香港城市大學
City University of Hong Kong

Department of Computer Science

BSCCS Final Year Project 2021-2022

Interim Report II

21CS068

Stock Market Price Prediction with a combination of NLP and Time Series

(Volume ___ of ___)

Student Name	:	**VOBBILISETTY Sai Navyanth**
Student No.	:	**55357700**
Programme Code	:	**BSCEGU4**
Supervisor	:	**Dr. SONG, Linqi**
Date	:	**January 31, 2022**

For Official Use Only

Table of Contents

1. Abstract

This project aims to provide retail investors a third party web application to navigate through the stock market and see buy/sell signals of any particular stock. This is achieved with different Machine Learning and Web Technologies. We built several stock price prediction models both using NLP on news data, time series analysis using closing price and technical indicators of the stock. The models include feedforward neural networks, recurrent neural networks, signals from technical analysis. All the models and hyperparameters are optimized.

2. Summary of Revisions and Additions since interim report 1

2. 1 Revision to interim report 1

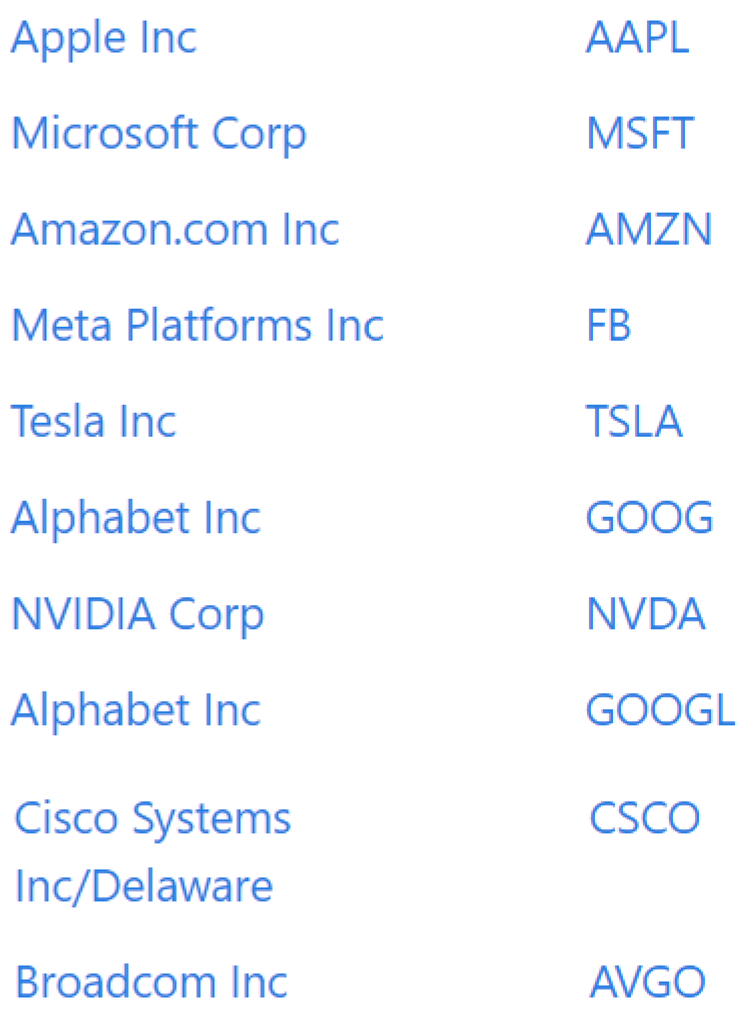
The professor has asked to add two changes to the report:

1. Describe the data collection process.
2. Implement simple algorithms to get benchmark results.

All the code worked so far has been pushed to GitHub. This link to it is here

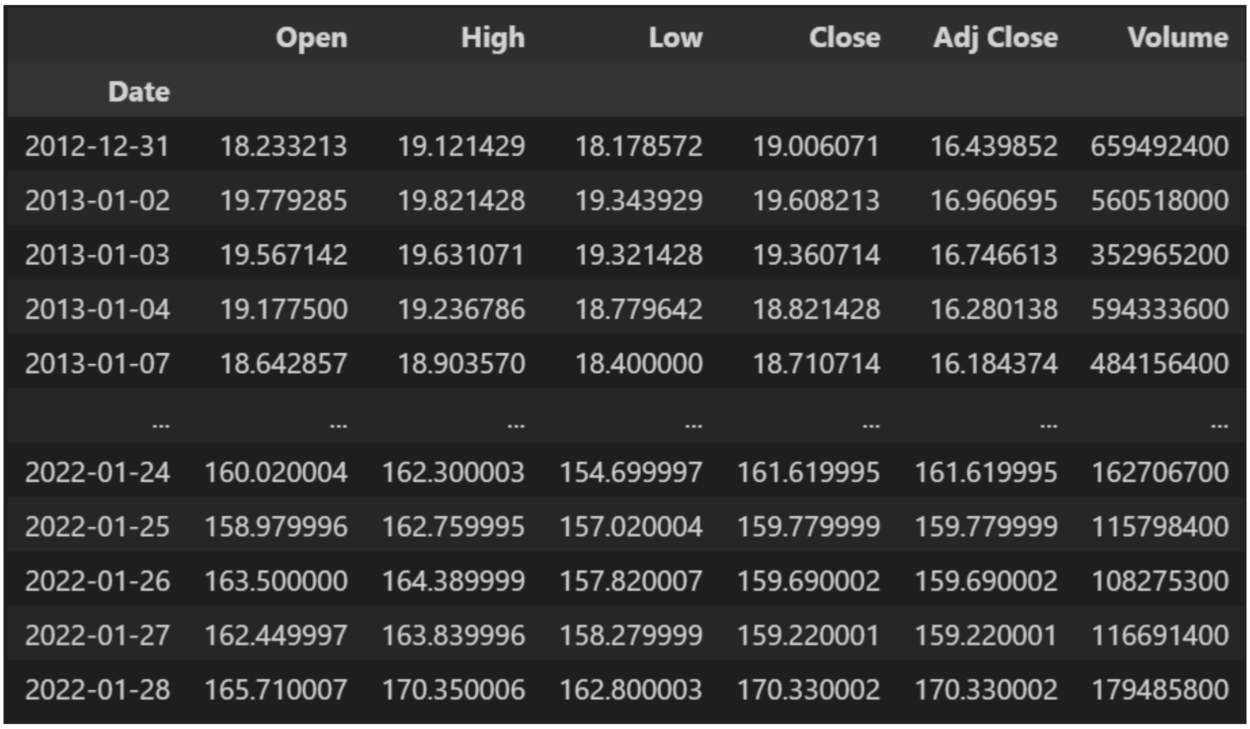
2. 1. 1 Data Collection

I've collected historical data of the top ten companies in the Nasdaq 100 index. These are the ten companies I've used:



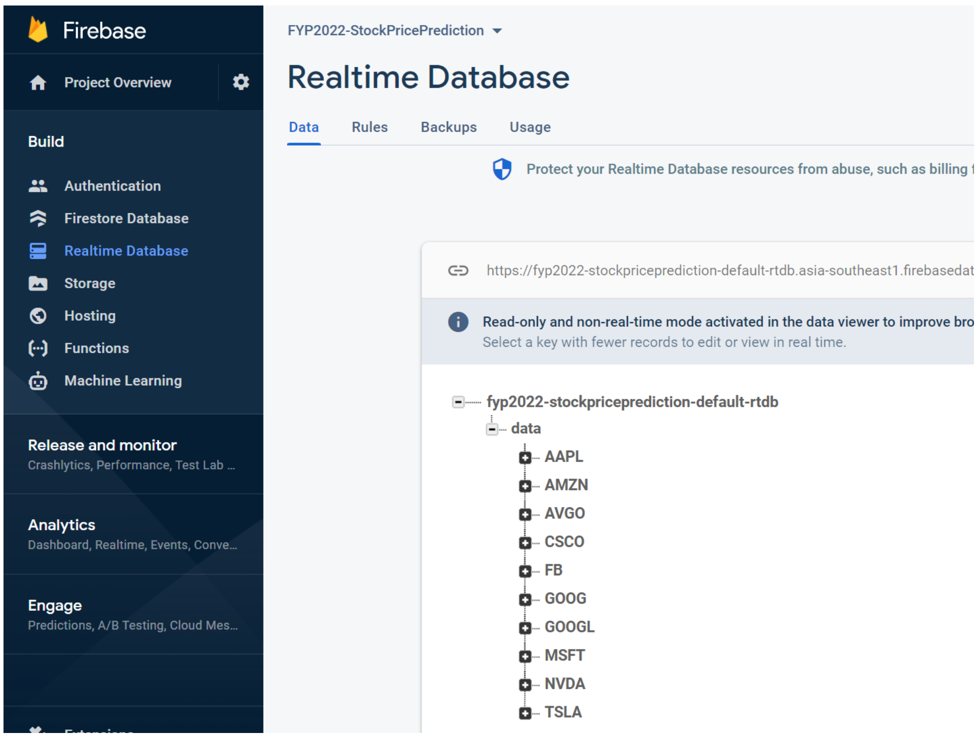
Apple Inc	AAPL
Microsoft Corp	MSFT
Amazon.com Inc	AMZN
Meta Platforms Inc	FB
Tesla Inc	TSLA
Alphabet Inc	GOOG
NVIDIA Corp	NVDA
Alphabet Inc	GOOGL
Cisco Systems Inc/Delaware	CSCO
Broadcom Inc	AVGO

The data collected for training is historical data of the stock from the date '2013-01-01' to '2020-11-30':



Date	Open	High	Low	Close	Adj Close	Volume
2012-12-31	18.233213	19.121429	18.178572	19.006071	16.439852	659492400
2013-01-02	19.779285	19.821428	19.343929	19.608213	16.960695	560518000
2013-01-03	19.567142	19.631071	19.321428	19.360714	16.746613	352965200
2013-01-04	19.177500	19.236786	18.779642	18.821428	16.280138	594333600
2013-01-07	18.642857	18.903570	18.400000	18.710714	16.184374	484156400
...	...	...	...	...	...	...
2022-01-24	160.020004	162.300003	154.699997	161.619995	161.619995	162706700
2022-01-25	158.979996	162.759995	157.020004	159.779999	159.779999	115798400
2022-01-26	163.500000	164.389999	157.820007	159.690002	159.690002	108275300
2022-01-27	162.449997	163.839996	158.279999	159.220001	159.220001	116691400
2022-01-28	165.710007	170.350006	162.800003	170.330002	170.330002	179485800

This data for the above mentioned 10 companies are collected and are being stored in firebase real time database.



Realtime Database

Protect your Realtime Database resources from abuse, such as billing f

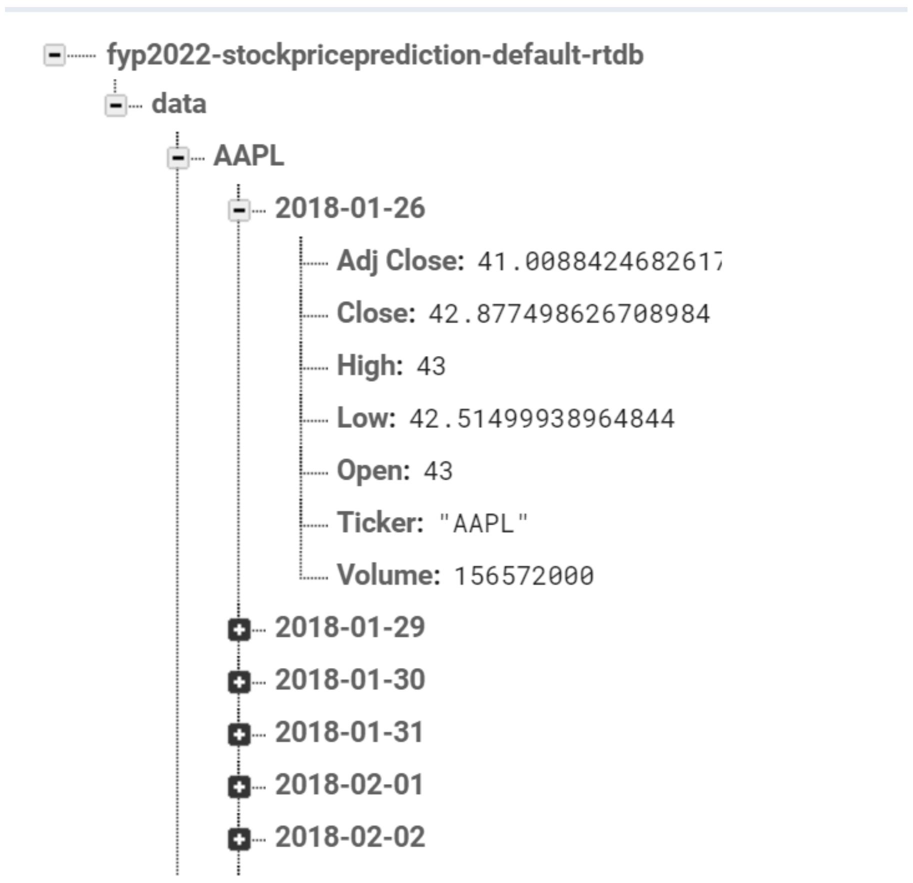
https://fyp2022-stockpriceprediction-default-rtdb.asia-southeast1.firebasedat

Read-only and non-real-time mode activated in the data viewer to improve bro
Select a key with fewer records to edit or view in real time.

```
fyp2022-stockpriceprediction-default-rtdb
  data
    AAPL
    AMZN
    AVGO
    CSCO
    FB
    GOOG
    GOOGL
    MSFT
    NVDA
    TSLA
```

Each of the documents contains information about the stock. The format of storage is:

data -> Ticker -> date -> {open, high, low, close, ticker, volume}



```
fyp2022-stockpriceprediction-default-rtdb
  data
    AAPL
      2018-01-26
        Adj Close: 41.0088424682617
        Close: 42.877498626708984
        High: 43
        Low: 42.51499938964844
        Open: 43
        Ticker: "AAPL"
        Volume: 156572000
      2018-01-29
      2018-01-30
      2018-01-31
      2018-02-01
      2018-02-02
```

From now on, we will be using only the data stored in firebase to train our models.

2. 1. 2 Implementing Algorithms to get results

So far, I have implemented two models completely and I am still working on two other models.

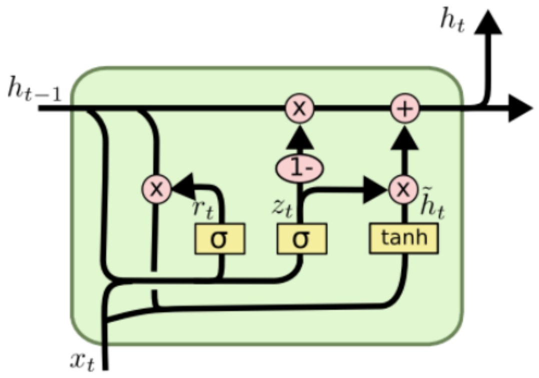
The models are:

1. LSTM on stock price data (completed)
2. Artificial Neural Network using different technical data of the stock (completed)
3. ARIMA (Still working)
4. Prophet (Still working)

5. Multivariate LSTM on different time-series data (need to work)

2. 1. 2. 1 LSTM on Stock Price Data:

LSTM (Long Short Term Memory) Neural Network models are a type of Recurrent Neural Networks which are used to analyze sequential data. A common LSTM unit is composed of a **cell**, an **input gate**, an **output gate** and a **forget gate**. The cell remembers values over arbitrary time intervals and the three *gates* regulate the flow of information into and out of the cell.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

