Stock Price Prediction Using Natural Language Processing and Machine Learning

by

Ahmed Ayman Amer

Bachelor of Science in Electrical Engineering, Arizona State University, 2018
A report submitted in partial fulfillment of the requirements for the degree of
Master of Engineering in the Department of Electrical and Computer Engineering

Stock Price Prediction Using Natural Language Processing and Machine Learning

by

Ahmed Ayman Amer

B.Sc., Arizona State University, 2018

Supervisory Committee

Dr. Fayez Gebali, Supervisor
Department of Electrical and Computer Engineering

Dr. Mohamed Watheq El-Kharashi, Supervisor
Department of Electrical and Computer Engineering

Abstract

Predicting the stock market is an infamous problem that many people have tried to solve. Can real time textual data in the form of tweets be used to predict stock movements? In this project, the use of different natural language processing methods are used to process twitter data to try to find out their sentiment. Furthermore, based on the sentiment, further analysis is done using machine learning techniques to try and predict next day returns for individual stocks. Two and Three different features were used to try and predict the next day's percentage change. The metrics used to assess the methodology were accuracy, precision and cumulative percentage gain or loss using a specific strategy or method. The results of this project suggest that using tweets as input for natural language processing and machine learning can achieve average accuracies and result in strategies that have consistently beaten the market in terms of cumulative returns.

Contents

List of Figures

List of Tables

Glossary

ACC……. Accuracy
BERT…….Bidirectional Encoder Representations from Transformers
KNN……...K-Nearest Neighbors algorithm
ML……. Machine Learning
NASDAQ……. National Association of Securities Dealers Automated Quotations
NLP……...Natural Language Processing
RF……. Random Forests algorithm
SLOT……...Self Supervised Learning of Tweets
TF-IDF……..Term Frequency – Inverse Document Frequency
VADER……. Valence Aware Dictionary for Sentiment Reasoning
YFINANCE……. ..Yahoo Finance

Acknowledgement

I would like to thank God for helping me throughout this process. I also am thankful to my family and friends who have supported my academic growth throughout the past 2 years.

I am also immensely grateful to Dr. Fayez Gebali for accepting me as part of his research group as a master's student. This is something that will change my life forever and for that I am very thankful.

Chapter 1 Introduction

“The term stock market refers to several exchanges in which shares of publicly held companies are bought and sold. Such financial activities are conducted through formal exchanges and via over-the-counter (OTC) marketplaces that operate under a defined set of regulations” [1]. The first ever stock market exchange (Amsterdam Stock Exchange) was created in 1611 [2]. It was only until the late 1700s where the New York Stock Exchange was formed in the USA [2].

Buyers would buy stocks when they thought or hoped that the stock price would go up and sellers would sell if they needed to liquidate their value into cash or when they thought that the stock price would go down. Discussions on what moves a stock price up or down are endless. However, across academic and some professional peers, general consensus has shown that there are two different frameworks for analyzing stock prices, which are known as fundamental and technical analysis. “Fundamental analysis evalues stocks by attempting to measure their intrinsic value” [3]. Fundamental analysts study everything from the overall economy and industry conditions to earnings and expenses reports of the particular company. Meanwhile, technical analysis involves the analysis of statistical trends such as movements in a stock’s price or volume [3]. Technical analysts do not attempt to measure intrinsic value, as they believe the intrinsic value is already reflected in the price, instead, they try to identify patterns in the charts or the numbers.

One technique to predict where a stock price will go is by using market sentiment analysis. Market sentiment refers to the overall consensus that all market participants agree on. Some factors that can affect market sentiment are news headlines and social media. Recent paper published by David Allen, Michael McAleer, and Abhay K.Singh (Daily market news sentiment and stock prices) has shown that “financial news sentiment has significant effects on Dow Jones constituent returns” [4]. In this project, social media sentiment rather than financial news sentiment will be measured using natural language processing techniques and further machine learning techniques will be used to answer the question of “whether social media sentiment can be used to predict stock prices”.

1.1 Motivation

The motivation for predicting a stock's price is simple. It is to make a profit on an investment, or at least not lose any money. It is important to note that the majority of the population of the USA and Canada are middle class everyday working people. Most of which have 401Ks or place their money in a retirement fund. Retirement funds almost exclusively raise the value of the fund by investing in the stock market. "Nearly three-quarters of all 401(k) money is held in stocks according to a vanguard report from 2021" [5]. Therefore, it is not unfounded to say that 70% of the money of the working class people is held up in one way or another in a certain stock. From this perspective, it becomes paramount to be able to make good trading decisions.

1.2 Related Work

Recent papers that are using natural language processing to try to predict stock prices include Alejandro's and Yuehana's [6], Hansen and Kazinnik [7], Xie, Han, Lai, Peng, and Huang (2023), Soun, Yoo, Choo, Jeon, and Kang (2022), and Mehtab and Sen (2021).

