Market Behavior Prediction via Open Domain Tweets

Kehao Guo kg2937
Tushar Gupta tg2749
Hitesh Agarwal ha2598

Abstract

In the age of social media, financial markets today are heavily influenced by public opinion. Digital media has exponentially increased the velocity of information flow, impact of which can be seen directly and in-directly on stock-prices and valuations of tech-giants. In the project, we aim to study the impact of signals like tweets on financial markets and build an automated system for aiding market decisions for investors.

1. Introduction

Big Data analytics plays a huge role in today's day and age due to the volume, velocity and variety of information generated each second. We aim to utilize tools specifically designed for Big Data analysis such as Pyspark, BiqQuery and Airflow to build a scalable data infrastructure & Flask for application development. For stock price prediction, we plan to experiment with various machine learning models to recommend the probable Close price for a company based on a variety of indicators. To go beyond, the traditional models readily available on the internet we aim to incorporate data from multiple sources like Tweets and reddit as most of the systems rely solely on one of the input methods. Other improvements include, an interactive web-application which loads information about the company taking in data from multiple sources for easy visualization.

2. Related work

Stock price prediction is an active research area in recent years. Better predictions over the stock prices lead to better returns in stocks. Therefore, significant efforts are put to build efficient and robust prediction models that can predict the future trend of a specific stock or the overall market. Research studies over the same topic, as well as events in the recent years provides evidences of a strong relationship between the new articles or tweets from top influencers about a company and its stock prices fluctuations. Following is discussion on previous researches on how future stock price can be predicted using historical data and sentiment on the news or twitter data about a particular company.

The interest in this topic was initiated by Bollen et al [1] in their research in 2010 where they attempted to predict the behaviour of the stock market by utilizing the mood of people on Twitter. The authors consolidated the tweet data of all twitter users in 2008 and used the OpinionFinder and Google Profile of Mood States (GPOMS) algorithm to classify public sentiment into 6 categories, namely, Calm, Alert, Sure, Vital, Kind and Happy. They verified and cross validated the resulting mood time series by comparing the ability of their model to detect the public's response to the presidential elections and Thanksgiving Day in 2008. Their results show a remarkable accuracy of nearly 87% in predicting the up and down changes in the closing values of Dow Jones Industrial Index (DJIA).

Mittal and Goel in their research [2] use twitter data to predict public mood and use the predicted mood and previous days' DJIA values to predict the stock market movements. They also propose a new cross validation method for financial data and obtain 75.56% accuracy using Self Organizing Fuzzy Neural Networks (SOFNN) on the Twitter feeds and DJIA values.

In the research [3], Nagar and Hahsler experimented an automated text mining based approach to aggregate news stories from various sources and create a News Corpus. The Corpus is filtered down to relevant sentences and analyzed using NLP. The count of positive and negative polarity words is used to generate a sentiment metric,NewsSentiment, which is used as a measure of the sentiment of the overall news corpus. Moreover, they have used various open source packages and tools to develop the news collection and aggregation engine as well as the sentiment evaluation engine.

In [4], Joshi et al created three different classification models (RF, SVM and Naïve Bayes) which depict the polarity of news articles being positive or negative, and then predict the price of the stock based on the news sentiment score.

3. Data

Our team has worked in parallel on the following 2 tasks. In the first one, we have focused on collecting data for multiple tech companies and storing it in a more processed format for easy model application. While in the second one, we have worked to build a browser based application.

3.1. Stock Price Data Collection: We have currently collected data for the top-5 tech giants in US as these

generate a huge amount of information on social media because of their popularity in the masses. Specifically, these are

- Apple (APPL)
- Google (GOOG)
- Microsoft (MSFT)
- Tesla (TSLA)
- Amazon (AMZN)

We recently changed our data source to the Yahoo Finance API same as the one mentioned in the HW4 due to the variety of options it provides. For collecting the data we use the max period parameter with a **1d** granularity to fetch all possible data.
Statistics on the data and stock schema can be found below,

Company Name	Data Range	Num Rows
Microsoft	Mar 1986 – YTD	9007
Apple	Dec 1980 – YTD	10333
Google	Aug 2004 - YTD	4355
Tesla	June2010 – YTD	2880
Amazon	May 1997 – YTD	6181