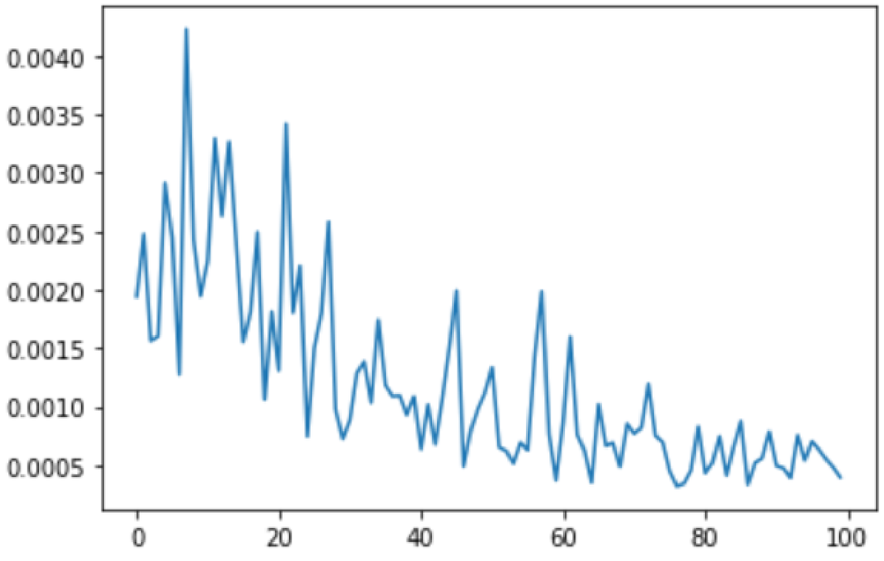
So, LSTMs can be used to analyze time-series data as well. In my project, for the LSTM model, the input would be a sequence of the closing price of ‘time_step’ number of previous days. The output prediction is the price of the stock the next day. Preprocessing step of scaling the data is done so that all the prices are in the same range.

Our model has three LSTM layers in our architecture. I’ve run it for 10 epochs by configuring the dependencies to run it on GPU provided by the CS lab. This is how it performed in the 10 epochs.

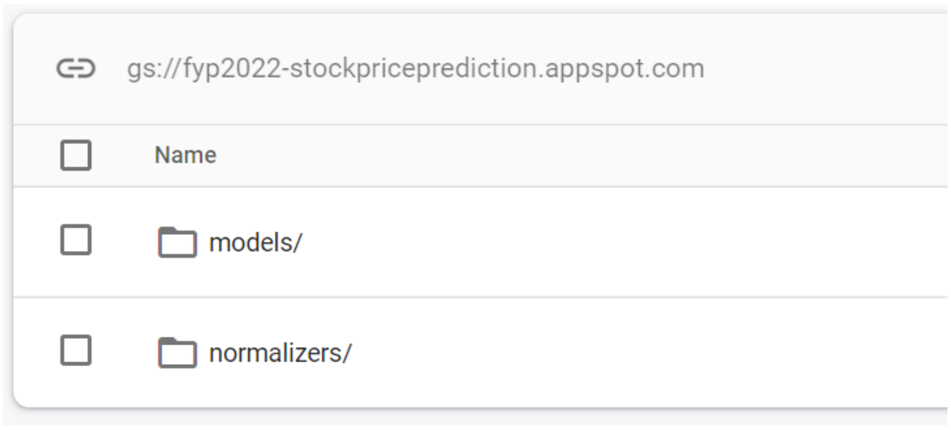
This is how the model performed after 100 epochs.

```
50/50 [==============================] - 1s 26ms/step - loss: 1.1204e-05 - val_loss: 3.9589e-04
```

This is a plot of validation loss.



As there is one model for one company, and one normalizer per company. As we need to use the same model and normalizer for predicting new data, we need to save the model and normalizers. So, I saved them both for each company in firebase:

