I. Definition

Project Overview

Modelling of stock markets have made major advancements since the inception of machine learning. Machine learning models have capabilities to consider large datasets, and analyse and record relationships between current and future prices. Through the use of current technical and charting methods used, we can predict prices with moderate success but machine learning models are preferred as they perform better and can base their prediction methodologies over training data acquired over a large time period. Hence, we may be even be able to spot infrequent trends which have occurred in the past.

Stock market is a well documented sector, and there are many API's and finance websites which provide daily price data. Yahoo finance, Google finance and Quandl, are amongst the leading websites which provide these services. S&P500 data from Yahoo finance. The data is available at <https://in.finance.yahoo.com/quote/^GSPC?p=^GSPC>.

Problem Statement

The objective is to predict future price action for the S&P500 counter using a regression. The tasks involved to achieve our objective are:

1) Import the downloaded data from a '.csv' extension file, index this data by date and produce a raw input data frame.
2) Add domain knowledge, and augment this data frame by adding multiple technical indicators. These indicators will reduce the noise in the data, by providing a distinctive value for an underlying occurring trend, and give a constant ranged value otherwise.
3) Standardise and normalise the data.
4) Apply feature extraction, to get the most relevant features out of our augmented data
5) Train our LSTM based model over this data. The LSTM based model would be suitable for this application, as it would back propagate through time, and learn specific prices changes which have occurred in the previous trading sessions, which lead a rise or fall in price of the stock after $t+5^{th}$ day.

6) Produce price predictions for t+5th day using this model

Metrics

Evaluation of our regression model, will be based on 5 major metrics. These metrics would evaluate how closely we predict future prices and optimistic our model is, and the possible gain in return compared to a naive strategy.

1) R^2 Score : This is a very commonly used metric for regression model. This models expresses the performance of our model as contrasted with a model which always outputs the mean value of the training target data, for all test instances.

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2,$$

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

2) Mean-squared Error : This metric is a summation of squared error between the target data and the predictions made by our data.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2$$

3) Return Ratio: The ratio between the dollar value return achieved by taking long or short positions on a stock based on the model predictions over a time period, and the return achieved if we purchased the stock at the start and sold it at the end of the time period.

4) Optimism Ratio : This is a self designed metric, which measures the number of times the model has produced a price prediction which is 1.5% greater than the actual price.

$$OR = \frac{OPT}{Length}$$

$Where,$

$OPT = Count\ of\ records\ where\ [\ predicted\_price > 1.015*actual\_price\ ]$

$Length = Total\ Number\ of\ Records$

5) Pessimism Ratio : This is a self designed metric, which measures the number of times the model has produced a price prediction which is 1.5% lower than the actual price.

$$OR = \frac{PES}{Length}$$

$Where,$

$OPT = Count\ of\ records\ where\ [\ predicted\_price < 0.985*actual\_price\ ]$

$Length = Total\ Number\ of\ Records$

---

1. *Data Source <https://in.finance.yahoo.com/quote/^GSPC?p=^GSPC>*
2. *LSTM <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>*
3. *LSTM <https://www.youtube.com/watch?v=6niqTuYFZLQ&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv&index=10>*
4. *Technical Indicators <https://www.tradingtechnologies.com/help/x-study/technical-indicator-definitions/list-of-technical-indicators/>*
5. *Technical Indicators <https://www.investopedia.com/articles/technical/04/041404.asp>*
6. *Singh, Ritika and Shashi Srivastava. "Stock prediction using deep learning." Multimedia Tools and Applications 76 (2016): 18569-18584.*
7. *Erkam Guresen, Gulgun Kayakutlu, and Tugrul U. Daim. 2011. Using artificial neural network models in stock market index prediction. Expert Syst. Appl. 38, 8 (August 2011), 10389-10397.*
8. *Wei Shen, Xiaopen Guo, Chao Wu, Desheng Wu, Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm, In Knowledge-Based Systems, Volume 24, Issue 3, 2011, Pages 378-385, ISSN 0950-7051*
9. *George S. Atsalakis, Kimon P. Valavanis, Surveying stock market forecasting techniques – Part II: Soft computing methods, In Expert Systems with Applications, Volume 36, Issue 3, Part 2, 2009, Pages 5932-5941, ISSN 0957-4174*
10. *Jigar Patel, Sahil Shah, Priyank Thakkar, K Kotecha, Predicting stock market index using fusion of machine learning techniques, In Expert Systems with Applications, Volume 42, Issue 4, 2015, Pages 2162-2172, ISSN 0957-4174,*

II. Analysis

Data Exploration

This project utilises the S&P500 tick data for prediction. The data is stored in form of a 2-dimensional data frame, where each column would represent the attributes. The raw dataset has been imported from 'Yahoo Finance'. The data contains records dated from '01-03-2000' to '10-30-2017'. This data is further divided into 3 individual segments. The training data has records, dating from '2000-03-07' to '2012-05-01'. Similarly, data for hold-out validation and testing date from '2012-05-02' to '2014-06-11' and '2014-06-12' to '2017-10-23'. The regression model will be built to predict the future 'Adj Close' parameter. This parameter is different from the 'Close' parameter of the dataset, as the 'Close' parameter does not compensate for events like stock split and dividends, and so does not have a smooth trend over time. Hence, we use 'Adj Close' parameter which is similar to the 'Close', but is adjusted for all these events.

The raw dataset consists of 5 columns. Each row in the dataset refers to a corresponding trading session.

1) 'Open ' : This attribute reports the opening price of the stock for that trading session.
2) 'Close' : This attribute refers to the closing price, that is the price at which the stock concluded is trading for the session.
3) 'Volume' : The number of trades that have been made on a particular stock.
4) 'High' : The maximum price the stock had spiked to, through the course of the trading session
5) 'Low' : The minimum price the stock had dipped to, through the course of the trading session
6) 'Adj Close' : The adjusted close parameters is similar to the close price, it is adjusted for share splits and dividend distributions. This parameter does not have N.A values, and is more consistent historically.

The last 5 records of the dataset are as follows:

	Open	High	Low	Close	Adj Close	Volume
2017-10-24	2568.659912	2572.179932	2565.580078	2569.129883	3.427330e+09
2017-10-25	2566.520020	2567.399902	2544.000000	2557.149902	3.874510e+09
2017-10-26	2560.080078	2567.070068	2559.800049	2560.399902	3.869050e+09
2017-10-27	2570.260010	2582.979980	2565.939941	2581.070068	3.887110e+09
2017-10-30	2577.750000	2580.030029	2568.250000	2570.939941	1.739902e+09

To model the dataset successfully, we would need

The number precision and storage format of tick data in the .csv file is identical to the above visualisation of data.

