Stock Price Prediction Using Machine Learning

Report Submitted for

ELECTRICAL & ELECTRONICS ENGINEERING SOCIETY
SUMMER MENTORSHIP PROGRAMME

BY

KEVIN MATHEW T (BE/10085/17)

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING BIRLA INSTITUTE OF TECHNOLOGY MESRA, RANCHI – 835215 (INDIA)

AKNOWLEDGEMENT

This work would not have been possible without the persistent support from my mentor, **Rudra Narayan Pandey**, for his constant support and who has taught me more that I can thank him for. I had the pleasure to work with him, and he taught me a great deal about researching in this field.

I am also indebted to **Mr. Subrat Kumar Swain**, Faculty Advisor, Electrical & Electronics Engineering Society (EEESOC), Birla Institute of Technology, for giving me this platform to learn through **Summer Mentorship** Programme organized by **EEESOC, BIT Mesra.**

I am also thankful to **Dr. P.R. Thakura**, Professor and Head, Department of Electrical and Electronics Engineering, Birla Institute of Technology for providing facilities and opportunities for participating in this programme.

I am also thankful to **Prof. (Dr.) M. K. Mishra**, Vice Chancellor, Birla Institute of Technology, Mesra, Ranchi for providing important facilities, required for carrying out this project.

ABSTRACT

Predicting the direction of stock prices is a widely studied subject in many fields including trading, finance, statistics and computer science. Investors in the stock market can maximize their profit by buying or selling their investment if they can determine when to enter and exit a position. Professional traders typically use fundamental and/or technical analysis to analyse stocks in making investment decisions. Fundamental analysis involves a study of company fundamentals such as revenues and profits, market position, growth rates, etc. Technical analysis, on the other hand, is based on the study of historical price fluctuations. Due to the nature of market forces, economies tend to follow a pattern of expansion and contraction, over long periods of time. The stocks trade within an overarching environment where economy moves from one phase of the business cycle to the next.

INTRODUCTION

Stock markets are hard to predict. Driven by supply and demand, macroeconomic changes such as inflation or political instability can affect whole markets, while local events like company financial announcements or product releases impact individual stocks. Traders also look for signals in the price development that may indicate future prices. Because some trading is based on these indicators, price movement by itself can also trigger trading activity, causing further price adjustments. Investors and traders should ideally consider all these factors when evaluating a stock, however market participants are often partitioned into two groups: Traders that make decisions based on news, facts and numbers are known as fundamental analysts, while those who look for signals in price history are using technical analysis.

Recurrent neural networks (RNN) have proved one of the most powerful models for processing sequential data.

Long Short-Term memory is one of the most successful RNNs architectures. LSTM introduces the memory cell, a unit of computation that replaces traditional artificial neurons in the hidden layer of the network. With these memory cells, networks are able to effectively associate

memories and input remote in time, hence suit to grasp the structure of data dynamically over time with high prediction capacity.

Some related research

Gencay (1996) and Gencay & Stengos (1998) examine how simple feedforward networks performs compared to an ARMA (1,1)-GARCH (1,1) model. They use moving-average indicators as inputs into the networks. Furthermore, they claim that the networks perform better than the ARMA (1,1)-GARCH (1,1) model when adopting buy/sell strategies. Both articles use daily Dow Jones Industrial Average Index data between 1963 and 1988.

Enke (2005) uses a feedforward neural network model together with a data mining7 framework in order to investigate predicting power of the input variables. He shows that given the right inputs a neural network model outperforms a buy and hold strategy. Qiu, Song and Akagi (2016) investigates the same problem but also how the result differs with different optimization methods. However, most research uses neural nets of feedforward architecture Maknickiene et al. (2011) is one example where they use recurrent networks for financial analysis. Rutkauskas (2010) is another example where LSTM networks are used to find a model for asset return prediction.

