Predict Stock Prices using Deep Learning

Machine Learning Engineer Nanodegree

Capstone Project

Definition

In this MLND capstone project, I will be attempting to predict future stock prices for selective stock symbols. I will take historical stock price data for specific Stock symbols from yahoo finance and use deep learning techniques to train a model. This model will then be used to predict prices for next couple of days.

As a reference, I will be using Machine Learning techniques taught in Udacity's Machine Learning NanoDegree and Machine Learning for Trading course.

Python language and Keras[6] library will be used for application development.

Domain Background

Predicting stock prices has long been a subject of interest for traders, mathematicians, statisticians and lately technologists. This has generated a lot of interest as one will be able to reap enormous profits if one is able to predict stock prices and stock market movements reflect, to some extent, the economic state of the country also. This has also been the field of research and interest as theoretical background of financial theory does not support stock market prediction.

Theoretically, there are two major hypotheses rejecting the predictability of future stock prices.

1. **Efficient Market Hypothesis**
2. **Random Walk Theory**

Efficient Market Hypothesis[1]

The efficient market hypothesis is an investment theory, proposed by Eugene Fama, which suggests that stock prices reflect all currently available information and so it is not possible to predict stock prices. This means that stock prices adjust quickly to any recently released information and past stock price history (which is publicly available) is also already reflected in the stock prices making it impossible to predict future prices. There are three versions of the efficient market hypothesis (EMH), all based on varying assumptions of price efficiency. The **weak** form of EMH claims that the prices of publicly-traded assets already reflect all available information, and past prices are of little value in predicting future trends. The **semi-strong** version of EMH holds that while prices are efficient, they react instantaneously to new information, while the **strong** version of EMH maintains that asset prices reflect not just public knowledge, but private insider information as well.

Random Walk Theory[2]

The **random walk theory hypothesis** is a financial theory which states that stock market prices evolve as per the random walk and thus cannot be predicted. The Random Walk Theory term was popularized by the 1973 book, *A Random Walk Down Wall Street*, by Burton Malkiel, and this was used earlier in Eugene Fama's 1965 article "Random Walks In Stock Market Prices".

There are both proponents and opponents to these hypotheses.

> Proponents believe that it is pointless to search for undervalued stocks or to try to predict trends in the market through either fundamental or technical analysis. They conclude that, because of the randomness of the market, investors could do better by investing in a low-cost, passive portfolio. A study done by Morningstar Inc found out that 25% of the top-performing active managers are able to consistently outperform their passive manager counterparts.
>
> Opponents believe that the market is predictable to some degree. Martin Weber, a leading researcher in behavioral finance, has performed many tests and studies on finding trends in the stock market. In one of his key studies, he found that stocks with high price increases in the first five years tended to become under-performers in the following five years. He cited this as a key contributor and contradictor to the random walk hypothesis. Other opponents give examples of investors such as Warren Buffett who has consistently beaten the market over long periods of time. Some give reference to events such as the 1987 stock market crash, when the Dow Jones Industrial Average (DJIA) fell by over 20% in a single day, as evidence that stock prices can seriously deviate from their fair values.

Prediction methodologies fall into three broad categories - Fundamental analysis, Technical analysis and Technological methods.

Fundamental analysis

In FA, basis of analysis is company's past performance. Fundamental analysis is built on the belief that human society needs capital to make progress and if a company operates well, it should be rewarded with additional capital which results in a surge in stock price. Fundamental analysis is widely used by fund managers as it is the most reasonable, objective and made from publicly available information like financial statement analysis. Following indicators related to past performance are used for FA.

Technical analysis

Technical analysis is not based on company's fundamentals. In TA, the future price of a stock is calculated solely on the basis of the (potential) trends of the past price (a form of time series analysis). Numerous patterns are employed such as the head and shoulders or cup and saucer. Alongside the patterns, statistical techniques are used such as the exponential moving average (EMA). Candle stick patterns are believed to be first developed by Japanese rice merchants, and nowadays widely used by technical analysts.

Technological methods

With the advent of the digital computer and recent developments in very high computing capacity, stock market prediction has evolved to a great extent. The algorithmic trading techniques such as Artificial Neural Networks (ANNs) and Genetic Algorithms are used to predict stock prices.

Personal motivation

Finance field has quite intrigued me since last couple of years. Seeing conflicting views about predicting stock prices, I myself wanted to check whether it is really possible to predict prices. During my MLND, I saw quite interesting implementations of Machine Learning techniques and wanted to utilize Deep Learning techniques to predict stock prices. Further motivation was to be able to apply the algorithm in real time to see if it will really result in stock market profits.

Problem Statement

The problem to be solved in this project is to predict the future stock prices for next couple (5, 10, 20) of days.

This problem will be structured as a supervised learning problem where the input dataset will be framed in such a way that it will have input and matching output values. A sliding window technique will be used to traverse input dataset and matching output. This will be considered a regression type of supervised learning as the output will be a continuous real value.

The input dataset will be the historical daily prices(of the chosen Stock symbol) dataset downloaded from the Yahoo finance website[8]. Adjusted Close price for each day will be considered as an input. The input values indexed on date will be treated as a time series(sequence). This dataset will be split into training, validation and test datasets. The training dataset will further be traversed using a sliding window technique. A sliding window with a fixed number of Adj Close values will be the input values and fixed number(depending upon the number of values to be predicted) of values immediately after the sliding window in sequence will be the matching output values. Validation dataset will be used to test model during training and Testing dataset will be used to test the final model.

A model will be developed matching input window values to the output values using Deep Learning. After developing the model using training dataset, stock price will be predicted for the period starting where the training dataset period ends. The predicted values can be compared against the testing dataset values for evaluation. The closeness of predicted values to actual values can be measured using Kera's evaluate method or using RMSE method.

Datasets and Inputs

Data for the project is taken from Yahoo finance website[8]. Historical data of following stock tickers is taken – S&P500, MSFT, XOM, MYL, WMT, PFE, IBM, AAPL, GOOG.

All available historical data for these stocks from yahoo finance website[8] will be taken. Different stocks have different periods of data available.

Start and End dates for each stock are as shown below. End date may change if more data is available at the time of submission of project.

Stock Symbol	Start Date	End Date
MSFT	1986-03-13	2017-07-07

XOM	1980-12-12	2017-07-07
MYL	1980-12-12	2017-07-07
WMT	1980-12-12	2017-07-07
PFE	1980-12-12	2017-07-07
IBM	1980-12-12	2017-07-07
SP500	1980-12-12	2017-07-07
AAPL	1980-12-12	2017-07-07
GOOG	2004-08-19	2017-07-07

The historical data will be collected for following events – Open, Close, High, Low, Adj Close, Volume. Here, we will use only Adj Close[5] price of the stocks. Adj Close price is similar to daily close price, only difference is that it takes care of past splits, reverse splits, dividends, rights offerings also.

Final dataset will contain Adj Close column with Date as the index. Dataset will be cleaned for any "null" values and it will be made sure that Adj Close data only have float values. Pandas dataframe dropna function will be used to drop any rows containing undefined data.

After data cleanup, it will be normalized so that its values are scaled between -1 and 1. It will help in convergence of machine learning algorithm and it will also help in bringing different stocks to the same scale.

Datasets for each stock will be split into training and test datasets in the ratio of 60/40 or 80/20.

A fixed window of data values from sequence will be taken as inputs and the a fixed number(depending upon the number of predicted values) of values just after the input window is taken as outputs. Window is rolled forward repeatedly to traverse the input dataset to get inputs and matching outputs.

Although chances of testing data information being trickled to training are negligible, as testing is done after model is fully trained using Keras

To avoid testing dataset information trickle down to training dataset -

- Testing data is used after model is fully trained
- Training and testing will be done on different stock symbols.

Solution Statement

In this project, attempt will be made to predict the prices using Deep Learning techniques of Recurrent Neural Networks. Recurrent Neural Network model LSTM (Long short term Memory) will be used to predict the stock prices.

Benchmark Model

Multiple benchmark models will be used for comparison:

1) As we are using existing data and we have test data available for which actual values are already available. We can test our model generated predicted values with these actual values and can see how accurate our model is.
2) Use Linear Regression prediction Model[9]

3) Find\develop prediction model using KNN(K nearest neighbor) regression model and run the model on same data as to be used on Deep Learning model. KNN model will be used as a benchmark.
4) Use Support vector Machine(RBF and Poly) methods

Evaluation Metrics

Predicted prices will be compared to Actual historical prices.

- Keras model.evaluate method will be used to find the training and test errors.
- MSE (Mean Square Error) will be calculated for predicted and actual values.

Project Design

Project will be implemented using programming language Python and neural network API Keras will be used for deep learning.

Input Data

- Data is downloaded from finance.yahoo.com[8] for selected tickers (S&P500, IBM, MSFT, WMT, PFE, XOM, MYL etc.) for date ranges (1980-12-12 to 2017-07-07)
- Data is read using pandas dataframe.
- Data is indexed using date value.
- Only Adj Close share price is finally used for analysis.
    - Exploration part shows that Open, Close, High, Low and AdjClose values have almost similar patterns,. So only AdjClose is kept out of these 5 features. Daily Return was also calculated based on AdjClose value but the results were almost close to AdjClose values. So to keep the model simple, only AdjClose is finally selected for analysis. As an improvement model, it is suggested to try including Volume for analysis.
- Data is cleaned so that is does not have any null and NA(undefined) values.
- For training and testing purpose, data will be split into training, validation and test datasets.
- SP500 is used as base dataset - model is trained on SP500 as SP500 is based on 500 stocks and it captures patterns related to wide variety of stocks.
- Model is finally tested on SP500 and other stocks such as IBM, MSFT, WMT, PFE, XOM and MYL.

