CAPSTONE PROJECT **ROHIT SINHA**

MACHINE LEARNING NANODEGREE **1st May 2018**

DEFINITION

Project Overview

The capstone project is to build a stock price indicator using Python. The investment and trading industry makes use of different tools to predict stock prices (Fig 1). Technical analysis is one of the means to evaluate stock prices using past prices and volume. Time series forecasting using ARIMA can also be used to predict future stock prices. The project evaluates machine learning techniques to predict daily stock price movement using time series modelling and features created from technical indicators. Numerous academic research have already been carried out in this field. Some of the relevant papers related to our project is listed below:

1. http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf
2. http://cs229.stanford.edu/proj2015/009_report.pdf
3. http://cs229.stanford.edu/proj2012/ShenJiangZhang-StockMarketForecastingusingMachineLearningAlgorithms.pdf

The papers discussed above make use of conventional machine learning models as well as deep learning to predict stock prices.

In the first part of the project we evaluate time series forecasting for stock prices to lay the ground work for more sophisticated machine learning models. We explore stationarity, autocorrelation and differencing of the time series data. Using these concepts we create a naïve model to predict stock prices based on ARIMA modelling. The second part of the project involves creating features from the raw time series data of stock prices. We introduce new features in the form of technical indicators, which will be used to predict the stock price. The final part of the project involves creating a trading strategy on the basis of new features created. We create a machine learning model to predict the stock price movement and create a Long-Short strategy on the basis of our predictions. We then compare our results to a Long only strategy.

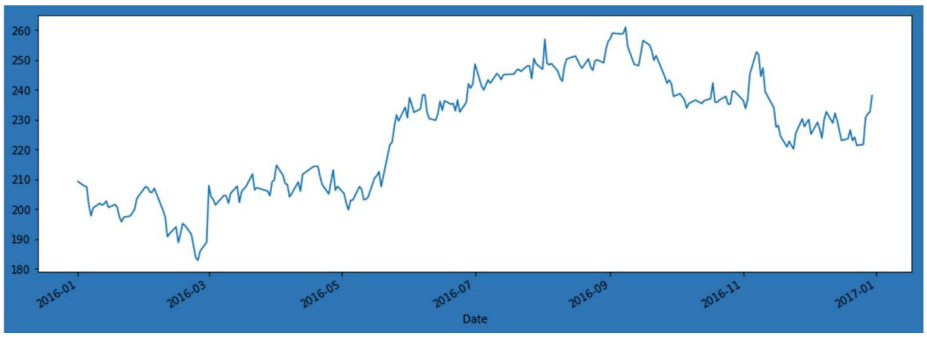

Fig 1. STOCK PRICE MOVEMENT

Problem Statement

The goal is to create a profitable stock investing strategy using Machine Learning. Before evaluating the strategy, we evaluate the time series using statistical techniques. A naïve model is created using the time series modelling and the stock prices predicted using the equation fitted for the model. Next we make use of *Classification* to predict the movement of the price for the next day (Up or down). The features used for the classification problem are the technical indicators of stock prices.

The tasks involved are the following:

- Download the stock prices
- Evaluate the stationarity of the data
- Create a naïve stock predictor using *ARIMA* modelling and predict the prices
- Create a stock predicting model using *Classification* (*Logistic Regression* and *Neural Network*)
- Create a Long-Short strategy using the models created
- Compare the Long-Short strategy performance vs a Long only strategy
- Compare the predictive power of the model over short, medium and long term horizons

Metrics

The metrics used to evaluate our supervised learning model is *Accuracy* of the model.

Accuracy is defined as follows:

$$\frac{\textit{Number of days the model correctly classified the testing data}}{\textit{Total Number of testing days}}$$

Accuracy alone is not a good measure to evaluate the trading strategy. To compare the performance of trading strategy, we need to evaluate the **Returns** and **Volatility** of our strategy.

Return is defined as *"The change in the value of investment expressed as a proportion of the original investment"*. To compare return across different time horizons, it is important to convert each return into annualized return.

To calculate the return of our strategy, we invest a notional of $100 in our long short strategy and compute the final value of our investment. The change in value of our investment as a proportion of our initial investment gives us our Return metric. It is then annualized to be compared across different time periods.

Volatility is a statistical measure of dispersion of Returns. It is measured as the standard deviation of the returns of the security. We compute the volatility of our strategy using the

standard deviation of the daily returns. Like Returns, Volatility is also annualized to be compared across different time periods.

To comment on the performance of a strategy, we have to look at the Returns and Volatility of the strategy combined.

The other statistical measures used to evaluate the time series model are:

- **Dickey-Fuller Test:** It is a statistical technique to evaluate the *stationarity* of a time series model. Stationary data has a constant mean and variance over time and does not have any trend or autocorrelation. It is important to have a stationary data to make predictions as the mean and variance are constant over time. *Dickey-Fuller* test has a null hypothesis that the data is non stationary.
- **ACF, PACF plots:** *Autocorrelation* and *Partial autocorrelation* is plotted for different time lags to ensure that the data is stationary. Autocorrelation is defined as *"Correlation between the elements of a series and others from the same series separated from them by a given interval."* While Partial autocorrelation is *"The partial autocorrelation at lag k is the correlation that results after removing the effect of any correlations due to the terms at shorter lags."* The time series which we are trying to predict should not have Autocorrelation or Partial Autocorrelation to avoid non stationarity of data.
- **ARIMA model**: Once the stationarity of the data is evaluated, we create a naïve ARIMA model. The ARIMA model stands for "Auto Regressive Integrated Moving Average".

More details about the statistical techniques is mentioned in the Algorithms and Techniques subsection of the Analysis section.

As mentioned above *Accuracy* alone is not a good measure of performance of trading strategy. If the strategy is accurate on days when there was a major market movement, the *Returns* of our strategy will be higher. So we can have better Returns even with lower accuracy. However accuracy should not be very low such that the model is always predicting wrongly.

ANALYSIS

Data Exploration

The data has been downloaded from Yahoo finance. (https://in.finance.yahoo.com/).We have downloaded data for three stock listed in the National Stock Exchange of India (NSE) INFOSYS (INFY.NS), HDFC BANK (HDFCBANK.NS), ITC LIMITED (ITC.NS).

The three stocks belong to different sectors, so that we remove any sector bias in our study. The data has been downloaded from 2000 onwards. The features available from the time series are the OPEN, HIGH, LOW and CLOSE prices. We create new technical factors using the existing features. The existing features will be highly correlated with each other (Fig 2) and hence we create new features which are not correlated (Fig 3) with each other and perform a better task at prediction.

