Stock Prediction using Neural Networks

Project Workbook 1

By

Mahesh Reddy Konatham [013823095]
Mohdi Habibi [010302851]
Shreya Hagalahalli Shrinivas [013821405]
Sneha [013850174]

March 4, 2020

Advisor: Gokay Saldamli

Project github link:

https://github.com/psnehas/Stock-Prediction-using-Neural-Networks

TABLE OF CONTENTS

Chapter 1. Literature Survey, State-of-the-Art

1.1 Literature Survey

Stock market prediction involves dealing with complex, deficient and highly skewed data as stated by Kim et al. [1], and because of the non-linearity problem, it becomes very hard to forecast. Tillakaratne et al. [2] also mention the imbalance of trading signals in the classification of data in to buy, sell and hold classes. Yu et al. [3] explored the conventional and standard machine learning models for predicting stocks like using genetic algorithms for feature extraction. The features extracted from the least squares support vector machine (LSSVM) learning model with optimal parameters and kernel techniques are used to predict the stocks. Zahedi et al. [4], used principal component analysis (PCA) for feature extraction, dimensionality reduction and with the help of artificial neural networks (ANNs) and various other accounting variables to predict new patterns in stock movements. Saad et al. [5] explained the problem of false alarm arising from a lack of short-term memory. The authors [5] also explored TDNN, RNN, and PNN for prediction to minimize risks and losses. Kwon et al. [6] stated that the hybrid algorithm based on a neurogenetic model, in which the data was generated by technical indicators, outperformed the average buy and hold strategy. Yu et al. in the most recent research [7], stated that integrating deep learning and neural networks helped tremendously in solving the nonlinear complications and thus justifying their results in achieving better accuracy for stock prediction.

Paper 1: ***Stock Market Prediction Using Artificial Neural Networks with Optimal Feature Transformation***

The comparison of artificial neural networks with heuristic genetic algorithms for feature extraction is very useful, especially in reducing the dimensional volume and unrelated components for forecasting the stock movement.

Paper 2: ***Modified Neural Network Algorithms for Predicting Trading Signals of Stock Market Indices.***

The problem of an imbalanced data set is addressed in this research paper. The parameters in the neural network are adjusted according to the forward movement pattern and the optimal least square error model.

Paper 3: ***Evolving Least Squares Support Vector Machines for Stock Market Trend Mining***

This research paper thoroughly analyzed the usage of the support vector machine model with the technique of minimizing the least square distance of each data point from the imaginary hyperplane sides.

Paper 4: ***Application of Artificial Neural Network Models and Principal Component Analysis Method in Predicting Stock Prices on the Tehran Stock Exchange***

Exploiting the Artificial neural networks with the help of principal component analysis seems to be very interesting. The optimal solution for finding the correct number of principal components is calculated by experiments on new data patterns.

Paper 5: ***Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks***

The idea of minimizing the risk ratio by reducing the amount of loss using algorithms like time delay, recurrent neural networks is very interesting.
Forecasting which is built on the analysis of daily closing prices of stocks is a good strategy for maximizing profits.

Paper 6: ***A Hybrid Neurogenetic Approach for Stock Forecasting***

This research paper proposes that we can generate an input feature vector by using technical measures from business analysts. After that this feature vector can be optimized by using a Genetic algorithm that can be further used to reduce the neural network weights using two-dimensional encoding.

Paper 7: ***Stock Price Prediction Based on Deep Neural Networks***

This research paper explores the concept of long short-term memory (LSTM) and deep learning to achieve better accuracy than traditional machine learning models. The problem of non-linearity and the solution to tackle that is discussed which can help my research on stock prediction.

1.2 State-of-the-art Summary

Modern-day researchers and analysts remarkably counted upon deep learning and neural network techniques after the notorious performance shown by conventional machine learning algorithms because of the problems arising from non-linearity in the stock data. The short term memory advantage of the Long Short Term Memory (LSTM) model is currently exploited a lot in forecasting the stock price along with the important domain indicators like price history. Artificial, probabilistic and convolution neural networks coupled with technical indicators like trading signals are most widely used for stock prediction presently.