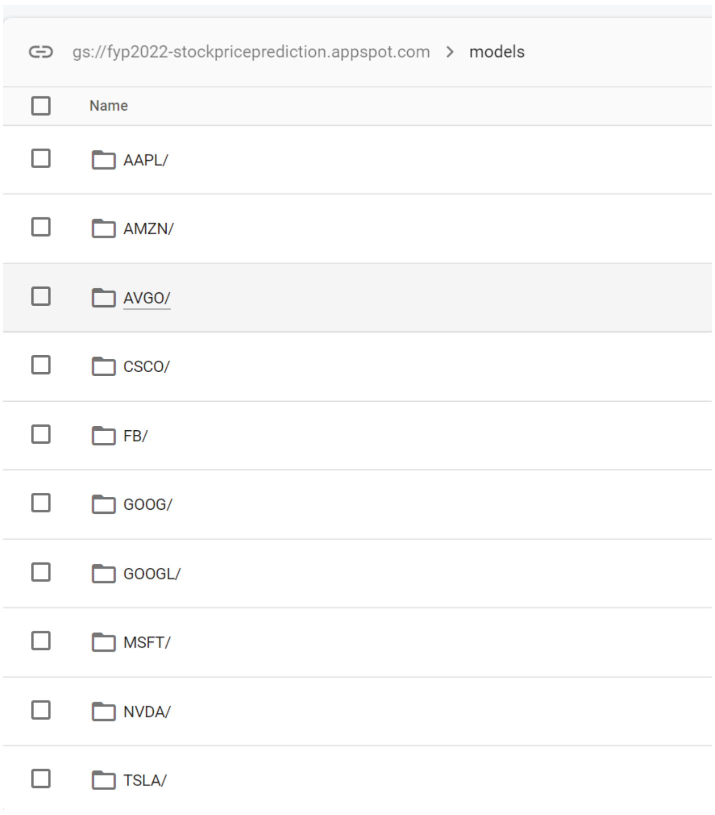


gs://fyp2022-stockpriceprediction.appspot.com

Name

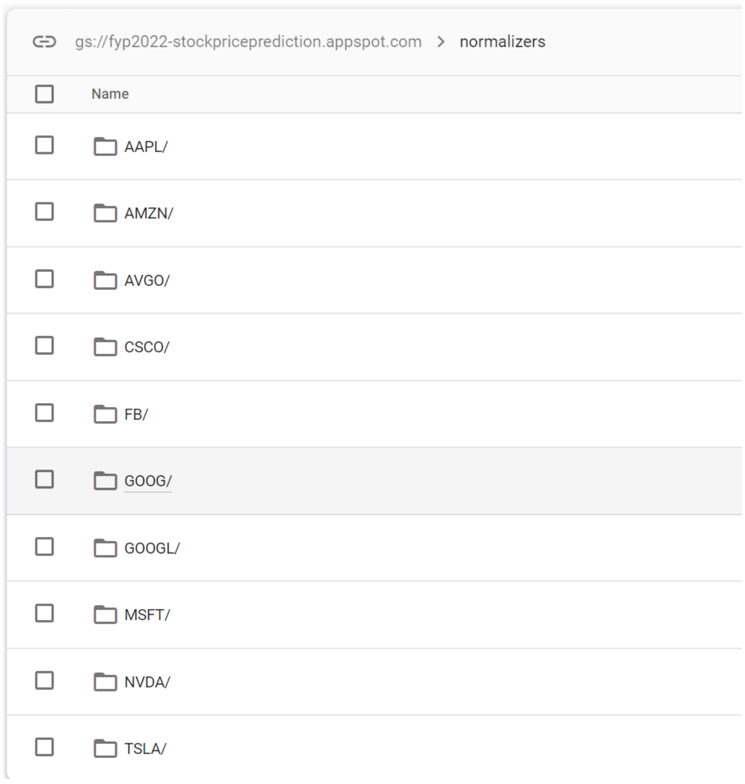
models/

normalizers/



gs://fyp2022-stockpriceprediction.appspot.com > models

Name
AAPL/
AMZN/
AVGO/
CSCO/
FB/
GOOG/
GOOGL/
MSFT/
NVDA/
TSLA/



gs://fyp2022-stockpriceprediction.appspot.com > normalizers

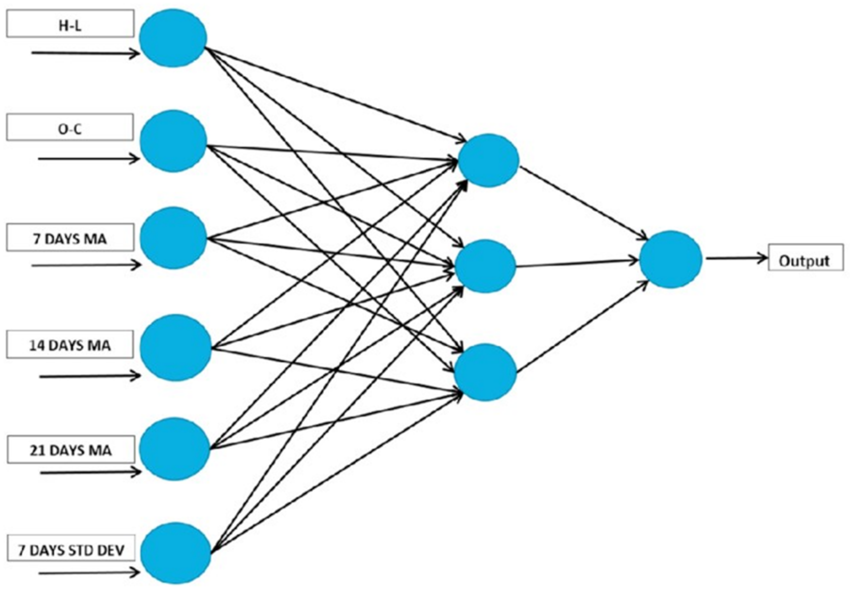
Name
AAPL/
AMZN/
AVGO/
CSCO/
FB/
GOOG/
GOOGL/
MSFT/
NVDA/
TSLA/

2. 1. 2. 2 Artificial Neural Network using different technical data of the stock

Predicting the price of the stock only based on the closing price of the stock is not that viable. We need to take into account much other information derived from the stock. So, I have calculated extra information from the given data like ‘High - Low’, ‘Open - Close’, ‘7 Day Moving Average’, ‘14 Day Moving Average’, ‘21 Day Moving Average’, ‘7 Day Standard Deviation’. As the range of all these features a lot, I scaled the dataset to be between 0-1.

```
df['H-L'] = df['High'] - df['Low']
df['O-C'] = df['Open'] - df['Close']
df['7MA'] = df['Adj Close'].rolling(window=7).mean()
df['14MA'] = df['Adj Close'].rolling(window=14).mean()
df['21MA'] = df['Adj Close'].rolling(window=21).mean()
df['7SD'] = df['Adj Close'].rolling(window=7).std()
```

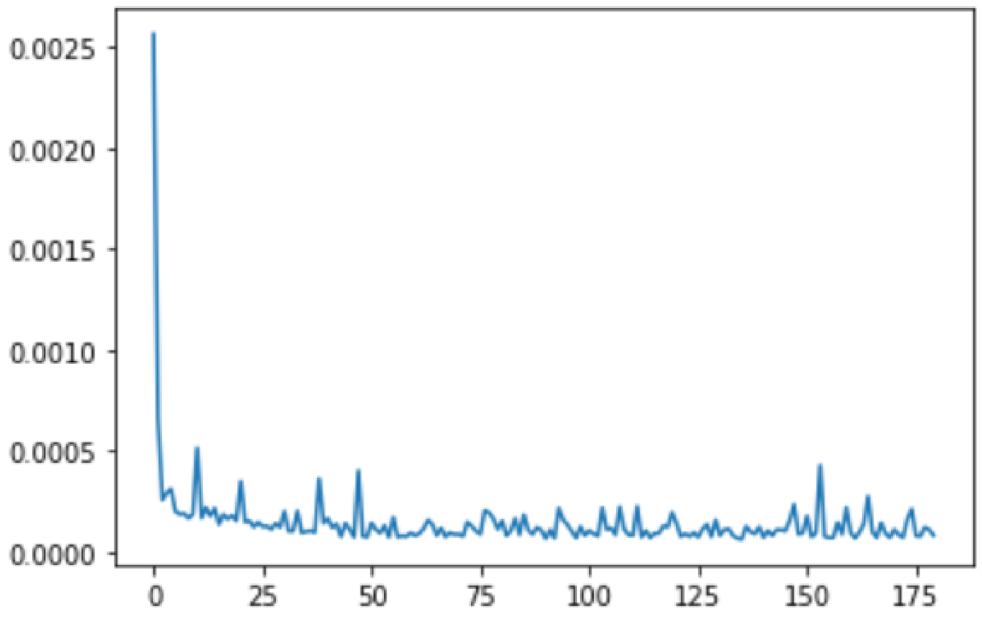
I built a standard neural network that takes in these seven inputs. The input would be all the values of the first day, and the output should be the price of the second day. This is how the neural network can be visualized:



I’ve run the model for 100 epochs. This is how it performed:

```
Epoch 11/180
163/163 [==============================] - 1s 3ms/step - loss: 2.2185e-04 - val_loss: 3.1171e-04
Epoch 12/180
163/163 [==============================] - 1s 3ms/step - loss: 2.1951e-04 - val_loss: 2.0703e-04
Epoch 13/180
…
Epoch 179/180
163/163 [==============================] - 1s 3ms/step - loss: 1.1136e-04 - val_loss: 1.2165e-04
Epoch 180/180
163/163 [==============================] - 1s 3ms/step - loss: 1.2434e-04 - val_loss: 1.1596e-04
```

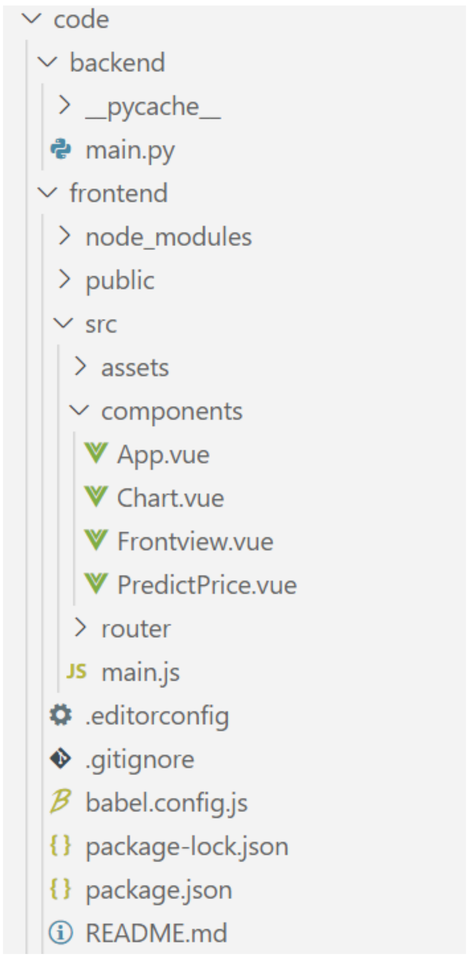
The plot below is the validation loss for epochs trained:



I’ve tuned the hyperparameters to see which model works the best. In the same way as the LSTM method, I’ve saved the models and normalizers online in firebase storage.

2. 2 Additions to interim report

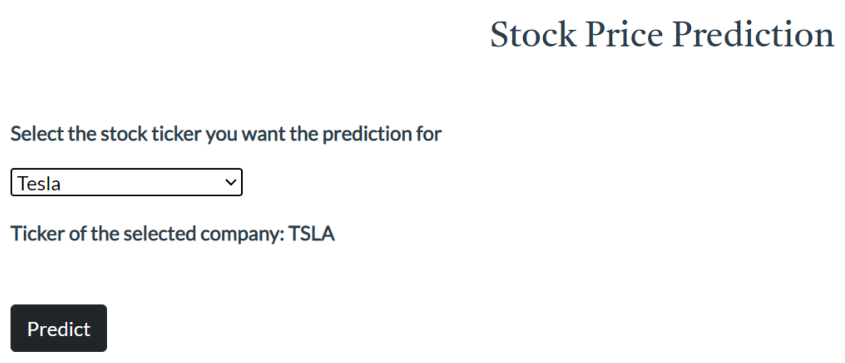
I’ve worked on the webapp as well to display my results. The webapp is implemented using HTML, CSS, JavaScript (VueJS framework) for the front end, and flask API to implement the backend. This is the folder structure of the webapp:



```
code
  backend
    __pycache__
    main.py
  frontend
    node_modules
    public
    src
      assets
      components
        App.vue
        Chart.vue
        Frontview.vue
        PredictPrice.vue
      router
      main.js
    .editorconfig
    .gitignore
    babel.config.js
    package-lock.json
    package.json
    README.md
```

In the frontend, I have four components to help code the frontend. This is what the webapp looks like now. I need to iteratively develop the webapp to perform better.

