AN INTELLIGENT AND SOCIAL-ORIENTED SENTIMENT ANALYTICAL MODEL FOR STOCK MARKET PREDICTION USING MACHINE LEARNING AND BIG DATA ANALYSIS

Muqing Bai[1] and Yu Sun[2]

[1]Brookfield Academy, 3215 N Brookfield Rd, Brookfield, WI 53045
[2]California State Polytechnic University, Pomona,
CA, 91768, Irvine, CA 92620

ABSTRACT

In an era of machine learning, many fields outside of computer science have implemented machine learning as a tool [5]. In the financial world, a variety of machine learning models are used to predict the future prices of a stock in order to optimize profit. This paper preposes a stock prediction algorithm that focuses on the correlation between the price of a stock and its public sentiments shown on social media [6].We trained different machine learning algorithms to find the best model at predicting stock prices given its sentiment. And for the public to access this model, a web-based server and a mobile application is created. We used Thunkable, a powerful no code platform, to produce our mobile application [7]. It allows anyone to check the predictions of stocks, helping people with their investment decisions.

KEYWORDS

Stock, Machine learning, Thunkable.

1. INTRODUCTION

Stock market plays an important role in the modern world [8]. It facilitates and efficiently allocates resources to the industries, or more specifically, companies who show promising outlook. There is no doubt that one can make a lot of money by investing in the stock market, which incentivize many to study the stock market, in hope of finding a better investment strategy [9]. Our application uses machine learning to figure out a stock's trend and then make a prediction on its price. The system can be run automatically, in comparison to analyzing a stock manually. This method is more advanced, efficient, and is friendly to the common users.

Many research papers have discussed the different factors that can affect a stock's prices [3], and the different methods to approach stock prediction [1]. Some focus on creating a more accurate sentiment analysis model. They implemented a "novel sentiment index", which weights each stock review differently and takes in consideration of many other potential effects on a stock's price [4]. However, these sources of data might not be good representations of the public, since the weight of each review is not equal.

As previous studies suggest, taking social sentiment into consideration can help improve stock predictions [3]. Our project follows the path of those who used sentiment analysis to predict stock

David C. Wyld et al. (Eds): AI, AIMLNET, BIOS, BINLP, CSTY, MaVaS, SIGI - 2022
pp. 213-223, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121819

prices. The difference is that we do sentiment analysis on any user's tweet from Twitter.com. We believe that Twitter.com is able to offer more globalized opinions of everyday investors. We then use TextBlob, a Natural Language Processing library, to analyze the tweets [10]. The number of tweets who held positive, negative, or neutral outlook on the stock is recorded respectively in a realtime database. We then put the data into a machine learning model, which is hosted on a web-server, to get a prediction. The web server allowed us to make something new — a simple, easy to use mobile application that is designed for everyday investors. It communicates with the web server and displays the prediction of a stock's price change when its ticker symbol is entered into the application by the user. Our application aims to help users make their investment decisions.

For the above-mentioned machine learning model, specifically, we are using Scikit-learn's Linear Regression model. We will prove to you that this model performs better than polynomial regression and Bayesian regression on a dummy dataset. And we will show you the relevance of each factor in the machine learning process.

The rest of the paper is structured as follows: Section 2 list out the challenges we encountered while doing this research; Section 3 focuses on the details of our solutions corresponding to the challenges that are mentioned in Section 2; Section 4 explains the experiments we did, which also acts as a supplement to Section 3 as it details part of the solution to the aforementioned challenges; Section 5 presents related works; Section 6 provide the conclusion and end this paper with future works.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Performing Large-Scale Data Collection and Sentiment Analysis

In order to build and train an accurate machine learning algorithm, a large amount of data is needed, since if we are only taking the comments from a couple people regarding the stock, the overall sentiment can be highly biased because each person has a great weight to the overall sentiment [11]. And on top of that, we need to be able to consistently extract data, since the project is aimed to be updating on a daily basis.

When the data is gathered, it will then be analyzed using a Natural Language Processing algorithm. The difficult part is to figure out which algorithm to use. Since our data is specific to stock trading, it has special terminologies and phrases such as "AAPL to the moon!!!" These may be difficult to analyze.