1.3 References

[1] Kim, Kyoung-Jae, and Won Lee. "Stock Market Prediction Using Artificial Neural Networks with Optimal Feature Transformation." *Neural Computing & Applications* 13.3 (2004): 255-60.

[2] Tilakaratne, C., M. Mammadov, and S. Morris. "Modified Neural Network Algorithms for Predicting Trading Signals of Stock Market Indices." *Journal of Applied Mathematics & Decision Sciences* 2009 (2009): 1-22.

[3] L. Yu, H. Chen, S. Wang and K. K. Lai, "Evolving Least Squares Support Vector Machines for Stock Market Trend Mining," in *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 87-102, Feb. 2009.

[4] Zahedi, Javad, and Mohammad Mahdi Rounaghi. "Application of Artificial Neural Network Models and Principal Component Analysis Method in Predicting Stock Prices on

the Tehran Stock Exchange." *Physica A: Statistical Mechanics and Its Applications* 438 (2015): 178-87.

[5] E. W. Saad, D. V. Prokhorov and D. C. Wunsch, "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks," in *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1456-1470, Nov. 1998.

[6] Y. Kwon and B. Moon, "A Hybrid Neurogenetic Approach for Stock Forecasting," in *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 851-864, May 2007.

[5] Yu, Pengfei, and Xuesong Yan. "Stock Price Prediction Based on Deep Neural Networks." *Neural Computing and Applications (2019).*

Chapter 2. Project Justification

Stock market historically has been very volatile with many factors having direct impact on the fluctuating nature of it. Many stock trading techniques involve technical analysis and fundamental analysis of each individual company's stock to understand the future of its value. Not only does a company's health and earnings have weight in deriving its stock value, but also many other external factors such as sentiment of the market and public reaction toward events have direct effect on the volatility.

With the vast amount of data available from the stock market, there has been lots of interest in using Machine Learning models to solve such problems. The idea is to extract knowledge from the previous stock market data to train such models so that they can predict the future price of a certain stock. There are many shortcomings in terms of using Machine Learning models to predict stock such as not taking into consideration the full previous history of a certain stock when prediction is forming. The history context in solving such problems is a crucial component that many Machine Learning models lack in.

Recently, Artificial Intelligence has shown promising results in Image Recognition and Natural Language Processing fields. The new branch of AI which is Deep Neural Networks has been the underlying process for getting better results combining with the powerful processing units out in the market. Recurrent Neural Networks (RNN) are a branch of Deep Learning which can bring the history context into picture. That is why it has been a popular tool to be applied to problems that need previous inputs for better prediction. The vanilla RNN models have a problem with keeping long term memory by design. Long Short-Term Memory is a type of Recurrent Neural Networks that is specialized in carrying over

information for a long time to allow neural networks rely on historical context and not only the previous inputs.

This research paper focuses on the previous movement of the stock market to predict the near future volatility without taking into consideration other external factors. The idea behind this approach is how to derive certain knowledge from stock market time series data using deep learning to understand near future positive or negative volatility. The automation of such a process will allow anyone to make quick decisions for making profit from the stock market volatility. Since LSTMs are specialized in keeping the historical context of inputs, it makes it a viable option for stock market prediction.

Chapter 3. Identify Baseline Approaches

The baseline chosen for this research paper is an implementation of stock prediction using LSTM discussed in the following paper "Stock Market's Price Movement Prediction With LSTM Neural Networks". In this paper the implementation is discussed as such, the approach taken is to predict the stock movement using LSTM with a limited number of dimensions for data. The difference between the start and end of each stock for every 15 minutes were taken into consideration for their implementation.