Alejandro and Yuehana found out that by using OpenAI's ChatGPT, they were able to process the sentiment of thousands of news headlines by prompting OpenAI's different GPT models. Using the result of the sentiment scores, they were able to develop a trading strategy that outperformed a regular buy and hold strategy by four times over [6]. Hansen and Kazinn in their paper showcase the different natural language processing methods and their varying abilities to decipher (Fed Speak). He finds that the Model GPT-3 has the highest Accuracy and the lowest mean absolute error followed closely by the BERT model. They also test out a variety of different (non-machine learning dictionaries) such as the Loughran and McDonald, the Henry Financial Dictionary, and the NRC Word-Emotion Association Lexicon of Mohammed and Turney which all lag the transformer models in terms of performance [7].

Xie, Han, Lai, Peng, and Huang from the Computer School of Wuhan University have run an abstract experiment where they prompt ChatGPT with a simple prompt based on a dataset that contains social media posts related to the stock and daily stock closing prices [8]. The prompt was "consider the data and tweets. Predict in one word whether the close price movement of the stock will rise or fall at the specified date. Only return Rise or fall". The results of the experiment were underwhelming, as ChatGPT in both zero-shot mode and chain of thought modes both underperformed relative to models that already exist out there such as state of the art method for multimodal stock price movement (SLOT), created by Yoo [9]. That being said, the paper does demonstrate that by using tweets as an extra parameter of input on top of stock closing prices, the accuracy of the predictions is significantly increased. This tells us that use of tweets as an input source for predicting stock price is a worthy task to consider.

Yoo, Soun, Cho, Jeon, Kang [9] have proposed a model known as (SLOT) or for capturing multi-level price trends. In this paper, the use of tweets as a data source is not used to directly

predict stock prices , rather, they are used to “understand the multi-level correlations between stocks in global and local views.” The difference between this model and previous models that use tweets as a data source is this, there is a limitation on the number of stocks that are mentioned in tweets. Most tweets only mention the top stock companies such as Google, Facebook, and Apple, meanwhile “23% of the lowest stocks have only 1% of all tweets [9].” Moreover, the SLOT model looks at tweets available as an indicator to global trends rather than direct correlation to a certain stock or security.

1.3 Report Outline

This report is structured as follows:

Chapter 1 gave a brief introduction of the stock market and its history and motivation for attempting to predict the direction of a stock price. Furthermore, it provided several different works related to the field and their general structure and methodology to try and predict stock prices.

Chapter 2 will give a brief introduction about the different natural language processing methods used in this project and how each NLP method functions. It also gives instructions on how to set up each NLP method in a python environment.

Chapter 3 introduces the different machine learning classifiers used in order to use the features available to try and make a prediction. This chapter also shows how to download and install the relevant libraries in the python environment to use these models.

Chapter 4 introduces the complete stock price prediction system from beginning to end and explains the methodology in detail. It explains each step in the system from data collection to feature formation and selection.

Chapter 5 introduces the performance evaluation which goes extensively over the different combinations of features and the corresponding results in terms of accuracy, precision, and cumulative returns. It also goes through a very relevant metric known as cumulative returns based on a simple strategy that uses the predictions in a simulated environment to see in real time if the machine learning classifiers can outperform the market.

Chapter 6 concludes this project and summarizes the results. It also gives suggestions for possible future work and improvement.

Chapter 2 Natural Language Processing

Natural Language process also frequently referred to as (NLP) is an “interdisciplinary field that lies at the intersection of computational science and artificial intelligence [10]”. NLP makes use of machine learning and deep learning techniques to accomplish tasks such as language translation. Traditional machine learning natural language processing involved the usage of Naive Bayes which was built on top of Bayes’ theorem, and other techniques such as RNNs (Recurrent Neural Networks) and LSTM (Long Short-Term Memory Units). However, RNNs and LSTMs have had a limitation where they are generally not able to understand the context of a sentence.

Modern machine learning techniques (using transformers) have completely revolutionized natural language processing where sentence context is much better understood which leads to better tasks such as language translation and sentiment analysis.

2.1 VADER

VADER (Valence Aware Dictionary and Sentiment Reason) is an NLTK (Natural language toolkit) module that provides sentiment scores based on the words used. “It is a rule-based sentiment analyzer in which the terms are generally labeled as per their semantic orientation as either positive or negative [11]”. VADER does not use machine learning, it’s rule based, meaning it’s based completely on logic (if-then-and) code. VADER is a module in NLTK that can be installed within a python environment using the following command(s):

To import VADER into the python environment, the following commands were run:

“Import nltk
nltk.download(‘vader_lexicon’)”

2.2 Naive Bayes

Naive Bayes is a supervised machine learning algorithm that is built on top of Bayes’ Theorem. “The Naive Bayes classifier is a popular supervised machine learning algorithm used for classification tasks such as text classification [12]”. In the context of this project, the Naive Bayes classifier is used to classify the sentiments of the tweets. The text is firstly tokenized, and each piece of text is given a TF-IDF score forming a vector. The vector is then fed into the multinomial naive bayes classifier for training. The classifier treats each element in the vector as a feature and learns how much weight to assign each feature in determining the class of a text. After the model is trained, it is then tested against the testing subject a few times.