Ref code:

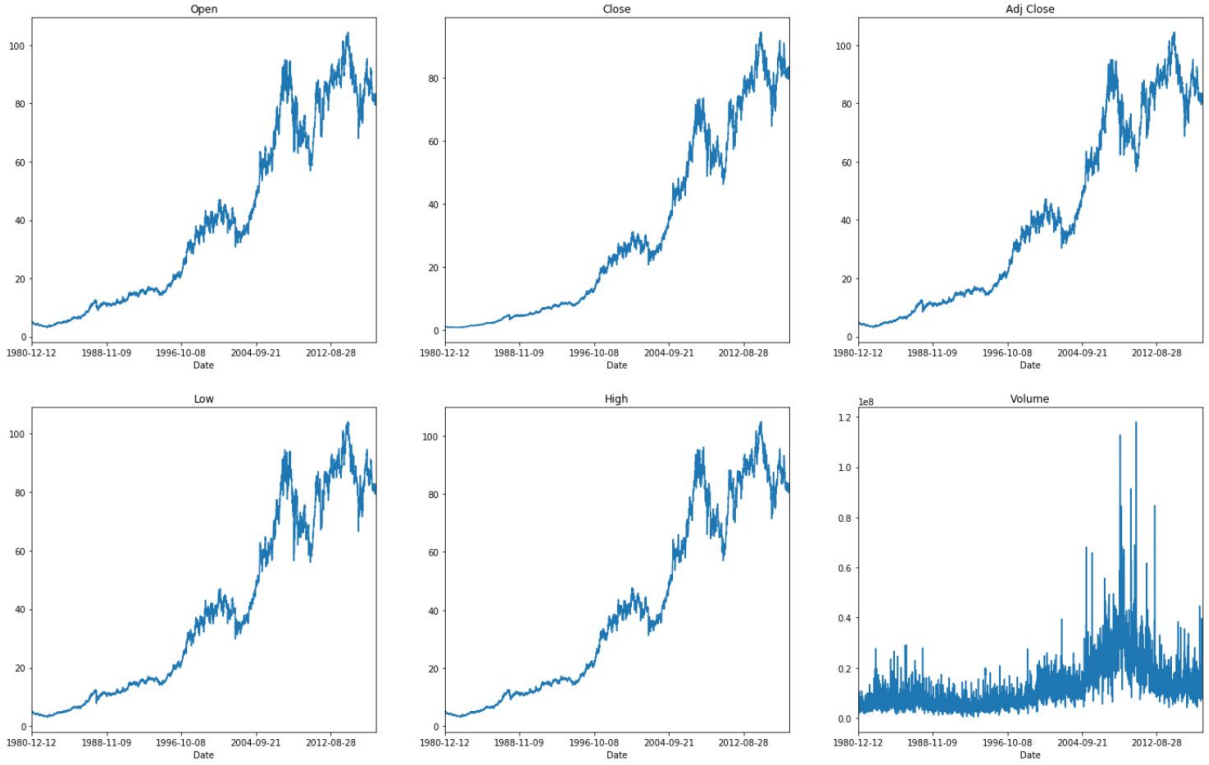
Reading data

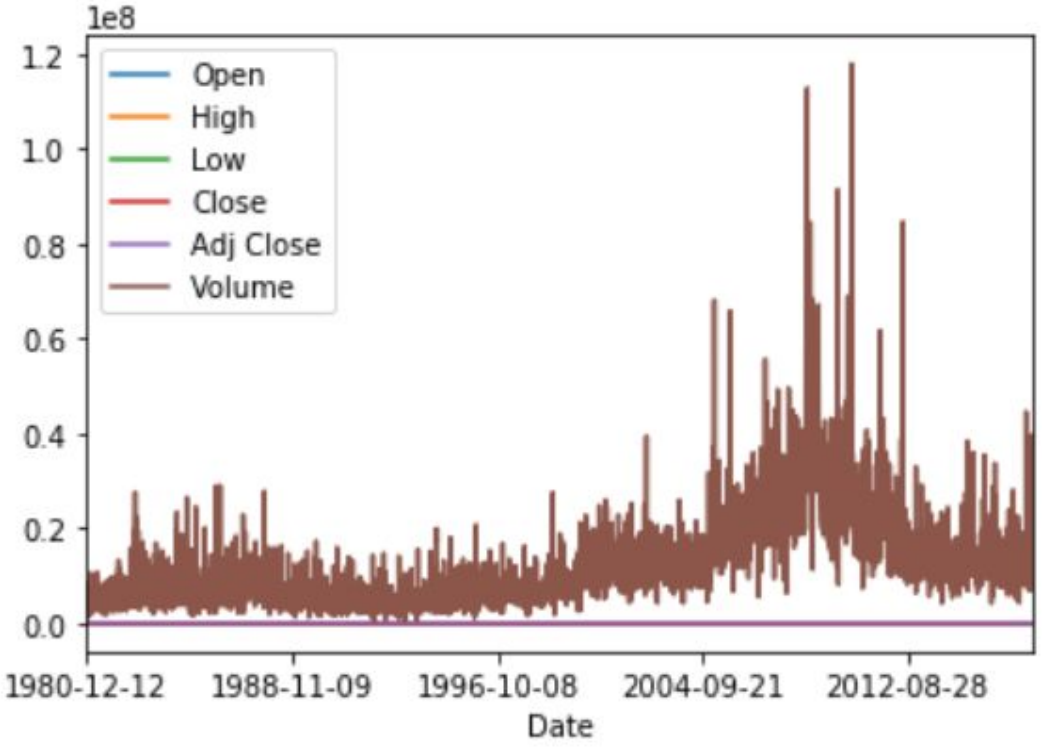
```
import pandas as pd
data_SP500 = pd.read_csv(SP500_file_path,index_col ='Date')
data_SP500.dropna()
```

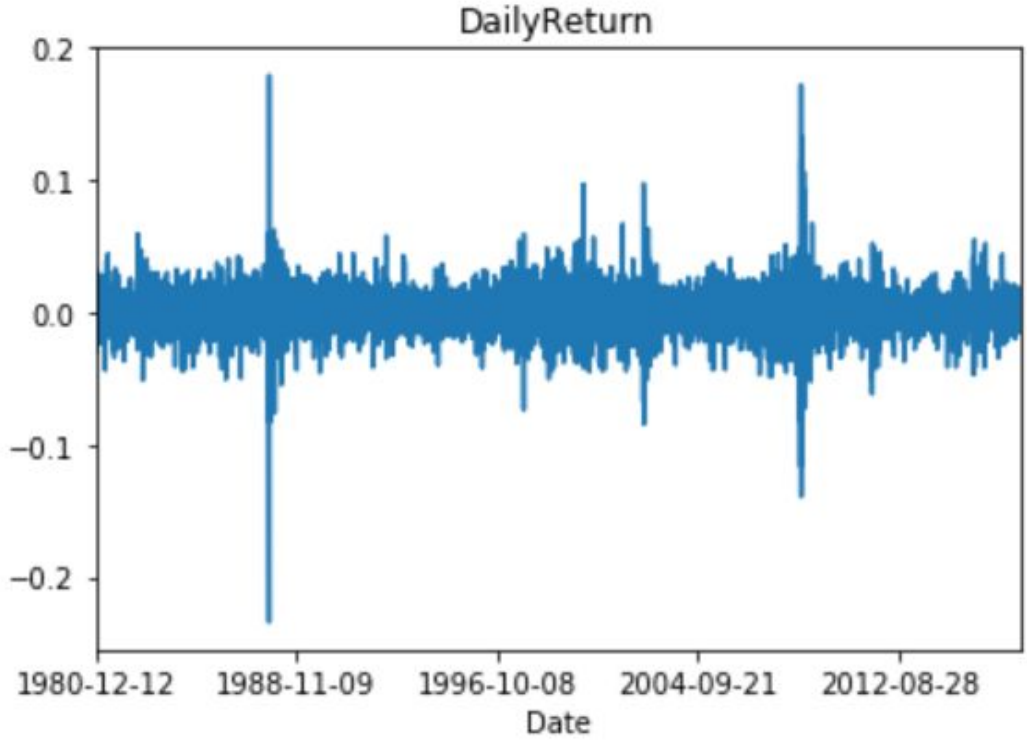
	Open	High	Low	Close	Adj Close	Volume
Date						
1980-12-12	4.882813	4.953125	4.882813	1.116966	4.921875	4892800
1980-12-15	4.921875	4.992188	4.921875	1.122285	4.945313	2446400
1980-12-16	4.945313	5.125000	4.945313	1.157744	5.101563	4040000
1980-12-17	5.148438	5.289063	5.148438	1.196749	5.273438	3816000
1980-12-18	5.273438	5.359375	5.210938	1.182565	5.210938	5790400

	Open	High	Low	Close	Adj Close	Volume
Date						
2017-06-30	81.080002	81.239998	80.699997	80.730003	80.730003	13696800
2017-07-03	80.790001	82.489998	80.720001	82.099998	82.099998	8422300
2017-07-05	81.680000	81.680000	80.519997	80.849998	80.849998	9830800
2017-07-06	80.860001	81.080002	80.089996	80.120003	80.120003	9822400
2017-07-07	80.199997	80.599998	79.809998	80.220001	80.220001	10119700

Data Exploration




1e8
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Open
High
Low
Close
Adj Close
Volume
1980-12-12
1988-11-09
1996-10-08
2004-09-21
2012-08-28
Date


DailyReturn
0.2
0.1
0.0
−0.1
−0.2
1980-12-12
1988-11-09
1996-10-08
2004-09-21
2012-08-28
Date

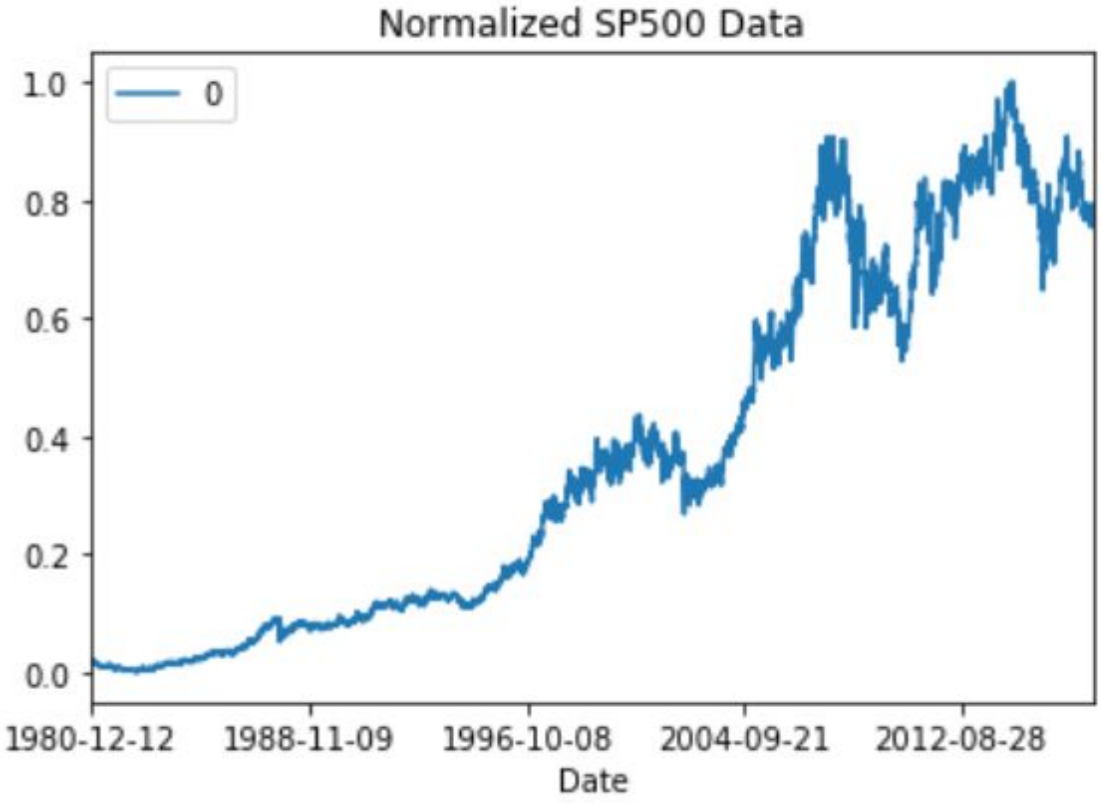
Normalize Data

sklearn minmaxscaler can be used for normalizing data

```
from sklearn.preprocessing import MinMaxScaler
minmaxScaler_SP500 = MinMaxScaler((0,1))

values = data_SP500_AdjClose.values
index = data_SP500_AdjClose.index

values = minmaxScaler_SP500.fit_transform(values)
normalized_data_SP500_AdjClose = pd.DataFrame(values, index = index)
```

Scaling is done between 0 and 1 as stock prices do not turn negative.