2.2. Optimizing the Prediction Model with High Accuracy

We were in face of a big problem — which machine learning model should we use to make the prediction? Also, which variables are actually relevant to a stock? Is it the public sentiment? The price? or other variables?

We conducted a total of three experiments to figure out the aforementioned questions. Experiment 1 compares the result of three different types of machine learning model. Experiment 2 compares the result of 3 different configurations of the polynomial-regression. Experiment 3 compares the accuracy of the linear-regression model with different categories of input data cut off. These experiments will be explained in detail in Section 4.

2.3. Making a User-Friendly Interface to Display the Data

The goal of our project is to help casual investors invest in the stock market. So naturally, there has to be a way that the end-users can access our prediction results of the stocks. Thus we created a mobile application. We aim to have a simplistic design and a user-friendly interface for the application. This part is further discussed in the Solution, Section 3.2.4.

3. SOLUTION

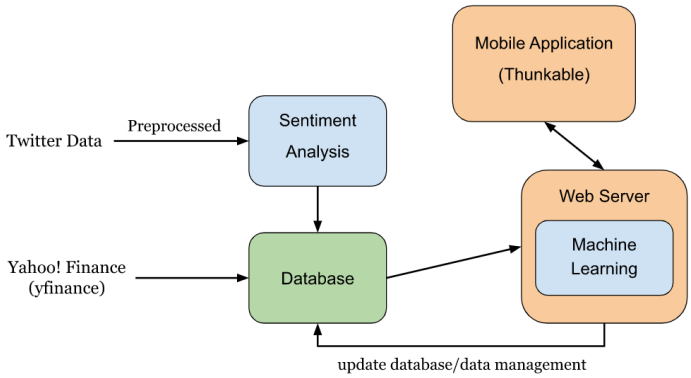

Figure 1. Overview of the solution

3.1. Web Scraping

In our case, Twitter.com is used to gather social sentiments. With Twitter's API, we can obtain the most recent 100 tweets about a particular stock. As shown in Figure 2, after authenticating using Twitter's API, the program passes the company's name or the company's ticker symbol as the keyword (query) to search for 100 matching tweets using the .search_recent_tweets() method.

```
38  # Twitter API Initialization
39  client = tweepy.Client(bearer_token=config.BEARER_TOKEN)
40  # Query
41  if 'coName' in locals():
42    response = client.search_recent_tweets(query=coName, max_results=100)
43  else:
44    response = client.search_recent_tweets(query=tickerS, max_results=100)
45
```

Figure 2. Screenshot of code 1

Alongside the tweets, the information about the stock, such as the market-closing price and the date, as well as the company's full name, are all collected through Yahoo! Finance's API. These data are stored in the database, which will be discussed in section 3.2.3.

3.2. Sentiment Analysis

After gathering the tweets, we need to analyze and distinguish them into 3 categories based on their polarity: positive, negative, and neutral. In order to achieve this, we need a Natural Language Processing (NLP) algorithm. Initially, we set up a model using TensorFlow, the configuration is as follows.

```
[ ] # building the neural nw model
    model = tf.keras.Sequential([
            tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_len),
            tf.keras.layers.GlobalAveragePooling1D(),
            tf.keras.layers.Dense(24, activation='relu'),
            tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
```

Figure 3. Screenshot of code 2

```
model.summary()

Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_16 (Embedding)    (None, 100, 256)          2560000

 global_average_pooling1d_16  (None, 256)              0
  (GlobalAveragePooling1D)

 dense_32 (Dense)            (None, 24)                6168

 dense_33 (Dense)            (None, 1)                 25

=================================================================
Total params: 2,566,193
Trainable params: 2,566,193
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Screenshot of model summary

However, it performed poorly with merely 26% accuracy [12]. But do note that the dataset itself may not be accurate.