The historic tick data has interesting features which can help us understand the nature of price change in the stock market. The daily return achieved in the markets average to 0% as seen in the graph below, but stock prices have risen. Stock markets are designed to rise over a long time period. The challenge for our model, is to predict small term price fluctuations, so that we can make profits in the long run.

Figure 2.3 -Daily Returns Visualisation

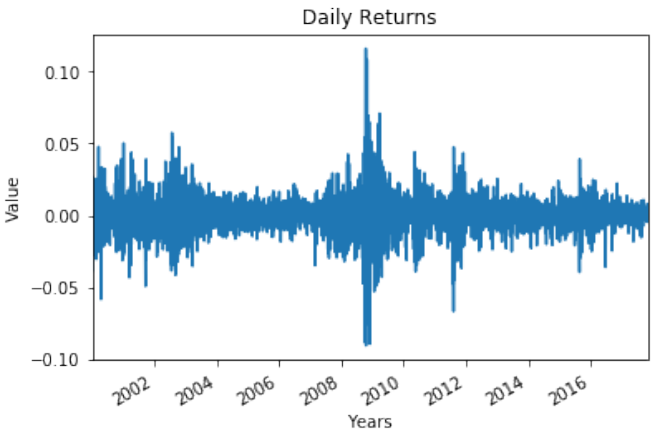
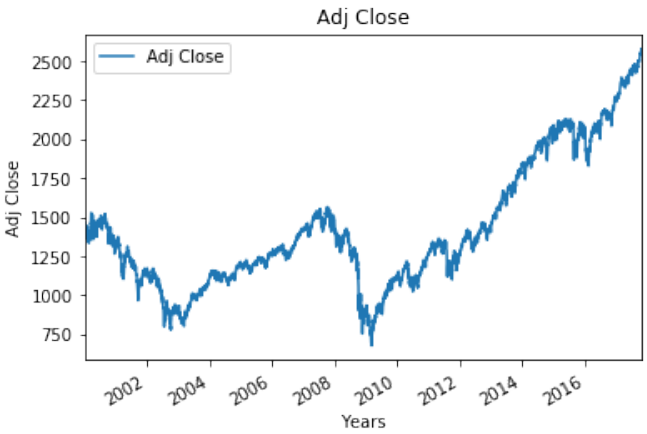


Figure 2.3 - Feature Visualisation



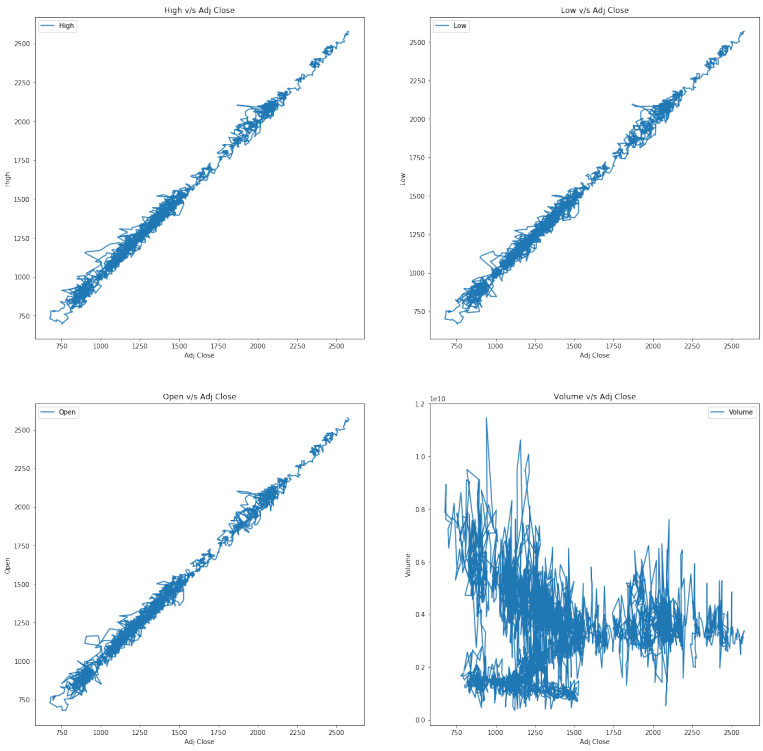
This reiterates the fact that if the model's predictions are slightly amiss, the probability of losing capital is extremely high.

The Figure 2.3, shows that as prices rise cumulatively, they do not rise in a smooth fashion. The curve has plentiful of jagged lines. This might be a hurdle for our model. To differentiate, which spike or fall will actual lead to a trend reversal and which one is random noise.

Exploratory Visualisation

The stock price prediction, is a complex problem, because the changes in values of attributes are very subtle and are barely distinctive.

Figure 2.3 - Feature Visualisation



In graph 2.1, all attributes in our naive dataset, have been plotted against the 'Adj Close' price after 5 days from our training period. The High, Low and Open data follows a clear trend and are directly proportional to the Target price. Intuitively, we may think that is a session has a high Closing price, it will have proportionately high 'Open' and 'High' price. But the straight line graph has a lot of aberrations and the line is very thick for the entire range of 'Adj Close' prices, this reveals volatility. When investing money, it is very important to first secure safety of the capital, and then look for price rise. The amount of volatility and deviation see and the required precision in prediction, makes stock price action a tough phenomenon to model.

Also, in the HLO graphs, we see that all these graphs are extremely similar to one another. The regions where the curve is thick, and thin regions are also similar. Hence, it is presumable that the information received from these 3 attributes is repetitive and is redundant. The 'Volume' parameter has no visible correlation with the 'Target'. It is scattered, and there are frequent outliers. The 'Volume' is completely scattered and a trend is not visible.

Due, to the repetitive and scattered information, our input dataset, is reduced to low dimensional input set. To counter this problem, the raw dataset has been transformed and various technical indicator inputs have been added. This provide clearer signals of a possible increase in price rise and prove to be more helpful to increase the accuracy of prediction.

Algorithms and Techniques

A Long Short Term Memory network along with fully connected dense layers have been used in this model. LSTM's are a special kind of recurrent neural network. Future stock prices are dependent on the stock prices of previous trading sessions. This data is highly sequential, and it is essential to determine underlying price activity in the previous trading sessions. The LSTM model unlike a multi-layer perceptron model uses back propagation through time, i.e. it along with the current data record it also looks back at a fixed number of records during back propagation to train the network. This property gives an LSTM to establish relationships between the previous data and the current target, and makes it a useful tool for time series analysis.