Different Scalers are used for each Stock symbol so that these scalers can then be used to recover original values.



Prepare Supervised Data

Data is arranged in the supervised fashion to show that any day AdjClose value of the stock is dependent upon the previous days values.

Different Window sizes of 5, 10, 15, 50, 75 and 100 are tried and it is found out that window size = 5 gives best results.

```
def window_transform_series(series, window_size):
    # containers for input/output pairs
    X = []
    y = []

    numWindows = len(series) - window_size

    for i in range(numWindows):
```

```
        lst = series[i:i + window_size]
        out = series[i + window_size]
        X.append(lst)
        y.append(out)


    # reshape each
    X = np.asarray(X)
    X.shape = (np.shape(X)[0:2])
    y = np.asarray(y)
    y.shape = (len(y),1)

    return X,y

window_size = 5
dataset = normalized_data_SP500_AdjClose.values
X,y = window_transform_series(series=dataset,window_size = window_size)
```

Train\Validation\Test split
Data is split into Train, Validation and test datasets

```
train_test_split = int(np.ceil(8*len(y)/float(10)))

X_test = X[train_test_split:,:]

y_test = y[train_test_split:]


train_valid_split = train_test_split - (len(y) - train_test_split)

X_train = X[:train_valid_split,:]

y_train = y[:train_valid_split]

X_valid = X[train_valid_split:train_test_split,:]

y_valid = y[train_valid_split:train_test_split]
```

Data Modelling

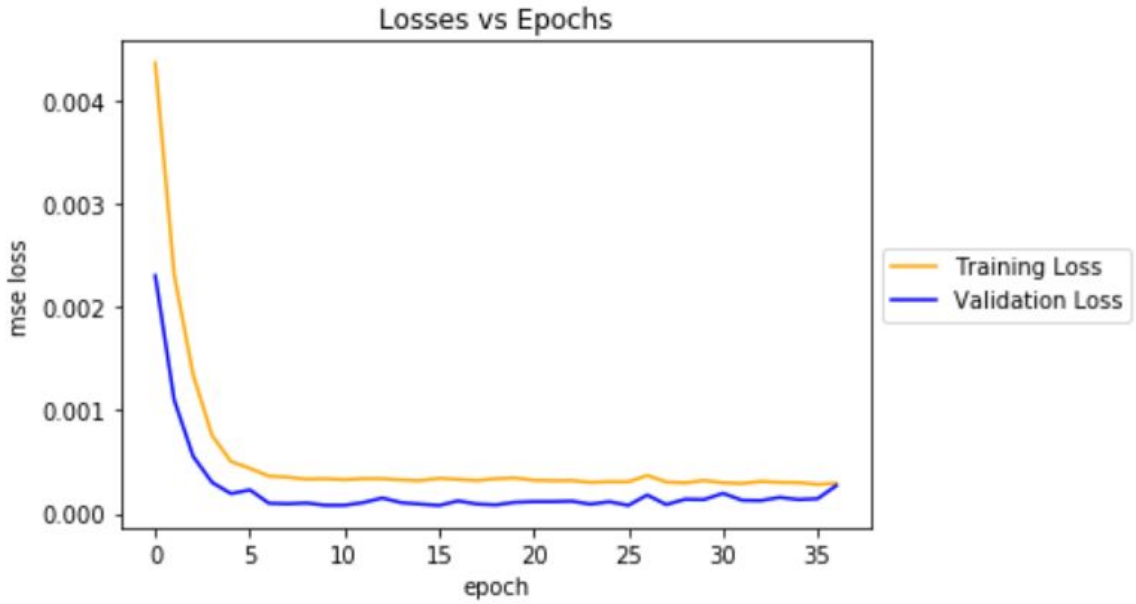
- Keras API will be used for model training and prediction testing.
- Keras sequential model[7] will used for training.
- LSTM layer will be added to this sequential model.
- Depending upon model performance, model LSTM layers will be added.
- Dropout layer will be added(if required) to handle data overfitting. Overfitting happens when training error keeps on decreasing but testing error starts rising after reaching bottom.
- Multiple Dropout layer will be added after each LSTM layer. (if required)

- Final linear layer with single output will be added at the end of model.
- Different Model Optimizers[7] (provided by Keras API) will be used to optimize the model.
- Model will be compiled with appropriate parameters values.
- Model will fit the training data – different parameter values for the fit function[7] will be provided to better fit the model.
- A high number(300) is assigned to epochs but best model is saved based on validation_loss and model fit is Early stopped if there is no improvement for 20 epochs.

Ref code :

```
model = Sequential()

model.add(LSTM(input_dim=1,output_dim=50, return_sequences=True))

model.add(Dropout(0.2))

model.add(LSTM(128,return_sequences=False))

model.add(Dropout(0.2))

model.add(Dense(output_dim=1))

model.add(Activation('linear'))

optimizer = keras.optimizers.Adagrad(lr=0.005, epsilon=1e-08, decay=0.0)

model.compile(loss='mse', optimizer=optimizer)

nb_epoch = 300

batch_size = 256

model.fit(X_train, y_train, batch_size=batch_size, epochs=nb_epoch, verbose=1, validation_data=(X_valid, y_valid), callbacks=callbacks_list, shuffle=False)

model.load_weights("Predict_Share_Prices_model_weights_best.hdf5")

model.compile(loss='mean_squared_error', optimizer=optimizer)
```

Losses during model training



Prediction

- After model fitting, prediction will be made for training and testing data to calculate training and testing errors. Keras Predict[7] and evaluate functions will be used to predict and evaluate[7].
- Actual values of testing dataset will be used to check the accuracy of Predicted values.

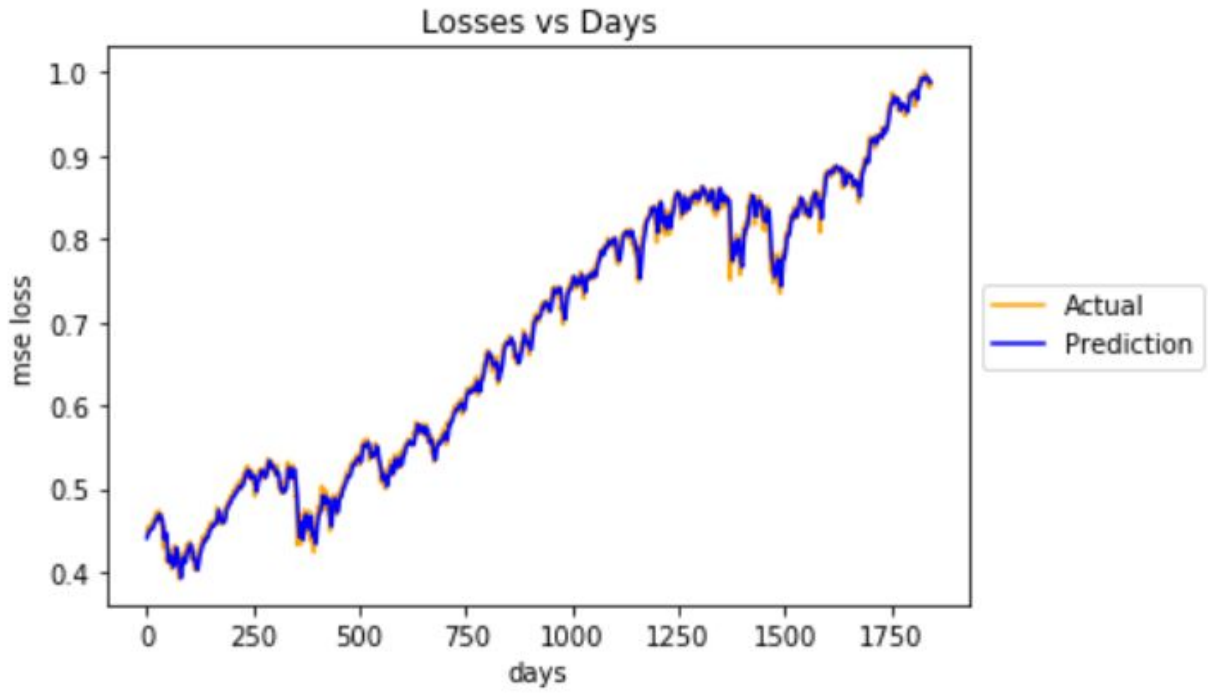
Ref code:

```
train_predict = model.predict(X_train)

test_predict = model.predict(X_test)

training_error = model.evaluate(X_train, y_train, verbose=0)

testing_error = model.evaluate(X_test, y_test, verbose=0)
```

Test, Validation and Prediction Errors - SP500

training error = 5.1344345161e-05
validation error = 7.87252354123e-05
testing error = 0.000103909208899