```
Epoch 26/50
135/135 - 4s - loss: -7.9707e+02 - accuracy: 0.2636 - val_loss: -1.3482e+02 - val_accuracy: 0.1993 - 4s/epoch - 30ms/step
Epoch 27/50
135/135 - 4s - loss: -8.8439e+02 - accuracy: 0.2657 - val_loss: -1.5342e+02 - val_accuracy: 0.2067 - 4s/epoch - 30ms/step
Epoch 28/50
135/135 - 4s - loss: -9.7684e+02 - accuracy: 0.2606 - val_loss: -1.6950e+02 - val_accuracy: 0.2073 - 4s/epoch - 30ms/step
Epoch 29/50
135/135 - 4s - loss: -1.0745e+03 - accuracy: 0.2674 - val_loss: -1.7475e+02 - val_accuracy: 0.1920 - 4s/epoch - 31ms/step
Epoch 30/50
135/135 - 4s - loss: -1.1769e+03 - accuracy: 0.2636 - val_loss: -1.9569e+02 - val_accuracy: 0.1973 - 4s/epoch - 30ms/step
Epoch 31/50
135/135 - 4s - loss: -1.2862e+03 - accuracy: 0.2594 - val_loss: -2.2550e+02 - val_accuracy: 0.2120 - 4s/epoch - 30ms/step
Epoch 32/50
135/135 - 4s - loss: -1.3975e+03 - accuracy: 0.2713 - val_loss: -2.3868e+02 - val_accuracy: 0.2053 - 4s/epoch - 31ms/step
Epoch 33/50
135/135 - 4s - loss: -1.5166e+03 - accuracy: 0.2597 - val_loss: -2.6920e+02 - val_accuracy: 0.2160 - 4s/epoch - 30ms/step
Epoch 34/50
135/135 - 4s - loss: -1.6378e+03 - accuracy: 0.2751 - val_loss: -2.7285e+02 - val_accuracy: 0.1987 - 4s/epoch - 30ms/step
Epoch 35/50
135/135 - 4s - loss: -1.7668e+03 - accuracy: 0.2662 - val_loss: -2.7654e+02 - val_accuracy: 0.1907 - 4s/epoch - 30ms/step
Epoch 36/50
135/135 - 4s - loss: -1.9017e+03 - accuracy: 0.2562 - val_loss: -3.1320e+02 - val_accuracy: 0.1993 - 4s/epoch - 30ms/step
Epoch 37/50
135/135 - 4s - loss: -2.0404e+03 - accuracy: 0.2702 - val_loss: -3.4411e+02 - val_accuracy: 0.2060 - 4s/epoch - 30ms/step
Epoch 38/50
135/135 - 4s - loss: -2.1822e+03 - accuracy: 0.2697 - val_loss: -3.6301e+02 - val_accuracy: 0.2020 - 4s/epoch - 30ms/step
Epoch 39/50
135/135 - 5s - loss: -2.3356e+03 - accuracy: 0.2613 - val_loss: -3.5802e+02 - val_accuracy: 0.1847 - 5s/epoch - 35ms/step
Epoch 40/50
135/135 - 4s - loss: -2.4907e+03 - accuracy: 0.2615 - val_loss: -3.9627e+02 - val_accuracy: 0.1920 - 4s/epoch - 30ms/step
Epoch 41/50
135/135 - 4s - loss: -2.6543e+03 - accuracy: 0.2699 - val_loss: -4.5797e+02 - val_accuracy: 0.2140 - 4s/epoch - 30ms/step
Epoch 42/50
135/135 - 4s - loss: -2.8206e+03 - accuracy: 0.2710 - val_loss: -4.4996e+02 - val_accuracy: 0.1920 - 4s/epoch - 30ms/step
Epoch 43/50
135/135 - 4s - loss: -2.9959e+03 - accuracy: 0.2683 - val_loss: -4.7724e+02 - val_accuracy: 0.1933 - 4s/epoch - 30ms/step
Epoch 44/50
135/135 - 4s - loss: -3.1782e+03 - accuracy: 0.2615 - val_loss: -5.4207e+02 - val_accuracy: 0.2107 - 4s/epoch - 30ms/step
Epoch 45/50
135/135 - 4s - loss: -3.3650e+03 - accuracy: 0.2669 - val_loss: -5.5043e+02 - val_accuracy: 0.2033 - 4s/epoch - 30ms/step
Epoch 46/50
135/135 - 4s - loss: -3.5617e+03 - accuracy: 0.2622 - val_loss: -5.8427e+02 - val_accuracy: 0.2027 - 4s/epoch - 30ms/step
Epoch 47/50
135/135 - 4s - loss: -3.7669e+03 - accuracy: 0.2583 - val_loss: -5.8690e+02 - val_accuracy: 0.1907 - 4s/epoch - 30ms/step
Epoch 48/50
135/135 - 4s - loss: -3.9742e+03 - accuracy: 0.2709 - val_loss: -5.9067e+02 - val_accuracy: 0.1780 - 4s/epoch - 33ms/step
Epoch 49/50
135/135 - 4s - loss: -4.1849e+03 - accuracy: 0.2555 - val_loss: -6.7279e+02 - val_accuracy: 0.2007 - 4s/epoch - 30ms/step
Epoch 50/50
135/135 - 4s - loss: -4.4108e+03 - accuracy: 0.2606 - val_loss: -7.3938e+02 - val_accuracy: 0.2107 - 4s/epoch - 30ms/step
```

Figure 5. Result accuracy

Then we tried using a RoBERTa model that was pre trained on 124 million tweets from January 2018 to December 2021. After testing it on a dataset of 1300 stock related tweets, the resulting accuracy is better, at about 55%. But still not a favorable result.