Because accurately predicting stock market returns is challenging a simpler binary classification method is often used. This is called direction of change forecasting, and tries to predict the direction of the market rather than the direction and magnitude. An interesting paper is Christoffersen et al. (2006) where they investigate the link between sign forecasting and the conditional variance, mean, skewness and kurtosis. More often, than investigating returns, are papers investigating the volatility of market returns which is easier to predict. For example, see Mantri, Gahan & Nayak 2010 and Hu & Tsoukalas 1999.

There is little research done on the forecasting of stock market returns and direction of change using long short-term memory (LSTM) recurrent neural networks. Although, LSTM's have been around since 1997 they have recently become popular because of the new AI wave, and the availability of programming interfaces that can handle them. In this thesis two different LSTM models, with two different activation functions, yielding four different models are used on three different datasets which exhibits different market characteristics. Thus, the experiment will yield interdisciplinary results regarding classic financial empirics, such as application of the

efficient market hypothesis (section 2.2) and advanced modern machine learning.

Background

2. Stock Markets

2.1 Stocks and Stock Market

Oxford Dictionaries: Definition of Market; defines a market as an area or arena in which commercial dealings are conducted. Financial markets can be described as aggregations of buyers and sellers involved in trading a financial instrument like stocks, bonds, commodities or currencies. A stock market is a financial market where company shares are traded. While the stock market is an abstract term, the actual trades may be executed over the counter, at a stock exchange, an electronic communication network or similar. Trades that are executed through stock exchanges are easily tracked, because of the centralized and transparent nature of the exchange. This makes stock markets more approachable for studying, unlike for instance foreign exchange markets where data has to be aggregated from multiple decentralized sources.

A stock exchange is a common hub for buyers and sellers to find each other and fulfil trades. Oxford Dictionaries: Definition of Stock Exchange; defines it as a market where securities are bought and sold, although note that the word market in this setting refers to a physical place,

unlike stock markets. Activity at the exchange is visible to other participants, which in turn will drive the price. It is the principle of supply and demand, put into action. A stock is regarded as liquid, if it sees significant amounts of trading activity. The more activity, the easier it is to find a buyer when someone tries to sell, and vice versa. For less liquid stocks some participants might struggle to complete trades. If the participant compromises on a less favourable price, chances of fulfilling the trade increase. When settling at another price point than intended, the price difference is known as slippage. By only considering liquid stocks, trades are usually filled almost instantly, provided normal market conditions. Because of the centralized, transparent nature of stock markets, gathering historical data from them is both easily and widely done. Some traders and analysts use these data in order to model future price movements. This is known as technical analysis, because it relies exclusively on the price development.1 Numerous stock exchanges exist today, each of them listing a different selection of stocks. Two well-known American exchanges are the New York Stock Exchange (NYSE) and the National Association of Securities Dealers Automated Quotations (NASDAQ). Except for both being based in New York, there are various differences related to the kind of companies they list, and how they execute trades. The NASDAQ is known to list companies related to technology, and runs algorithms to automate the trading process. The NYSE

on the other hand, runs a more traditional auction-based trading. The differences between the NASDAQ and the NYSE are discussed further in Investopedia.

2.1.1 Stock Market Data

Stock market data are available in various forms, ranging from fine-grained information concerning each trade, to one data point every month. Exactly what the data points contain may vary, but a widely used composition consists of six variables: Time, open, high, low, close and volume. Time is a reference to when the data point is from. Open and close is the share price at the beginning and end of the period, while high and low refers to the highest and lowest point the value reached throughout the period. Lastly, volume is the number of shares that were traded in total. Deciding the time span of each data point plays an important role; long periods prevent making short-term forecasts, while short periods are noisier due to their detailed nature. For clarity, this report will sometimes refer to data points as stock prices, even though more information is actually contained in each point.