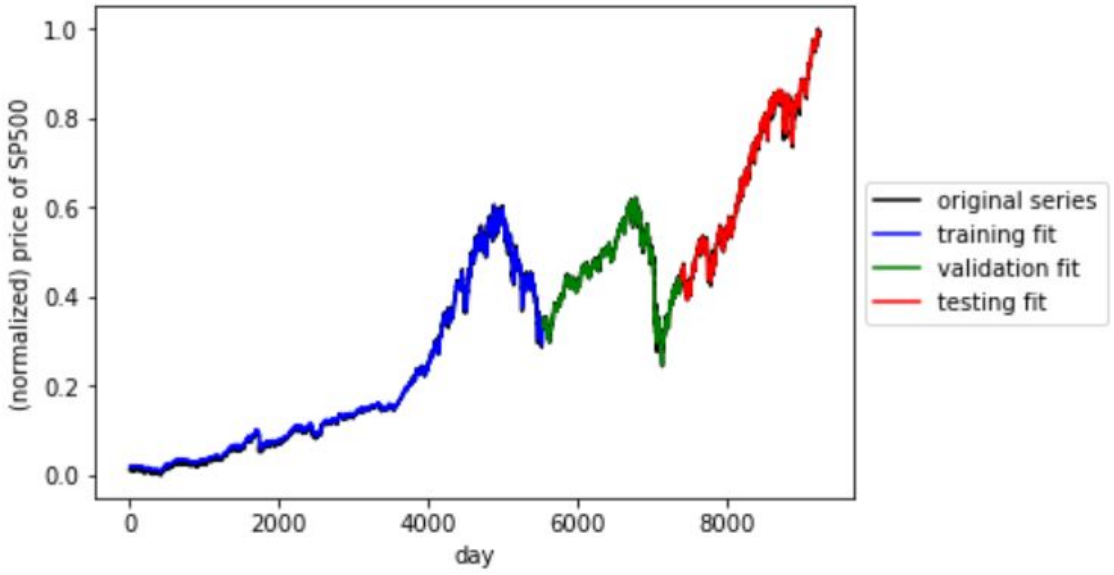
Prediction on SP_500 Test data



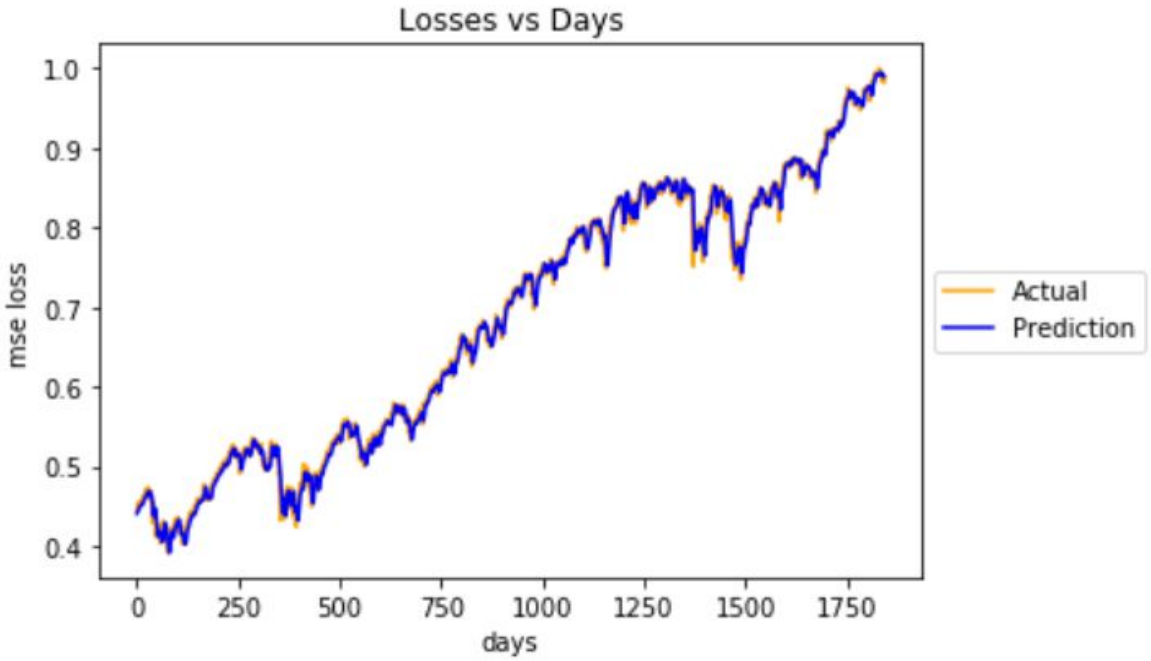
Visualization

Actual training and testing data will be plotted alongside the predicted data using matplotlib library functions. Visualization should be able to show the closeness of predicted values to actual values.

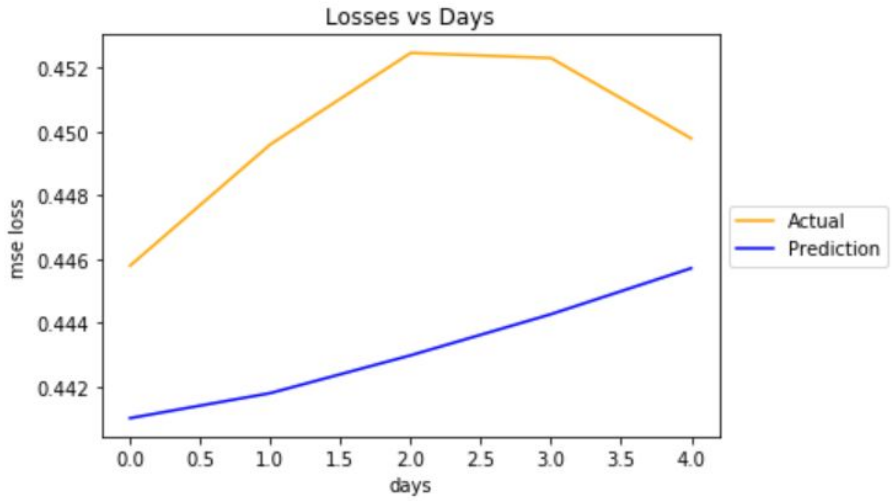
Fig below shows data for stock ticker-SP500. Blue part shows the training data. Green part shows the Validation data. Red part shows the testing data. Black data shows the actual data. Closeness of Blue\Red points to the black point lines will show the degree of fit.



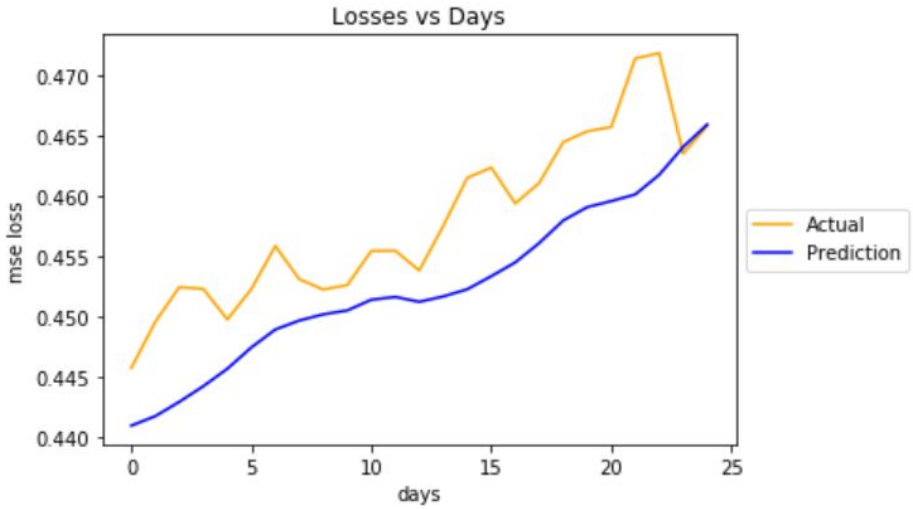
Predicting 1 value at a time and using this predicted value to predict future data