Firstly, the user selects the company for which he/she wants to predict the price. Later, they select ‘predict’ button.

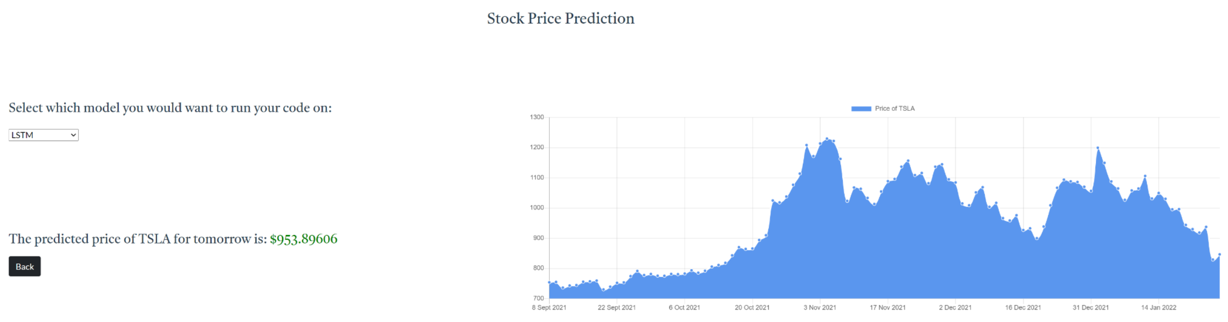

Stock Price Prediction

Select the stock ticker you want the prediction for

Ticker of the selected company: TSLA

Once the user presses ‘predict’, the ticker symbol is being sent to the backend. Flask python api gets the model and normalizer from the firebase api. Once the model is loaded, test data is being created by getting the past 100 days of stock using ‘yfinance’ library, and preprocess the dataset using the normalizer loaded. Once we have the normalized data, use the loaded model to predict the price. A json response is being sent to the frontend to display the required information.

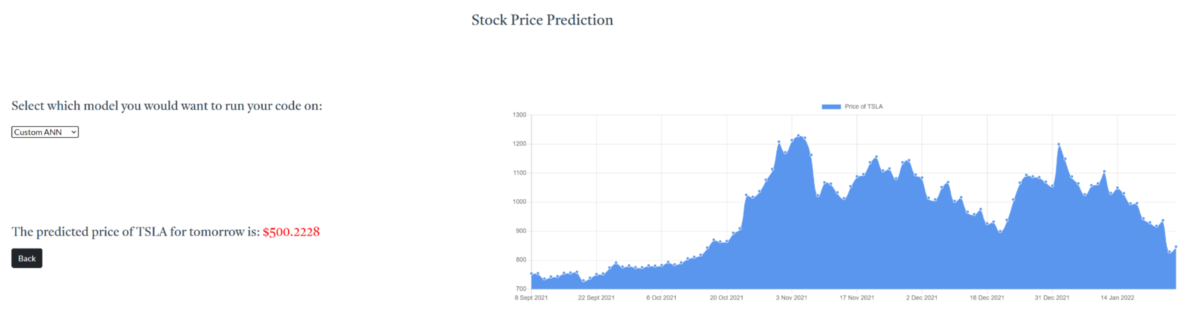
Here, there is a model dropdown list, where by default it selects the LSTM model. On the right, there is a graph showing all the past 100 days' stock prices.


Stock Price Prediction

Select which model you would want to run your code on:

The predicted price of TSLA for tomorrow is: $953.89606

Here, I changed the model to Custom ANN, and it loads the prediction using that model.


Stock Price Prediction

Select which model you would want to run your code on:

Custom ANN

The predicted price of TSLA for tomorrow is: $500.2228

Back

If the price is more than yesterday, then it'll display in green color or else in red.

2. 3 Tasks need to be done

I need to iteratively develop the frontend, backend, and machine learning models to make it perform better.

I need to backtest the strategies and find out how much return can be generated if these strategies were implemented in real life, taking into consideration of real-life scenarios.

Need to test iteratively the webapp and machine learning models to see how it works.

[From here, it is a reference to the interim report 1]

1. Introduction

In recent years, the number of retail investors investing in the stock market has significantly increased [4]. This rise can be attributed to the pandemic since retail investors became aware of and better informed about stock markets. Another factor that led to this is influential people like Elon Musk, Chamath Palihapitiya, etc.

Most retail investors invest based on the crowd/momentum and end up incurring losses. I have witnessed the losses of many people who had invested their own money into the market as first-time retail investors with little knowledge about the intricate workings of the market.

Moreover, it is very time-consuming for retail investors to keep track of the various companies and industries in the stock market.

As a solution to this problem, my groupmate (Varun) and I decided to work on this topic for our final year project as a group, where we aim to assist retail investors to predict better as to whether they should invest in a particular company by checking the company's recent news and historical data to forecast the price of the stock.

The two leading technologies we will be using in our FYP are:

1. Time Series Analysis/Forecasting (implemented by me)
2. NLP (Natural Language Processing) (implemented by my groupmate)

1. 1 Time Series Analysis

Time series is a series of observations recorded at regular time intervals. The time interval can be hourly, daily, weekly, monthly, quarterly, and annually. Depending on the problem, there might be time series analysis where time intervals can be as small as seconds or minutes [10].

In our project, we would use time series to predict/forecast the price of a particular company's stock in the future, using the stock's price in the past. This is a quantitative way of predicting the price of a stock and does not consider any sentiment of how the company is performing in the market.

1. 2 Natural Language Processing

According to IBM [6], Natural Language Processing is a branch of Artificial Intelligence, which gives computers the ability to understand the text and spoken words in much the same way as humans.

NLP is used in multiple scenarios [1]. Here are some of them:

1. Text Search
2. Machine Translation
3. Summarizing group of text

4. Sentiment Analysis

In this project, we will be focusing mainly on sentiment analysis of news data. Sentiment analysis is the process of calculating the sentiment of a given text using different methods. The sentiment can be either positive, negative, or neutral.

1. 3 Project Objective & Scope

Objective: We are creating a fully working framework to predict the price of a stock, from acquiring historical data from yahoo finance to cleaning the data to train our algorithms (mainly LSTM and ARIMA). In the end, create a simple web app to display the results.

Scope: We intend to create multiple models and test which model is efficient. The main models we plan to implement are:

1. LSTM (Long Short Tem Memory) Neural Network
2. ARIMA (Auto-Regressive Integrated Moving Average)

We will backtest both these methods and decide which one should be used in our web app.

For the data, we will be using the top 10 companies present in the NASDAQ 100 index as of 19th September 2021.

2. Literature Review

The recent works in this area are mainly focused on predicting the price of a stock using NLP on stocks' news [8] or through time-series analysis of the historical prices of the stocks [5,7,12]. However, predictions solely based on this information might not be perfect. In the following section, I explain the reasons why these methods are not excellent.

2. 1 Stock Price Prediction Using Time Series Models

2. 1. 1 Analysis

There are different models one can use to perform time series analysis.

Some of them are:

1. ARIMA Model
2. PROPHET
3. KERAS with LSTM

The paper (Chouksey, 2019) has implemented all the above three methods for stock price prediction. The author conducted a detailed analysis of all three methods by comparing their Root Mean Square Errors (RMSE). In other words, the lower the error rate, the better.

They have analyzed five different Indian banks for 20 years taken from the Yahoo Finance library.

The detailed analysis shows that the KERAS with LSTM model performs the best with an average RMSE of 6% compared to 151%, 173% for PROPHET and ARIMA Model, respectively.