To import the Naive Bayes model into python, the following commands were run:
“Pip install sklearn
From sklearn.naive_bayen import MultinomialNB”

2.3 BERT

BERT stands for “Bidirectional Encoder Representations from Transformer” [13] which is a language representation model developed by Google in 2018. BERT utilizes transformer technology mentioned in the paper “Attention is all you need paper.” [14] While the BERT model is already pretrained, it’s a good practice as per the original paper [13] to finetune (further train) the model to the specific task at hand. In this project, the finetuning and use of the BERT model is explored.

To import the pre-trained BERT model, the transformers library has to be installed and imported to the python environment. To install the transformers library, the following command is run:

“! pip install transformers”

Then the BERT model is imported by running the following commands:
From transformer import BertTokenizer, BertModel

BertModel: This is the base BERT Model. It takes as inputs tokens produced by the BertTokenizer and outputs a representation

BertTokenizer is the module responsible for the tokenizer that will convert the text into tokens that the BERT model can understand. Tokenizing is the process of splitting the input into words, subwords, and mapping these words to their index in the BERT vocabulary.

Torch: To run the BERT model, a deep learning library such as Torch is necessary to run it. After the text is tokenized or represented into tokens, the module tensors inside the torch library are used to manipulate the tokens as necessary. Furthermore, the module (Neural Network) in the torch library is used to define the neural network architecture, including the layers and forward propagation logic.

To install and import the Torch library, the following commands are used:

“!pip install torch”

“Import torch
Import torch.nn as nn
From torch.utils.data import TensorDatasert, Dataloader”

Chapter 3 Machine Learning

The different Machine learning classifiers used in this project can be found below.

3.1 Random Forests

“Random forests is a type of supervised machine learning algorithm made up of decision trees, where it is used for both classification and regression [15]”. The random forests classifier works by first training the model on a couple of features. Once the model is trained, the new testing data is simulated on decision trees. For example if this sentiment is positive or negative? If the sentiment is positive, another feature is tested by asking a question such as, is the tweet volume above the designated threshold or not? If yes, then perhaps the return is positive, based on the kind of data present in the training.

The random forest classifier can be imported into a python environment by utilizing the sklearn library. Sklearn is short for scikit-learn which is one of the most popular machine learning libraries in python. “It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistent interface in python [16].”

To install and import the sklearn library the following commands are used:

“pip install -U scikit-learn”
“import sklearn”

To import the random forest classifier from sklearn, the following command is used:

“From sklearn.ensemble import RandomForestClassifer”
“**From sklearn.model_selection import train_test_split”**

Where data is split using the train_test_split module from the sklearn library.

3.2 K-Nearest Neighbour (KNN)

KNN is a simple and powerful supervised machine learning algorithm. “While it can be used for either regression or classification problems, it is typically used as a classification algorithm [17]”. In the case of this project, it is used for classification. The values of the features of the testing dataset are compared to all the instances of the training set and the difference between these values, usually measured by calculating the Euclidean distance between the numbers and each other. The results of the nearest neighbors in the training set to the testing set, ends up being the result for the testing set as well. In the context of this project, the default number of k-nearest neighbors taken into consideration is 5.

The KNN classifier is also imported from the sklearn library. To import the library, the following command is used:

“From sklearn.neighbors import KNeighborsClassifier”

Chapter 4 Stock Price Prediction System

Firstly, data is acquired and collected. Secondly, some mild preprocessing is done onto the datasets to get them a good enough format for step number 3 (Natural Language processing). Thirdly, three different natural language processing methodologies are assessed. Thee three are VADER, TF-IDF with Naïve Bayes, and finally, a transformer based neural network model is tried which as well is known as BERT. The accuracies for each of the models are calculated to see which one performs the best. The best-performing model is finetuned using the first dataset so that it can then be used against the main dataset (which contains the bulk of the data that is used to derive the prediction from). More processing is done to the main dataset and the final dataset to derive the meaningful features from them. The new dataset with the chosen sets of features is then passed into two different machine learning classifiers and the accuracy, precision, cumulative returns are calculated. A step-by-step stock price prediction system figure can be seen in figure 4.1.