This research paper will focus on the same implementation but with more dimensions to bring a holistic view of the data into account such as volume and total trades. These two factors can result in decisive improvement toward the results since the volume and total trades within a certain amount of time can be an indication of positivity or negativity fluctuation. There are some fundamental investment strategies that can be used as a baseline also which are frequently used by stock traders such as holding on to a stock in hope of seeing a near future positive fluctuation in order to make profit.

Several other papers used Machine Learning models such as Random Forest and Support Vector Machine which will be used as a baseline for this paper. The idea is how to improve the result using LSTM and bringing the historical context into the view for neural networks to obtain improved results in predicting near future stock prices.

Chapter 4. Dependencies and Deliverables

4.1 Dependencies

1. Collecting good quality dataset:

Stock related datasets are not publicly available. They must be purchased by authentic organizations. The quality of the data is also an important factor for better predictive models.

2. Data preprocessing and feature extraction:

Although data preprocessing is a trivial step in machine learning, it is a very important step because training the model on the wrong set of features might lead to low or wrong results degrading the prediction accuracy.

3. High performance computing devices and long training hours:

The dataset collected contains data points of 1 year having around 40 tickers in each file; each sample recorded with a time difference of 1 minute. For such heavy data, ordinary CPUs will take several days to perform one iteration of training. High performance computing devices like HPC or GPUs are necessary for faster computation.

4.2 Deliverables

1) A RESTful API for end users. A web application using React at the frontend and Flask at the backend. While React provides a user friendly and interactive user interface, Flask provides an easy API endpoint to merge with frontend.
2) A derivative paper for publication.