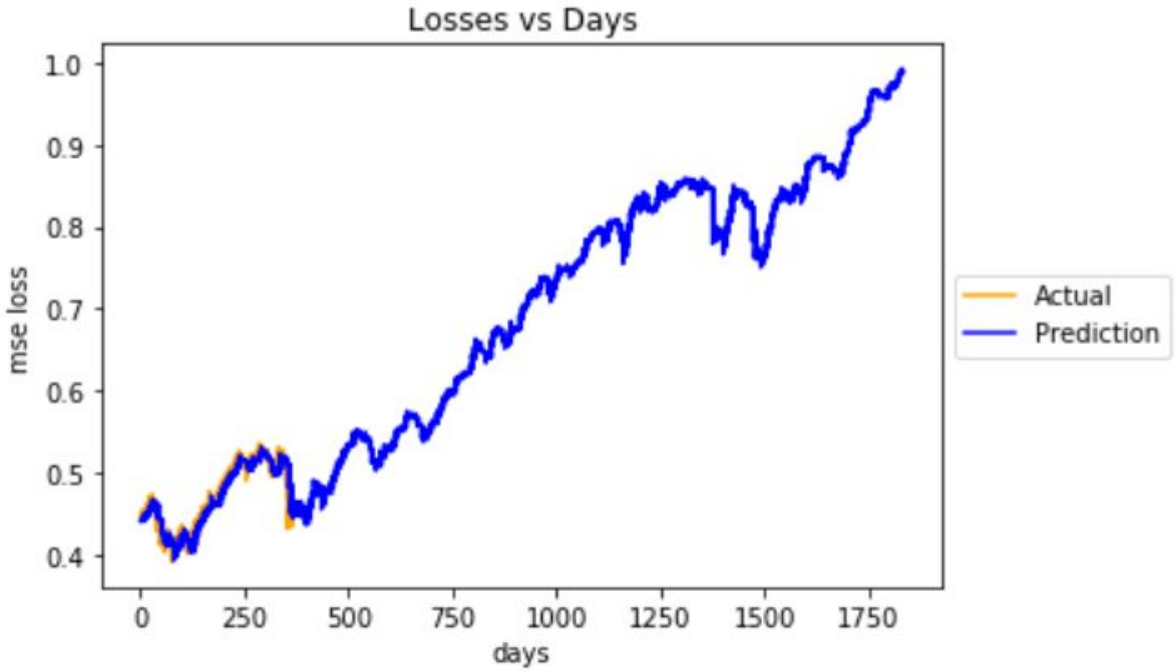
First five values



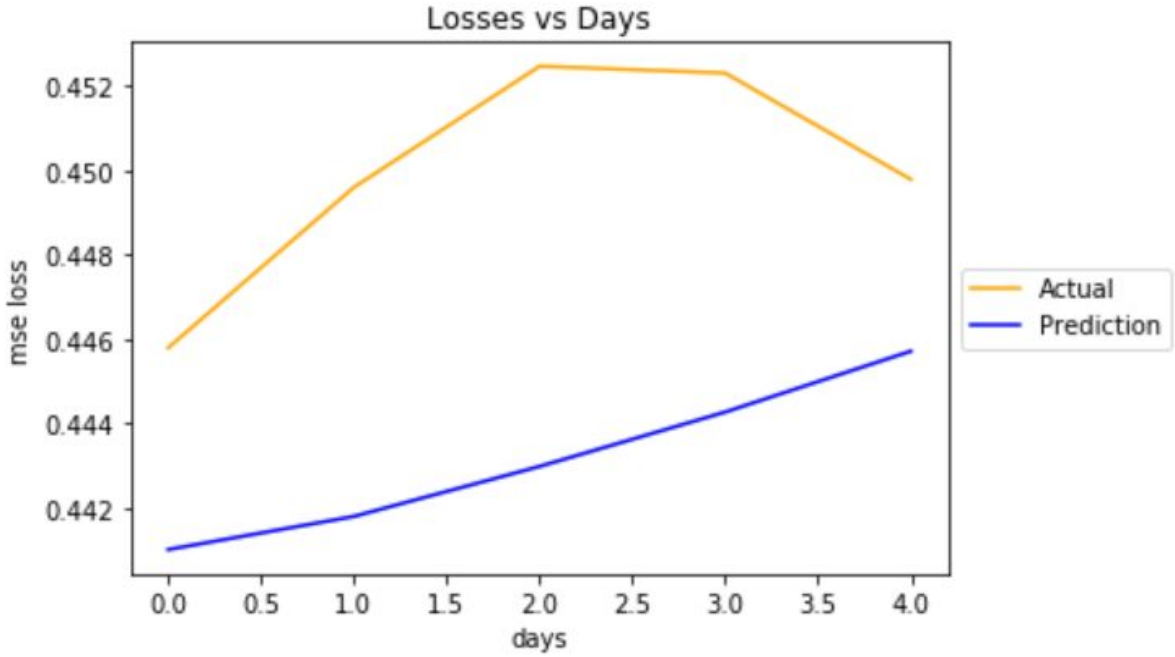
First twenty five values



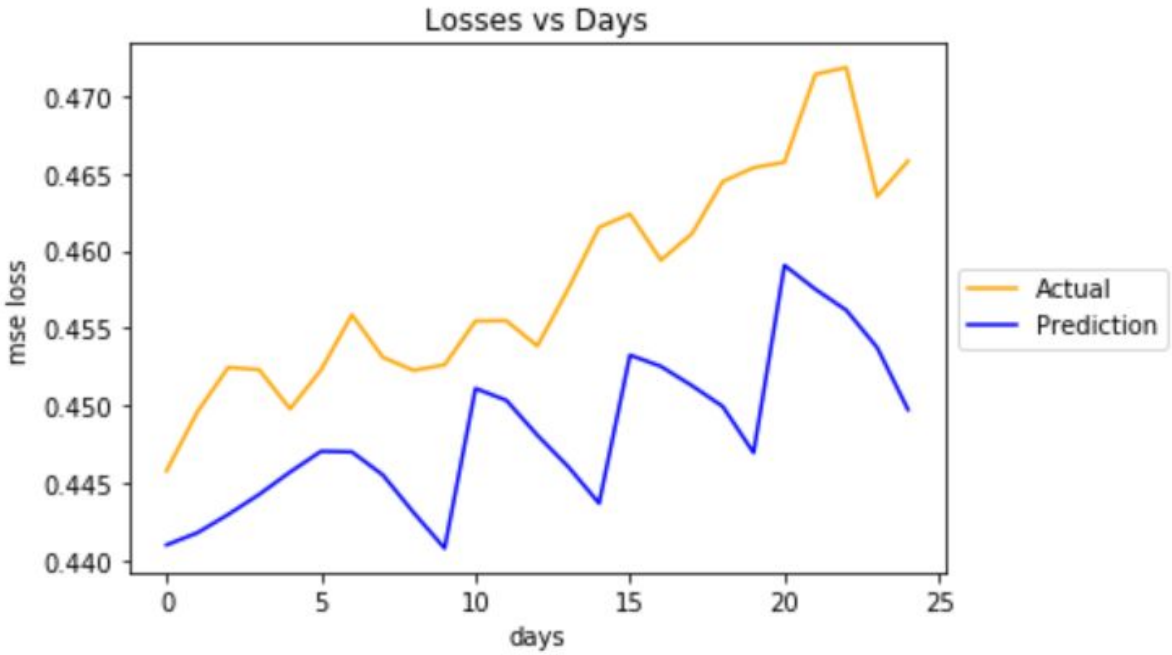
Predicting 5 values at a time and using these 5 predicted values to predict future data



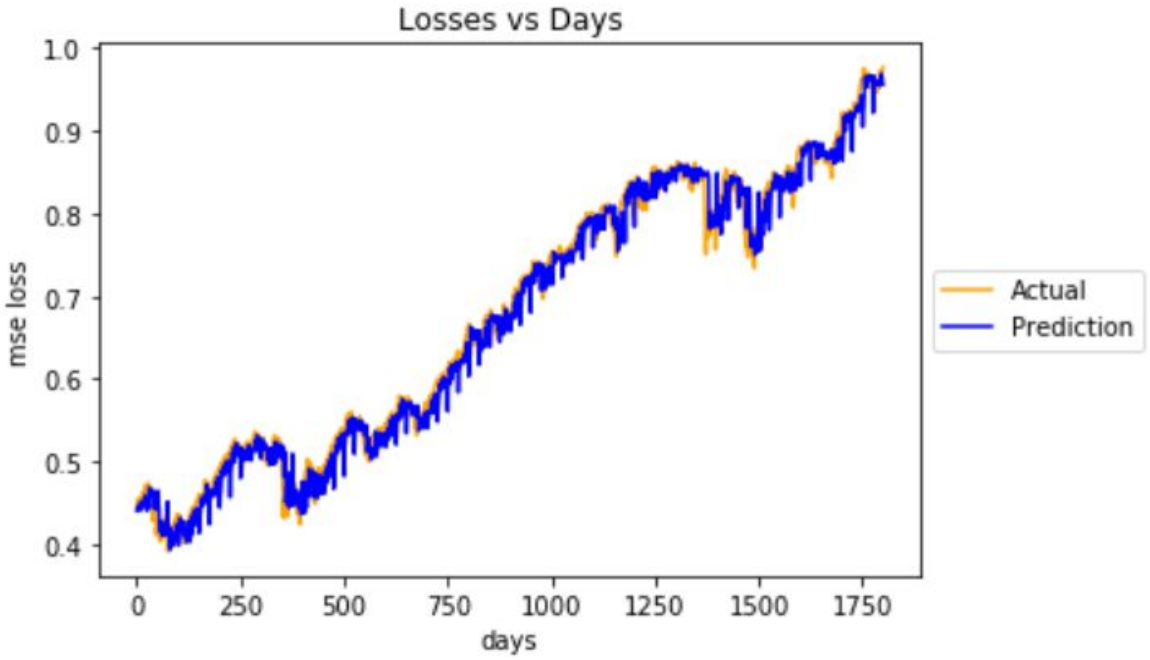
First five predicted values



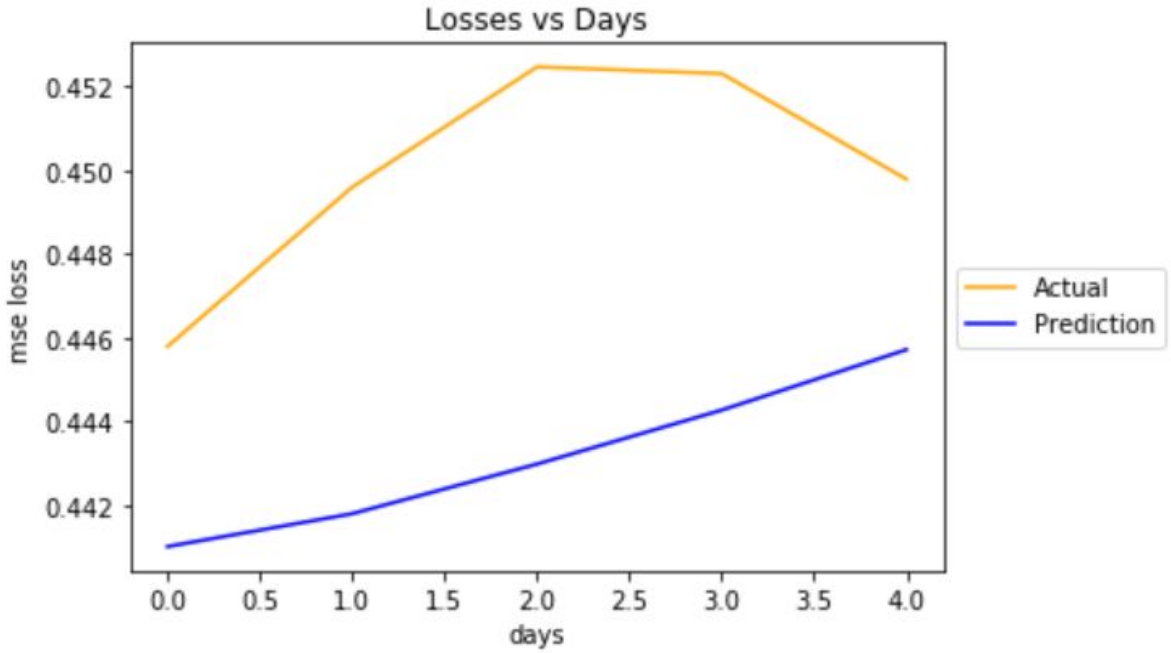
First twenty five predicted values



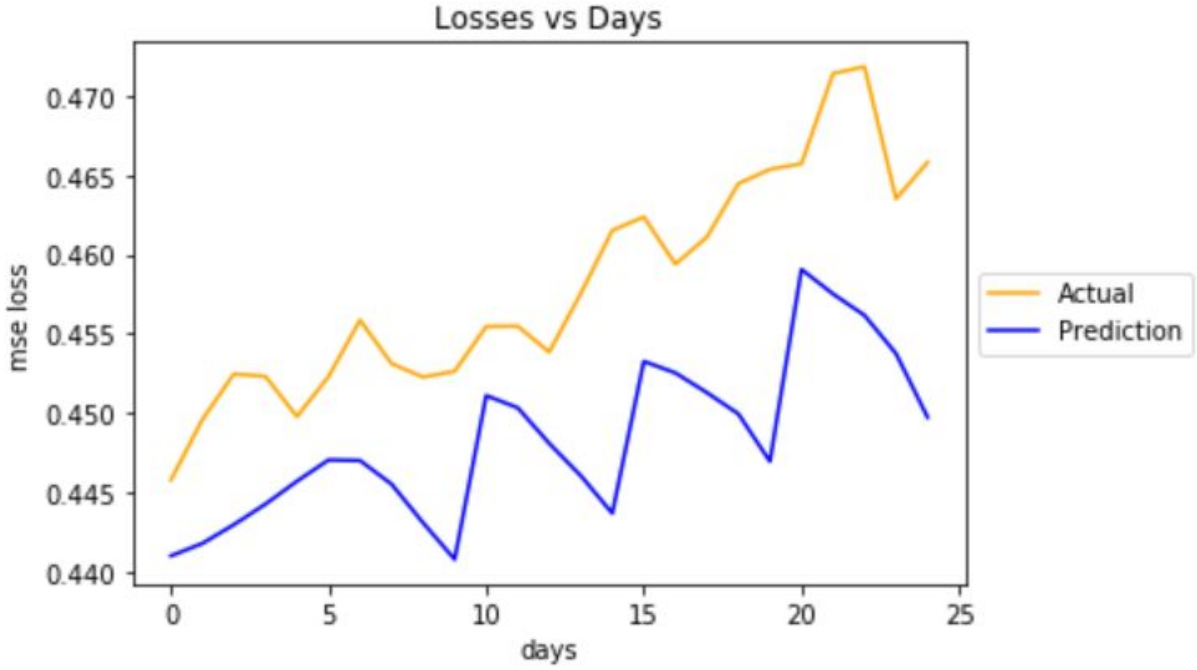
Predicting 5 values at a time and using these 5 predicted values to predict future data