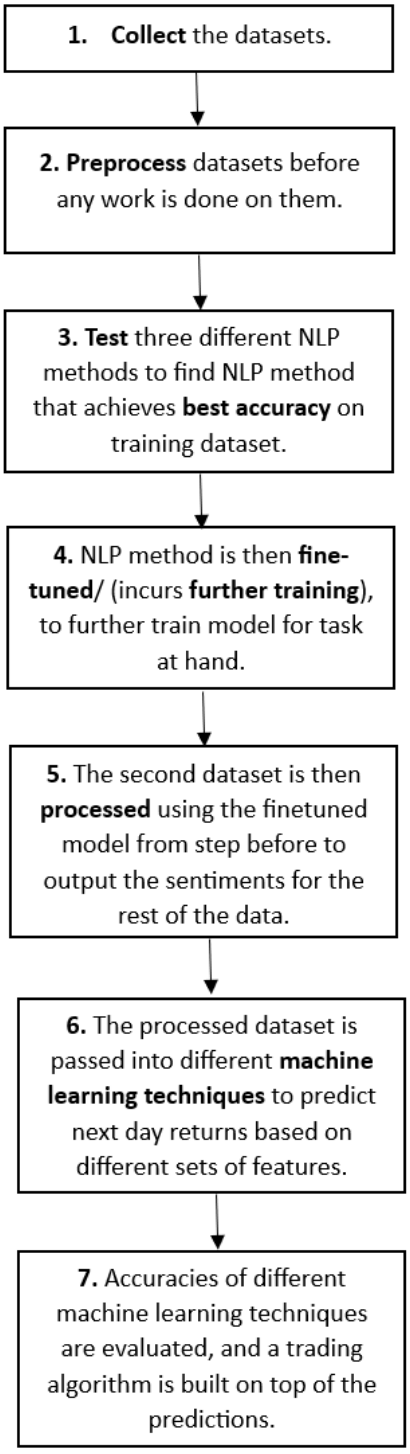

Figure 4.1 Stock price prediction methodology

4.1 Data Collection

In this project, three different datasets are acquired from three different sources.

First dataset
First dataset is a dataset acquired from Kaggle [18] which contains two features, textual ‘tweets’ and the sentiment of that tweet as given by a human being. The sentiment is given either a score of ‘1’ denoting positive or ‘-1’ denoting a negative sentiment. The purpose of this dataset is two-fold. Firstly, to find out the accuracies of the three different natural language processes and secondly, to train (Fine-Tune) the BERT model.

A snippet of first dataset can be figure 4.2, The dataset contains 5791 rows of tweet text data and their corresponding sentiment as rated by a human being.

	A	B
1	Text	Sentiment
5	MNTA Over 12.00	1
6	OI Over 21.37	1
7	PGNX Over 3.04	1
8	AAP - user if so then the current downtrend will break. Otherwise just a short-term correction in med-term downtrend.	-1
9	Monday's relative weakness. NYX WIN TIE TAP ICE INT BMC AON C CHK BIIB	-1
0	GOOG - ower trend line channel test & volume support.	1

Figure 4.2 Snippet of First dataset

Second Dataset

This is the main dataset in the project. It is used to provide the data from which the sentiment will be calculated from and from which the trading decisions will be derived as well. This dataset is acquired from data.world [19] and contains stock tweet data from the date 15-03-2016 → 15-06-2016 for 104 of the top NASDAQ stocks in 104 different csv files. Each file contains the relevant data for the stock. This dataset contains features such as the 'Ticker Name', 'Time of Tweet', 'Text', 'Number of likes', 'Number of RTS', and 'Followers of Person Tweeting'.

Number of features in original dataset	Number of features extracted for further processing	Number of features after processing used for machine learning	Number of rows in each file (Average)	Total Number of Rows of Data in all 104 stock files
26	9	2 & 3	6500 rows/stock file	676,000 Tweets

Table 4.1 Description of second dataset

The original second dataset had many unnecessary features and data such as the latitude and longitude of the tweet. Such information is unnecessary, the 26 original columns in each stock file are filtered into 8 which contain information that can be used. These columns include the following: Date, Hour, Tweet Content, TweetID, Favs, RTs, Followers, Following, and Symbols. More processing will be done on those 8 features/columns to extract the relevant 2 and 3 features that will be used as input for the ML classifiers.

Third Dataset

This dataset is acquired from yahoo finance [20]. It is a popular online stock price site. Yahoo finance has a python library named 'yfinance' and it can be imported into the python environment using the following command:

"Import finance as yf"

This dataset contains the adjusted closing price of the stocks which will be used as a metric to see if the twitter sentiment can have an effect on it.

4.2 Natural Language Processing

During this section, the use of the 3 different natural language processing methods are used in order to process the tweets and provide an either positive or negative sentiment. The first dataset is passed into each of the three libraries and the accuracy of each method is calculated by calculating how many of the output sentiments are actually close to the real sentiment in the original dataset.

4.2.1 VADER

VADER is a module inside of NLTK (Natural Language ToolKit). To import VADER into the python environment, the NLTK must first be installed and imported. To install and import the NLTK, the following commands are used:

"!pip install nltk
Import nltk"

The VADER module is then downloaded into the python environment by running the following commands:

"nltk.download('vader_lexicon')
From nltk.sentiment.vader import SentimentIntensityAnalyzer'

The 'SentimentIntensityAnalyzer' is a class within the VADER package in NLTK that calculates the sentiment scores. The score can range from -1 (negative) to +1 (positive). As mentioned above, each word of the text is tokenized and given a score using this method, the final score of the text is then output.