Schema of the data can be found below:

Date	Open	High	Low	Close	Volume
1980-12-12	0.100453	0.100890	0.100453	0.100453	469033600
1980-12-15	0.095649	0.095649	0.095213	0.095213	175884800
1980-12-16	0.088661	0.088661	0.088224	0.088224	105728000
1980-12-17	0.090408	0.090845	0.090408	0.090408	86441600
1980-12-18	0.093029	0.093466	0.093029	0.093029	73449600
...	...	...	...	...	...
2021-11-29	159.369995	161.190002	158.789993	160.240005	88748200
2021-11-30	159.990005	165.520004	159.919998	165.300003	174048100
2021-12-01	167.479996	170.300003	164.529999	164.770004	152052500
2021-12-02	158.740005	164.199997	157.800003	163.759995	136739200

Figure 1

Keeping in mind the cost of cloud resources and the size of the data, our current model creation is being done on local computer. We plan to shift the entire data to BigQuery for the final application.

3.2 Tweet Data Collection

So far, we have collected Tweets for the above-mentioned companies from the Kaggle data set [5]. The dataset, as a part of the paper published in the 2020 IEEE International Conference on Big Data under the 6th Special Session on Intelligent Data Mining track, is created to determine possible speculators and influencers in a stock market. The dataset contains tweets from 2015 to 2020 for each company. It contains over 3 million unique tweets with their information such as tweet id, author of the tweet, postdate, the text body of the tweet, and the number of comments, likes, and retweets of tweets matched with the related company. Tweets are collected from Twitter by a parsing script that is based on Selenium.

3.4 Methods

Using the collected data on stock prices, we wanted to see if it is possible to predict the Close price of the stocks based on simply stock historical information like High, Low, Open, Volume, Close etc., similar to what we do in the Assignment 4. Using the notion, we have performed the following analysis.

3.4.1 Close Price Prediction based on stock information and tweet data

3.4.1.1 Data Pre-Processing

- The following section depicts the best performing statistical models on the company Apple for which we observe the most improvement in predictions when we include tweets.
- Pre-processing includes removing duplicates, dropping NULL values
- We experimented with multiple lag values of 5, 10,20 days from which 10 days give the best possible results.
- Using the past 10 days information a rolling mean is calculated for each column in the data
- The current Close price is taken into a new data frame and shifted by 1 day to concatenate with the rolling mean information. By this, we have current Close price and the corresponding past 10 days rolling mean of stock prices and tweet count with sentiment.
- Data is split into train and test in a 70/30 ratio
- For the analysis, we have only taken past 2-year information into account to remove any ambiguous financial events

3.4.1.2 Statistical Models

The error metrics of all the models can be found in Table 1.

a. **Linear Regression:** Figure 3 below depicts a graph between the real and predicted values. We can see they indeed follow a relationship.

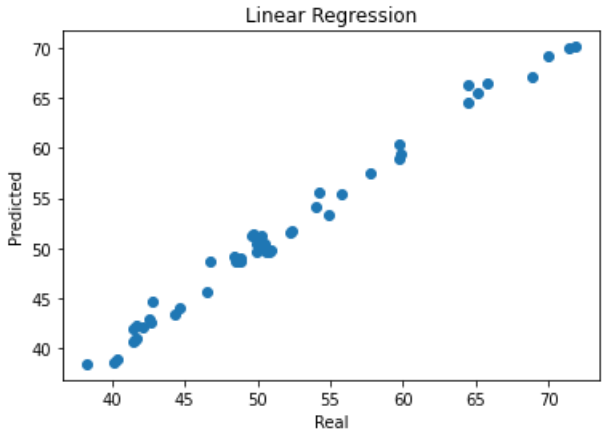


Figure 2: Predictions vs Real LR

b. **Random Forest:** We now move to a more advanced algorithm which is also able to select the most optimum features from the dataset thus helping to remove any kind of overfitting if any. We currently do the analysis in python's sklearn as it contains the library RandomizedSearchCV for parameter search. We vary the number of features from 1, 5 and do a 5-fold cross validation. As a result, the model contains 4 features. The relationship between real and predicted can be seen below:

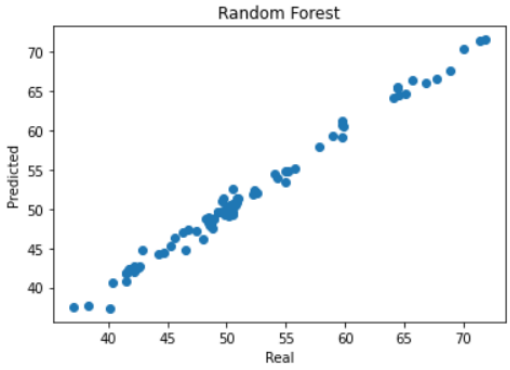


Figure 3: Predictions vs real RF

The model shows an improvement over Linear Regression which was expected.

c. **Gradient Boosted Trees:** Boosted trees are a type of ensemble methods which use of a combination of weak-learners: Decision trees to predict the target variable. They are generally known to perform better than Random forests. The resulting predictions can be visualized below:

We again use the RandomizedSearchCV to determine the best hyperparameters to the model. The best model has 2 features and performs worse that Random Forest models. In the end we also, visualize a plot of the complete set of predicted values against the real stock price using the best performing RandomForest model.

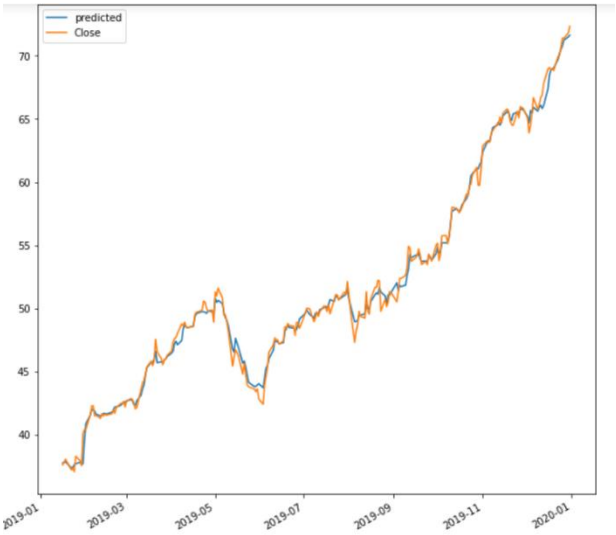


Figure 4: RF model for Apple

3.4.1.3 Deep Learning model (LSTM)

LSTM (Long-Short term memory networks) are recent advancements in Deep learning which are adept at dealing with sequential information and have been used in variety of use cases. We use the same here to check if Close can be predicted using its past values. Moreover, the ability of LSTM model to remember information from the past to generate current predictions add the extra functionality to our tool to predict the trend of prices in the next 5 days.

Data Preprocessing:

- For the task, we first scaled the Close Price using sklearn's MinMaxScaler in the range of (0,1)
- The data used is APPL stock data for its entire period.
- Here, we experimented with a feeding in the past 60 days of information to the current day as LSTM are able to remember information for a longer period of time
- The first 95% information is used for training the LSTM model in keras and the rest for validation

Model:

We use a 2-layer LSTM with 128 and 64 size of hidden units. After that we add a full-connected layer of size 25 which finally gives a single output. Mean Squared error is used as a metric to improve and adam optimizer to optimize.

In 1 epoch the loss decreases to 9*(10^-5)

Figure 6 in Appendix shows the predictions along with the validation set.

4. Experiments

We have now seen the performance of LSTM as well as the stock price prediction utilizing tweet data for Apple. We now present the complete set of results for mean-absolute error as obtained from the experiments on Amazon, Microsoft, Tesla and Google.

Model	**AAPL**	**GOOG**	**MSFT**	**TSLA**	**AMZN**
Linear Regression	0.8	16.65	1.39	1.69	24.5
Random Forest	0.54	16.11	1.20	1.92	23.7
Gradient Boosted Trees	0.56	17.53	1.35	1.89	21.9
LSTM	0.43	15.04	1.15	1.75	22.04

5. System Overview

The software architecture consists of three major modules: the models, the automotive backend data pipeline, and the interactive application. An overview of the system can be found at Figure x. Python is the programming language used in most stages of the implementation, while some SQL was used during database operations.