```
0.546923076923077
{'Positive': 317, 'Neutral': 846, 'Negative': 137}
```

Figure 6. Results from the RoBERTA model

Eventually, we choose the more accessible and easier to use TextBlob as the NLP model. It is also pretrained. We pre-process the text for better results, which removes “RT”, words that include “@” or “#”, “https” hyper links, and any punctuation. This text is then fed to Textblob for sentiment analysis and returns a BaseBlob object. BaseBlob.polarity then returns a float between -1.0 and 1.0, where -1.0 is very negative, 1.0 is very positive, and 0.0 is defined as neutral. These data are then sent and stored in the database, ready to be accessed.

3.3. Database and Machine Learning

As mentioned in the previous sections, the backbone of this project, the database, stores a stock’s prices and its sentiment analysis by date [13]. It has a data structure where the information about the stock is arranged by their date and is listed as subsets of that stock, as shown in Figure 7. The dataset can store multiple entries of stocks at the same time. It communicates with the web server frequently.

Figure 7. Screenshot of database

Before doing machine learning, we first request data from the database. A machine learning model is created for each individual stock. So we extract the data under that specific stock’s ticker name. The data we use as inputs are the dates and the 3 types of sentiments: positive, negative, and neutral. They are put into a list, which is then put into the 2D array called input_data (Figure 7). input_data contains all the data from previous days except for today’s, because today’s data doesn’t have a corresponding output. We will be using today’s data as the input for our prediction once the model is trained appropriately. The output_data, however, has only one category, which is the change in price of the stock in the next day. Essentially, it takes the price of the corresponding date and the next price value on the list to find the change in price from the day where the sentiment values are recorded to the next day. This combo of input_data and output_data is now our training data set for the linear regression machine learning model. We are using linear regression because it is the best performing model, proven in previous experiments (Section 4). After training using the method model.fit(), we can now predict the change in price between today and tomorrow. As shown in Figure 8, model.predict() produces the prediction, which is then turned into a percentage and bundled with ticker and name in the dictionary named final_res. final_res will be finally returned at the end of this method.

```
71  #
72  model = linear_model.LinearRegression()
73  model.fit(input_data, output_data)
74  res = model.predict([today_data])#[0]
75  print(res)
76
77  # add stock info to join prediction
78  final_res = {
79    "prediction" : str(round(res[0] * 100, 2)) + "%",
80    "ticker" : ticker,
81    "name" : name
82  }
```

Figure 8. Screenshot of code 3

3.4. Web Server and Mobile App

A simple Flask server is setted up and open for access. It facilitates the database and hosts the machine learning models. In Figure 9 is the setup of the web server. Its first method *stockUpdate* simply updates all the stocks that are already recorded in our database. It scrapes Twitter to find the new sentiment values, along with the price of the stock at the moment of update. The second method, *stockUpdateWithList* updates the stocks that are on a specific list. And if they were never recorded in the database, then they will be added to it. Method *stockInfo* takes in a parameter, the ticker symbol of a stock, in the form of a https hyperlink. It goes through the process of machine learning and predicting — a dictionary structured like *result* from Figure 9 is returned.