Chapter 5. Project Architecture

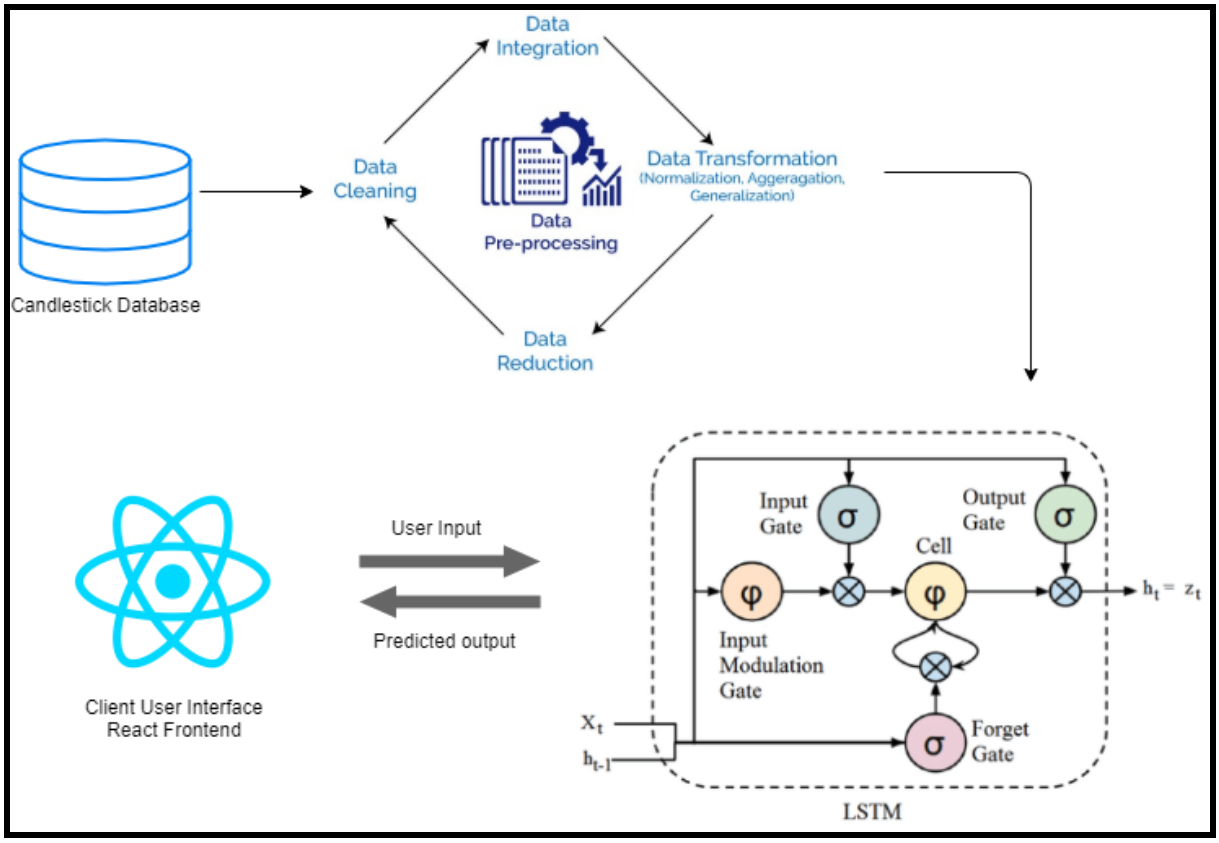


Figure 1: *High-level architecture*

A candlestick data chart is a financial chart that is used to describe price movements. A candlestick represents one day of data. It is very similar to a bar chart with four important chunks of information for that day: open, close, high and low as shown in figure 2. Each candlestick represents a datapoint in our dataset. A snapshot of the dataset is in figure 3. Each sample contains a date, ticker, timebar, first trade price, high trade price, low trade price, last trade price, volume weight, volume and total trades attributes. Data preprocessing (data cleaning, data transformation) are performed on the data to make it fit for processing.

After feature extraction, the final dataset is passed through the LSTM model for training. Long short term memory (LSTM) networks are a type of artificial recurrent neural networks that are capable of learning order dependence in sequential prediction problems. In sequential data problems or the time series problems, the data changes occur over time and it becomes necessary to keep track of changes in order to make accurate predictions. For instance, in a stock prediction problem which is a classic time series problem, the prediction of whether the stock is bearish or bullish depends on the information learned over years. Classical machine learning models have no memory per say, which makes them incapable of referring back over time to learn patterns. When there is a significant change, these models must be retrained with new datasets to keep them updated and the process is very costly for huge datasets.

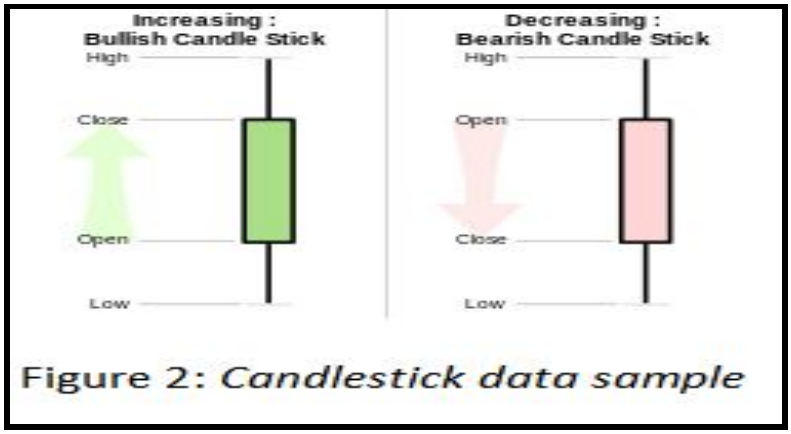


Figure 2: *Candlestick data sample*

Neural networks, especially recurrent neural networks were introduced to solve such time dependent problems. Although RNN's performed better than the classical models, RNNs could retain memory upto certain levels only. Going back several steps and connecting information was difficult for RNNs. LSTMs in this regard, have a special ability to add or remove information to the cell state, regulated carefully using gates. Gates are composed of sigmoid neural net layer and a pointwise multiplicative operation. LSTMs use tanh function for activation of the states and sigmoid activation function for the node output. The outputs of this layer are zero and one specifying how much information of each cell must be let

through. After successive epochs, the trained model is then tested for accuracy. The measure for accuracy is the loss function that trains the model for lower losses and increases the prediction accuracy.