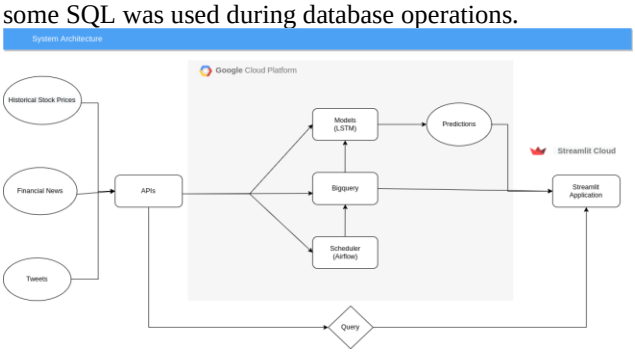


Figure 5: System diagram

The models were built and trained using Python libraries such as Keras TensorFlow and scikit-learn. The backend involves two parts: data fetching and data storage. These functionalities are achieved by Twitter API, Yahoo Finance API (yfinance) [**7**], Airflow [8] and BigQuery [**9**]. Twitter API and Yahoo Finance API are the data sources of stock data and tweets data, respectively. Deployed on a virtual machine on Google Cloud Platform (GCP), Airflow is used as a workflow automation tool that uses the APIs to fetch needed data, process them if necessary, make predictions with the trained models, and write to corresponding data tables in BigQuery on GCP. Finally, the application was implemented with Streamlit [**r**], which is highly integrated with python libraries and provides abundant features for rendering dynamic web applications. Our web application is deployed on Streamlit Cloud with a public URL.

The interactive UI of the application can be divided into two parts, user inputs and visualization. An overview of the UI is shown in Figure **6**. On an user's inputs, the application would display actual stock price information and the predicted close prices of the given stock made with different models. It is worth noting that the application would automatically rerun if the user changes inputs to the application, so that the results are always in sync with user inputs.

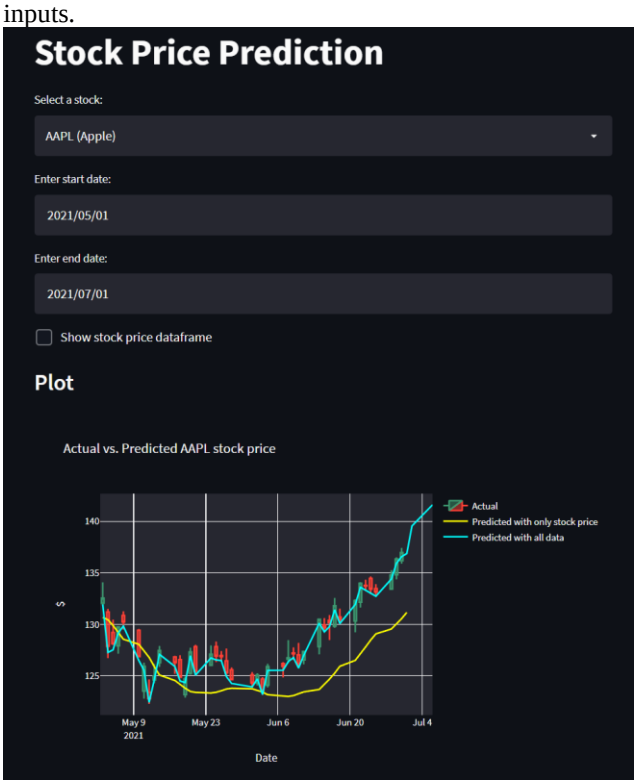


Figure 6: Overview of interactive UI of the web application

The upper half of the UI (Figure **xxx**) is where users select the stock and a range of dates in interest according to the prompts. Users can select one of the five stocks in a drop-down list at a time: Apple (AAPL), Tesla (TSLA), Microsoft (MSFT), Amazon (AMZN), and Google (GOOGL). They also can select a start date and an end date

and historical stock data and predictions made for these dates will show up in the later section. By default, the end date is today's date in real time and the start date is 5 days before it. The users cannot select future dates and can only select a start date up to 5 years ago.