Figure 2.4 - Simple Artificial Neural Net

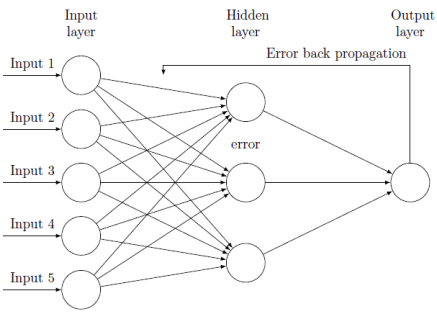


Figure 2.5 - LSTM Hidden Layer

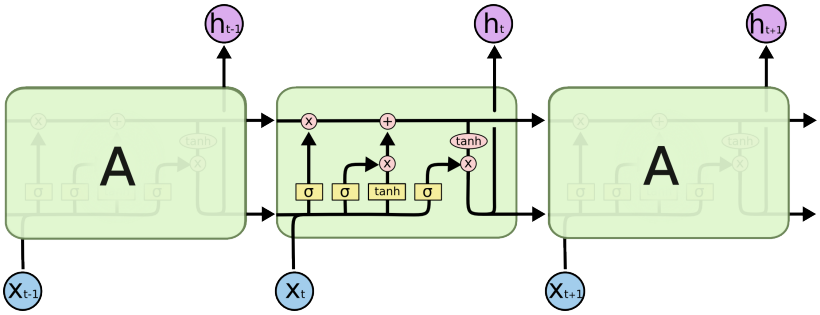


Figure 2.4 shows a simple neural network. Each node is trained using back-propagation based on the current record only. The inputs feed forward in the network, they are multiplied by the weights of each hidden layer, and the resulting output is passed on to the next layer. During back propagation, the output for the specific input is compared with the actual value, and we calculate the error. This error is then fed backwards through the network and all the resulting weights change values based on this error, such that the error is minimised. This operation will take place for each set of inputs and their resulting output. This would complete one 'epoch'. The neural net can be run for a desired number of epochs, till we attain our desired accuracy.

Figure 2.5 shows a LSTM layer node. Each node is trained by back-propagation like the neural network through node h_{t+1}. The error passes from the output node to h_{t+1} eventually to

the LSTM node. The inputs are similarly fed forward like the neural network. Additionally, unlike a simple neural network, at each training step, along with error coming from a node at a higher layer, we also receive error from the same node at instance t-1. This is back propagation through time. The node learns from its own value in the previous time instance. In LSTM's there is a special cell state regulated by 3 different gates c, o, g. These gates, will determine the degree of previous information to be recorded into the cell state and the degree of information to be passed from the cell state into our node. A LSTM network thus trains both vertically through layers, and horizontally through time.

Along with 2 LSTM layers, the model has 2 fully connected Dense Layers. The hyper parameters for the model are:

Training Parameters

1) Epochs - Number of iterations for which the model should back propagate.
2) Batch Size - Number of records to be considered for back propagation in one training instance.
3) Loss Function - The function to be optimise during training to minimise loss.
4) Validation Data - This data can be provided if we want our model to perform hold out validation.

Network Architecture Parameters

1) Number of layers
2) Number of nodes in each layer
3) Type of Layer (LSTM, Dense)
4) Layer specific parameter(see links above)

Data pre-processing Parameters (check Data Preprocessing section)

The dataset was divided sequentially into training and testing segments. Each mini-batch used for training was sequentially loaded into the model, and the resulting weights from each batch were carry for

Benchmark

As a standard benchmark for our model, we use machine learning techniques of Support Vector Machines and a Multi-layer Perceptron Model and the GRU Model for price prediction. ARIMA, GARCH, GRU, SVM and MLP models were considered for use as benchmark models, amongst which the GRU, SVM and MLP model provided the highest return ratio and R2 score, and thus were chosen as final benchmark models.

Each model produced results at a very high rate, so we have not considered training and testing time as a constraint. Further details of the fine tuned model are presented below:

The data parameters for all benchmark models are the same. Optimally tuned data parameters have been used to train and test the benchmark model, so that the model can perform optimally.

Data Parameters:

11. Transforming Window: 11 days
12. Look-back Window: 5 days
13. Feature Extraction Technique: ICA
14. Number of Attributes: 12

GRU Model: The GRU model architecture, and the data and model parameter settings have been listed below.

Figure 2.6 - GRU Model Summary.

Layer (type)	Output Shape	Param #
gru_1 (GRU)	(None, 5, 64)	14784
dropout_5 (Dropout)	(None, 5, 64)	0
gru_2 (GRU)	(None, 5, 128)	74112
dropout_6 (Dropout)	(None, 5, 128)	0
flatten_2 (Flatten)	(None, 640)	0
dense_4 (Dense)	(None, 256)	164096
dropout_7 (Dropout)	(None, 256)	0
dense_5 (Dense)	(None, 512)	131584
dropout_8 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 1)	513

Total params: 385,089
Trainable params: 385,089
Non-trainable params: 0

Model Parameters:

1. Epochs: 100 Training Epochs
2. Loss Function: Mean Squared Error function
3. Optimiser: Adam optimiser.
4. Kernel Initialiser: The dense layers in the network have been initialised using 'he_normal' initialiser found in Keras.
5. Activation Function: Linear

Support Vector Machine Regressor: This is the most commonly used technique for stock price prediction. The model parameters for this regressor are as follows:

Model Parameters:

1. Kernel: Sigmoid
2. Degree: 3
3. Epsilon: 0.001
4. Gamma: 0.7
5. C: 2

Multi-layer Perceptron Model: This model is a densely connected artificial neural network model. The model has 2 hidden layers and the following architecture:

Figure 2.7 - Multi-layer Perceptron Model Summary.