```
1  import random
2  from flask import Flask
3  import tweet
4  import data_processor
5  import json
6
7  app = Flask(__name__)
8
9
10 @app.route("/stockUpdate")
11 def stockUpdate():
12   data_processor.updateAllStocks()
13   return "Updated"
14
15 @app.route("/stockUpdateWithList")
16 def stockUpdateWithList():
17   data_processor.updateAllStocksWithList()
18   return "Updated with List"
19
20 @app.route("/stockInfo/<ticker>")
21 def stockInfo(ticker):
22   ticker = ticker.upper()
23   if ticker not in all_stocks:
24     result = {
25       "prediction" : "N/A",
26       "ticker" : ticker,
27       "name" : "Stock Data N/A"
28     }
29   else:
30     result = data_processor.getStockData(ticker)
31
32   return json.dumps(result)
33
34 stock_list_file = open("stock_list.txt")
35 all_stocks = []
36 for line in stock_list_file.readlines():
37   all_stocks.append(line.strip())
38
39 app.run(host = '0.0.0.0')
```

Figure 9. Screenshot of code 4

The web server allowed us to make a mobile application. It is targeted for people who don't know anything about coding or even investing. Thus, the app uses a very simplistic user interface.

This app is created with Thunkable. As shown in Figure 10, the panel on the left is our first screen, where the user types in a ticker symbol and presses the Search button, it navigates to the 2nd panel and sends a get request to our web server to get the information regarding this stock, including the predictions. After the web server responds, the information will be displayed on panel 2 in the order of *prediction*, *name*, and *lastly ticker*.

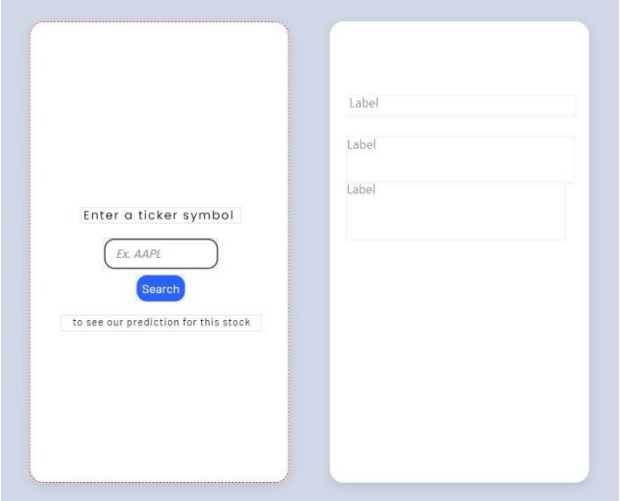

Figure 10. Home screen

Do note that Thunkable is a non coding app creation platform. They use logic statements to substitute for coding. Figure 11 shows the "code" for the first panel.

Figure 11. Screenshot of Thunkable

4. EXPERIMENT

4.1. Experiment 1

Certain aspects of machine learning can significantly improve its performance/accuracy if the right module is used. In the following experiments, we will look at the different factors that contribute to the accuracy of a machine learning module, and figure out which has the best performances.

All the following tests involve data that are artificially manipulated. Since we couldn't gather a big enough dataset of the real world twitter users' sentiments on certain stocks, we are using a dummy dataset to train the algorithms; and ultimately test each algorithm on their accuracy. The dataset consists of 2 categories: input data and output data. Input data includes Positive Sentiment, Negative Sentiment, Neutral Sentiment, Month, and Day. Positive Sentiment, Negative Sentiment, and Neutral Sentiment each takes a randomly generated integer between 0 and 50, while Month takes an integer between 1 and 12, and Day takes an integer between 1 and 30. The output data, however, is a single number that is the sum of Positive Sentiment, Negative Sentiment, and

Neutral Sentiment plus a randomly generated integer between -15 and 15. (should I explain why the output data doesn't take in Day and Month as its factors?)