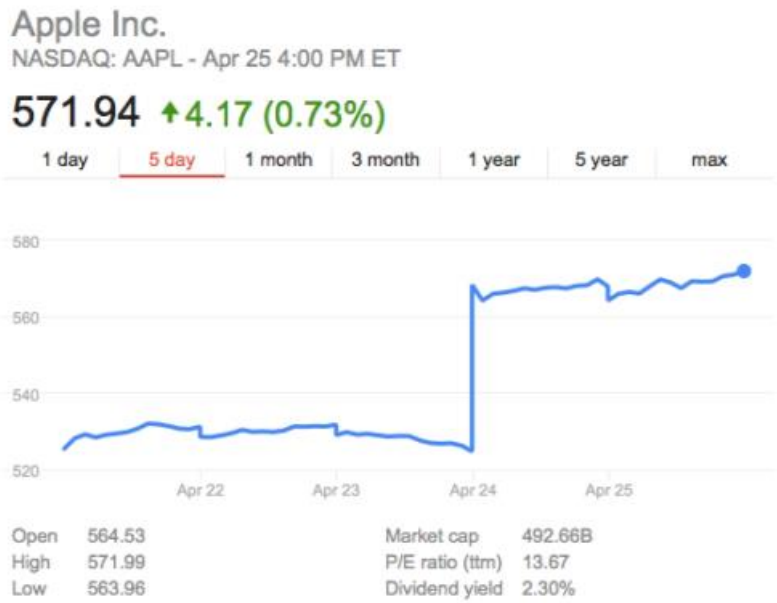


Fig 2.1: After-hours trades cause a jump discontinuity.

Furthermore, although it is possible to plot all the variables except volume in what is known as a candlestick chart, I will plot only closing prices, using line charts. This is for the sake of brevity and clarity, but be aware that there are more variables in the actual data.

2.2 Long Short-Term Memory (LSTM)

LSTM [Patterson, 2017] is a kind of Recurrent Neural Network (RNN) with the capability of remembering the values from earlier stages for the purpose of future use. Before delving into LSTM, it is necessary to have a glimpse of what a neural network looks like.

2.2.1 Artificial Neural Network (ANN)

A neural network consists of at least three layers namely:

1) an input layer,

2) hidden layers, and,

3) an output layer.

The number of features of the dataset determines the dimensionality or the number of nodes in the input layer. These 7 nodes are connected through links called “synapses” to the nodes created in the hidden layer(s). The synapses links carry some weights for every node in the input layer. The weights basically play the role of a decision maker to decide which signal, or input, may pass through and which may not. The weights also show the strength or extent to the hidden layer. A neural network basically learns by adjusting the weight for each synopsis. In the hidden layers, the nodes apply an activation function (e.g., sigmoid or tangent hyperbolic (hyperbolic tangent)) on the weighted sum of inputs to transform the inputs to the outputs, or predicted values. The output layer generates a vector of probabilities for the various outputs and selects the one with minimum error rate or cost, i.e., minimizing the differences between expected and predicted values, also known as the cost, using a function called SoftMax. The assignments to the weights vector and thus the errors obtained through the network training for the first time

might not be the best. To find the most optimal values for errors, the errors are “back propagated” into the network from the output layer towards the hidden layers and as a result the weights are adjusted. The procedure is repeated, i.e., epochs, several times with the same observations and the weights are re-adjusted until there is an improvement in the predicted values and subsequently in the cost. When the cost function is minimized, the model is trained.

2.2.2 Recurrent Neural Network (RNN)

A recurrent neural network (RNN) is a special case of neural network where the objective is to predict the next step in the sequence of observations with respect to the previous steps observed in the sequence. In fact, the idea behind RNNs is to make use of sequential observations and learn from the earlier stages to forecast future trends. As a result, the earlier stages data need to be remembered when guessing the next steps. In RNNs, the hidden layers act as internal storage for storing the information captured in earlier stages of reading sequential data. RNNs are called “recurrent” because they perform the same task for every element of the sequence, with the characteristic of utilizing information captured earlier to predict future unseen sequential data. The major challenge with a typical generic RNN is that these

networks remember only a few earlier steps in the sequence and thus are not suitable to remembering longer sequences of data. This challenging problem is solved using the “memory line” introduced in the Long Short-Term Memory (LSTM) recurrent network.