Layer (type)	Output Shape	Param #
dense_7 (Dense)	(None, 64)	832
dropout_9 (Dropout)	(None, 64)	0
batch_normalization_1 (Batch	(None, 64)	256
dense_8 (Dense)	(None, 128)	8320
dropout_10 (Dropout)	(None, 128)	0
batch_normalization_2 (Batch	(None, 128)	512
dense_9 (Dense)	(None, 256)	33024
dropout_11 (Dropout)	(None, 256)	0
batch_normalization_3 (Batch	(None, 256)	1024
dense_10 (Dense)	(None, 1)	257

Total params: 44,225
Trainable params: 43,329
Non-trainable params: 896

Model Parameters:

1. Epochs: 1500 Training Epochs
2. Loss Function: Mean Squared Error
3. Optimiser: Adam optimiser.
4. Kernel Initialiser: The dense layers in the network have been initialised using ‘he_normal’ initialiser found in Keras.
5. Activation Function: Rectified Linear

III. Methodology

Data Preprocessing

The steps used to prepare the data for the final model are given below:

1. Raw price data is imported from 'Yahoo Finance'

2. This data is fed into a 'transform_data' function which will add additional technical indicators as attributes. These added technical indicators, are added as they remove noise from the price action data. These indicators will take a distinctive value for specific break-through points in the data, and have a negligible or constant value for all other values. The technical indicators added to the data are:
    1. Daily Returns
    2. 5-day Momentum
    3. Simple Moving Average
    4. Exponential Moving Average
    5. Upper & Lower Bollinger Bands
    6. 3-day Rate of Change (ROC)
    7. Moving Average Convergence Divergence (MACD)
    8. Pivot Point
    9. Stochastic Indicators
    10. 8-week Weekly Returns
    11. 2-month Monthly Returns
    12. 14-day Relative Strength Index (RSI)
    13. 14-day Average True Range (ATR)
    14. On Balance Volume (OBV)
    15. Fibonacci Retracement over 7, 14, & 21 days with 38.2, 61.8 and 50% thresholds.
    16. 14-day Average Directional Movement

*All indicators for which the time window has not been specified, the time window can be fixed by the us. It is a hyper-parameter we can tune.

3. The data attributes are standardised and then normalised using MinMax scaling technique. To standardise each parameter, the following methodology is used :

$$Z = \frac{\bar{X} - \mathrm{E}[X]}{\sigma(X)/\sqrt{n}}.$$

4. A ‘Target’ column is added to our data frame which contains the regression target. The ’n_day_prediction’ functions produces this attribute. The number of days for which the prediction is to be made is specified as a parameter for this function. Hence, we can produce predictions for multiple time windows.

5. After preparing our dataset, for further processing, the data is split into 3 parts using the ‘create_dataset’ function. Roughly, 70% of the data is held out for training. The final 20% of our data is reserved for hold our cross validation. The rest of our data is used for testing the model.

6. After transforming our data, we have 47 attributes in it. To reduce the number of epochs needed for our model to provide similar accuracy, we can perform feature extraction on our data. This would also eliminate all indicators which provide little information, and only relevant attributes in our dataset are maintained. We have applied Principal Component Analysis, Independent Component Analysis and Factor Analysis and compared the result for each of the 3 methods using a LSTM model, predicting prices 5 days in advance. The results, post applying each technique are discussed in detail in the Refinement section. Based on these results we have chosen the most suitable technique for our data, which produced the best results.

7. LSTM model takes as input a 3-Dimensional dataset. In the function ‘reshape’, we convert our 2-Dimensional price data into 3-Dimensional Dataset. Records from ’n' previous trading session are added as the third dimension for every record in our dataset. So in our data frame, we add an additional dimension which stores data for past ’n' trading session. The number of loopback days, or the ’n’ can be tuned as a hyper parameter and its optimal value can be found.

Implementation

A Recurrent Neural Network model with Long Short Term Memory layers has been used to predict stock prices

Figure 3.1 - Optimal LSTM Model Summary.

Layer (type)	Output Shape	Param #
lstm_5 (LSTM)	(None, 5, 64)	29184
dropout_9 (Dropout)	(None, 5, 64)	0
lstm_6 (LSTM)	(None, 5, 128)	98816
dropout_10 (Dropout)	(None, 5, 128)	0
flatten_3 (Flatten)	(None, 640)	0
dense_7 (Dense)	(None, 256)	164096
dropout_11 (Dropout)	(None, 256)	0
dense_8 (Dense)	(None, 512)	131584
dropout_12 (Dropout)	(None, 512)	0
dense_9 (Dense)	(None, 1)	513

Total params: 424,193
Trainable params: 424,193
Non-trainable params: 0

Model Parameters:

1. Epochs: 125 Training Epochs
2. Loss Function: The loss function used is the mean squared error loss function and the metric used is also mean squared error. This has been used because, in a regression based problem, this metric would lead to best result.
3. Optimiser: Adam optimiser.

4. Kernel Initialiser: The dense layers in the network have been initialised using 'he_normal' initialiser found in Keras.
5. Dropout: To ensure that all layers have been trained equally well a dropout of 0.3 has been applied to each layer.
6. Activation Function: Linear

Refinement

The performance of our model heavily depends on our training data. To refine our performance we can tune these hyper-parameters. Tuning can be done to hyper parameter in the input dataset, which can produce better regression results. The control variables while tuning these parameters are as follows:

1) Epochs -100
2) Optimiser - Adam
3) Look back period - 5 days
4) Data Transformation Window - 14 days
5) Feature Extraction Method - ICA
6) Number of Attributes - 12

The list of tuneable hyper-parameters and the resulting change is the results of the model are given below.

1) Feature Extraction : Extraction techniques like Independent Components Analysis, Principle Components Analysis and Factor Analysis, can be applied to the transformed dataset. The results of applying each of these 3 parameters, and the resulting change in accuracy is following:

Figure 3.2 - Performance Visualisations with different Feature Extraction Methods