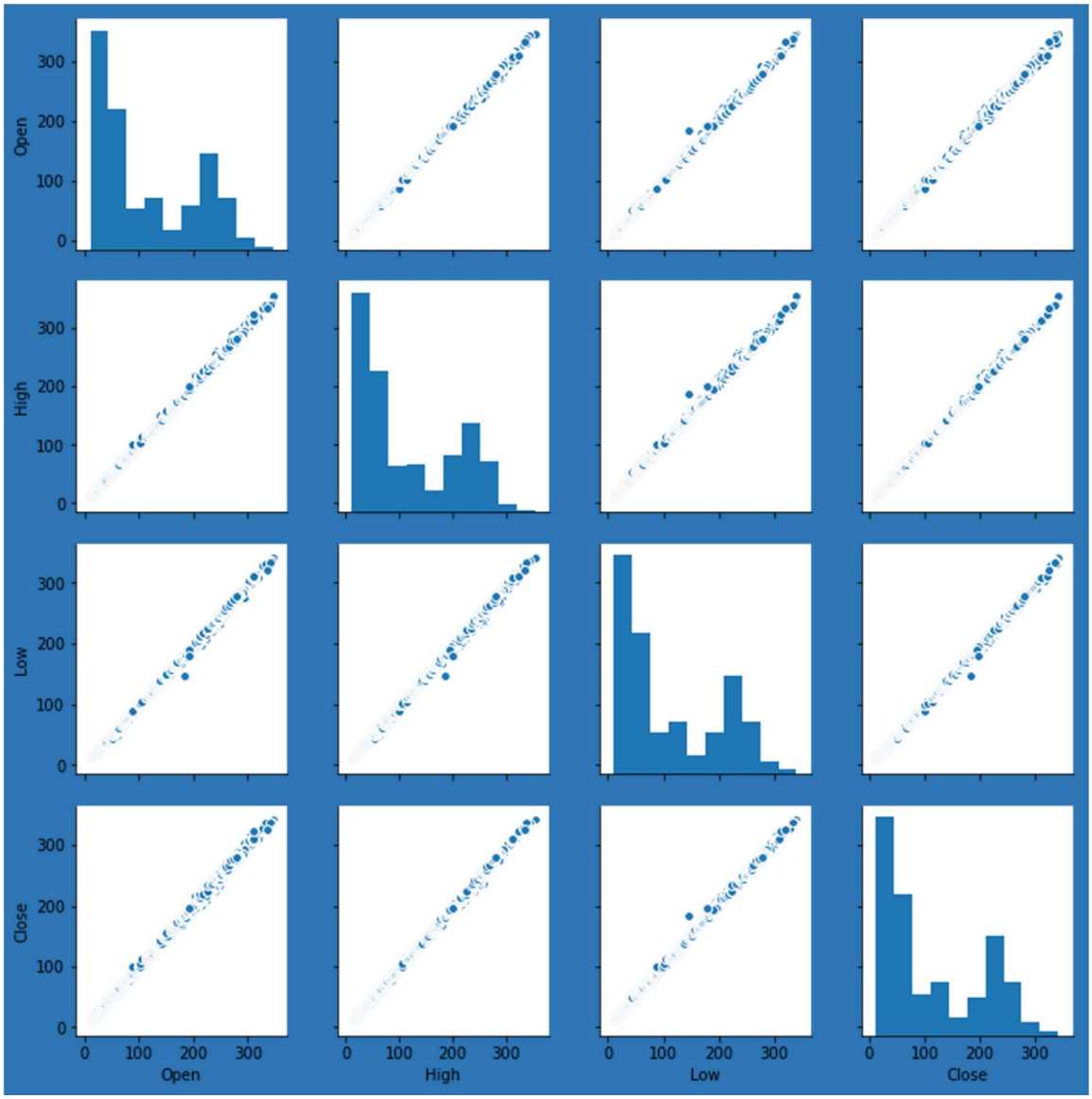

Fig 2. PAIRPLOT OF ORIGINAL FEATURES

We can see the features are *Lognormally* distributed with a negative *skew*. This is the typical distribution of Stock prices. We need to look at features which are non-correlated and then check the distribution of the features.

The new technical features are created (using *TA-Lib* library from www.quantopian.com)

- High minus Low (H-L), Open minus Close (O-C), Relative Strength Indicator (RSI)
- Average Directional Index (ADX), Parabolic SAR, Williams Percentage Range (Williams-R)

The correlation matrix of the above technical indicators is shown in Fig 4. The technical indicators are used to predict the next day's price movement (Up or down).