The pre-processed text is passed into the Sentiment Intensity Analyzer function which gives it a score from -1 to 1. An extra line of code is used to classify the scores into either positive or negative. A score of 0 to 1 is noted as positive, while -1 to -0.01 is noted as negative. The accuracy of VADER is calculated by comparing the outputted scores to the original sentiment scores of the document. The following formula is used:

$$\text{Accuracy} = \frac{Number\ of\ similar\ sentiment\ scores}{Total\ Number\ of\ rows} \times 100 \qquad \textit{Equation 4.2.1}$$

The **VADER accuracy** found is: **66.48%**, which is a relatively low score.

4.2.2 NAIVE BAYES and TF-IDF

In this section, TF-IDF is used in tandem with Naïve Bayes to derive the predictions. TF-IDF is short for Term Frequency – Inverse Document frequency.

To import the Naive Bayes classifier into the python environment, the sklearn (scikit-learn) library in python must be installed and imported.

The library is then loaded into the environment and these 2 specific modules are loaded into the environment.

From sklearn.feature_extraction.text import TfidVectorizer

TfidVectorizer is a function within the sklearn library that will tokenize the text data and turn it into the form of numbered vectors which can then be inserted into the machine learning algorithm.

From sklearn.naive_bayes import MultinomialNB

The first dataset is split into 80/20 training/testing ratio. Similar to VADER, the accuracy of Naïve Bayes is calculated by using equation 4.2.2. The **accuracy** of the **NAIVE BAYES** model for sentiment analysis is **66.78%**. Which is about the same as the VADER model.

$$\text{Accuracy} = \frac{\text{True Positives}}{\text{Total Number of Tries}} \qquad \textit{Equation 4.2.2}$$

4.2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pretrained NLP model developed by google and created by hugging face. The pretrained model has 110 million parameters. The model used in this project is the 'Bert-base-uncased' model which uses the RELU activation function. The Bert-Tokenizer is used to tokenize the text-data (which returns pytorch tensors), and the data is split into batches (In this project batch size is set to 16), then input into the training. The AdamW optimizer was used with a learning rate of 5e-5. This optimizer is used to update the weights and the biases of the original transformer neural network and set the weights and biases for the extra finetuning layer as well.

The extra finetuning layer has an input layer of 768 nodes (Equal to the output layer of the Bert-base uncased model), 50 nodes in the hidden layer (this is a hyperparameter), and 2 output nodes to denote the positive and negative sentiment.

4.2.3.1 Training of the BERT model

The training and testing dataset is also split to an 80:20 ratio. And the first dataset is passed through the BERT model 4 times (4 Epochs). The test accuracy and training losses can be seen in table 4.2.

Epoch	Test Accuracy	Training Loss
1	78.23%	0.53
2	81.48%	0.30
3	82.68%	0.14
4	83.28%	0.06

Table 4.2 Accuracies and test losses in first 4 epochs for BERT model

After incurring this further training (fine-tuning) process, the BERT model is saved for the usage of the next step. Which is the actual interpretation of the sentiment of the second dataset which contains the data that will be used for the prediction process.

4.2.3.2 Running the second dataset on the BERT model.

The second dataset which contains the tweets for the 104 stocks from the period of 15th of March - 15th of June - is run against the fine-tuned BERT model to output the sentiment of those tweets.

4.3 Further processing

After the sentiment of the 6500 rows of data (tweets) of the 104 different stock files are calculated, the dataset now looks as follows:

	A	B	C	D	E	F	G	H	I	J	K
1	Tweet Id	Ticker	Datetime	Text	Text_Clean	Favs	RTs	Followers	Following	Is a RT	Sentiment
2	7.43E+17	aal	2016-06-15 9:26	Why Ameri	why ameri	0	0	2039	108	FALSE	1
3	7.43E+17	aal	2016-06-15 8:18	Yesterday's	yesterdays	0	0	1792	80	FALSE	0
4	7.43E+17	aal	2016-06-15 8:06	SA_Quicklc	saquickide	0	0	1589	1376	FALSE	0
5	7.43E+17	aal	2016-06-15 8:04	JPMorgan (	jpmorgan c	0	0	771	8	FALSE	1
6	7.43E+17	aal	2016-06-15 8:01	$DAL	dal aal 5 la	0	0	788	6	FALSE	1
7	7.43E+17	aal	2016-06-15 7:51	JPMorgan (	jpmorgan c	0	0	1171	58	FALSE	1
8	7.43E+17	aal	2016-06-15 7:40	RT @bnkin	rt insiders	0	1	91	443	TRUE	0
9	7.43E+17	aal	2016-06-15 7:13	$AAL With	aal with v	0	0	10	0	FALSE	0
10	7.43E+17	aal	2016-06-15 5:28	$AAL Amer	aal america	0	0	33	2	FALSE	1
11	7.43E+17	aal	2016-06-15 5:07	Biggest	biggest los	0	0	183	29	FALSE	0
12	7.43E+17	aal	2016-06-15 4:15	| $AAL | Ar	aal america	0	0	0	0	FALSE	1
13	7.43E+17	aal	2016-06-15 3:24	TARO Day	taro day hi	0	0	77	40	FALSE	1
14	7.43E+17	aal	2016-06-15 3:06	$AAL top t	aal top ten	0	0	117	90	FALSE	1
15	7.43E+17	aal	2016-06-15 3:04	airlines bre	airlines bre	0	0	130	179	FALSE	1
16	7.43E+17	aal	2016-06-15 2:12	RT @airlin	rt airline st	0	4	812	1271	TRUE	0
17	7.43E+17	aal	2016-06-15 0:53	We calcula	we calculat	0	0	1398	1495	FALSE	1