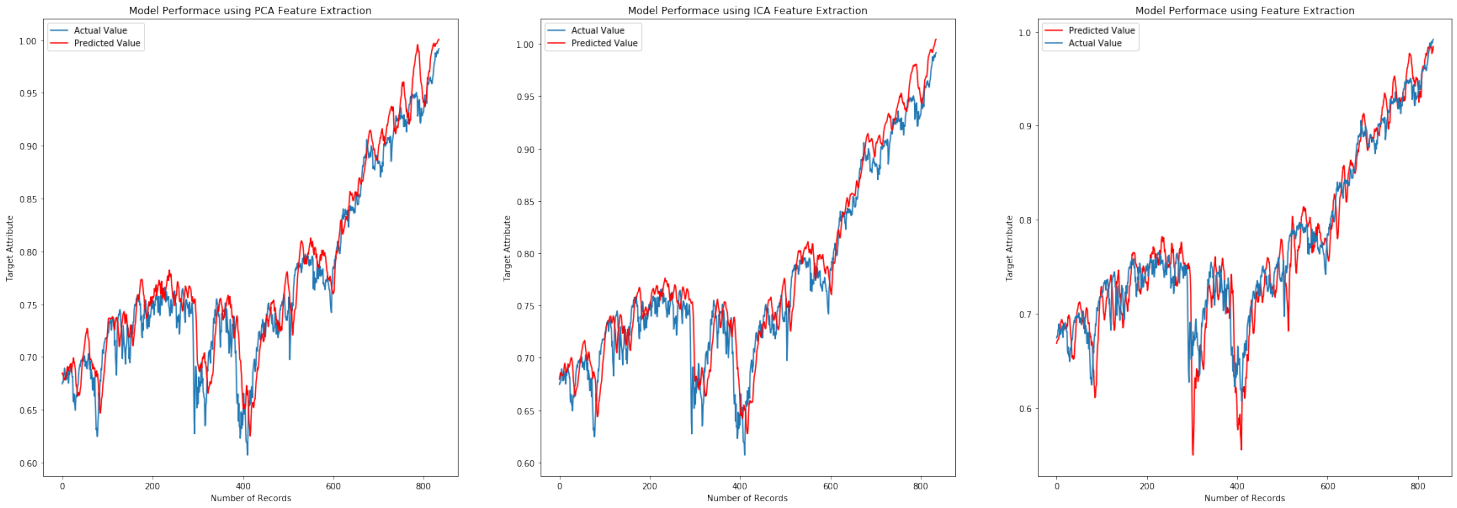


Table 3.1 - Performance Metrics with different Feature Extraction Methods

Method	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
PCA	0.000583	0.926666	1.884503	0.428742	0.158084
ICA	0.000551	0.930741	2.045787	0.431138	0.168862
FA	0.000724	0.908963	0.249341	0.301796	0.314970

2) Transforming Window: Majority of our attributes in our transformed dataset are technical indicator attributes. Hence, we need to discern the number of days for which these technical indicators should be calculated. Various window sizes and their corresponding impacts on the result have been reported.

Table 3.2 - Performance Metrics with different Transforming Windows

Window Period	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
3	0.000555	0.930104	2.810439	0.441916	0.171257
7	0.000424	0.946651	4.831541	0.250299	0.247904
11	0.000447	0.943751	5.962667	0.240718	0.262275
14	0.000660	0.916967	2.221774	0.520958	0.153293

3) Number of attributes: The number of attributes in the data decide how well the model will perform. Post data transformation, we have 47 data attributes. Ideally we would like to retain all the components, but this would cause the curse of dimensionality problem. Hence, it is essential to determine a suitable number of components, for which the model, provides maximum efficiency.

Table 3.3 - Performance Metrics with different Number of Attributes

No. of Attributes	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
7	0.000503	0.936713	2.945463	0.246706	0.298203
12	0.000553	0.930429	4.921270	0.398802	0.176048
18	0.000618	0.922229	1.988380	0.522155	0.140119
25	0.000536	0.932515	1.705043	0.334132	0.238323
32	0.000470	0.940888	3.763431	0.247904	0.273053
45	0.000733	0.907813	1.070164	0.439521	0.201197

4) Look Back Window: A LSTM model can back propagate through time unlike a normal ANN. This makes it very suitable for use in time series analysis. Although, we have to decide how many days should the model back propagate through in time. Ideally, we would want that we back propagate through the entire training time series, but this would cause the vanishing gradient problem. Although a LSTM eliminates the vanishing gradient problem, we can tune a this window value, such that it is optimises the model predictions.

Table 3.4 - Performance Metrics with different Look Back Windows

Look back Period	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
No look back	0.000387	0.951347	3.1390147	0.329762	0.165476
3 Days	0.000546	0.931349	4.546969	0.427718	0.173238
5 Days	0.000534	0.932749	4.428151	0.408383	0.171257
8 Days	0.000690	0.913146	1.675390	0.537259	0.131010
11 Days	0.000595	0.925140	5.003220	0.451146	0.154403
14 Days	0.000694	0.912629	2.407061	0.529055	0.138014

Figure 3.3 - Performance Visualisation with different Number of Attributes. The graphs correspond to: (1,1) : 7, (1,2): 12, (1,3): 18, (1,4): 25, (1,5): 32, (1,6): 45 attributes

Figure 3.4 - Performance Visualisation with different Look Back Windows. The graphs correspond to: (1,1) : 0, (1,2): 3, (1,3): 5, (1,4): 7, (1,5): 11, (1,6): 14 look back days

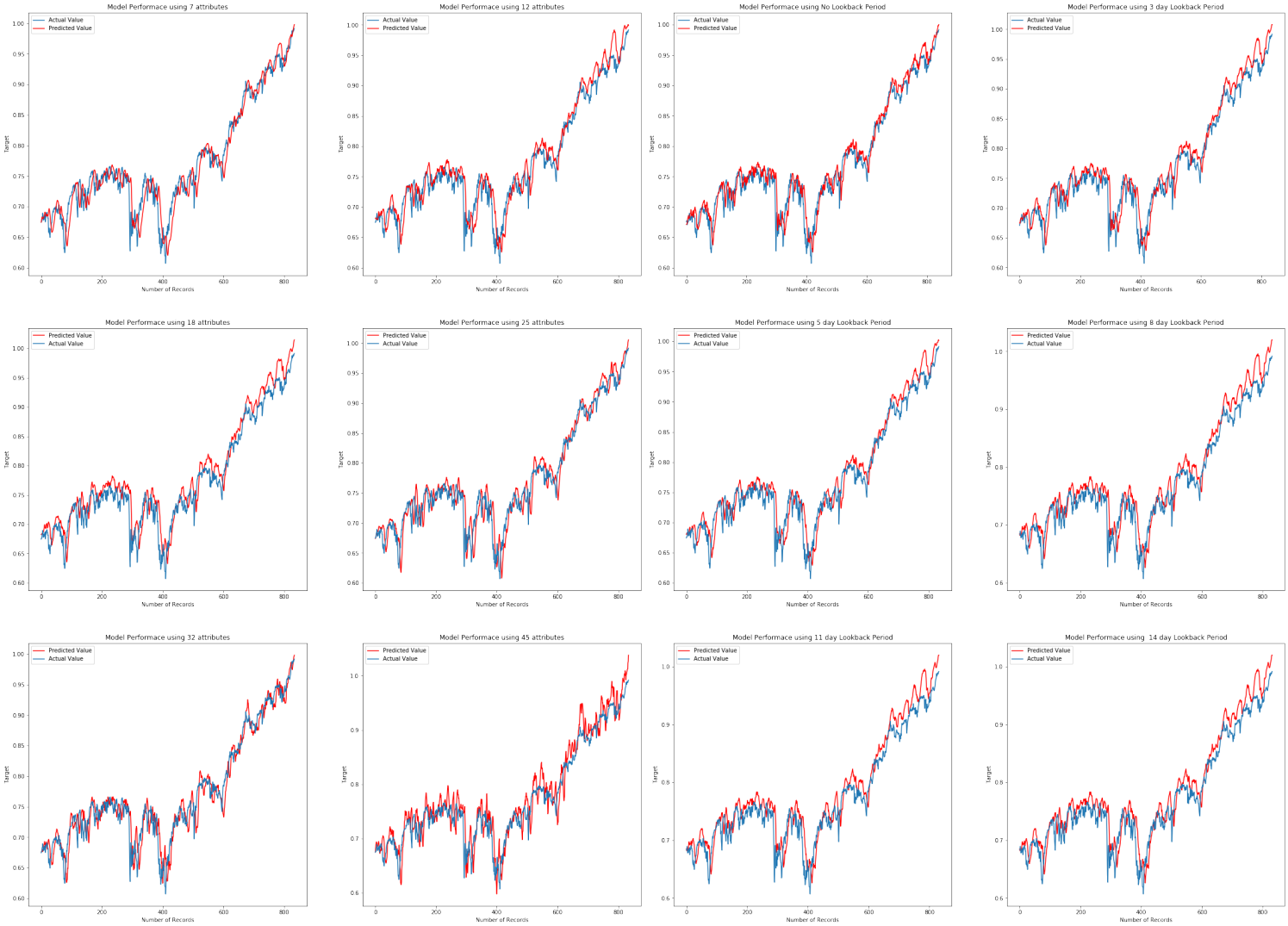
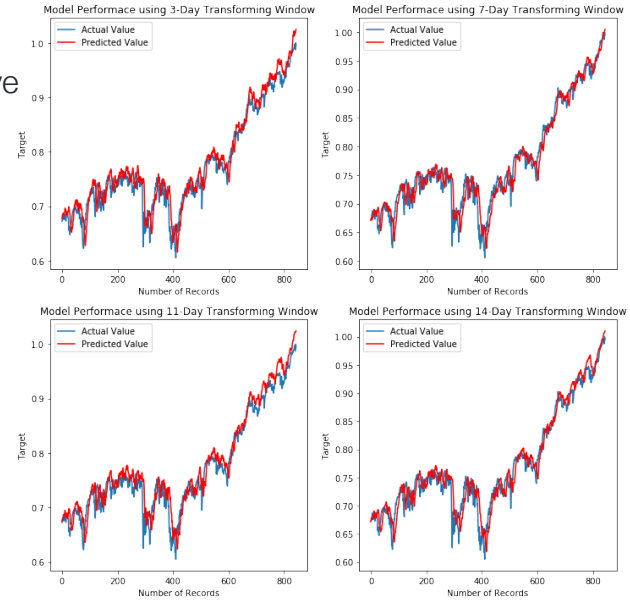