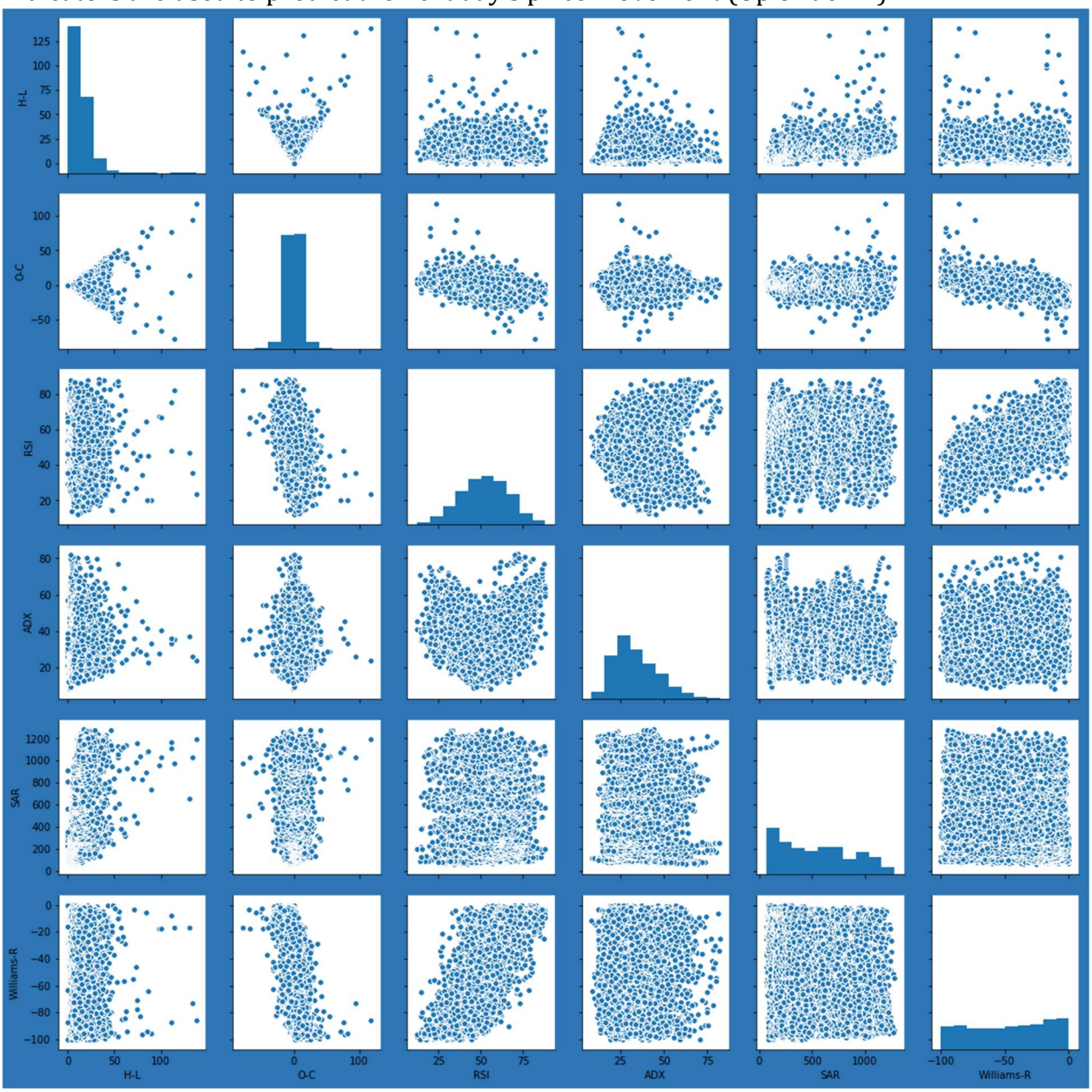

Fig 3. PAIRPLOT OF TECHNICAL INDICATORS

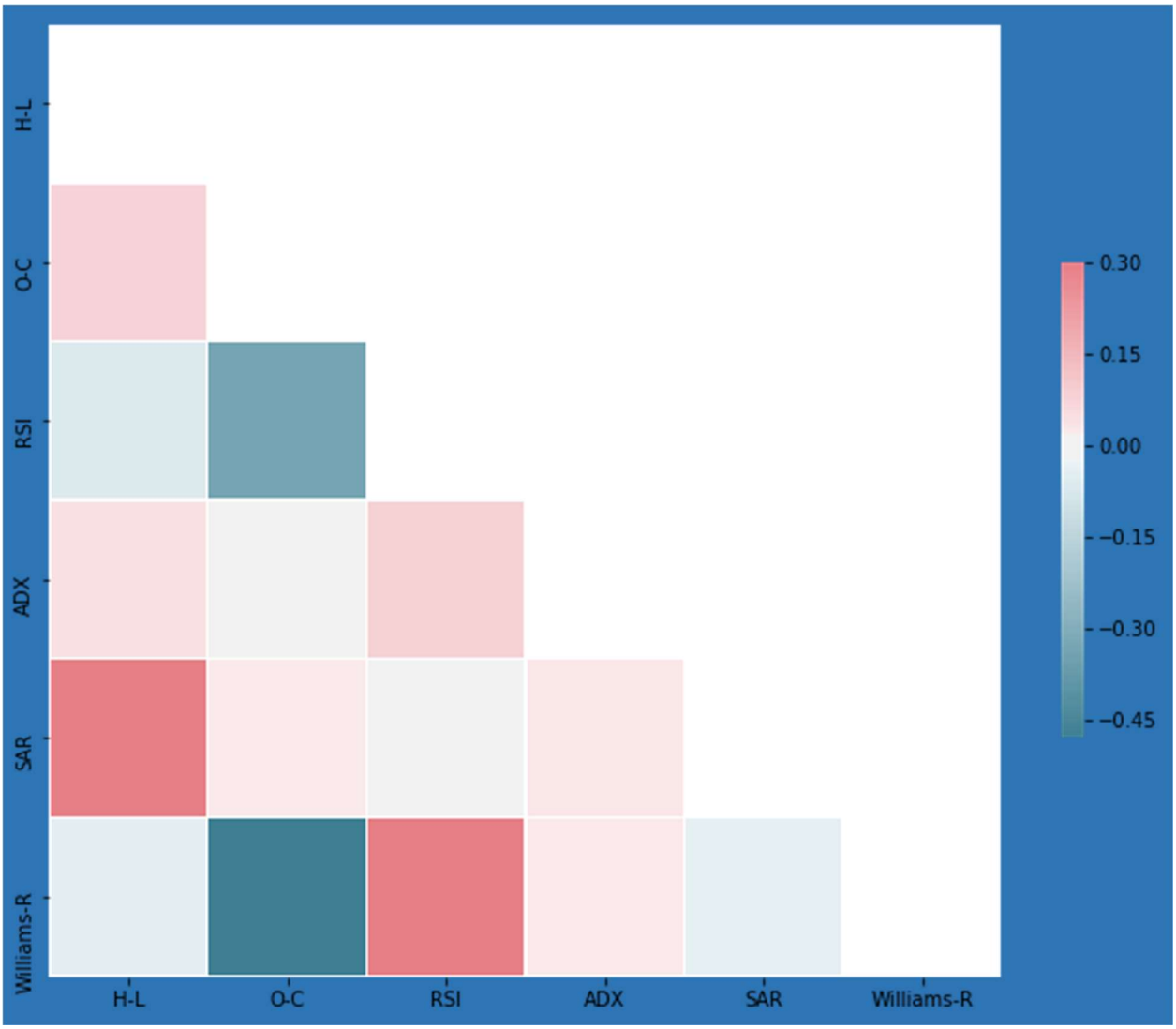

Fig 4. CORRELATION MATRIX OF TECHNICAL INDICATORS

The pairplot of the new features clearly show the change in the distribution of the features from a negative skewed distribution to a more even distribution. Only the feature High minus Low (H-L) is still have a negative skew. The correlation plot above also shows that the features are not highly correlation with each other.

Algorithm and Techniques

DICKY-FULLER Test is used to check for the stationarity of the data. A stationary time series is one which has a constant mean and variance and no trend and autocorrelation exists in the time series.

The actual time series of stock prices is not stationary but if we transform the data and take daily difference, the new resulting series is stationary. Prediction is correct only on

stationary data because the mean and variance of the series does not vary over time. *Hence we should always forecast a change in the stock price rather than the absolute stock price due to the stationary nature of the daily difference series.*

We have plotted below the time series of daily difference of log transformed stock prices. The test statistic for DICKY-FULLER test rejects the null hypothesis and suggests that the data is stationary in all the three log transformed daily difference series as shown by the constant mean and standard deviation of the series.

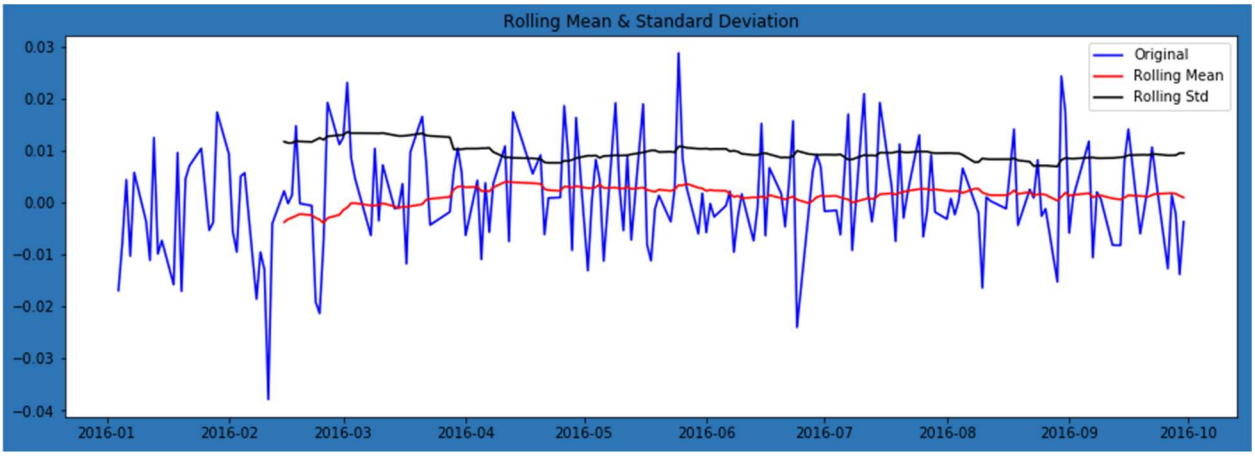

Fig 5. Stationarity of the Log Transformed Daily Difference HDFC Price

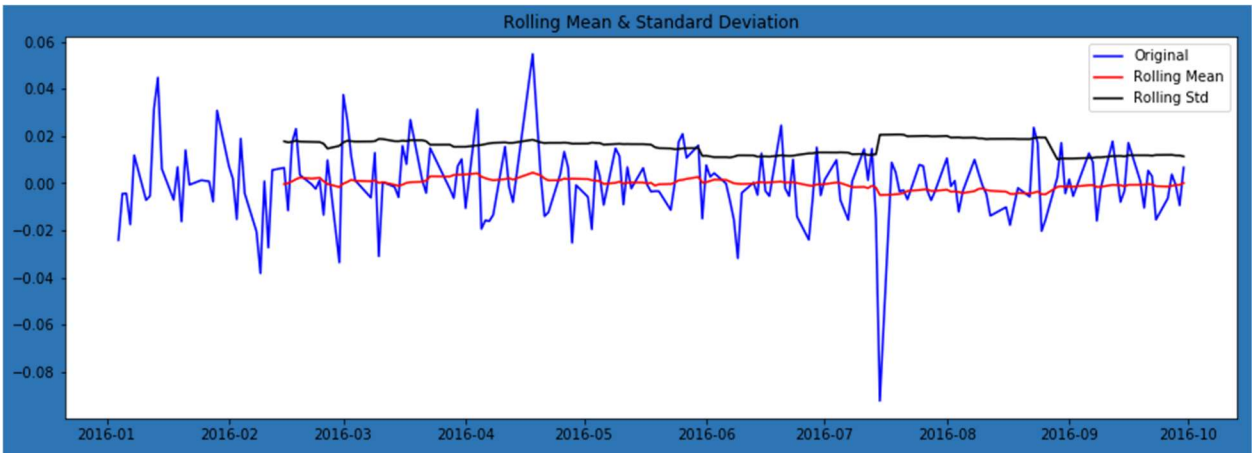

Fig 6. Stationarity of the Log Transformed Daily Difference INFOSYS Price

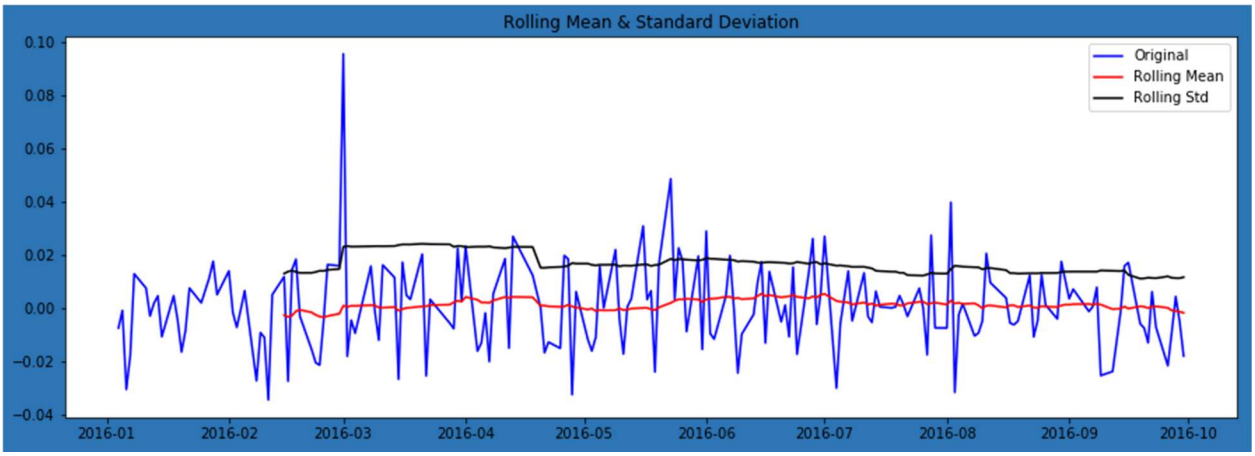

Fig 7. Stationarity of the Log Transformed Daily Difference ITC Price

The ACF and PACF plots for the time log transformed daily time series of the three stocks also suggests that no significant autocorrelation or partial autocorrelation exists for any lags. *Hence it is safe to predict the change of stock prices for any lag (T+1, T+2 etc).*