First Five Predicted Values



First 25 Predicted values



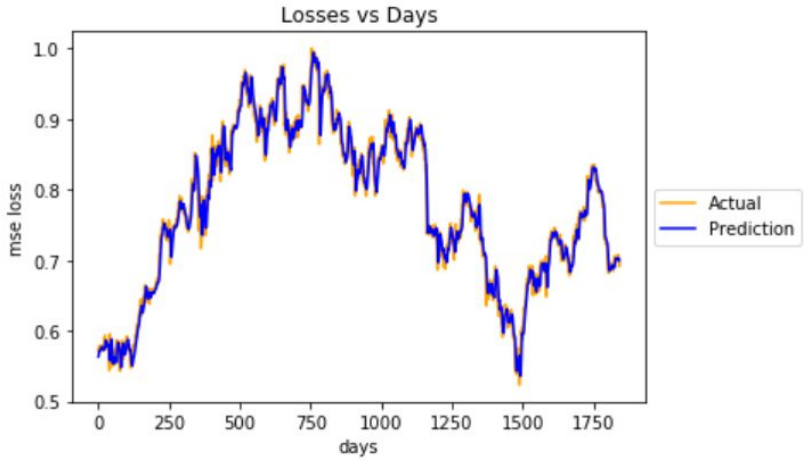
Above graphs show that if we use predicted values in future predictions, the predictions fall apart and correct results are only upto a few days. This happens because errors accumulate and trickles down to future values.