Figure 3.5 - Performance Visualisation with different Transforming Windows

Optimal Model

The model for our problem statement has all the above defined model parameters and the following data hyper parameters:

1) Feature Extraction Technique: ICA
2) Look-back Window: 5 days
3) Number of Attributes: 12
4) Transforming Window: 11 days



IV. Results

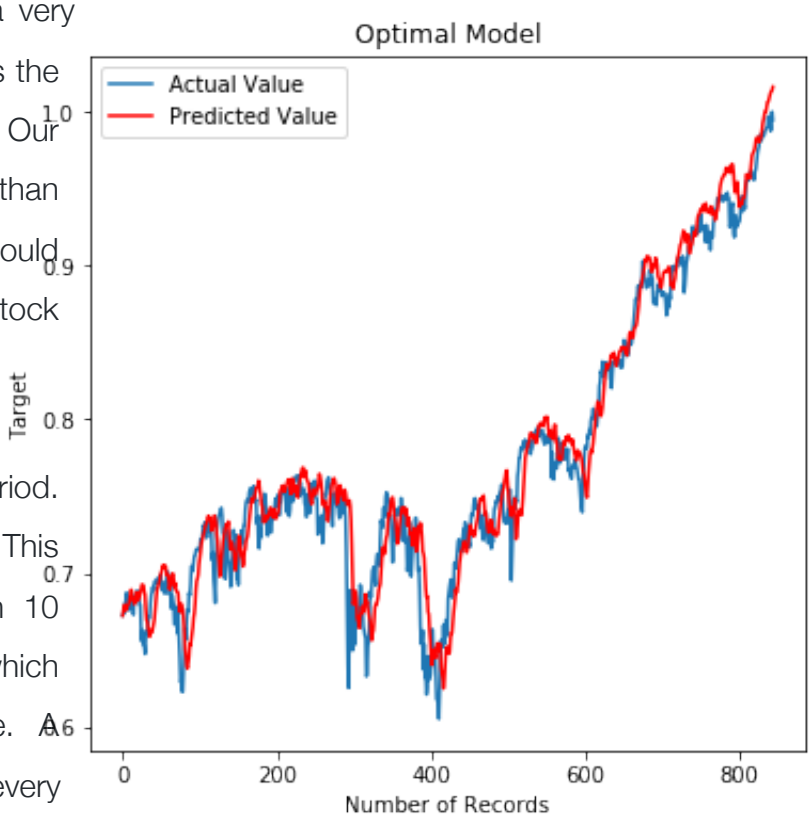
Model Evaluation and Validation:

Through tuning previously mentioned parameters a final model has been developed. The final model provides the following testing metrics.

The Performance Results for the Optimal LSTM Model for S&P 500 is :