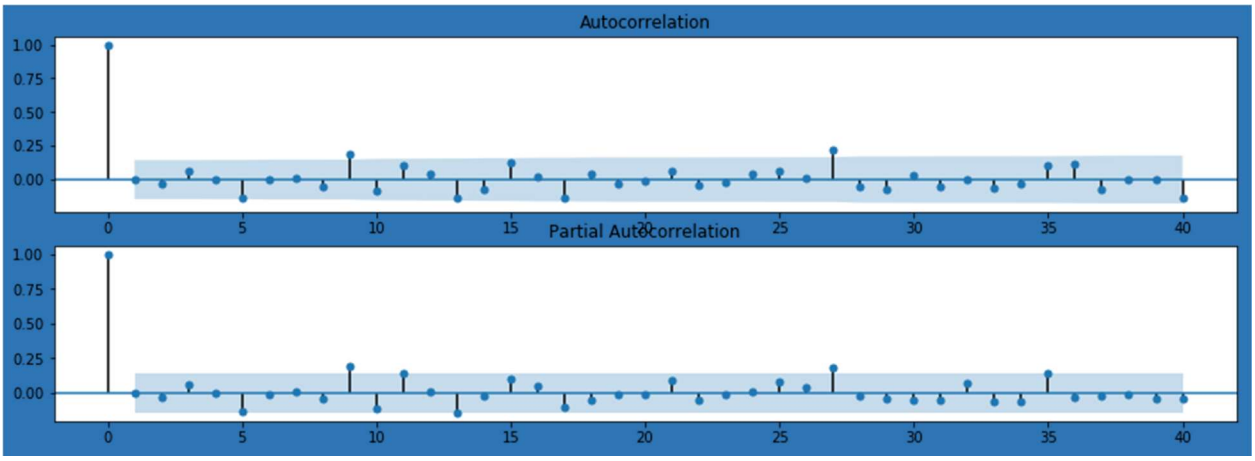

Fig 8. ACF and PACF plot of the Log Transformed Daily Difference HDFC Price

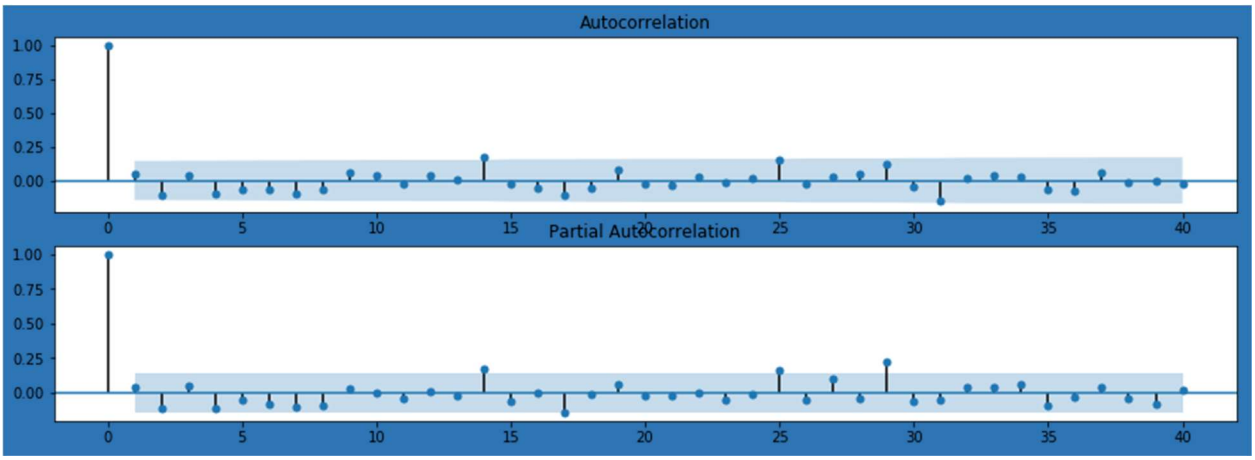

Fig 9. ACF and PACF plot of the Log Transformed Daily Difference INFOSYS Price

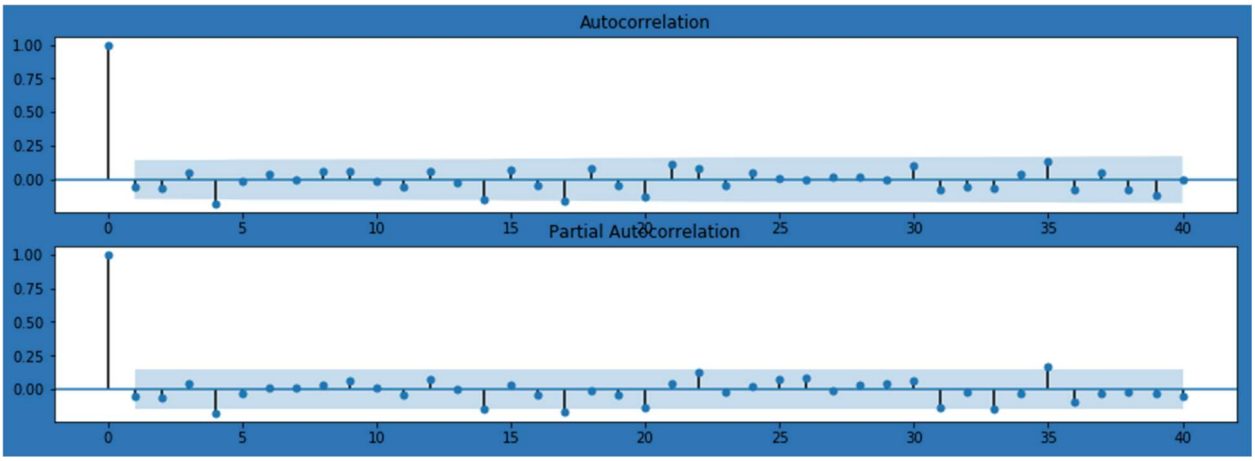

Fig 10. ACF and PACF plot of the Log Transformed Daily Difference ITC Price

Once we are sure about the stationarity of the data, we try to predict stock prices using a naïve ARIMA (2, 1, 1) model. We have only used data for 2016 to train and test the model. The out of sample results are from October 2016 to Dec 2016, while the model is trained from January 2016 to September 2016. *ARIMA model is a linear equation dependent on number of auto regressive terms, number of terms in moving average and the order of differencing. As we will see in our results below, ARIMA is used to create a very naïve model which close to a linear equation.*

The tools for time series analysis are available in the *stasmodels* package of Python.