2.2.3 Long Short-Term Memory (LSTM)

LSTM is a special kind of RNNs with additional features to memorize the sequence of data. The memorization of the earlier trend of the data is possible through some gates along with a memory line incorporated in a typical LSTM. The internal structure of an LSTM cell is demonstrated in Diagram 1: 8 Diagram 1. The internal structure of an LSTM [Colah’s Blog, 2015]. LSTM is a special kind of RNNs with additional features to memorize the sequence of data. Each LSTM is a set of cells, or system modules, where the data streams are captured and stored. The cells resemble a transport line (the upper line in each cell) that connects out of one module to another one conveying data from past and gathering them for the present one. Due to the use of some gates in each cell, data in each cell can be disposed, filtered, or added for the next cells. Hence, the gates, which are based on sigmoidal neural network layer, enable the cells to optionally let data pass through or disposed.

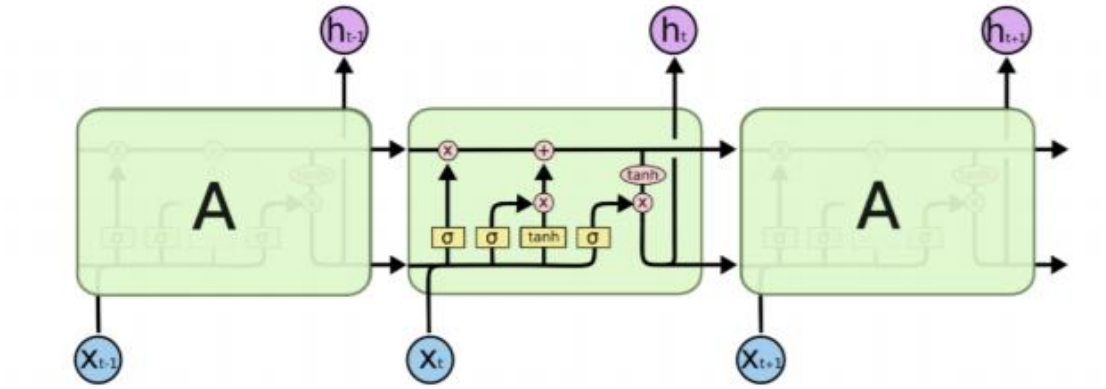


Fig 2.2: The internal structure of an LSTM

Each sigmoid layer yields numbers in the range of zero and one, depicting the amount of every segment of data ought to be let through in each cell. More precisely, an estimation of zero value implies that "let nothing pass through"; whereas; an estimation of one indicates that "let everything pass through."

Three types of gates are involved in each LSTM with the goal of controlling the state of each cell:

- **Forget Gate** outputs a number between 0 and 1, where 1 means "completely keep this"; whereas, 0 implies "completely ignore this."

-**Memory Gate** chooses which new data need to be stored in the cell. First, a sigmoid layer, called the "input door layer" chooses which values will be modified. Next, a hyperbolic tangent layer makes a vector of new candidate values that could be added to the state.

- **Output Gate** decides what will be yield out of each cell. The yielded value will be based on the cell state along with the filtered and newly added data.

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Python Implementation

For this project, we have considered Recurrent Neural Network and Long Short-Term Memory. Our system consists of several stages which are as follows: -

• **Stage 1:** Raw Data: In this stage, the historical stock d ata is collected using the Quandl API available on Pytho n and this historical data is used for the prediction of fut ure stock prices. Using the API, we collected data which consisted of Open, High, Low, Close, Volume Traded, E x-Dividend, Split-Ratio, Adjusted Open, Adjusted High, Adjusted Low, Adjusted Close and Adjusted Volume of the Apple Inc. stock from **2000-01-03** to **2018-03-27**, w hich is equivalent to 4585 days of data for the Apple Inc stock price.