Figure 4.3 Dataset after processing sentiment and retrieving relevant features for further processing

Figure 4.3 showcases the new main dataset after the sentiment processing using BERT with the sentiment column added to the far right where 1 denotes positive and 0 denotes a negative sentiment. The dataset then incurs further processing by doing the following:

1. The mean and the standard deviation of the ‘Number of Followers’ and ‘RTs’ columns are calculated.
2. The tweets that have an author that has a number of followers higher than the mean is given more weight.
3. The tweets that have more ‘Retweets’ than the mean is also given more weight.
4. A new metric is calculated to assign a weight to each tweet. Where tweets that have more followers and have more retweets are giving more weight. The new weight is multiplied by the sentiment either 1 for positive or -1 for negative, getting a weighted sentiment.
5. The tweets for one day are grouped together and a mean weighted sentiment is calculated. This mean weighted sentiment denotes the average sentiment across many different tweets for that tweet, where a higher weightage is given to tweets that have a higher ‘RT’ count and a higher ‘Followers’ count.
6. An extra metric is calculated, known as the sentiment moving average. The sentiment moving average calculates the average of the sentiment for the last 3 days. This is important as stock returns may not always be sudden.

7. The daily adjusted closing price for the next day is acquired from the third dataset by using yfinance. The next day's adjusted closing price is assigned to the same row as today's tweets as the tweets are going to be used to assess tomorrow's predictions.
8. The percentage change for daily stock prices is calculated and are assigned one of three different bins. If the daily return is negative, it is assigned to bin number 0. If the daily return has a small positive return (between 0% and 1% return) it is assigned to bin number 1. If the daily return has a high positive return (Higher than 1% return) it is assigned to bin number 2.
9. All the tweets for one day are combined, and the sentiment weighted, and Sentiment MA are averaged out for the day. Furthermore, the count for the number of tweets is logged in a column called 'Tweets', which is the number of tweets per day for a certain stock. This will be used as the third feature in the second set of features.

The new dataset can then be seen as shown in figure 4.4, each ticker now has a Sentiment_Weighted and a Sentiment MA for a specific day. The Tweets column also logs the frequency of tweets for that ticker and for that specific day. These 3 different features will be used next to input to the different ML classifiers.

	A	B	C	D	E	F	G	H
1	Ticker	Date	Sentiment_Weighted	Sentiment_MA	Tweets	Adj Close	Percent_Change	Percent_Change_Bin
2	aal	2016-03-11	0.30	0.30	33	40.70	0.28	1
3	aal	2016-03-14	0.89	0.43	19	40.81	-1.13	0
4	aal	2016-03-15	0.38	0.48	45	40.35	0.47	1
5	aal	2016-03-16	0.19	0.49	85	40.54	-0.50	0
6	aal	2016-03-17	0.02	0.20	41	40.34	3.18	2
7	aal	2016-03-18	0.38	0.20	61	41.63	0.07	1
8	aal	2016-03-28	0.30	0.49	30	39.20	1.08	1
9	aal	2016-03-29	0.43	0.48	87	39.62	0.07	1
10	aal	2016-03-30	0.68	0.47	65	39.65	-0.89	0
11	aal	2016-03-31	0.37	0.49	41	39.30	-3.63	0
12	aal	2016-04-01	0.38	0.47	48	37.87	-0.38	0

Figure 4.4 Dataset after processing new features

4.4 Using Machine learning

In this section, two different machine learning classifiers are used to try to predict the next day returns which is the output metrics using two different sets of features across two different data splits. In the first set of features, Sentiment_Weighted and Sentiment_MA are to be used in order to try to predict the next day returns, while in the second set of features, Sentiment_Weight, Sentiment_MA, and the 'Tweets' are to be used to make the next day returns predictions. The two classifiers used are KNN and RF.

Since stock price prediction usually has non-linear relationships with it's input features, KNN and RF are chosen, as they both do a good job at modelling non-linear relationships between features and output.