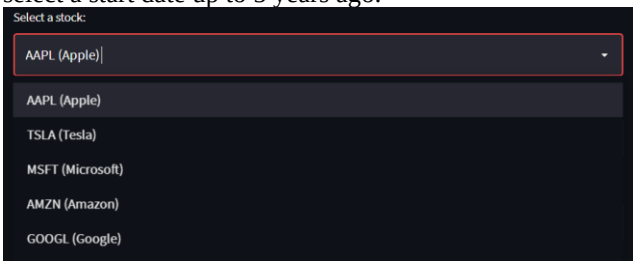
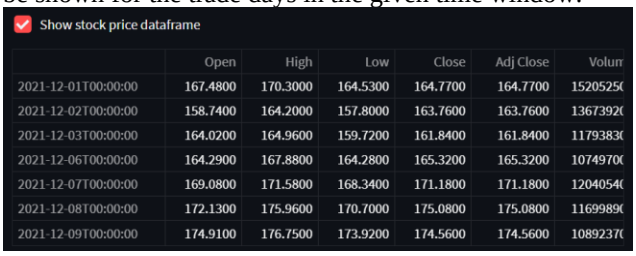


Figure 7: Stock selection

Given user inputs of dates and the stock, the application will fetch the available stock information in the specified time window with yfinance. The stock information is saved in a dataframe and cached by Streamlit. Within 10 minutes, if the data returned by Yahoo Finance API does not change after a refresh or on a new set of user inputs, it is cached in local memory in order to speed up repetitive access to the application. Meanwhile, the application will perform queries on BigQuery tables to fetch predicted close prices using BigQuery's API and libraries such as pandas-gbq. For predictions made with the LSTM model that takes past stock prices and tweets data as inputs, predictions for next 5 days are fetched, whereas only next day's prediction is fetched for the LSTM model that considers just past stock prices. This will be reflected in the plot as well.

With data prepared, the application will then interactively display the actual data and the predictions. First of all, users will have the option to check the actual stock price information according to their queries. By default, this information is hidden. However, if a user chooses to toggle the checkbox (Figure **8**), actual stock price data with features including open price, close price, high price, low price, adjusted close price and volume will be shown for the trade days in the given time window.



	Open	High	Low	Close	Adj Close	Volum
2021-12-01T00:00:00	167.4800	170.3000	164.5300	164.7700	164.7700	1520525(
2021-12-02T00:00:00	158.7400	164.2000	157.8000	163.7600	163.7600	1367392(
2021-12-03T00:00:00	164.0200	164.9600	159.7200	161.8400	161.8400	1179383(
2021-12-06T00:00:00	164.2900	167.8800	164.2800	165.3200	165.3200	1074970(
2021-12-07T00:00:00	169.0800	171.5800	168.3400	171.1800	171.1800	1204054(
2021-12-08T00:00:00	172.1300	175.9600	170.7000	175.0800	175.0800	1169989(
2021-12-09T00:00:00	174.9100	176.7500	173.9200	174.5600	174.5600	1089237(

Figure 8: Displaying historic stock price data

At last, the actual stock prices and the predicted ones are plotted in the same graph (Figure **9**) made with Plotly. Actual stock prices are plotted in the form of a candlestick chart, whereas the predicted prices are plotted in yellow and blue lines according to the legend. The UI consists of a variety of interactive functions that enhance users' visualization experience when comparing the stock prices. For instance, users can set the plot to fullscreen by toggling a button in the upper right corner. If the mouse is placed on the line or the candlesticks in the plot, textual price labels will be displayed for more clarity. Users can also choose to toggle on the legend to show or hide a set of predicted close prices or actual prices to enable better comparison. Moreover, functions such as zoom in/out, download and scaling are supported as well.

Figure 9: Interactive plot of actual vs. predicted stock prices

The major limitation of the application is that it provides predictions for only 5 pre-selected stocks. These stocks may not be of interest to some users, thus the potential user's population of the application is limited. There are two ways of potential improvement. We can provide predictions for a wider range of stocks (e.g., those in S&P 500) scaling up and scaling out the system. Due to significantly higher volume of data, we would need more hardware resources on GCP to store predictions and perform computation, leading to a higher cost. We can also scale up by using frameworks such as Spark or Dask and introduce techniques such as parallel processing to speed up operations such as data extraction. On the other hand, another solution to rectify the problem is to make predictions on any stock given by the users in real time. While bringing more flexibility, this method reduces use of storage but would result in significantly longer wait time due to real time computation and can potentially harm user experience.