We are using a relatively small batch of 100 generated samples of dummy data for each trial in the following experiments (add test results from 1000 data samples later)

The first experiment compares the accuracy between 3 different machine learning models: linear regression, polynomial regression, and Bayesian regression. As shown in Figure 12, polynomial regression (in green) slightly under-performed compared to linear regression and Bayesian regression. Do note that linear regression and Bayesian regression have very identical accuracy in each of the 20 trials, although they are not exactly the same.

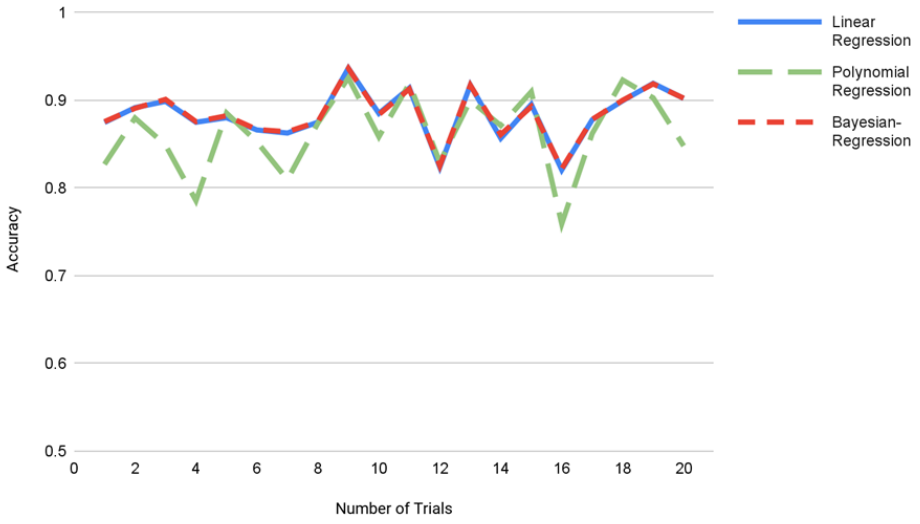

Figure 12. Graph of accuracy vs number of trials

However, the difference in accuracy between these 3 will turn to be neglectable when a bigger sample size is used.

4.2. Experiment 2

The second experiment focuses on polynomial regression. We want to find out which of its configurations produces the most favorable result. To give a short explanation, the *sklearn*polynomial regression takes in a Degree parameter and "generates a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree."

In Figure 13, we can see that *Degree 2* is performing significantly better than the other two, considering that the best accuracy we can get is 1; however it can go far down into the negatives. The farther it is into the negatives, the less accurate the model is. To better display all the values in one graph, we dropped out the result from Trial 15 of *Degree 4*, which came in at an astonishing -399.005058028454 accuracy rating. We can see that *Degree 3* and *Degree 4*'s accuracy not only underperform, but also fluctuate a lot throughout the trials, especially for *Degree 4*. Both of their results came out as unfavorable, staying below 0 for the most part. *Degree 2*'s performance is similar to the polynomial regression model in experiment 1, since that is also using degree 2, which is the default configuration for polynomial regression.

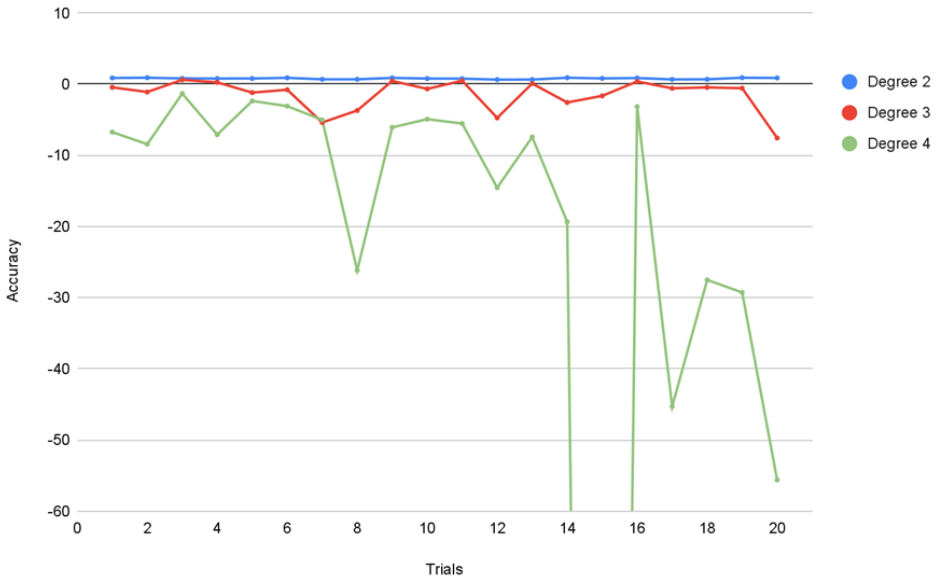

Figure 13. Graph of accuracy vs trials

4.3. Experiment 3

Figure 14 shows the result from experiment 3. In this experiment, we are trying to see what happens when we emit one of the five inputs. We are using a linear regression model for this experiment.

By emitting Month or Day, the accuracy stays virtually the same. Emit Month yields an average accuracy of 0.8614 from 20 trials, Emit Day yields 0.8595. Both are very close to the average accuracy of linear regression from experiment 1, which turns out to be 0.8836. This means that the output is likely not affected by neither Month or Day, which matches with our process of creating the dummy data. But it also proves that the linear model can function well even when one dimension (is it dimension? or) of the data is missing. But do note that Emit Month and Emit Day fluctuate with a wider range (at about 0.25) than that of the regular linear regression with the full dataset from experiment 1 (at a little more than 0.1). Still, these two results dusted the rest in their accuracy. Below them are the results from Emit Positive Sentiment, Emit Negative Sentiment, and Emit Neutral Sentiment, which are very inconsistent in their accuracy. Notably, Emit Neutral Sentiment displays a wider range of accuracy, although the three types of sentiment have the same weight when creating their matching outputs.