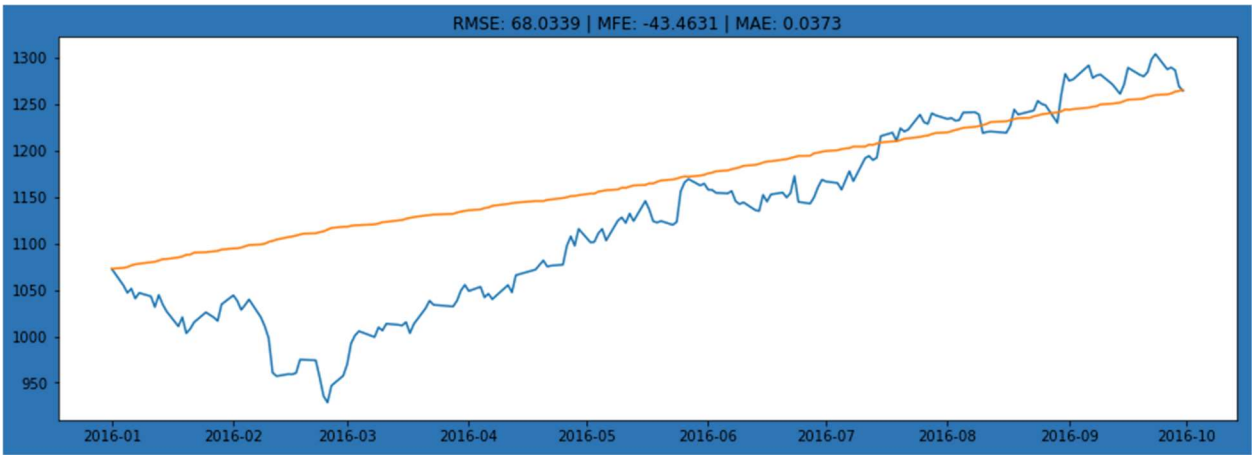

Fig 10. ARIMA Model on HDFC Training Data

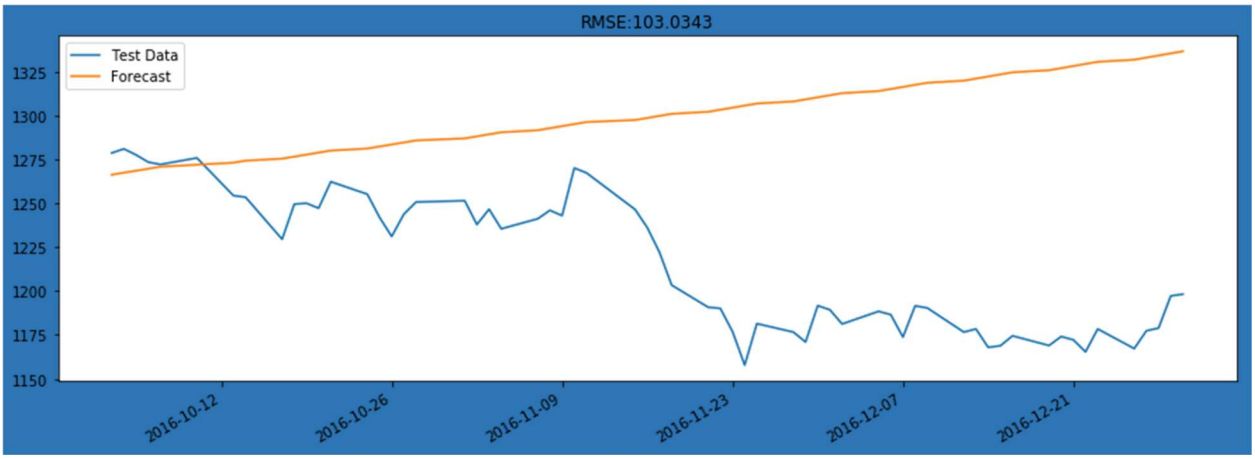

Fig 11. Predictions based on ARIMA model for HDFC

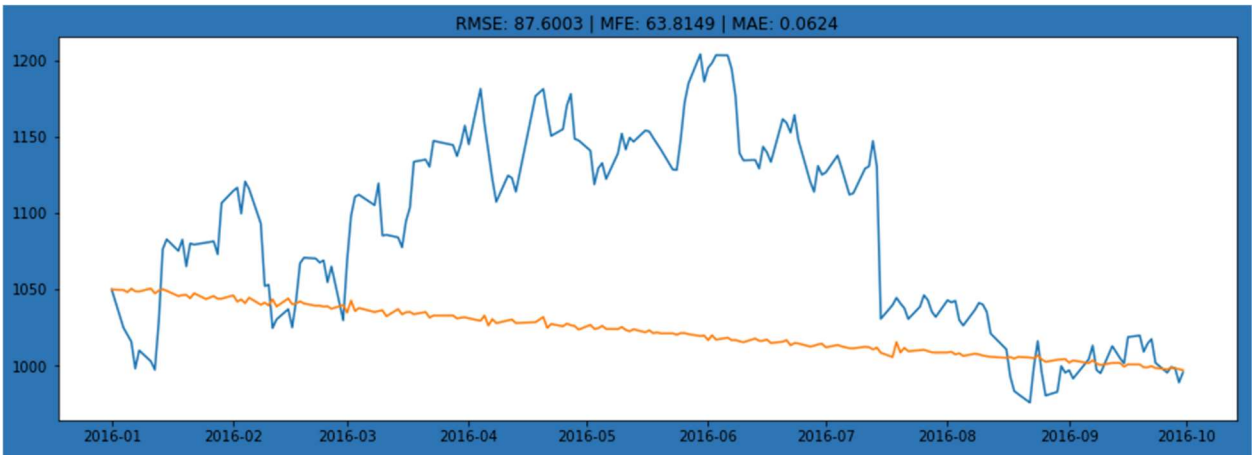

Fig 12. ARIMA Model on INFOSYS Training Data

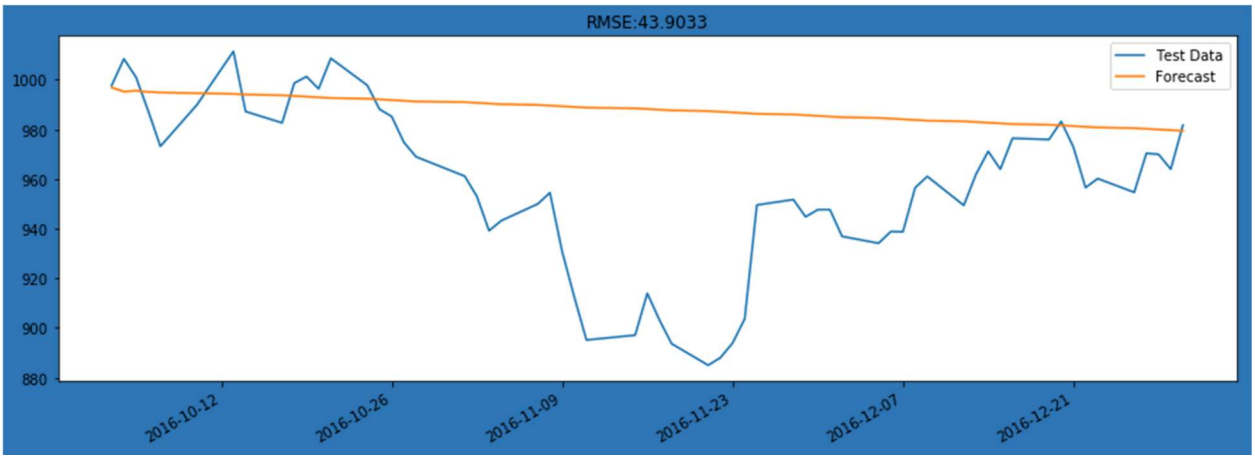

Fig 13. Predictions based on ARIMA model for INFOSYS

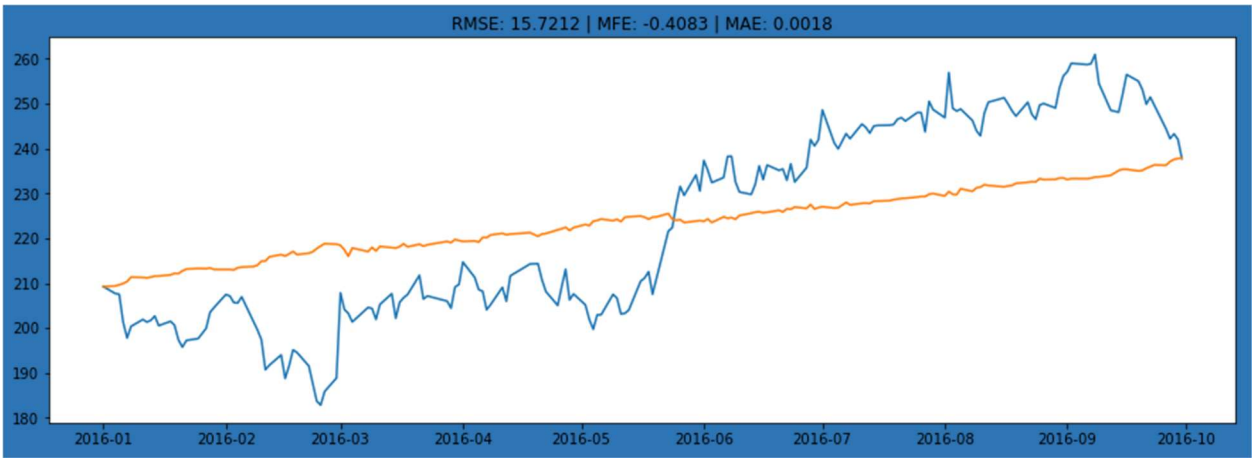

Fig 14. ARIMA Model on ITC Training Data

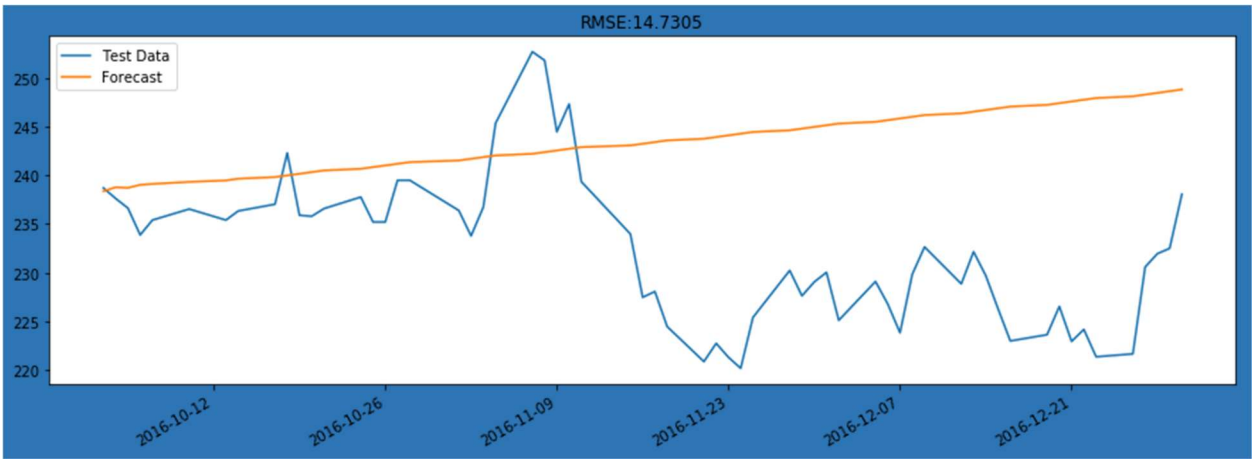

Fig 15. Predictions based on ARIMA model for ITC

The ARIMA modelling approach shows that:

- It is important to predict the change in stock price and not the level itself
- Time Series Forecasting can be used to create Naïve models which can be used to predict recent stock change but not fit for creating a sophisticated trading strategy

Keeping the above points in mind, we move from time series modelling to a supervised machine learning. We have used two techniques to create a model which predict the stock change for the next day.