Moreover, users may experience a few delays when accessing the current application. The reason is that the query results returned by BigQuery are not being cached. Streamlit has a strict requirement on the format of data to be cached and it would be difficult to convert data to required format. A solution to convert data to the dict-like format would provide users a more fluent experience, especially after the first access to the application.

6. Conclusion

In the end, we can say that not all companies benefit equally via the tweets data, as being an open-domain

source of information, it contains a lot of variability. Apple and Microsoft benefit the most from tweet sentiments which can be attributed to cleaner information available for product-based companies. On the other hand the deep learning approach involving LSTM performs best in all domains and is correctly able to capture the price change trends of the data.

7. References

[1] J. Bollen and H. Mao. Twitter mood as a stock market predictor. IEEE Computer, 44(10):91–94 https://doi.org/10.1016/j.jocs.2010.12.007

[2] A. Mittal and A. Goel Stock Prediction Using Twitter Sentiment Analysis - https://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf

[3] Anurag Nagar, Michael Hahsler, Using Text and Data Mining Techniques to extract Stock Market Sentiment from Live News Streams, IPCSIT vol. XX (2012) IACSIT Press, Singapore

[4] Stock Trend Prediction using news sentiment analysis – Joshi et al. https://arxiv.org/pdf/1607.01958.pdf

[5] Tweets about the Top Companies from 2015 to 2020 h https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020?select=Company_Tweet.csv

[6] Github repo: https://github.com/tushargupta14/StockPricePredictionBigData

[7] https://pypi.org/project/yfinance/

[8] https://airflow.apache.org/

[9] https://cloud.google.com/bigquery

8. Project Contribution

Name	Contribution
Tushar Gupta(tg2749)	LSTM model and tweet data extraction
Hitesh Agarwal (ha2598)	Statistical model experiments
Kehao Guo (kg2937)	Web application development

9. Appendix

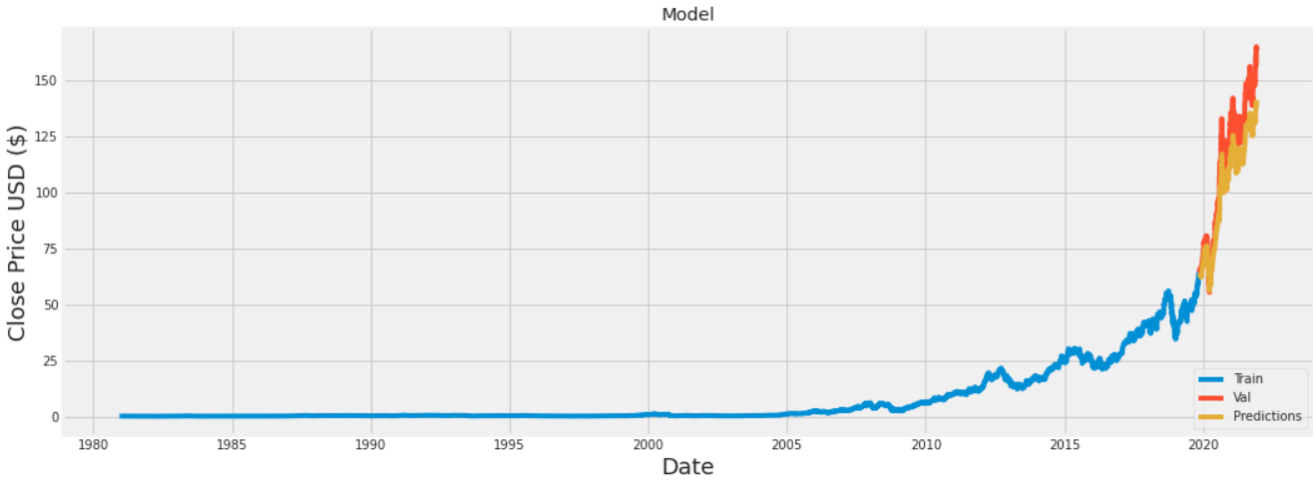


Figure10: LSTM predictions on Apple

Figure 11: Future trend prediction on stock price using LSTM