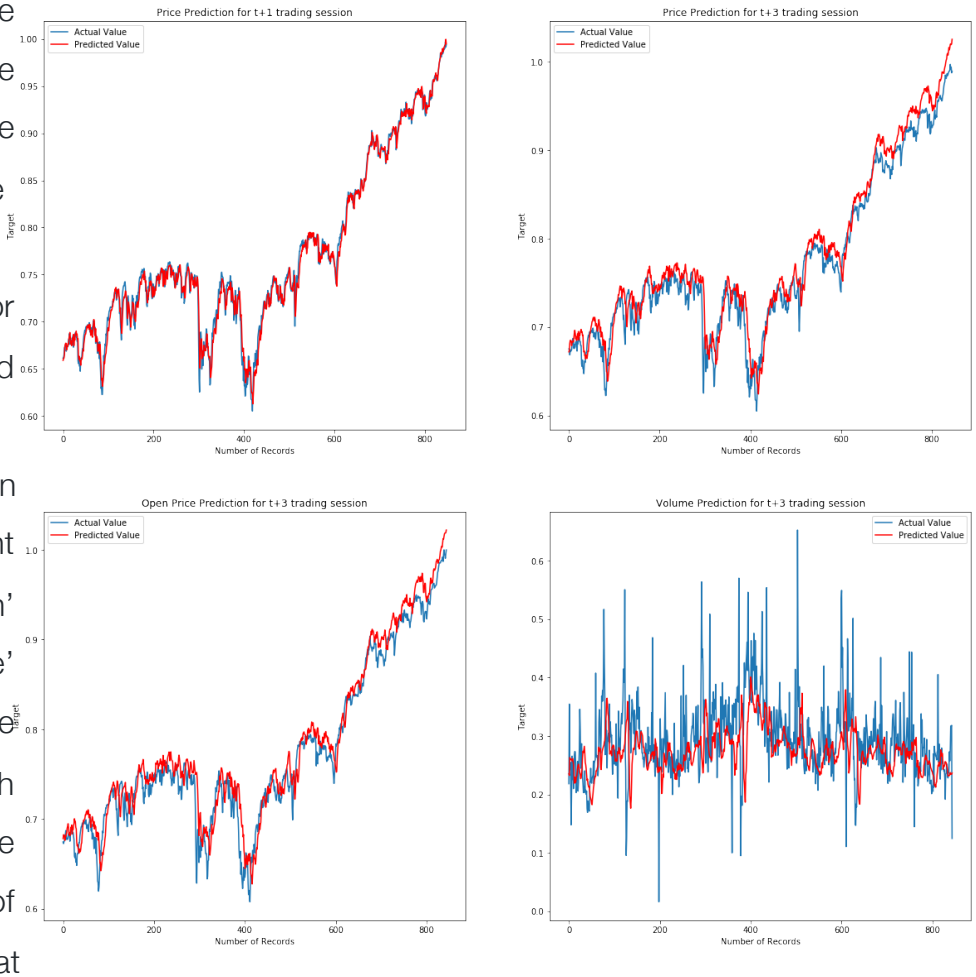
- The Return ratio is 4.308454
- The R^2 score is 0.948616
- The Mean Squared Error is 0.0004285
- The Optimism ratio is 0.31043
- The Pessimism ratio is 0.203791

Figure 4.1 - Performance Visualisation of the Optimal Model



The model has a high R^2 Score and a very low RMS Error. The model hence, predicts the underlying trend with great accuracy. Our model provides a result 4.3 times better than the naive strategy. This means that if we would make a profit of $1 by simple buying a stock and hold it for a time period, if we buy/ short using predictions of our model we would make $4.3 over the same time period. The model has an optimism ratio of 0.3. This means that every 3rd price prediction in 10 price predictions has a price prediction which is 1.5% greater than the actual price. A pessimism ratio of 0.2 means than 2 in every 10 predictions have a prediction lower than 1.5% of the actual value. These 2 parameters determine the confidence level of our prediction. This parameter is not optimal, and can be worked upon. But, we can attribute this to the volatility in prices, and sudden prices spikes. Hence, we can conclude that our model is successful in most instances, and is reliable. This is a satisfactory result and our model can be trusted.

Figure 4.2 - Performance Visualisations for Robustness Test



To test the robustness of the model, we use the same model to make price predictions for different time windows, and for attributes like 'Open' and 'Volume', for which it has not been trained for.

The model succeeds in predicting prices for a different time window and the 'Open' attribute. But, the 'Volume' prediction has a low R^2 score and an unconventionally high 'Return Ratio'. The visualisation of predictions of 'Volume' attribute show that our model can successfully capture the trends seen but cannot model the high volatility in this attribute. Because of this high volatility, this parameter is not suitable for prediction.

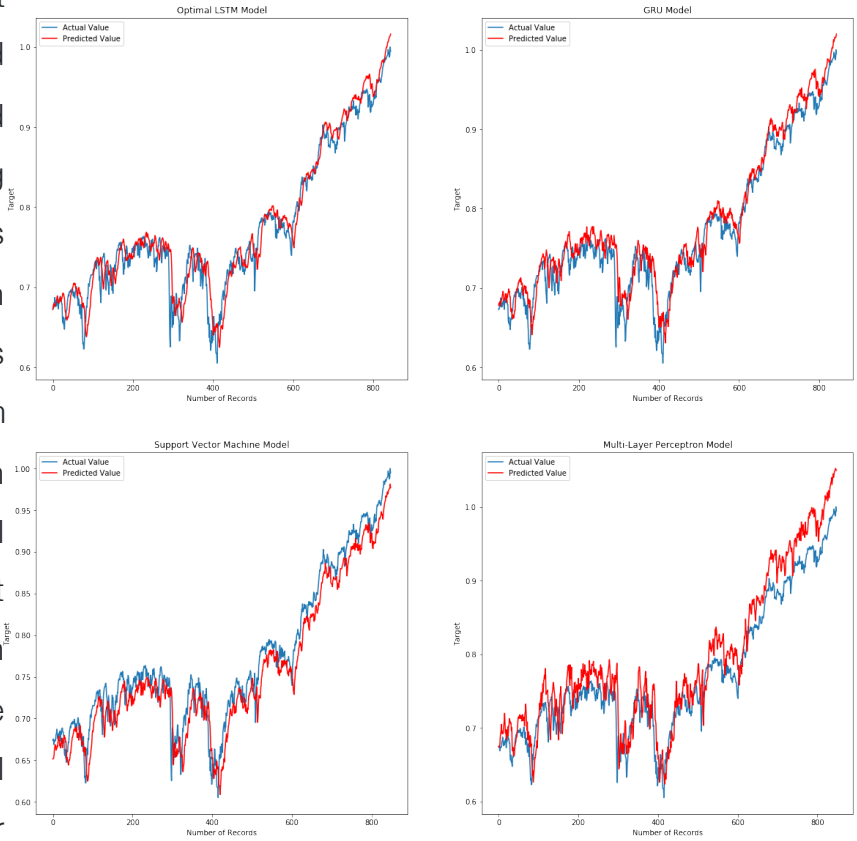
The model thus, proved to be both versatile and robust as it can adapt to different training sets and provide satisfactory results.

Table 4.1: Performance Metrics of Robustness Test

Method	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
1-Day Prediction	9.08175E-05	0.988888	0.962664	0.080094	0.127208
3-Day Prediction	0.000427	0.948221	2.254592	0.484633	0.106383
Open Price Prediction	0.000424	0.949290	3.712512	0.4158768	0.127962
Volume Prediction	0.004678	-0.105015	152.6911	0.3447867	0.587677

Justification:

Figure 4.3 - Performance Visualisation of Benchmark and Optimal Models



In our analysis, all important aspects of a regression result, and the monetary returns achieved have been reported. Trading imposes a monetary risk, and thus the confidence level for each prediction by the model is extremely important. The optimism ratio of our optimal model, is much better and lower than all other all other model. This would mean that the probability of the prediction being closest to the actual price will be highest in the optimal model, as compared to other models. Our model has the highest R^2 score. This means that our model follows price action most closely than all other benchmark models. The lowest mean squared error re-confirms this. Our model although, has a sub-optimal returns ratio, and the GRU model outperforms our model in this parameter. This is a very important metric, but because our model does better in other important aspects like optimism ratio, our model has more consistency and would minimise risk better. Although the pessimism ratio of our model is sub-optimal to the GRU model, owing to better results in other performance metrics, we can disregard this. Hence, our optimal model provides the highest reward while minimising risk.