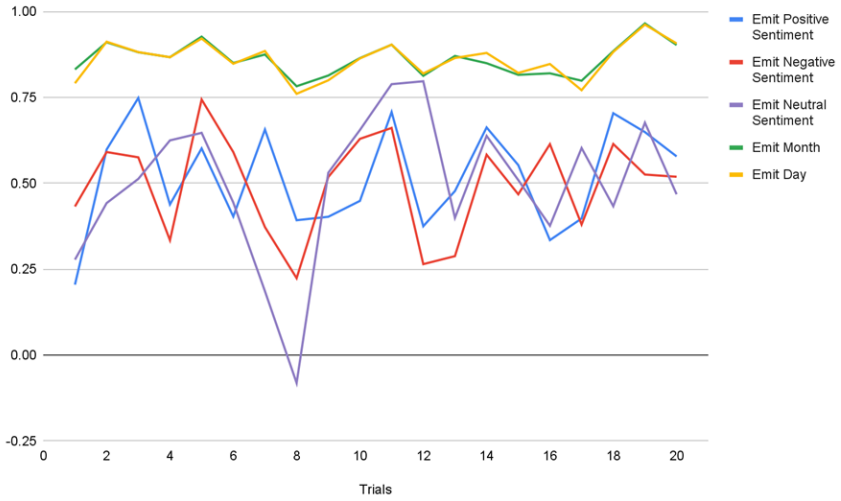

Figure 14. Graph of trials

5. RELATED WORK

Nguyen, T. H., & Shirai, K. [2] introduce a new feature called the TSLDA to the prediction model. The TSLDA model captures the topic and sentiment simultaneously. Their model was able to outperform a prediction model that only uses historical prices by 6.07% in accuracy. However, their work is different from ours in that they "chose SVM with the linear kernel as the prediction model", which only predicts if the stock price goes up or down. In comparison, we used a linear regression model to predict the % change in the stock price.

Qiu, Y., Song, Z., & Chen, Z. [3] introduce a sentiment index that assigns different weights to different sources of sentiments. On top of that, their model takes in consideration stock market anomalies: the day-of-the-week effect and holiday effect. In comparison, our model takes in data as an input variable, which with a large amount of training data, the model can find out about these effects on its own. But do note that our paper doesn't include a system that assigns weights to the sentiments. Every tweet weighs the same.

Porshnev, A., Redkin, I., & Shevchenko, A. [4] analyzed Twitter user's sentiment in a detailed way: they "evaluate presence of eight basic emotions" in each tweet, which is then used to predict DJIA and S&P500 indicators [14]. While their work is focused on predictions on two of the most received stock market indicators, our application allows the public to search up any stock that is available in Yahoo! Finance. On top of that, our method simply analyzes the polarity of the tweets and divides them into 3 different categories: positive sentiment, negative sentiment, and neutral sentiment, instead of analyzing detailed emotions of each tweet like Porshnev and others did in their study.

6. CONCLUSIONS

The stock market is constantly evolving, to which machine learning might be its best suite for making predictions [15]. Among previous approaches that involve machine learning, social sentiment based stock predictions are definitely proven to have increased accuracy on stock price predictions. By experimenting with a dummy dataset, we found the best machine learning model to be linear regression, despite that Bayesian regression produced almost identical results. By

implementing the best performing machine learning model, and with the easy-to-use mobile application, more investors can now receive more accurate stock market predictions.