Model Robustness

Model testing on other Stocks' Test Data

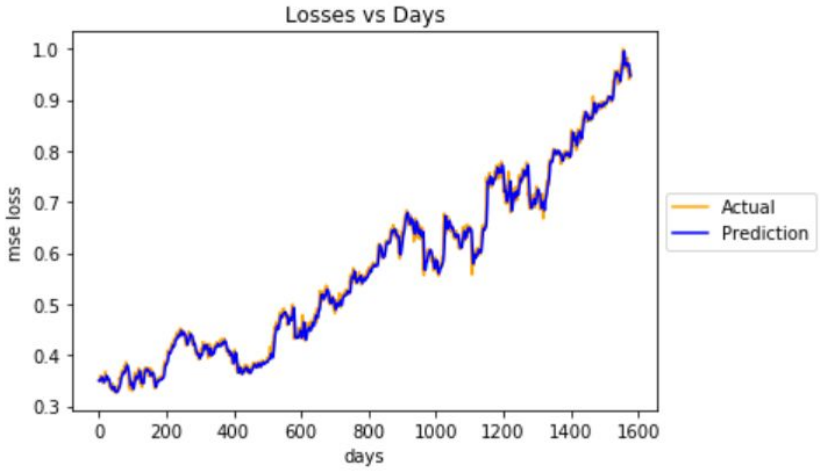
IBM

Testing error = 0.000260660544341



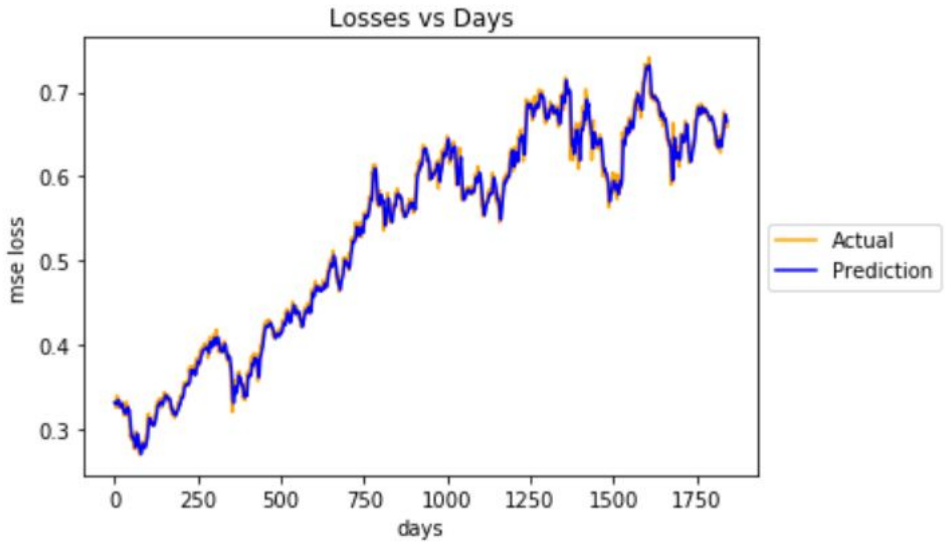
MSFT

Testing error = 0.000184823278927



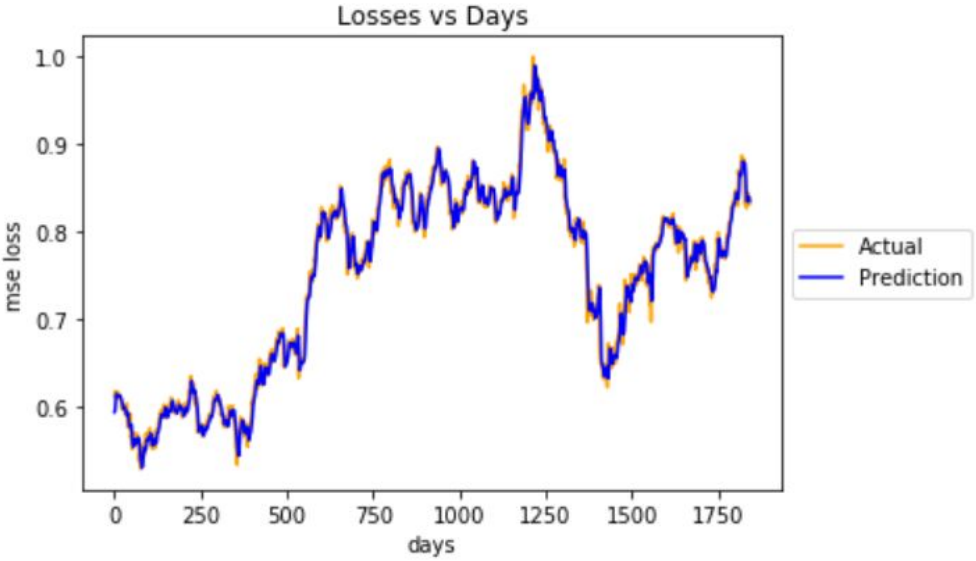
PFE

Testing error = 0.000108408666386



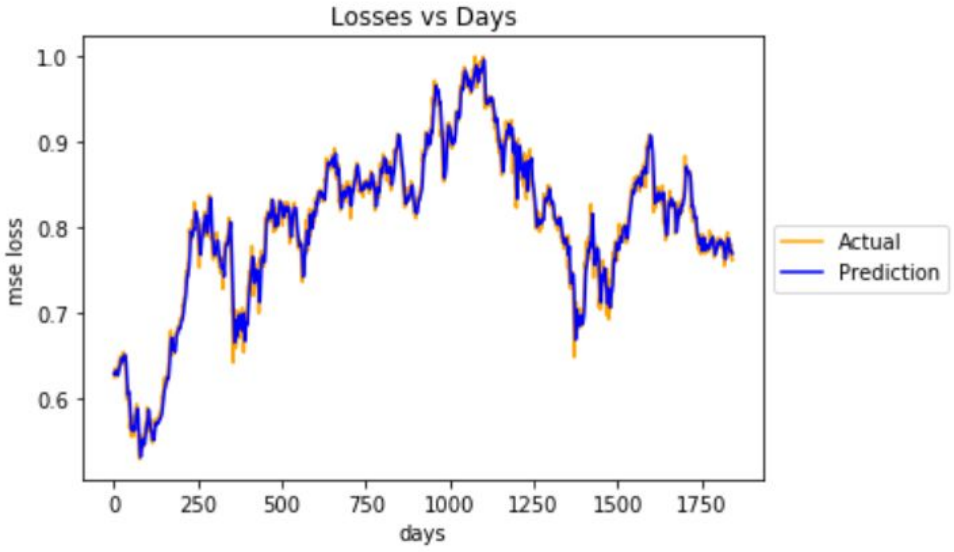
WMT

Testing error = 0.000151255406694



XOM

Testing error = 0.00022837143543



MYL

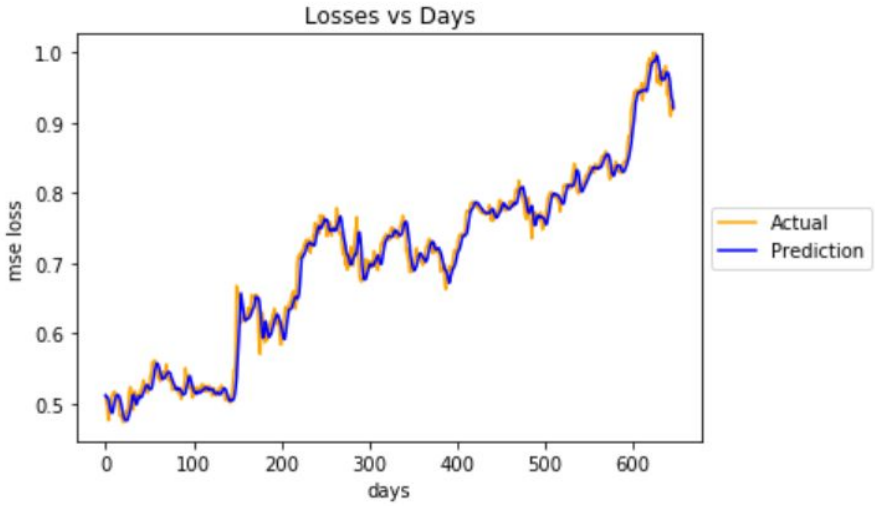
Testing error = 0.000343096786705

AAPL

Testing error = 0.000212655573821

GOOG

Testing error = 0.000333827781351



Benchmark

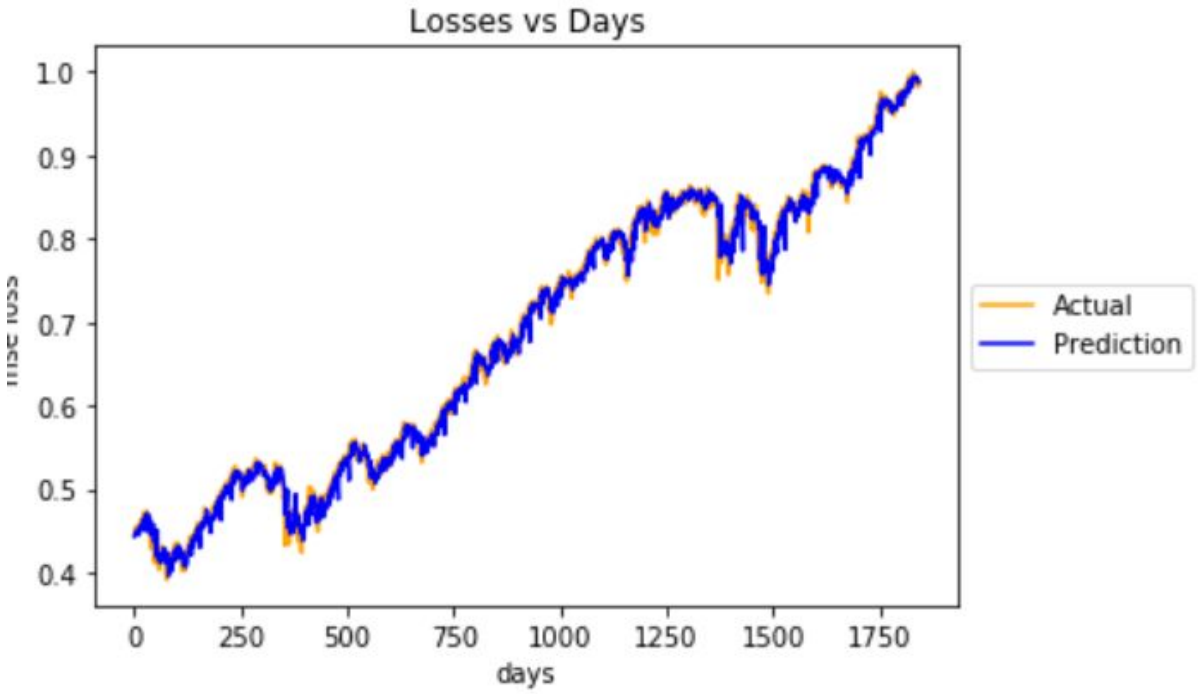
Following benchmark models are considered¶

- **Moving Averages**
- **Linear Regression**
- **KNN**
- **Support vector machines (RBF)**
- **Support vector machines (Poly)**

Testing for these benchmarks show that only Linear Regression and Moving Averages are close to the performance obtained with Deep Learning LSTM model. Infact Linear Regression seems to give better results than Deep Learning model

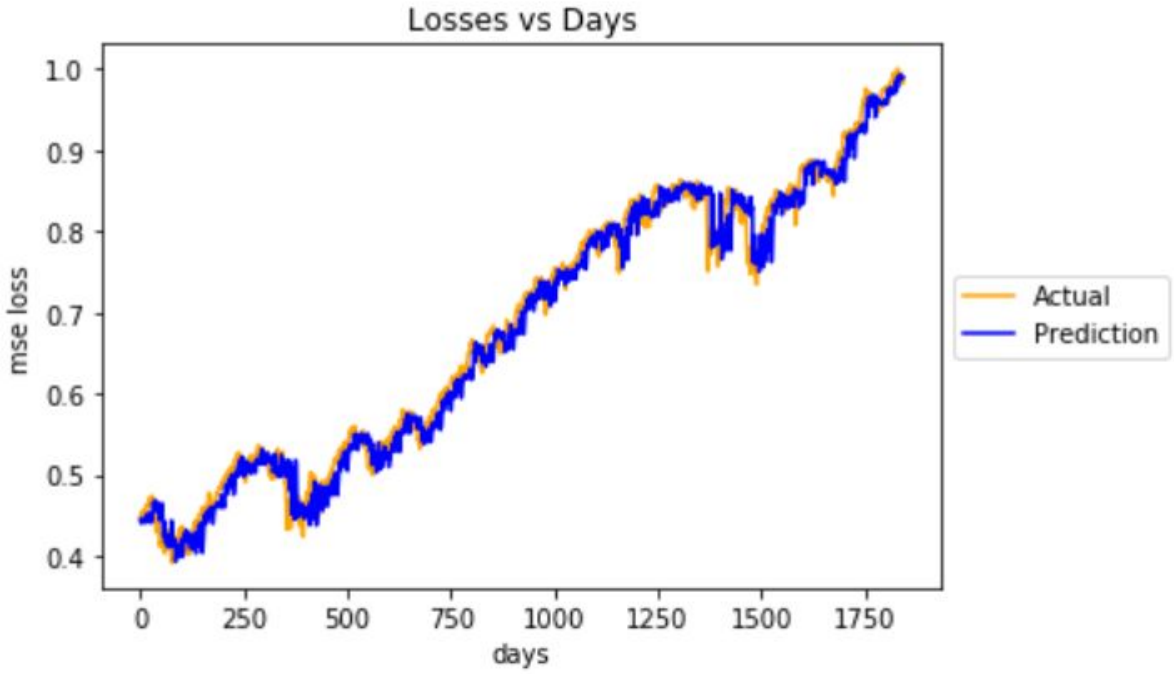
Moving averages

Test Error : 0.000328648905233



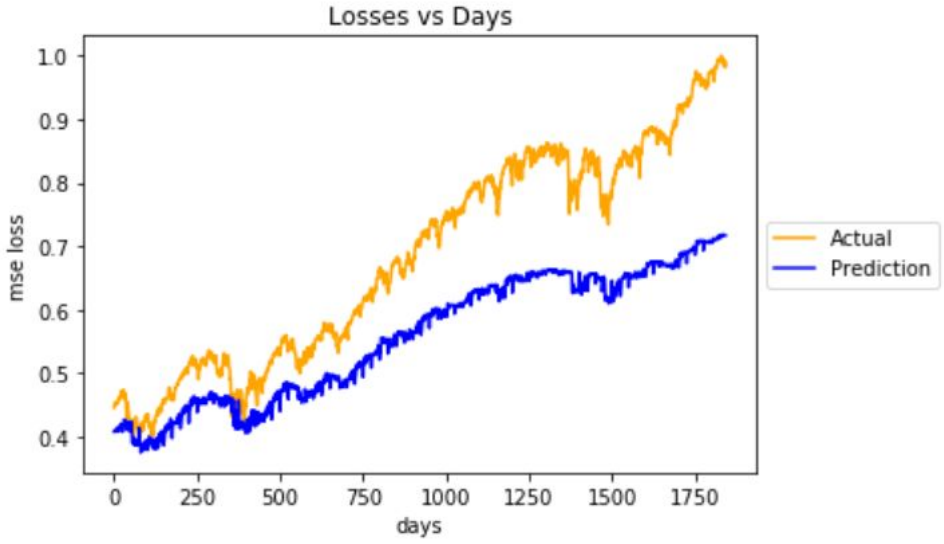
Linear Regression

Test Error : 0.00043885



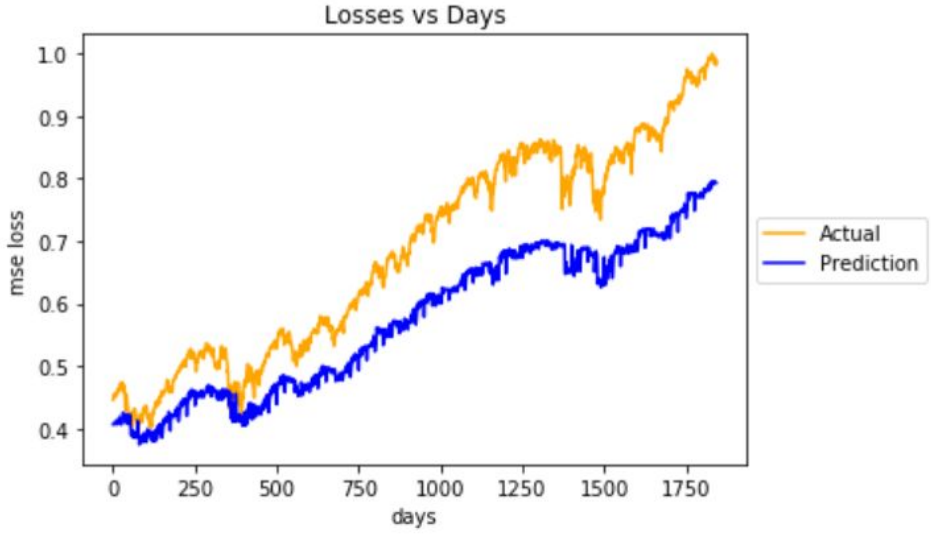
Support Vector Machine(Kernel=rbf)

Test Error : 0.02073039



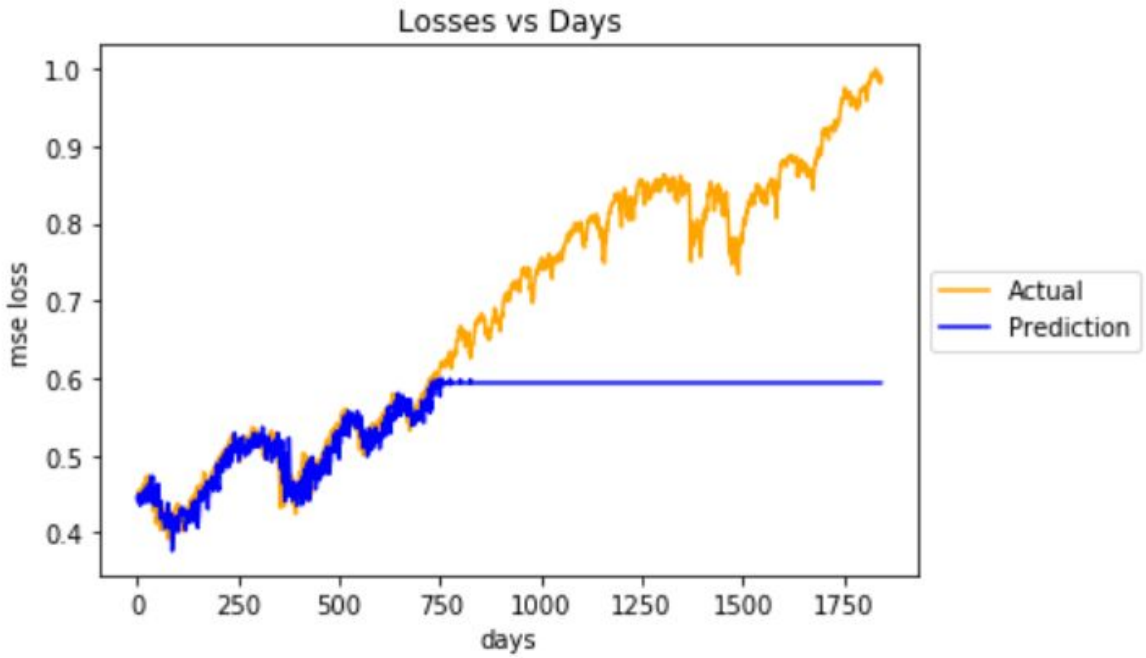
Support Vector Machine(Kernel=poly)