Table 4.2: Performance Metrics of Benchmark & Optimal Model

Model	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
LSTM	0.000428	0.948616	4.308454	0.310426	0.203791
GRU	0.000511	0.938698	5.722242	0.447867	0.120853
SVM	0.000543	0.934952	-1.858130	0.080094	0.631331
MLP	0.001052	0.874004	2.478719	0.689046	0.115430

V. Conclusion

Free-Form Visualisation

This model, has been trained and build specifically for S&P500. It has only looked at the price change patterns and learned from S&P500. To test the universality and how well the model generalises for stocks it has not encountered before, it should be tested on different stocks. Different stocks have a different price and trading patterns as traders with different styles trade it. Hence, the model is not expected to produce dependable results. Testing the performance of the model for different equities would represent that how well has the model been able to capture general trading strategies. These strategies correspond to the retail traders and investors who use basic skills to buy equities.

Table 5.1: Performance Visualisations with Other Stocks

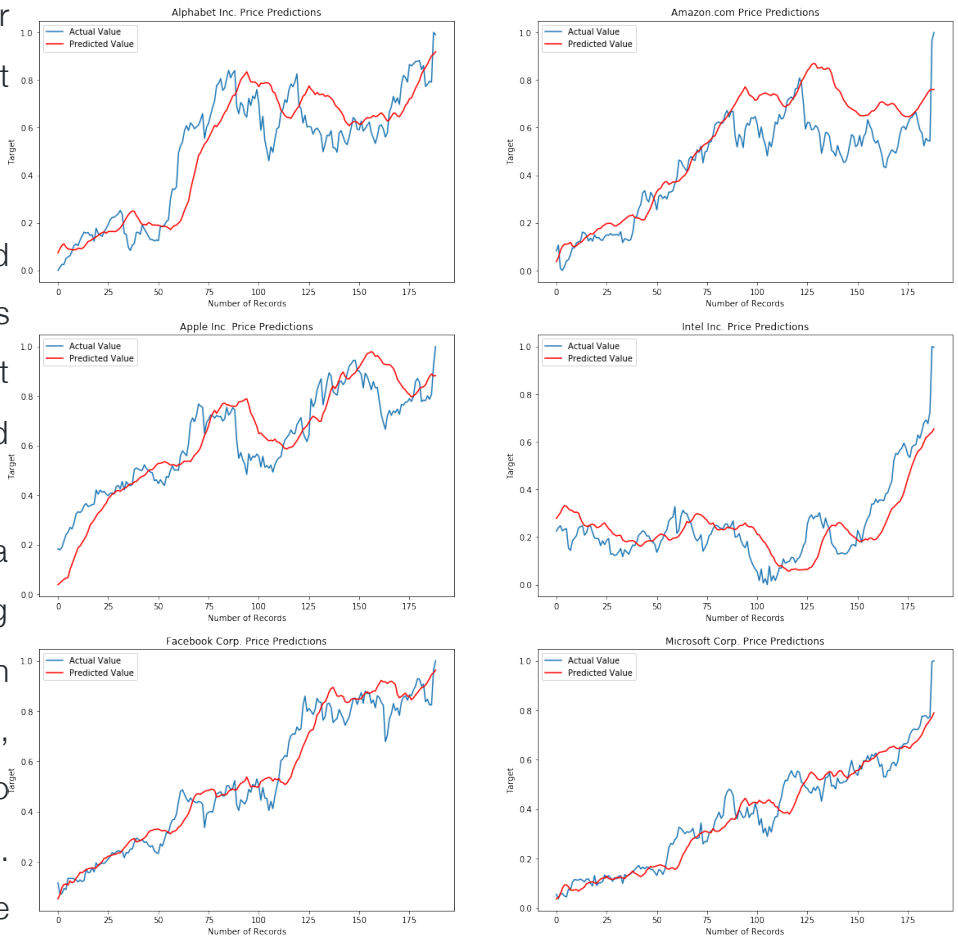


Table 5.1: Performance Metrics with Other Stocks

Model	RMSE	R^2 Score	Return Ratio	Optimism Ratio	Pessimism Ratio
Alphabet Inc.	0.015346	0.769627	1.297281	0.507936	0.460317
Apple Inc.	0.010488	0.698532	1.102767	0.470899	0.444444
Facebook Inc.	0.00477	0.930562	2.738894	0.529101	0.317460
Intel Corp	0.010292	0.622686	-3.661072	0.507936	0.470899
Amazon.com	0.018111	0.574846	7.490526	0.719577	0.206349
Microsoft Corp.	0.00391	0.910585	4.123487	0.444444	0.470899

Reflection

The project can be broken down into the following phases:

1. Definition of the problem statement, the input dataset, the target variable and the nature of solution. (classification or regression)
2. Download of the data from open-source and licensed sources.
3. Importing the data, transforming its attributes, adding more attributes to the data.
4. Standardise and normalise the data.
5. Applying feature selection to choose appropriate number of attributes.
6. Splitting data into training and testing and converting it into a model compatible format.
7. Definition of model network architecture.
8. Tuning of all data and model hyper parameters to find the optimal solution
9. Preparing benchmark models for performance comparison.
10. Producing optimised results and comparing with that of benchmark models.

I found stage 3, stage 7 and stage 8 very interesting. By determining appropriate features and their time scale of calculation, as we can inject domain knowledge into our model. Stage 7 and stage 8 were the most important stages for model performance. The quality of features added and the correct time window for them control the model performance. Thus, it is essential to determine the correct mix of features. I used a grid search approach to find optimal features. The most interesting part was discovering how and degree by which each hyper parameter influences the final model performance.

Improvement

Further refinements in this project can be made in the data preprocessing part of the model. Genetic algorithms could have been used for feature extraction. This could have resulted in better results for our model. The normalisation of the model could have been performed using Box-Cox transformations. Implementation of genetic algorithms was very intricate, and hence was not used. Because of standardisation of the dataset, records took negative values. Hence, box-cox transformation was not feasible. Difficulties were also faced while coding up the statistical models. Despite smoothening of the data series, by differentiation, the statistical model failed to produce adequate results.