- *Logistic Regression* using *sklearn.linear_model* library
- *Neural Network* using *Keras* library

Logistic Regression – It is similar to Linear Regression as it is a generalized linear model. The aim of the logistic regression is to estimate the values of the model coefficients and fit the training data and minimize the squared error term and predict y. The dependent variable

in our model is a binary variable 0 or 1, signifying whether the market has gone Up or Down. The independent features are the technical factors.

The linear model can be expressed as:

Y = w0 + w1*feature1 + ……wn*featuren

The logistic regression computes the weighted sum of the input variables after computing w0, w1….wn. It runs the result through a special non-linear function called the sigmoid function to produce the output y. Here the output is in the binary form.

Neural Network – To understand the working of the neural network for our model consider a simple example where our features are the input parameters, there are two hidden layers and the output consists of prediction of the stock price. The final activation function of the output layer is the sigmoid function. The neural network is trained by initializing the weights of the input features as zeros. Repeating the steps of forward propagation, error measurement and backward propagation on the entire data set we eventually find a solution to our cost function which is a measure of how far the predicted value is from the actual value.

Benchmark

Since we are solving a Classification problem *Accuracy* of the Logistic Regression can serve as the benchmark for our model. However, as mentioned earlier we cannot compare the Long Short strategy on the basis of *Accuracy* alone. The correct measure of performance will be the *Excess Return* of the Long Short Strategy over the Long only strategy. By looking at excess return we can ascertain that our model is adding value and we are definitely performing better than simple strategy of going Long the stock.

The *Excess Return* of our strategy using Logistic Regression will be compared against Neural Network. The comparison will also be cross validated against different sample sizes to ascertain which model performs well in general.

Methodology

As discussed earlier our independent variables are based on technical indicators while our dependent variable is based on our closing price. Before training the model we preprocess our dataset to make the mean of all independent variables equal to 0 and variance to 1. This ensures that there is no bias while training the model.

If next day closing price is greater than today, our dependent variable is 1 while if it is lower, our dependent variable is 0.

- Logistic Regression – It estimates the value of the model coefficients using the Sigmoid function. If the value of the output is greater than 0.5, we classify the outcome as 1 (TRUE) or else 0 (FALSE).

- Neural Network – We use *Sequential* method from the *Keras* library. We add layers to the neural network with Sigmoid function as the activation function of the output layer. If the value of the prediction is greater than 0.5, we classify the outcome as 1 (TRUE) or else 0 (FALSE)

The output of the test dataset gives us the prediction to go long (1) the stock or go short the stock (0). Using the daily returns we can compute the cumulative returns of the Long-Short Strategy vs the Long only strategy. The daily returns also help us compute the annualized return and volatility for the strategies. The results are compared for test sizes of 1%, 5% and 20% to confirm if the strategy is successful over short term or long term. Since it is a time series data, test data starts from the end of the training data.

Another important feature of your methodology is tuning of the hyperparameters of the model. For logistic regression the hyperparameters used were:

- **Penalty** : 'L1' and 'L2' and **C** : 0.1,0.5,1.0,2.0

For every Logistic regression the best model was chosen on the basis of the hyperparameters selected. The optimized model had a better accuracy than the original model selected without any tuning of the hyperparameters. The neural networks took a lot of time to train the model and hence I have not tuned the hyperparameters. However the two models have been compared for different test sizes to compare the results using timeseries cross validation. K-Fold Cross Validation will not be correct choice here because it ignores the temporal component inherent in the data.

I would like to highlight some issues which I faced while going through the data and the application of statistical and machine learning model on the data.

- The raw data available from Yahoo has prices missing for some days. We need to drop those days before computing returns. The Pandas library has a useful function of dropna which helps to get rid of missing data.
- ARIMA model will fail to find parameters if the time series is very long. We need to play with multiple combinations of parameters to get the ARIMA model if we are modelling a very long time series. Since the ARIMA discussion was primarily aimed to create a naïve model we chose a smaller time period on only one year to create our model.
- The Neural Networks takes time to converge to find an optimal solution, so ideally the code should be deployed on a powerful machine or AWS. We will be able to introduce more hidden layers and increase the complexity of our model.
- While computing the returns and volatility of our model it is very important to check for any missing data before computing the metrics.

RESULTS

The results of our benchmark model (Logistic Regression) is compared vs Neural Network. Both Logistic Regression and Neural Network is run with different random states to verify the robustness of the model

Benchmark Performance

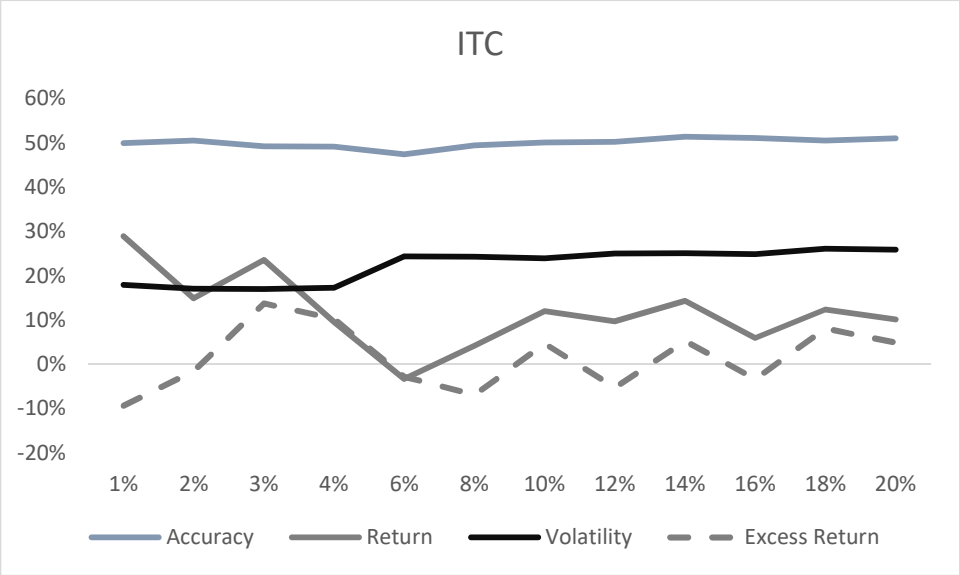

Fig 16. ITC Performance (Logistic Regression)

The Cross Validation results of our benchmark model using Logistic Regression shows that our strategy does give positive excess returns for certain sample sizes (3%, 4%, 10%, 18% and 20%). It is the 3% sample size which performs best with an excess return of 14%.

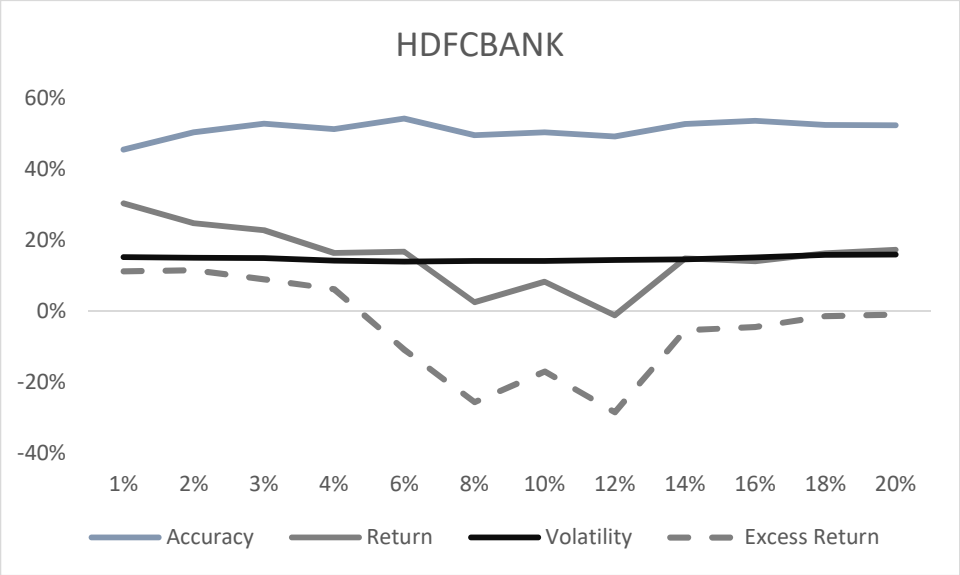

Fig 17. HDFC Performance (Logistic Regression)

Similarly while looking at the results for HDFC we can observe that the model in general performs well with excess returns for smaller sample sizes (1%, 2%, 3% and 4%).