2. 1. 2 Limitations

There are multiple limitations in approaches like this. They are:

1. Lack of backtesting:

While implementing a trading strategy, backtesting is very important. Backtesting is basically to determine how good a trading strategy is by testing the strategy with historical data. In this way, one can find out how a particular strategy works [7].

2. Merging algorithms:

The research paper described above has only worked with one algorithm at a time. For example, in one scenario, ARIMA might be better than LSTM, while it might be different in another. If the author had tried to combine the outputs of both these algorithms, the output might have been better. This process of combining models is called ensemble learning.

Ensemble learning combines multiple models or classifiers to improve the accuracy of a particular problem. So, models trained with ensemble learning are generally better than models which are static [11].

3. Performance it in real life:

The author doesn't implement the strategy in real life to show how it performs. For example, a paper trading account can be opened, dedicated to a trading strategy that would determine how the portfolio is performing.

In this way, we can financially analyze how the strategy would perform in real life if implemented with real money. But, this is not done in most research papers, as they mainly focus on the technical implementation.

2. 2 Improvements in our potential solution

We mainly tackle the three limitations that are described above. To overcome the first problem, we do extensive backtesting to our algorithm. As we already have a stock's historical price, we can backtest the strategy using the data.

To overcome the second limitation, both the algorithms developed by my groupmate and I would be combined in the end to provide better results. In this way, we won't have to depend on only one algorithm.

Finally, to overcome the third limitation, we implement our strategies in real life using online stock brokers that provide free paper money, which can be used to test.

3. Preliminary design, solution, and system

3. 1 Data Collection and Storage

We will be using the 'yfinance' library in python to get the historical data of stock. The primary focus in this project would be the top 10 companies of NASDAQ 100. They are:

We will extract the past ten years of data of these ten companies and use it to train our algorithms.

We will be using a MongoDB database located in a local Heroku server. The data we extract will be stored in this MongoDB database to be referenced anytime when needed.

In addition, I will be helping my groupmate in extracting news data, as it is comparatively difficult. We will be extracting news data from different websites like Bloomberg, Reuters, etc., and save it in the MongoDB database. We will build a scraper in python and run it every day so that it can extract the required data every day.

3. 2 Training Algorithms

We will be implementing two kinds of algorithms:

1. ARIMA
2. LSTM

3. 2. 1 ARIMA

Auto-Regressive Integrated Moving Average is a statistical method to analyze and forecast time series data. Lets breakdown ARIMA to understand better:

Auto-Regression (AR): This algorithm determines the relationship between the present value or observation and few lagged values or observations.

Integrated (I): Subtracting an observation/value from an observation/value at another time step. This is used to make the time series stationary. A stationary time series data is a series of data whose mean and standard deviation is constant throughout the series. ARIMA cannot work if the data is not stationary. That is why ARIMA converts the data to stationary beforehand.

Moving Average (MA): This is an algorithm that uses the relationship between an observation or value and a residual error by applying the moving average of previous observations or values.

ARIMA mainly consists of 3 parameters p,d, and q.

'p', also known as 'lag order' represents the number of previous observations or lagged observations.

'd' represents the order of differencing needs to be done in Integrated (I).

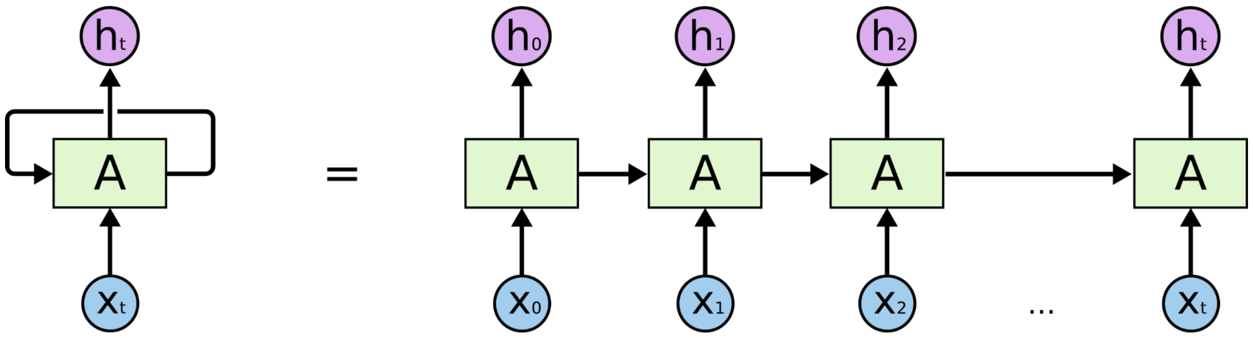
'q' represents the size of the moving average window.

We will be tuning these three parameters and finding the best combination that would fit our training data.

We will mainly use the 'statsmodels' library in python, which provides many statistical models, where ARIMA is one of them. We will use hyperparameter tuning using 10-fold cross-validation to find the best values of 'p', 'd', and 'q'.

3. 2. 2 LSTM (Long Short Term Memory)

LSTM networks are a type of RNN (Recurrent Neural Network, which is used to learn from the experience for tasks like speech recognition, language translation, etc.) used to learn and deal with long-term dependencies. Traditional RNNs generally fail when there are long-term dependencies. For example, if asked to translate a very long sentence, an RNN would fail to do so. This is where LSTMs are utilized. They come with all the utilities an RNN can provide and have additional features that an RNN fails to provide.



It is important to learn from past experience in this project and provide it to future data. That is why we will be using LSTMs. We will use the closing price of a stock and train it to our LSTM network using a moving window of 30-40 steps.

In other words, using the past 30-40 days of stock price, our trained LSTM model will predict the following day's price. The number of days, moving period is a hyperparameter. With our current knowledge, we think it would be better to have 30-40 days as our training set, but it would change as we backtest our algorithm.

We will use the PyTorch framework to build our LSTM network. PyTorch was developed by Facebook. Our LSTM network would have initially five hidden layers and 50 neurons, and our data will be normalized to be between 0-1 using the MinMax Scaler from the 'scikit-learn' library. This will make our data normalized. Later, we will update our training data in such a way that the input would be a sequence of prices of the previous 30-40 days, and the output would be the price of the following day. The hyperparameters which need to be tuned for the network would be the number of layers, number of epochs, mini-batch size. etc.

Once both the algorithms are done training, we will backtest our models and find which one works better, either LSTM or ARIMA. Later, we will store the model as a pickle file in our MongoDB database. This final model will be used in the web app for the predictions.