The Hyperparameter for K-Nearest Neighbors is set to 5, while in RF, minimum datapoints in leaves is = 10, and minimum number of trees is set to a 100. In RF, no bootstrapping is done in order to maintain chronological order of the data. Furthermore, no cross-validation is done with the dataset as this is a time-series data where chronological order is of upmost importance

4.5 Calculating cumulative trading returns based on predictions

As explained in the previous section, the machine learning classifiers were used to predict the percent change bin, where there are three possible bins the next day returns can be assigned. Either 0 (For negative), 1(Slightly positive), and 2 (Very positive).

Some simple code was written to simulate a real-life example, where a decision would be made to buy a stock if the prediction was 2 (Very positive), and short the stock when the prediction was 0 (Very Negative).

The code was tested against the two different data splits and the total average cumulative returns across all the stock tickers were compared to a simple buy and hold strategy. A buy and hold strategy are a simple strategy where a stock would be bought and held. This is done as a control measure to show if the other strategies can outperform this simple strategy.

The results of this logic can also be seen in the next chapter.

Chapter 5 Performance Evaluation

The metrics used to measure the effectiveness of the techniques used are accuracy, precision, and cumulative returns compared to a buy and hold strategy. The results of the use of two different sets of features across two different data splits can be seen in sections 5.1-4

5.1 Performance with a 34/66 data split using 2 features

In the instance of using only two features 'Sentiment Weighted' and 'Sentiment_MA' to predict the Percent_Change_Bin the following were the results:

	Accuracy	Precision	Cumulative Returns
Random Forests	0.476	0.472	1.03%
K-Nearest Neighbors	0.450	0.450	1.29%
Buy and Hold	N/A	N/A	-2.00%

Table 5.1 Metrics for 34/66 data split using 2 features.

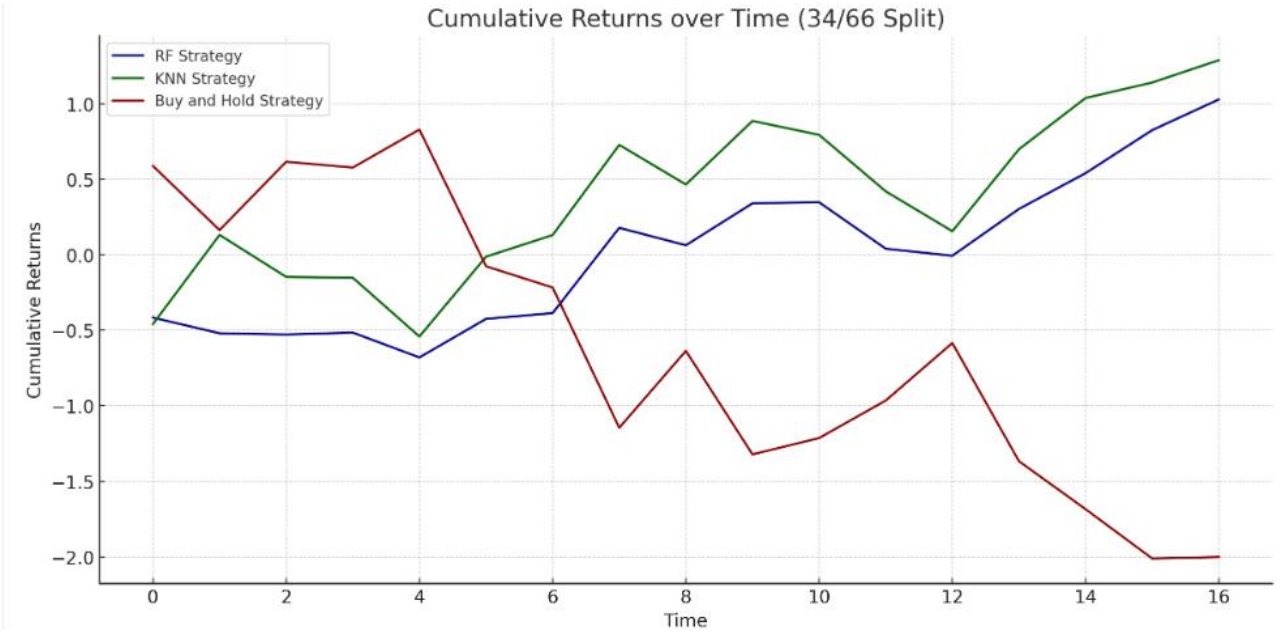

Figure 5.1 Cumulative returns over time for the different ML classifiers vs. buy and hold for 34/66 split using 2 features.

For the use of 2 features and 34/66 data split, Random Forests outperforms KNN in both accuracy and precision. Meanwhile K-Nearest neighbors outperform RF in cumulative returns as seen in figure 5.1. This means that K-Nearest neighbors is able to make predictions that make more profitable trades than RF and more than the market as well.

5.2 Performance with a 66/34 split using 2 features

In this section, the same 2 features are used, which are ‘Sentiment_Weighted’ and ‘Sentiment_MA’ to try to again predict the ‘Percent_Change_Bin’. The following were the results:

	Accuracy	Precision	Cumulative Returns
Random Forests	0.479	0.465	0.34%
K-Nearest Neighbors	0.458	0.437	0.54%
Buy and Hold	N/A	N/A	-0.64%

Table 5.2 Metrics for 66/34 data split using 2 features.