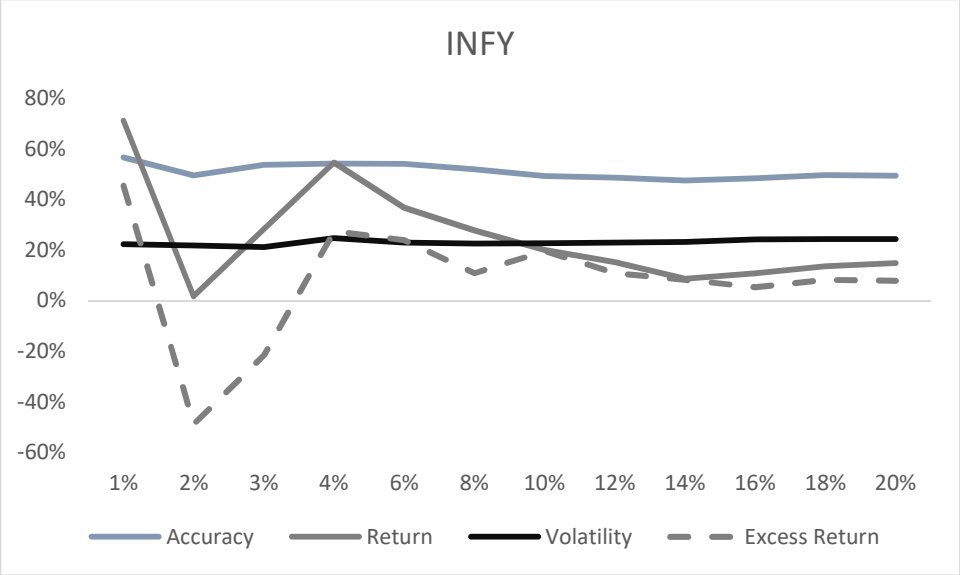

Fig 18. INFOSYS Performance (Logistic Regression)

Finally for INFOSYS, we can see that the model has performed well in general across different sample sizes except for 2% and 3% sizes.

The results of the Logistic Regression are consistent across different random states. Although the three stocks compared give us mixed signals, it is obsereved that in general lower sample sizes tend to give a better model.

Neural Network Model

The neural network model result is compared vs our benchmark model. It gives inconsistent results when run with different random states. In general the model is not able to perform better than our benchmark model discussed above.

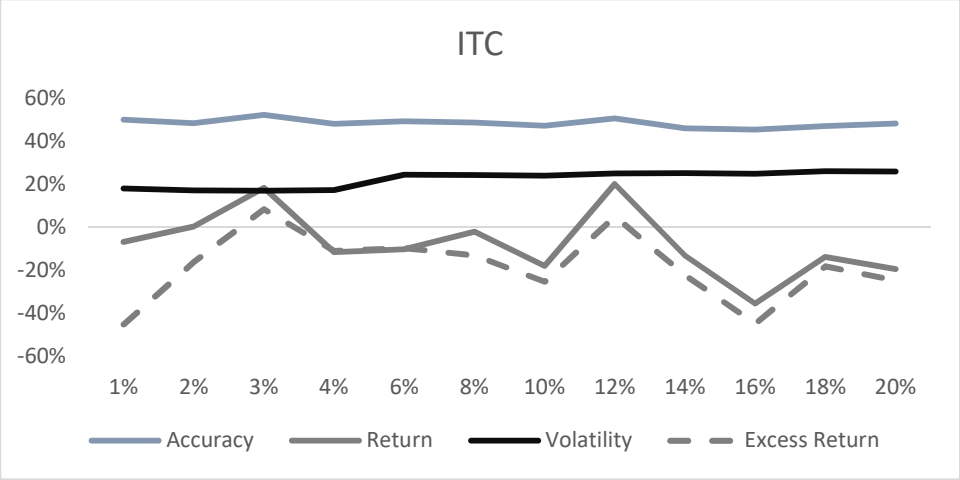

Fig 18. ITC Performance (Neural Network)

The neural network model for ITC is not able to generate excess return in general across different sample sizes.

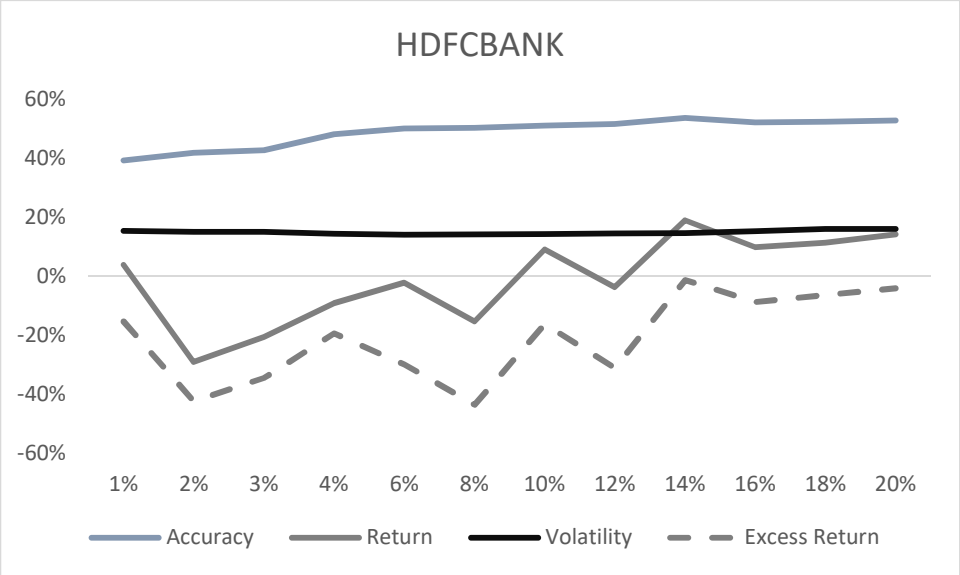

Fig 19. HDFCBANK Performance (Neural Network)

The Neural Network model for HDFC performs even worse consistently generating negative excess returns across all the sample sizes.

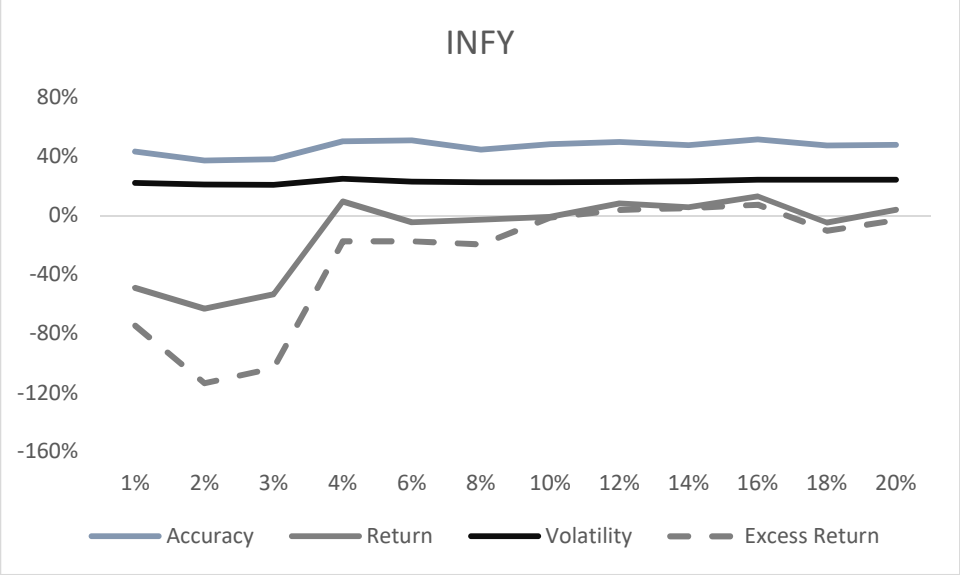

Fig 20. INFOSYS Performance (Neural Network)

Finally the neural network model for INFOSYS is also not able to generate excess return in general across different sample sizes.

