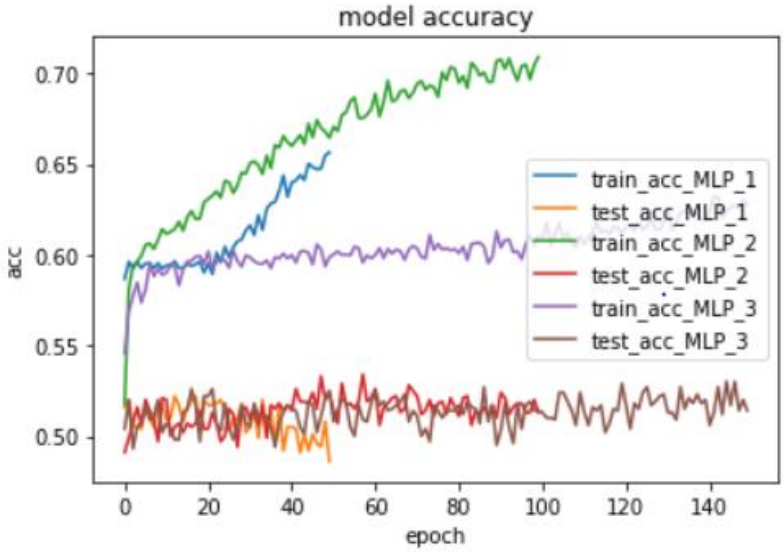
c)Type 3: Basic MLP Layer with a multiple hidden dense layer along with regularizes and dropouts to avoid the overfitting. Adam Optimizer and SoftMax activation for output. Expecting our model to avoid overfitting and also to bridge the gap between training and testing accuracy.



Results:

From the above Model Accuracy graph, we can observe that the train accuracy went up to 62.5 and test accuracy fluctuated around 49 to 53. We could see that regularizations included helped the model to reduce overfitting and test accuracy seems to near the training accuracy, which implies that the variance is reduced by limiting overfitting.

Comparison: Comparing training and testing accuracies of the above MLP models.

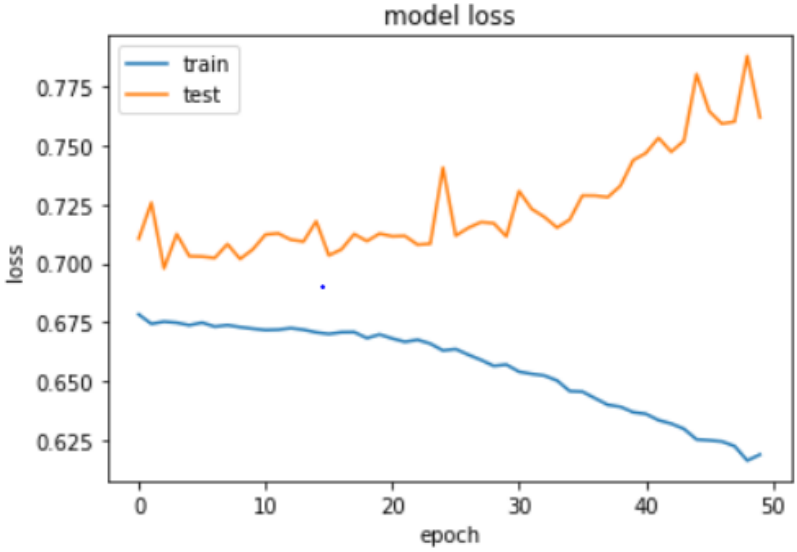


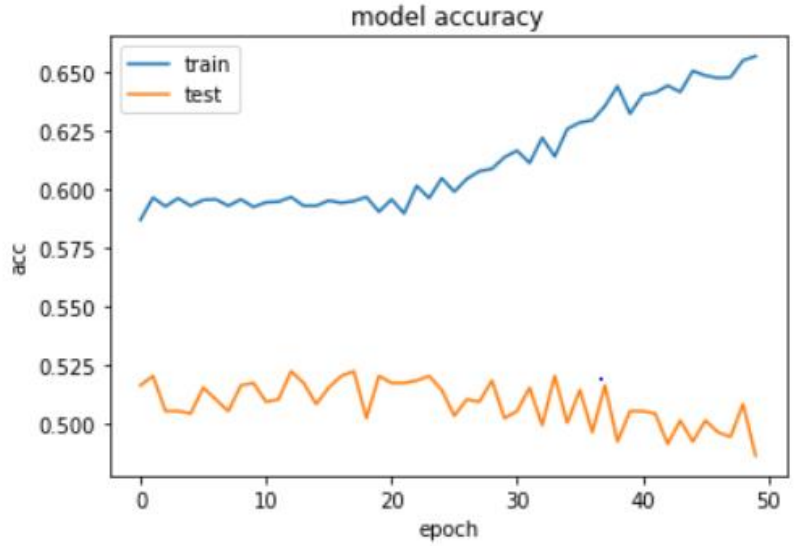
We can observe that effect of regularization on the training accuracy and better accuracy for test in MLP 3.

Applying LSTM architecture to predict the value along with Accuracy and Loss graphs:

```
Model: "sequential_13"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 30, 64)            16896
_________________________________________________________________
dropout_6 (Dropout)          (None, 30, 64)            0
_________________________________________________________________
lstm_3 (LSTM)                (None, 30, 64)            33024
_________________________________________________________________
dropout_7 (Dropout)          (None, 30, 64)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1920)              0
_________________________________________________________________
dense_29 (Dense)             (None, 2)                 3842
_________________________________________________________________
activation_11 (Activation)   (None, 2)                 0
=================================================================
Total params: 53,762
Trainable params: 53,762
Non-trainable params: 0
```

Performance Analysis:





Results:

We can observe that test accuracy varied between 49 to 52.5 and training accuracy increasing from 58 to 65.

Results of predicting the values using different deep learning structures:

From the results obtained from LSTM and various MLP architectures, we can derive that MLP architectures are more likely overfitting the model compared to LSTM. And investing time on getting better test Accuracy of the LSTM model looks feasible than MLP architectures.