	Date	Ticker	TimeBarStart	FirstTradePrice	HighTradePrice	LowTradePrice	LastTradePrice	VolumeWeightPrice	Volume	TotalTrades
2	20070103	AAPL	4:15	86.33	86.33	86.01	86.01	86.2093	4300	8
3	20070103	AAPL	6:07	86.25	86.25	86.25	86.25	86.25	100	1
4	20070103	AAPL	6:08	86.05	86.05	86	86.04	86.01666	4860	8
5	20070103	AAPL	6:49	86.04	86.04	86.04	86.04	86.04	200	2
6	20070103	AAPL	7:03	86.2	86.2	86.2	86.2	86.2	1000	2

Figure 3: *Sample records for APPLE*

The user interface consists of REST API calls that interact with the backend model and display prediction results to the user. The interface uses React library at the frontend, which is highly interactive and user friendly. It takes the company name or ticker as its input and provides the future prediction value as its output. The backend is developed using Flask that exposes the prediction model endpoints for prediction and sends the result back to the frontend for smooth computation and easy consumption.

Chapter 6. Evaluation Methodology

In the current implementation, we are using Root Mean Square Error (RMSE) to evaluate the model. The idea behind RMSE is to calculate the square root of the mean of actual values minus predicted values divided by the number of observations. Mathematically it is defined by the following formula:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

With Minimising RMSE, one can observe that the performance of the model increases. Usually in the training step, the RMSE starts to be a higher number but through different epochs, the number should become smaller. Based on our current implementation, the RMSE of our model is 1.90. This is a considerably low number but our implementation needs to be improved and perform the same way for other stocks. As it was discussed, this is a reiterative process and our goal is to keep the RMSE low through different changes we make to the model.

To start with the baseline approaches we will be implementing the prediction model using traditional approaches such as Random Forest, Multi-Layer Perceptron etc. On the Deep Learning Side Our goal is to implement a prediction model using LSTM Neural Networks with significant improvement in performance. LSTM is a type of Recurrent Neural Network that has the ability to differentiate between recent and early samples by assigning different weights to them. This can act as a major advantage for the dynamic and complex nature of the stock market data.

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{F}_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP}$$

The performance metrics which will be used for analysis are accuracy, precision, recall and F-1 measure. These measures are calculated based on the outcome of the true positive, true negative, false positive and false negative classes. These metrics are calculated for both classical machine learning models such as Random Forest, Multi-Layer Perceptron and the advanced deep learning model i.e. using LSTM. The results are compared to prove significant improvement in the performance of the prediction model using LSTM.