Important points to note when comparing the results of our models are:

- In terms of hyperparameters for logistic regression L1 penalty measure and a smaller C value of 0.5 gives better accuracy.
- The data is trained over a long period (2000-2014) and has incorporated many outliers or extreme movements (Crisis period of 2000, 2008 and 2012) and hence it generalize well to unseen data or extreme outliers.
- When we compare our results across different random states, Logistic Regression gives consistent results compared to Neural Networks.
- Accuracy in general remains constant over different sample sizes.
- Volatility of our strategies also remain constant over different sample sizes.
- Each stock has its own idiosyncratic behavior and we need to calibrate our model individually for each stock
- The timing (entry and exit) of strategies is very important

Since the Neural Network models have given inconsistent results we chose our initial Logistic Regression model as our final model. To be more specific the best models using Logistic regression are:

- ITC (Test Size = 3%)
- HDFC (Test Size = 2%)
- INFOSYS (Test Size = 1%)

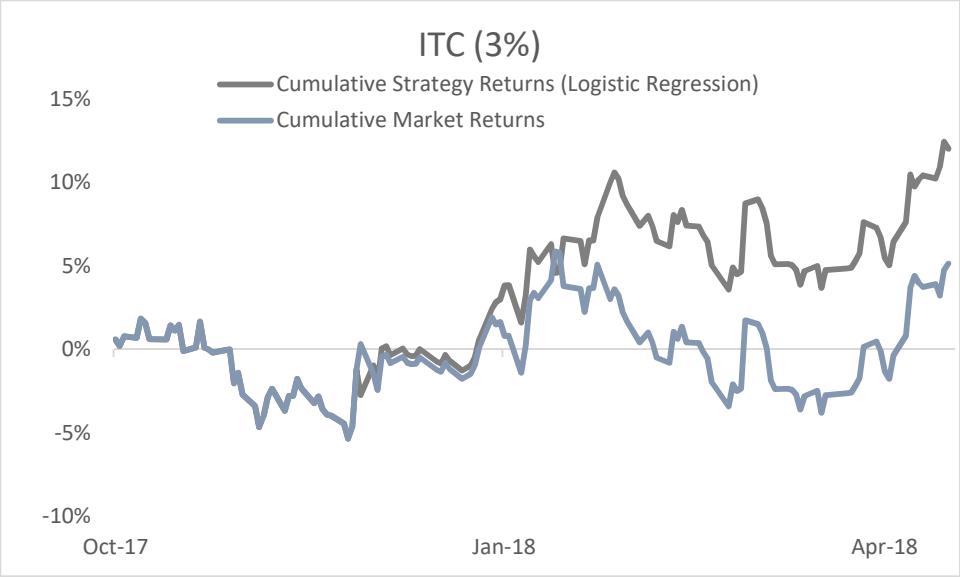

Fig 21. ITC Performance (Logistic Regression, Test Size = 3%)

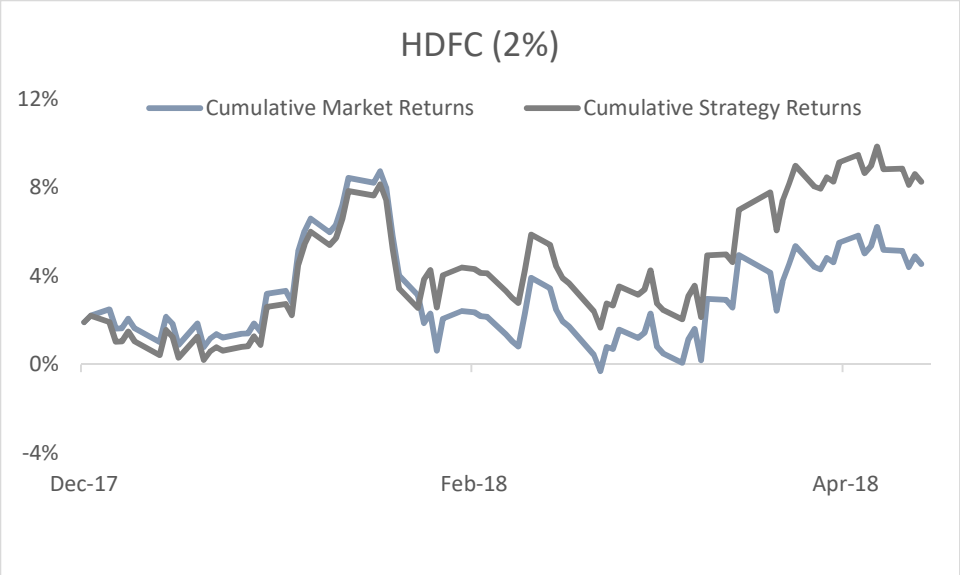

Fig 22. HDFC Performance (Logistic Regression, Test Size = 2%)

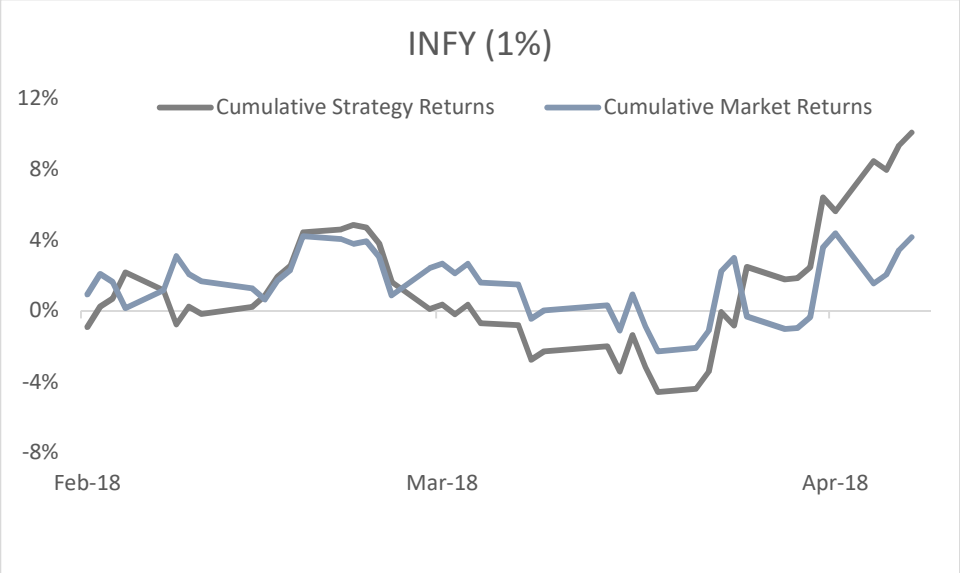

Fig 23. INFOSYS Performance (Logistic Regression, Test Size = 1%)

From our results we can conclude that:

- Logistic Regression can help us to predict stock price movements. In general the model works well for shorter time frame but the predicting power fades over longer time horizon.
- Neural Network models are not able to perform as well as the Logistic Regression. In general they give inconsistent results and needs to be carefully evaluated over

different hyperparameters to find a stable solution. Introduction of deeper layers and use of different optimizers may help us to fine tune our model.

CONCLUSION

The prediction of stock prices is a very difficult problem to solve due to inherent data issues. The time series data needs to be stationary with no autocorrelation to be used in a machine learning or time series model. We first started with simple time series modelling of stock prices to understand the nature of data. The time series modelling helped us understand that the stock price itself is not stationary but if we take the daily difference and of stock prices it can introduce some stationarity in the data. We build a naïve ARIMA model on the basis of daily difference in the price of the data to predict short term price change. The resulting ARIMA model was very simplistic and represented as a linear combination of parameters of ARIMA model. We next moved to build a supervised learning model and generated a Long Short trading strategy on the basis of the prediction of stock price movement. The results from supervised learning makes us understand that every stock has got idiosyncratic behavior and there cannot be a generalized supervised learning model to predict stock prices. To have a more consistent results across all sample sizes we need to model more fundamental factors apart from the technical ones. Here sentiment based factors can help us to capture some of the stock behavior. The stock under goes many regime changes with mean and variance changing over time. We can also introduce a new factor which captures the given regime of the stock and expectation maximization algorithms can be used to capture the regimes.

The daily Long-Short strategy can be expensive and we have not deducted the cost of daily rebalancing from the strategy returns. An ideal strategy should move from daily signals to a systematic monthly or fortnightly strategy which is more based on fundamental factors than technical ones.

Machine learning is an exciting way to predict stock prices. We need to evaluate our model on many stocks from different sectors with introduction of more fundamental behavior to create a more generalized model.