• **Stage 2:** Data Pre-processing: The pre-processing stag e involves
a) Data transformation: The data was scaled using Min-Max scaler available in the Scikit-Learn API in Python.

Transforms features by scaling each feature to a given range.

This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e. between zero and one.

The transformation is given by:

```
X_std =
(X - X.min(axis=0)) / (X.max(axis=0) - X.
min(axis=0))
X_scaled = X_std * (max - min) + min
```

where min, max = feature_range.

This transformation is often used as an alternative to zero mean, unit variance scaling.

b) Data Splitting: The data was slip into two parts – one for Training and the other for Testing. The training part consists of ~80% of the data (OHLC average for 3667 days for the apple stock), and the testing part consists of the later ~20% of the data (OHLC average for 918 days for the Apple stock).

- **Stage 3:** Feature Extraction: In this layer, only the features which are to be fed to the neural network are chosen. We will choose the feature from Date, open, high, low, close, and volume. Then we converted these features to form a single feature known as OHLC – an average of the Opening, High, Low and Closing prices for each day.

• **Stage 4:** Training Neural Network: In this stage, the data is fed to the neural network and trained for prediction assigning random biases and weights. Our LSTM model is composed of a sequential input layer followed by 2 LSTM layers and dense layer and then finally a dense output layer with linear activation function.

The code of the Neural Network implemented in Keras is as follows:

```
from keras.layers import Dense
from keras.layers import LSTM
from keras.models import Sequential

regressor = Sequential()
regressor.add(LSTM(units=16,
     input_shape= (None, 1),
     return_sequences=True))
regressor.add(LSTM(units=32, activation='relu'))
regressor.add(Dense(units=1))
regressor.compile(optimizer='adam',
     loss='mean_squared_error')
regressor.fit(X_train, y_train, batch_size=32,
     epochs=100)
```

Analysis: For analysing the efficiency of the system we are used the Root Mean Square Error (RMSE). The error or the difference between the target and the obtained output value is minimized by using RMSE value. RMSE is the square root of the mean/average of the square of all of the error. The use of RMSE is highly common and it makes an excellent general-purpose error metric for numerical predictions.

$$RMSE = \sqrt{\frac{1}{N}\sum\left(\hat{Y}_i - Y_i\right)^2}$$

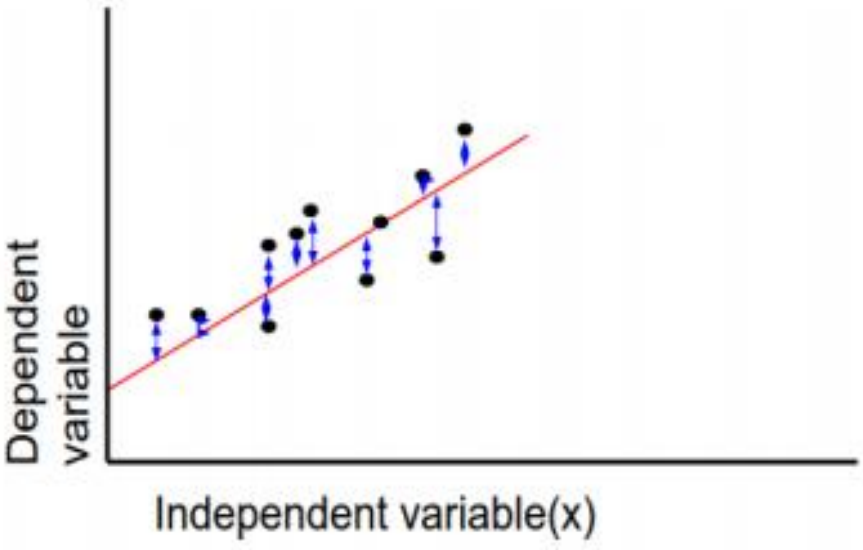


Fig 3.1: RMSE Value Calculation

Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors.