Chapter 7. System Design/Methodology

Solution Overview

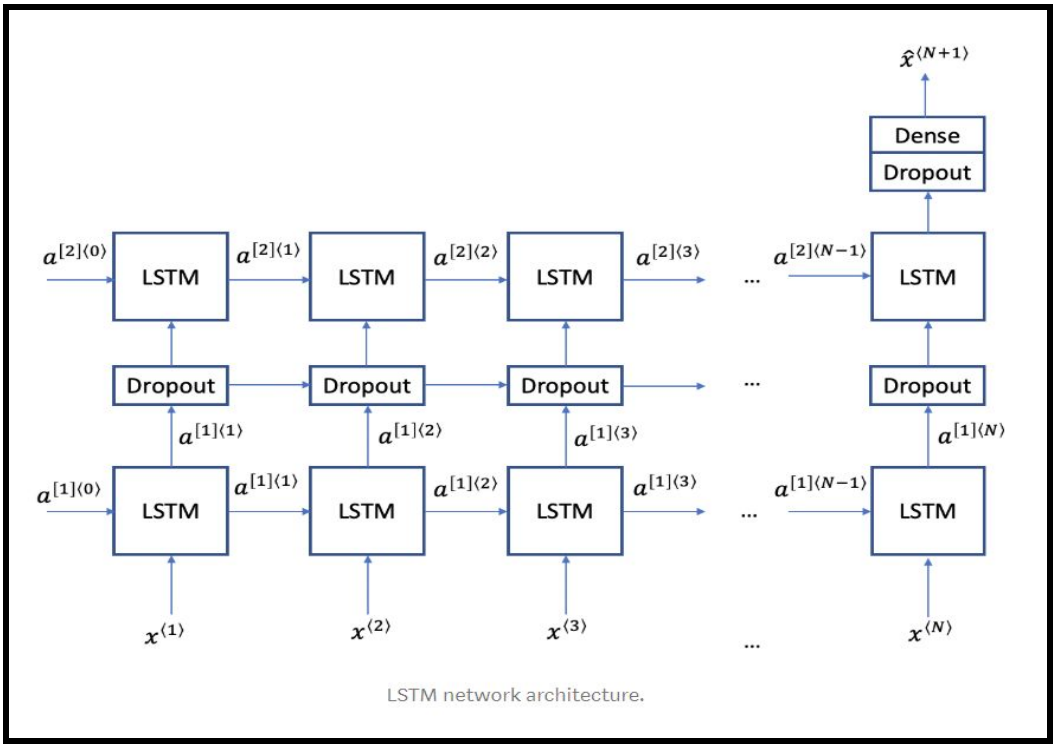
The system design for the project is classified into short achievable milestones. Each milestone is briefly described in this chapter. The Implementation Plan and Progress section contains all the necessary details of milestones achieved.

Setting up the project execution environment and acquiring the development tools were the two initial tasks that served as a prerequisite for the project methodology and implementation. Obtaining access to HPC, setting up the required libraries like Numpy, Pandas, Matplotlib, Keras, Tensorflow and obtaining development tools like Microsoft Visual Studio for React front end programming, Flask for backend integration were the first steps towards this goal. We have used Python 3.7 and the latest versions of all other libraries.

After the setup, our next goal was to ***obtain quality data***. Our dataset includes several stocks like Apple(AAPL), Baidu Inc(BIDU), Chipotle Mexican Grill (CMG), Ebay(EBAY) and so on. The dataset consists of 1 year samples of all these stocks for the year 2007. This dataset involves several .csv files where each csv file represents a day data for a particular stock. To sum up, the dataset we have at present is for *250 days(weekday data) x 41 stocks per day x 500 candlestick data samples*. Incorporating all the data files into a single csv file to feed the model is a challenging task. Current implementation focuses only on the AAPL stock. After the initial phase, we intend to expand this dataset to include at least 3 years of data.

The dataset we collected had to be pre-processed as it contained some fields that weren't helpful in analysis. So, ***data preprocessing*** was the next step in our implementation. Current steps for data preprocessing include removal of unwanted columns, obtaining the mean of the fields to create a single columnar data and converting it into two column time series data, 1st column consisting stock price of time t, and second column of time t+1. Further steps in preprocessing include binning to combine data on a per day basis, per week basis and per month basis to compare and validate the model predictions.

Now that the data is ready, the next step is to ***choose a suitable model***. From bibliographic research, we came across various approaches to solve the stock prediction problem - Linear Regression, Moving Average, Last Value method, Gradient Boosting techniques to name a few. Novel approaches among them involved deep neural networks and reinforcement learning methods. For our project, we use Long Short Term Memory - a deep learning approach that can combat the vanishing gradients problem encountered in long sequences.



LSTM network architecture.

We have currently used two sequential LSTM layers stacked together with a dense layer to build a Recurrent Neural Network (RNN) model using Keras. The final layer uses a linear activation function. As we expand our dataset, we will increase the number of sequential and dense layers for better model training.