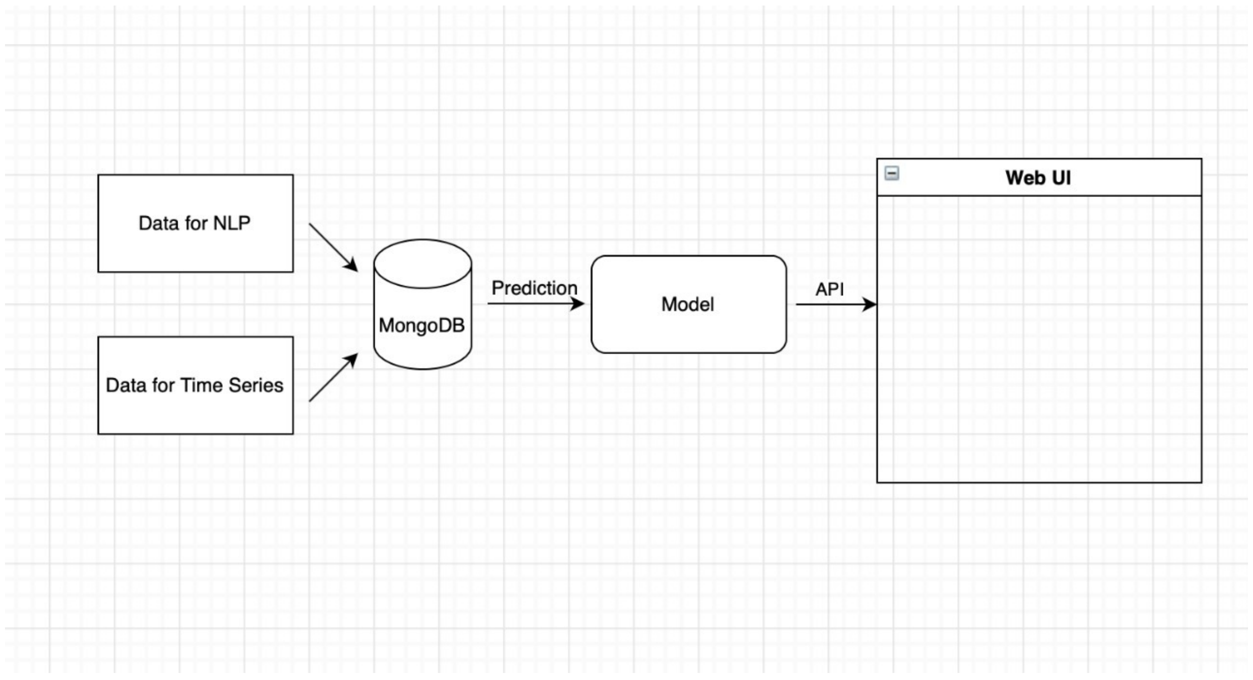
3. 3 Web Application Development

Once the model training part is done, we will build a simple web app using the JavaScript framework Vue.js for the front-end and a python framework called flask for the backend.

In the front end:

- We will allow the user to select a company from a dropdown list of the ten companies.
- A graph showing how the stock performed in the past 30-40 days.
- The prediction of the stock's price, along with a signal showing whether to buy/sell/hold the particular stock.

In the backend, we will connect the web app to the MongoDB database, containing the final models and the data. In addition, the predictions will be stored in another database in order to let the user view the predictions over time. This would be the final workflow of our project:



4. Testing Procedures

We test the whole framework in two different ways:

1. Unit and Integration Testing
2. Backtesting by Using a Virtual Trading Environment

4. 1 Unit and Integration Testing

Our framework consists of multiple interdependent modules. So, it is crucial to perform Unit Testing of smaller modules and Integration testing whenever needed (when combining multiple smaller modules). If one of the modules fails to work, the whole framework will break down. So, it is important to do these steps.

We will be using two python libraries, ' unittest' and 'integrate', to perform Unit Test and Integration tests, respectively, in python. But other than these two, we will also write custom unit test cases, in order to test our machine learning model, if it returns good results for base cases. For example, one test case can be, if a hypothetical stock stays constant everyday, which means, the price of the stock is constant everyday, then the prediction should also be the same constant. But, if our model fails to do so, it means it failed the uni test. Similarly, we can add more test cases, to test our model.

4. 2 Backtesting by Using Virtual Trading Environment

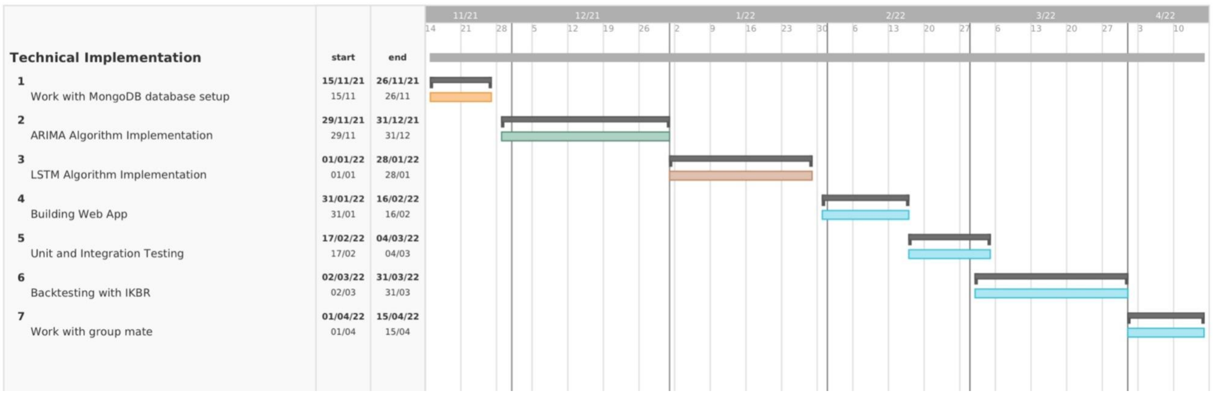
Any Machine Learning algorithm or research paper dedicated to predicting stock prices (for example, the research paper explained in Literature Review) would calculate the MSE (Mean Squared Error) of their predictions and analyze how their algorithm worked. But, in real-life trading scenarios, there are various factors to consider if a trading strategy works well, like transaction costs, bid-ask spread, time lag, etc.

This is our primary source of analyzing and testing the performance of our models. Interactive Brokers (IKBR) is a US trading platform used to trade various kinds of securities like stocks, options, futures, currencies, bonds, and funds. IKBR provides a free API and allows users to have a virtual/paper trading account.

In IKBR, we will open a paper trading account with virtual cash of $100,000. We will maintain the portfolio of the ten stocks of NASDAQ 100, and we will make trades solely based on the predictions from our model. We will continue this process for every trading day in a month. In the end, we will calculate different financial ratios, like the Sharpe ratio, hit ratio, the net return, etc., to evaluate our portfolio in real life.

5. Tasks to be done

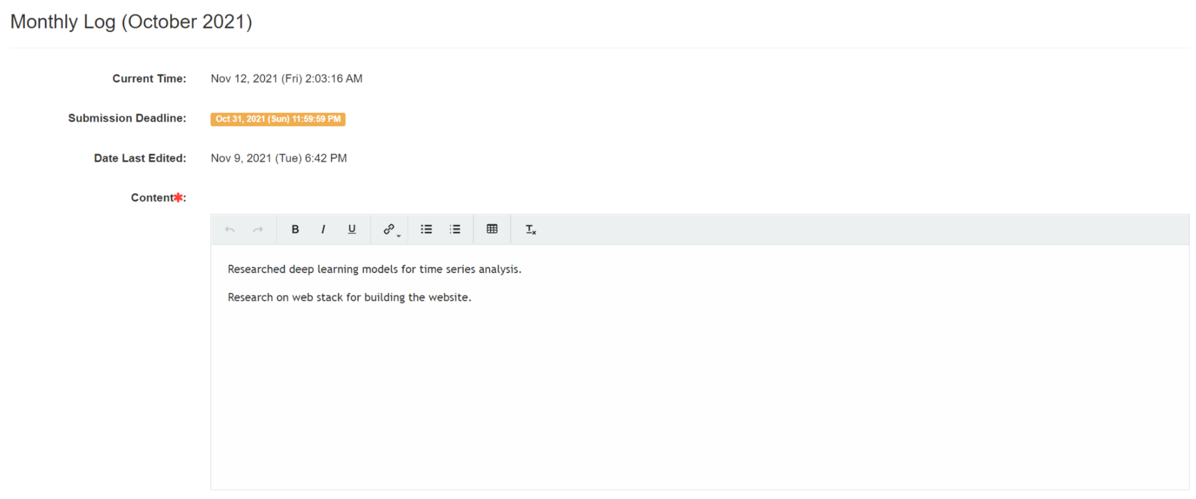
At the time of writing this report, research on various frameworks for web apps and neural networks has been done. After today, I will be following the tasks as mentioned in the below Gantt chart.



Technical Implementation	start	end
1 Work with MongoDB database setup	15/11/21 15/11	26/11/21 26/11
2 ARIMA Algorithm Implementation	29/11/21 29/11	31/12/21 31/12
3 LSTM Algorithm Implementation	01/01/22 01/01	28/01/22 28/01
4 Building Web App	31/01/22 31/01	16/02/22 16/02
5 Unit and Integration Testing	17/02/22 17/02	04/03/22 04/03
6 Backtesting with IKBR	02/03/22 02/03	31/03/22 31/03
7 Work with group mate	01/04/22 01/04	15/04/22 15/04

According to the schedule, ARIMA and LSTM algorithms need to be implemented by the end of January 2022. By the end of April 2022, Testing and Webapp implementation needs to be done.