Results

R^2 Score

Wikipedia defines R^2 Score as following; In statistics, the **coefficient of determination**, denoted R^2 or r^2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s).

It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

There are several definitions of R^2 that are only sometimes equivalent. One class of such cases includes that of simple linear regression where r^2 is used instead of R^2. When an intercept is included, then r^2 is simply the square of the sample correlation coefficient (i.e., r) between the observed outcomes and the observed predictor values.[4] If additional regressors are included, R^2 is the square of the coefficient of multiple correlation. In both such cases, the coefficient of determination ranges from 0 to 1.

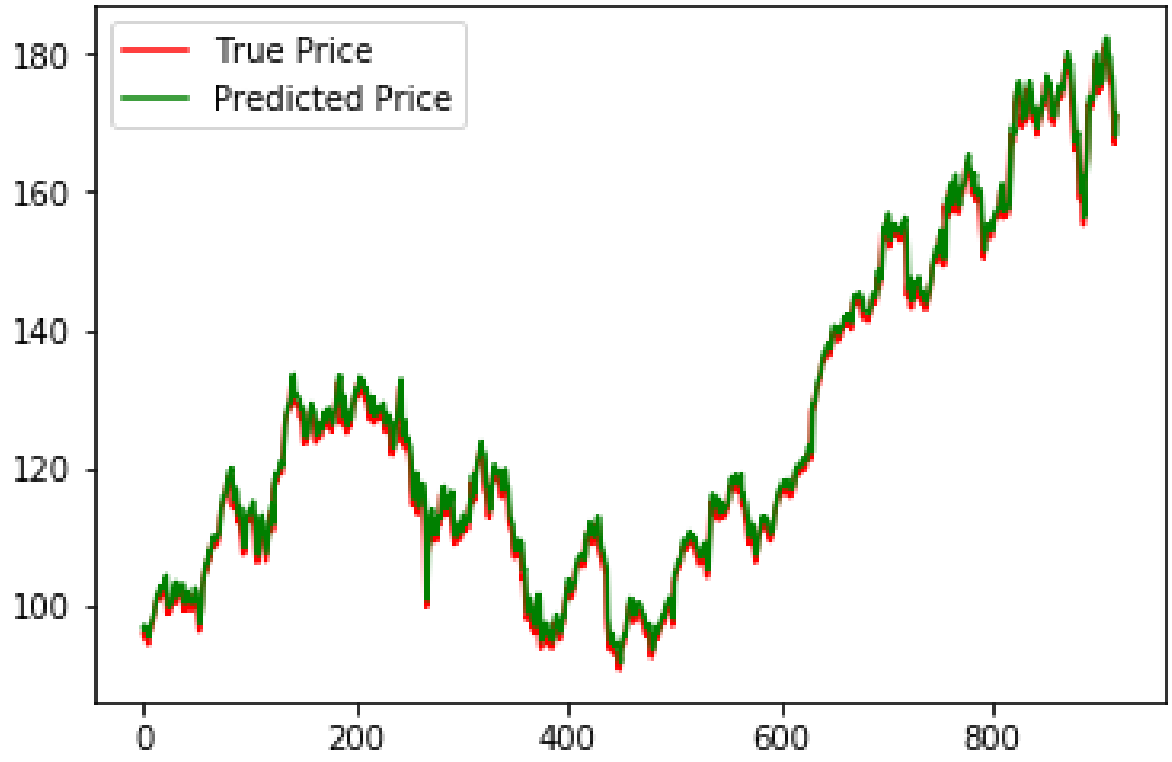
R^2 Score can be calculated as follows:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{\text{samples}}-1}(y_i - \bar{y})^2}$$

where

$$\bar{y} = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} y_i$$

After prediction the model gives an R^2 Score of 0.99534



Prediction for the training data by the model and the real stock prices plotted using matplotlib library in Python.