Training, Testing and Evaluating the predicted results is the next step in the process. The dataset is split into 75% training and 25% testing data. Adagrad (adaptive gradient algorithm) is used as the optimizer for faster convergence. The predicted test results are evaluated using Root Mean Square Error as the evaluation metric.

Our future work towards the project is - more data preprocessing steps to incorporate all the data, implement and evaluate several binning methods to be able to analyze current stock price window and predict the near future raise/drop in the prices. Model implementation, training and evaluation methodologies are likely to remain the same except for the change in number of RNN layers as needed.

Chapter 8: Implementation Plan and Progress

8.1 Setting up a programming and execution environment:

In order to try out different techniques and get ourselves familiar with the dataset, a small subset of the data was used to set up the pipeline. In this project, we used Python 3.6 along with the following packages for the environment setup:

Pandas dataframe is one of the widely used tools in the field of Data Science for ease of access and having a huge set of APIs that makes working with a big amount of data subtle.

Matplotlib is used for drawing different diagrams and plots to analyze data and understand feature correlations in a dataset.

Numpy is another tool used along with Pandas for working with big data.
Sklearn is used periodically for ease of use specifically in the preprocessing step of the project. Different packages such as MinMaxScaller and mean squared error were used in this project.

Keras is an open source library running on top of TensorFlow which makes working with Neural Networks much easier. This tool is used when the data is ready to be trained and a model needs to be created. In this project, LSTM was used to build a model.

8.2 Acquiring development tools:

The development tools apart from the python libraries are React for frontend and Flask for backend programming. For React, we will be using Microsoft Visual Studio, a free IDE by Microsoft. For the endpoint testing, we will use Mocha, a tool to test API endpoints for accurate functionality.

8.3 Understanding /executing example programs or hardware simulations:

So far, we have been working on understanding the dataset and trying to build a pipeline for data preprocessing and training. For the initial implementation, a year's worth of stock data for “AAPL” was used to train a model. There are a few different approaches taken by the stock traders but for our implementation we took OHLC (Open High Low Close) mean for each reading to be the stock amount. Some of the initial preprocessing steps are calculating the aforementioned index and split the data between the testing and training set.

The approach currently taken was every row is considered to be the X value at time t and the next row being Y value at time t+1. After observing the results, we realized that this approach is not very promising and our hypothesis was wrong. Since then we changed our hypothesis to find a period of time in each day that many day traders are trading. Once we are able to filter our daily data, we are hoping that we will get a better result and be able to prove our hypothesis.