6. Monthly Log

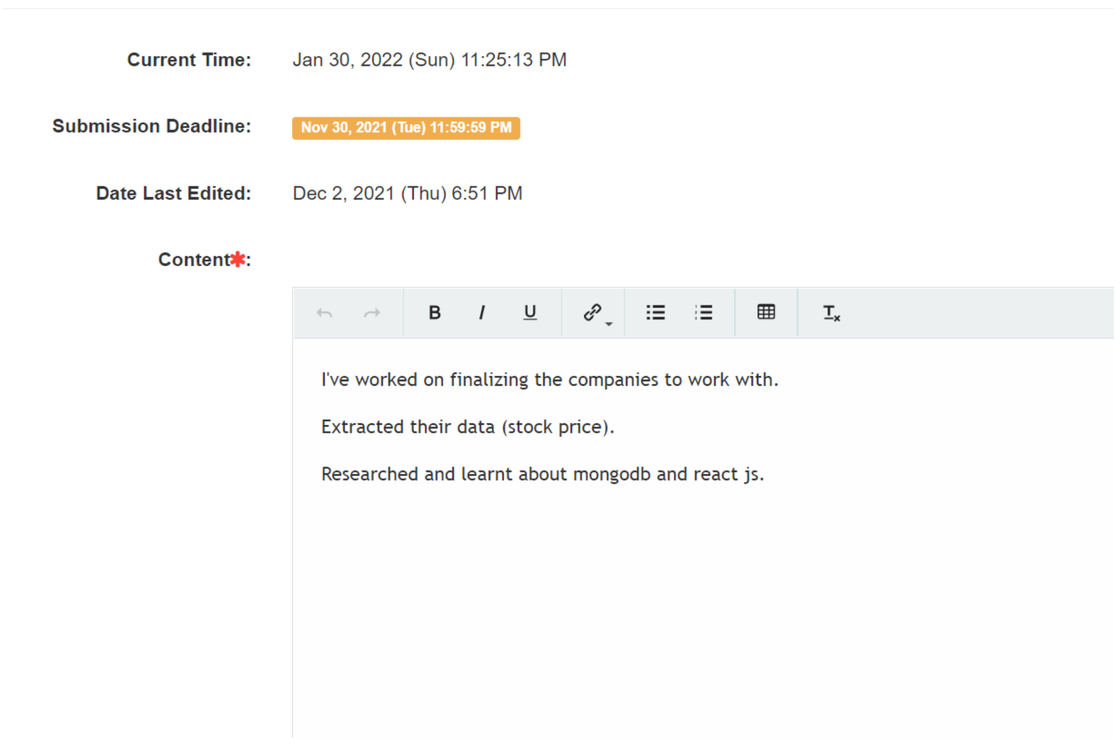


Monthly Log (October 2021)

Current Time:	Nov 12, 2021 (Fri) 2:03:16 AM
Submission Deadline:	Oct 31, 2021 (Sun) 11:59:59 PM
Date Last Edited:	Nov 9, 2021 (Tue) 6:42 PM
Content✱:	

Researched deep learning models for time series analysis.

Research on web stack for building the website.

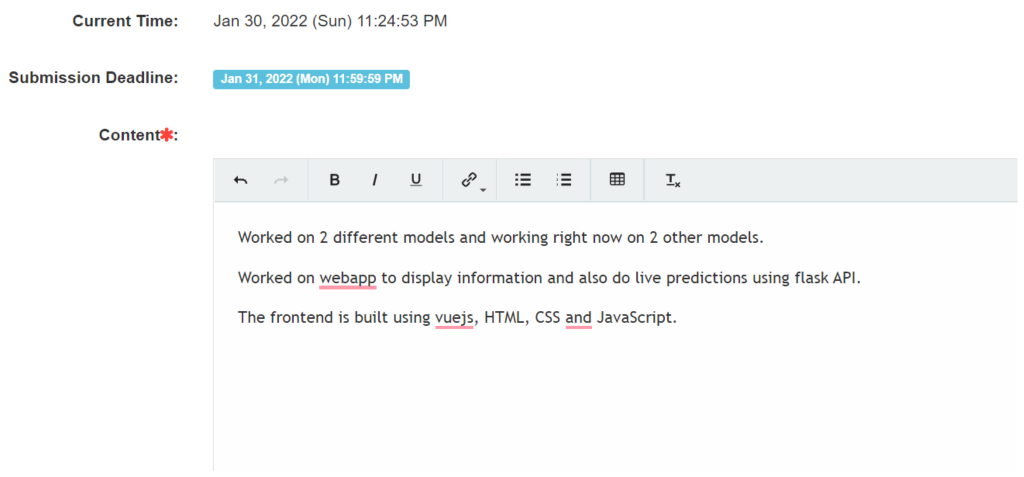


Current Time:	Jan 30, 2022 (Sun) 11:25:13 PM
Submission Deadline:	Nov 30, 2021 (Tue) 11:59:59 PM
Date Last Edited:	Dec 2, 2021 (Thu) 6:51 PM
Content✱:	

I've worked on finalizing the companies to work with.

Extracted their data (stock price).

Researched and learnt about mongodb and react js.


Current Time: Jan 30, 2022 (Sun) 11:24:53 PM

Submission Deadline: Jan 31, 2022 (Mon) 11:59:59 PM

Content*:

Worked on 2 different models and working right now on 2 other models.

Worked on webapp to display information and also do live predictions using flask API.

The frontend is built using vuejs, HTML, CSS and JavaScript.

7. References

1. Avenga. (2021, October 12). *Natural language processing: Tasks and application areas*. Avenga. Retrieved October 24, 2021, from https://www.avenga.com/magazine/natural-language-processing-application-areas/.
2. Chouksey, S. (2019). *Stock price prediction using time series models - DBS esource home*. Stock price prediction using time series models. Retrieved October 24, 2021, from https://esource.dbs.ie/bitstream/handle/10788/3830/msc_chouksey_s_2019.pdf?sequence=1&isAllowed=y.
3. Chouksey, S. (2019). *Stock price prediction using time series models - DBS esource home*. Stock price prediction using time series models. Retrieved October 24, 2021, from https://esource.dbs.ie/bitstream/handle/10788/3830/msc_chouksey_s_2019.pdf?sequence=1&isAllowed=y.
4. Fitzgerald, M. (2021, August 18). *A large chunk of the retail investing crowd started during the pandemic, Schwab survey shows*. CNBC. https://www.cnbc.com/2021/04/08/a-large-chunk-of-the-retail-investing-crowd-got-their-start-during-the-pandemic-schwab-survey-shows.html
5. Hoare, T. (2019, July 11). *Stock price prediction using time series models*. Dublin Business School. https://esource.dbs.ie/handle/10788/3830
6. IBM Cloud Education. (n.d.). *What is natural language processing?* IBM. Retrieved October 24, 2021, from https://www.ibm.com/cloud/learn/natural-language-processing.

7. Kuepper, J. (2021, September 13). *The importance of backtesting trading strategies*. Investopedia. Retrieved October 24, 2021, from https://www.investopedia.com/articles/trading/05/030205.asp.

8. Mehtab, S. (2019, December 9). *A Robust Predictive Model for Stock Price Prediction Using Deep. . .* ArXiv.Org. https://arxiv.org/abs/1912.07700

9. Mehtab, Sidra & Sen, Jaydip. (2020). A Time Series Analysis-Based Stock Price Prediction Framework Using Machine Learning and Deep Learning Models.

10. Prabhakaran, S. (2021, October 8). *Time series analysis in python - A comprehensive guide with examples - ML+*. Machine Learning Plus. Retrieved October 24, 2021, from https://www.machinelearningplus.com/time-series/time-series-analysis-python/.

11. Polikar, R. (n.d.). Ensemble learning. Scholarpedia. Retrieved October 24, 2021, from http://www.scholarpedia.org/article/Ensemble_learning.

12. Shen, J. (2020, August 28). *Short-term stock market price trend prediction using a comprehensive deep learning system*. Journal of Big Data. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00333-6