However, the dummy data that we used is artificially manipulated, which means that we assumed a relationship between the sentiments and stock prices. That is why the accuracy is as high as 0.8836. But due to the policy of Twitter not allowing us to obtain past tweets from any period of time, our actual dataset is too small to produce sufficient outcome. We had to work with the limitation and resulted in dummy data.

But in the future, once we gathered a big enough dataset. We can then perform Experiment 3 again. By cutting out a category of data and seeing how that affected the accuracy of prediction, we can tell how relevant that variable is to the price of the stock.

REFERENCES

[1] Agrawal, J. G., Chourasia, V., & Mittra, A. (2013). State-of-the-art in stock prediction techniques. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(4), 1360-1366.

[2] Nguyen, T. H., & Shirai, K. (2015, July). Topic modeling based sentiment analysis on social media for stock market prediction. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers) (pp. 1354-1364).

[3] Qiu, Y., Song, Z., & Chen, Z. (2022). Short-term stock trends prediction based on sentiment analysis and machine learning. Soft Computing, 26(5), 2209-2224.

[4] Porshnev, A., Redkin, I., & Shevchenko, A. (2013, December). Machine learning in prediction of stock market indicators based on historical data and data from twitter sentiment analysis. In 2013 IEEE 13th International Conference on Data Mining Workshops (pp. 440-444). IEEE.

[5] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.

[6] Hendler, Glenn. Public sentiments: Structures of feeling in Nineteenth-Century American literature. Univ of North Carolina Press, 2001.

[7] Hoehle, Hartmut, and Viswanath Venkatesh. "Mobile application usability." MIS quarterly 39.2 (2015): 435-472.

[8] Aggarwal, Rajesh K., and Guojun Wu. "Stock market manipulations." The Journal of Business 79.4 (2006): 1915-1953.

[9] Gidofalvi, Gyozo, and Charles Elkan. "Using news articles to predict stock price movements." Department of Computer Science and Engineering, University of California, San Diego (2001): 17.

[10] Gujjar, J. Praveen, and Prasanna Kumar HR. "Sentiment analysis: Textblob for decision making." Int. J. Sci. Res. Eng. Trends 7.2 (2021): 1097-1099.

[11] Phan, Xuan-Hieu, Le-Minh Nguyen, and Susumu Horiguchi. "Learning to classify short and sparse text & web with hidden topics from large-scale data collections." Proceedings of the 17th international conference on World Wide Web. 2008.

[12] Redman, Thomas C. "Measuring data accuracy: A framework and review." Information quality. Routledge, 2014. 33-48.

[13] Dubitzky, Werner, et al. "The open international soccer database for machine learning." Machine Learning 108.1 (2019): 9-28.

[14] Porshnev, Alexander, Ilya Redkin, and Alexey Shevchenko. "Machine learning in prediction of stock market indicators based on historical data and data from twitter sentiment analysis." 2013 IEEE 13th International Conference on Data Mining Workshops. IEEE, 2013.

[15] Kourou, Konstantina, et al. "Machine learning applications in cancer prognosis and prediction." Computational and structural biotechnology journal 13 (2015): 8-17.