8.4 Implementing a prototype of your proposed method so the use of all technologies is understood:

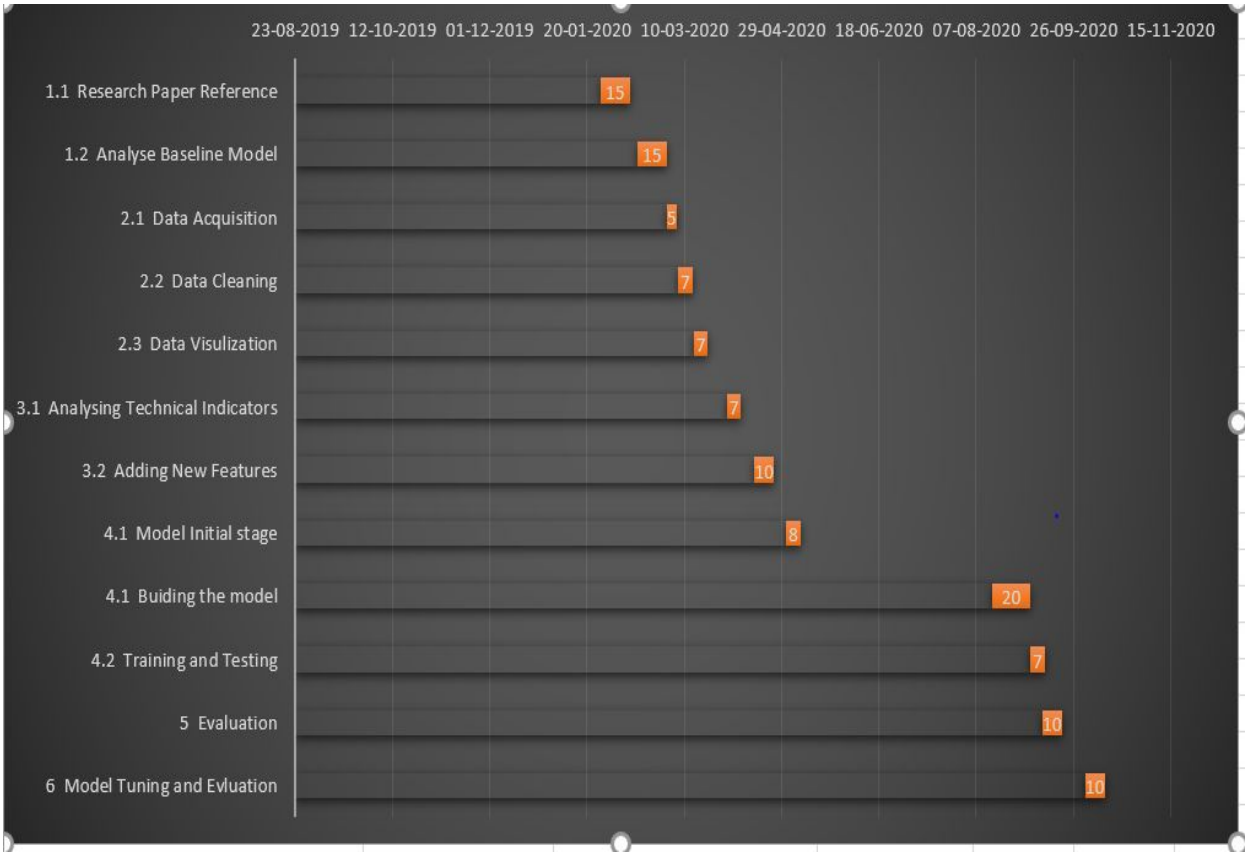
After extensive research, we found some github tutorials that are relevant to the problem stated, that use similar methodology as a solution. We are integrating the development tools and libraries required to run the code snippets. Parallelly, we have performed a basic run of the code scripts on google colab to understand the elements of the script. Also, we have tried plotting various plots to understand and visualize the dataset.

Our current implementation does not represent the bigger picture we are working toward. The end goal of this project is to allow the user to submit a snapshot of a certain stock within specified time of the day, and provide a price prediction of the near future. The results we have achieved so far are promising but the core part of the project is coming up

with the strategy that we need to train our model toward. This step is going to be repetitive until we obtain a result that we are aiming for.

Chapter 9. Project Schedule

9.1 Gantt Chart



TASK	Person Responsible
CMPE 295A	
Phase1 : Research Paper	
1.1 Research Paper Reference	Team
1.2 Analyse Baseline Model	Team
Phase2 : Exploratory Data Analysis	
2.1 Data Acquisition	Shreya,Sneha
2.2 Data Cleaning	Shreya,Sneha
2.3 Data Visulization	Shreya,Sneha
Phase3 : Data Engineering	
3.1 Analysing Technical Indicators	Mahesh,Mohdi
3.2 Adding New Features	Mahesh,Mohdi
CMPE 295B	
Phase4 : Implementation of Model	
4.1 Model Initial stage	Team
4.1 Buiding the model	Team
4.2 Training and Testing	Team
Phase5 : Evaluation	Shreya,Sneha
Phase6 : Model Tuning and Evluation	Mahesh,Mohdi

9.2 Pert Chart