Test Error : 0.01477115



KNN

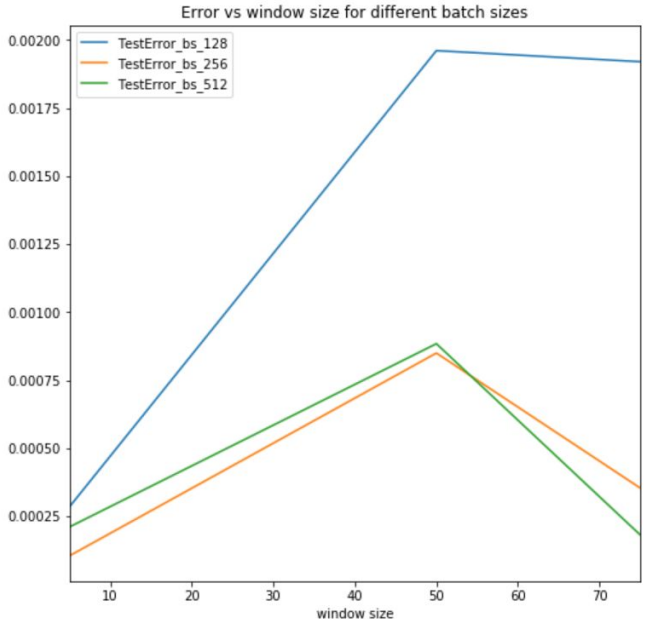
Test Error : 0.03221490



Testing Errors for Different Window_Sizes

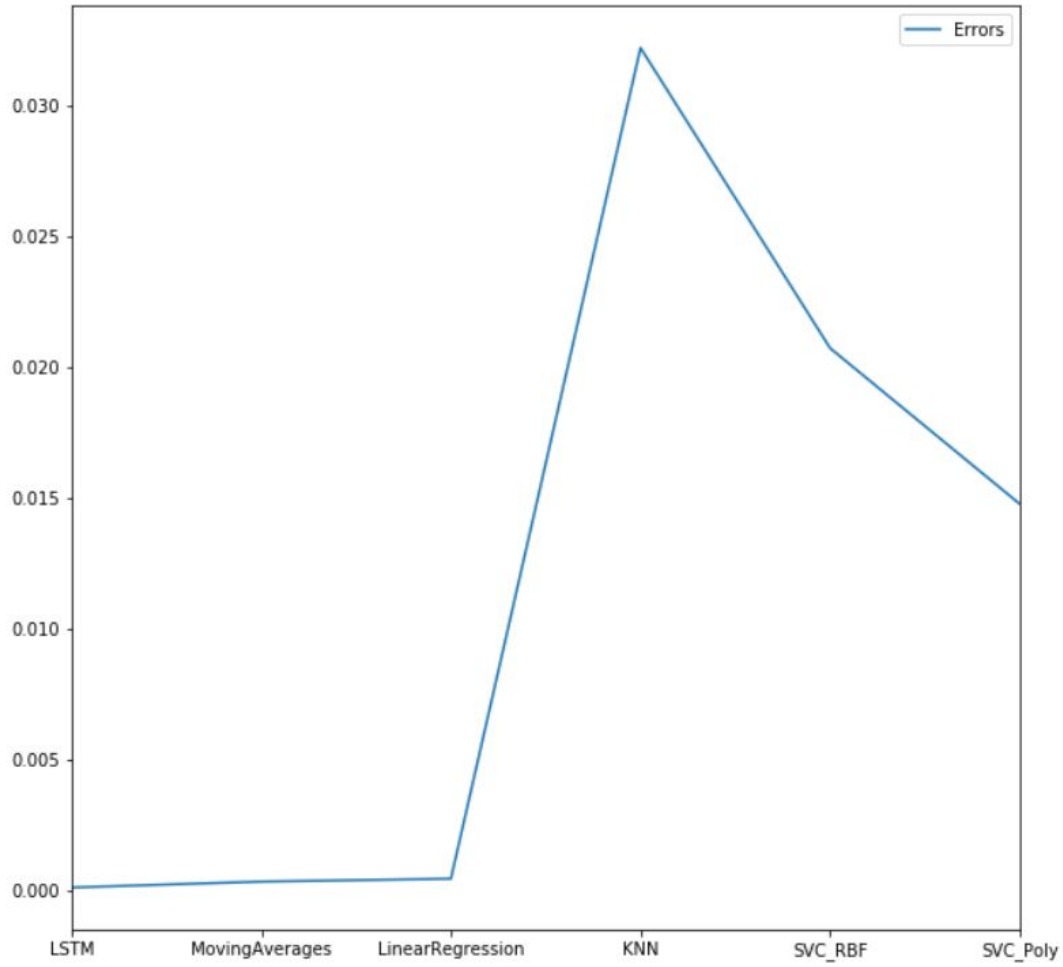
As per above data, window_size = 5 gives best performance for all 3 types of errors

Testing Errors for Different Batch_Sizes

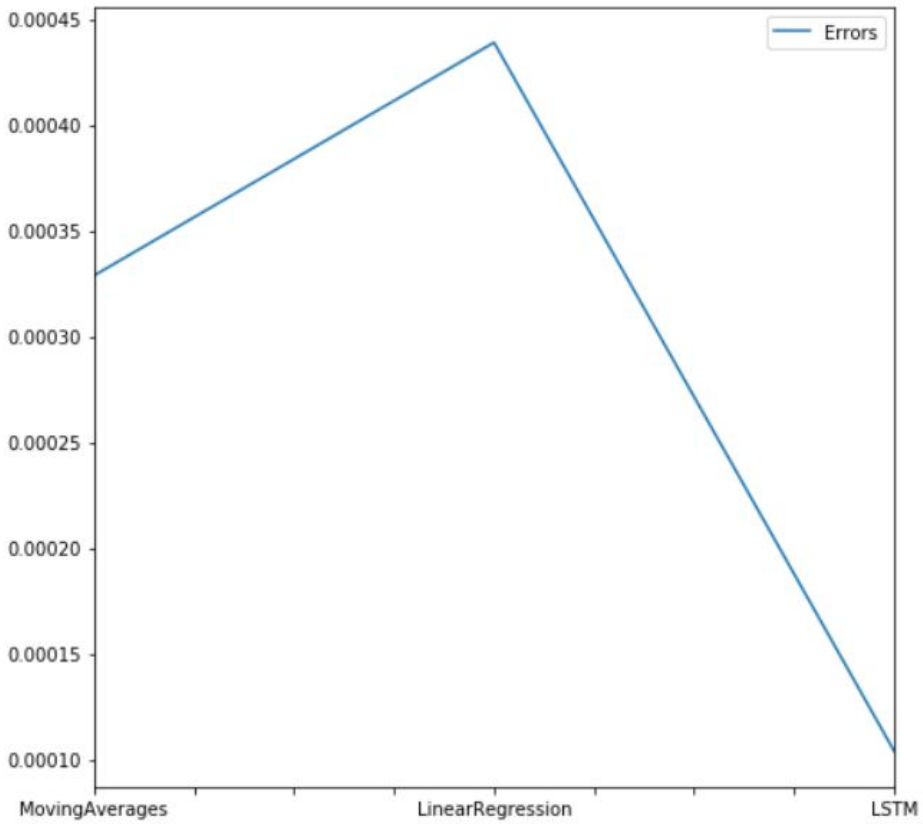


As per above graph, Test Error is lowest for batch size=256 for window sizes below around 55 and Test Error is lowest for batch size = 512 and window sizes above 55.

Testing Error for Different Models



Lowest Testing Errors for models



Conclusion¶

As seen in above plots, Deep Learning LSTM Model with window_size = 5 and Batch_size = 256 gives best results

Possible Improvements

1) Try other features such as Volume of trading for data dependency
2) Although I tried various models with different number of layers, different number of LSTM units per layer, tried Bidirectional LSTMs (to check current price dependency not only on past values but future values also), tried combination of Convolution layers and Bidirectional and simple LSTM units but these did not give better results. GAN(Generative Adversarial Networks) can be tried for predicting stock prices.
3) Removing stationarity - on a long term basis , there is a trend of increasing values of stock prices. It can be tried to remove this trend and then predict the prices.

Appendix

Artificial Neural Networks[3]

Artificial neural networks are computing systems inspired by the biological neural networks constituting human brains. An ANN is a collection of connected units (logical units) called artificial neurons. In a neural network, thousands or millions of neurons, are organized in layers. Each layer may perform different kinds of transformations on their inputs. ANN learn by forward propagation (matching inputs to outputs) and backpropagation (matching outputs to inputs).

RNN (Recurrent Neural Network)[4]

A **recurrent neural network** (**RNN**) is a class of artificial neural network in which neurons have directed cycle connections. Unlike feedforward neural networks(where neurons are connected in only one direction from input to output), RNNs have backward connections also. RNNs can use their internal memory to process arbitrary sequences of inputs.

LSTM (Long Short Term Memory) Network[4]

Long short-term memory (LSTM) is specific type of RNN that attempts to remember and forget previously identified patterns. LSTMs have "forget" gates to discard unwanted patterns. LSTMs try to remove the vanishing or exploding gradient problems and this helps LSTM learn tasks that require memories of events that happened thousands or even millions of discrete time steps earlier.

References

1) https://en.wikipedia.org/wiki/Stock_market_prediction
2) https://en.wikipedia.org/wiki/Random_walk_hypothesis
3) https://en.wikipedia.org/wiki/Artificial_neural_network
4) https://en.wikipedia.org/wiki/Recurrent_neural_network
5) http://www.investopedia.com/terms/a/adjusted_closing_price.asp
6) https://keras.io/
7) https://keras.io/models/model/
8) https://finance.yahoo.com/
9) https://www.quantstart.com/articles/Forecasting-Financial-Time-Series-Part-1