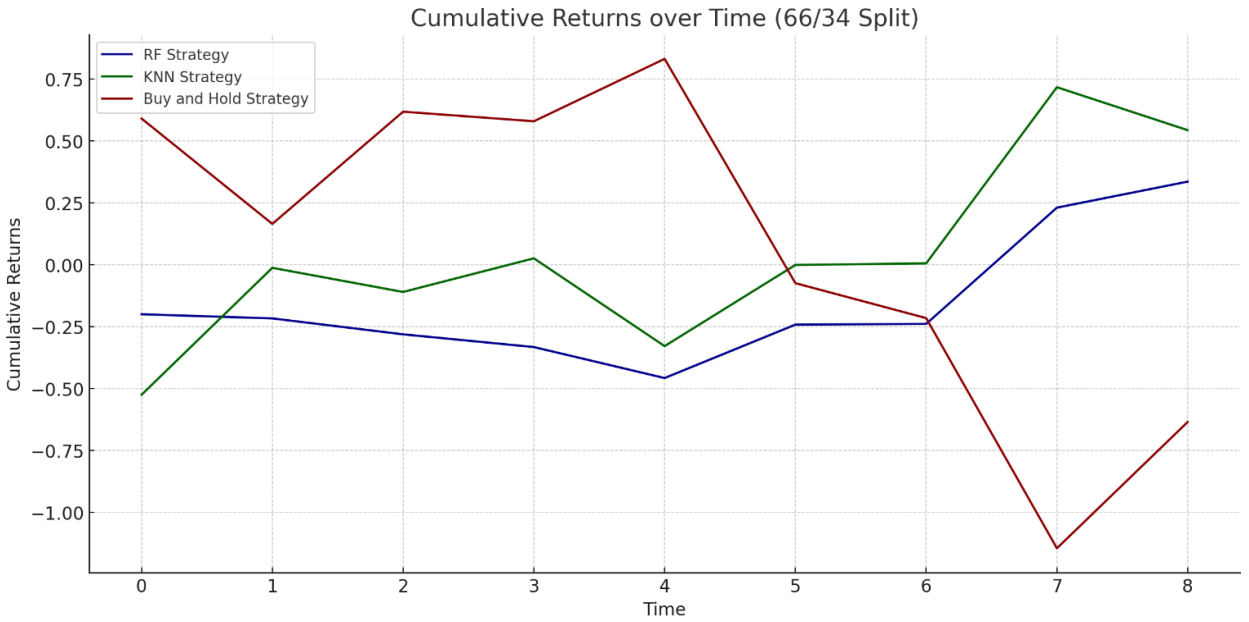

Figure 5.2 Cumulative returns over time for the different ML classifiers vs. buy and hold for 66/34 split using 2 features.

Using a 66/34 data split with the same features as 5.1, RF still outperforms KNN in terms of accuracy and precision as seen in table 5.2. Both classifiers are still able to make predictions that matter in terms of cumulative returns. This is in part means that both classifiers can detect negative returns a lot better than positive returns. In a down market, these classifiers do a really good job of detecting negative returns and thereby returning signals that can outperform the market.

5.3 Performance with a 34/66 split using 3 features

In this section, 3 features are used to try to predict the percent_change_bin, which are the 'Sentiment_Weighted', 'Sentiment_MA', and the 'Tweets' which denote the volume/number of tweets for a single day for a specific stock with 34% of the data trained, while 66% used for testing.
The performance of the different machine learning classifiers can be seen below:

	Accuracy	Precision	Cumulative Returns
Random Forests	0.481	0.465	1.178%
K-Nearest Neighbors	0.510	0.472	1.374%
Buy and Hold	N/A	N/A	-2.00%

Table 5.3 Metrics for 34/66 data split using 3 features.

Figure 5.3 Cumulative returns over time for the different ML classifiers vs. buy and hold for 34/66 split using 3 features.

When using 3 features and a 34/66 data split, KNN outperforms RF in terms of accuracy, precision, and cumulative returns. Both classifiers still can beat the market in terms of cumulative returns by a wide margin of at least 3%. This shows the predictive power of machine learning classifiers especially when the overall market is going down.

5.4 Performance with a 66/34 split using 3 features

In this section, 3 features are used to try to predict the percent_change_bin, which are the 'Sentiment_Weighted', 'Sentiment_MA', and the 'Tweets' which denote the volume/number of tweets for a single day for a specific stock with 66% of the data trained, while 34% used for testing.
The performance of the different machine learning classifiers can be seen below:

	Accuracy	Precision	Cumulative Returns
Random Forests	0.491	0.467	0.353%
K-Nearest Neighbors	0.515	0.478	0.638%
Buy and Hold	N/A	N/A	-0.636%

Table 5.4 Metrics for 66/34 data split using 3 features.

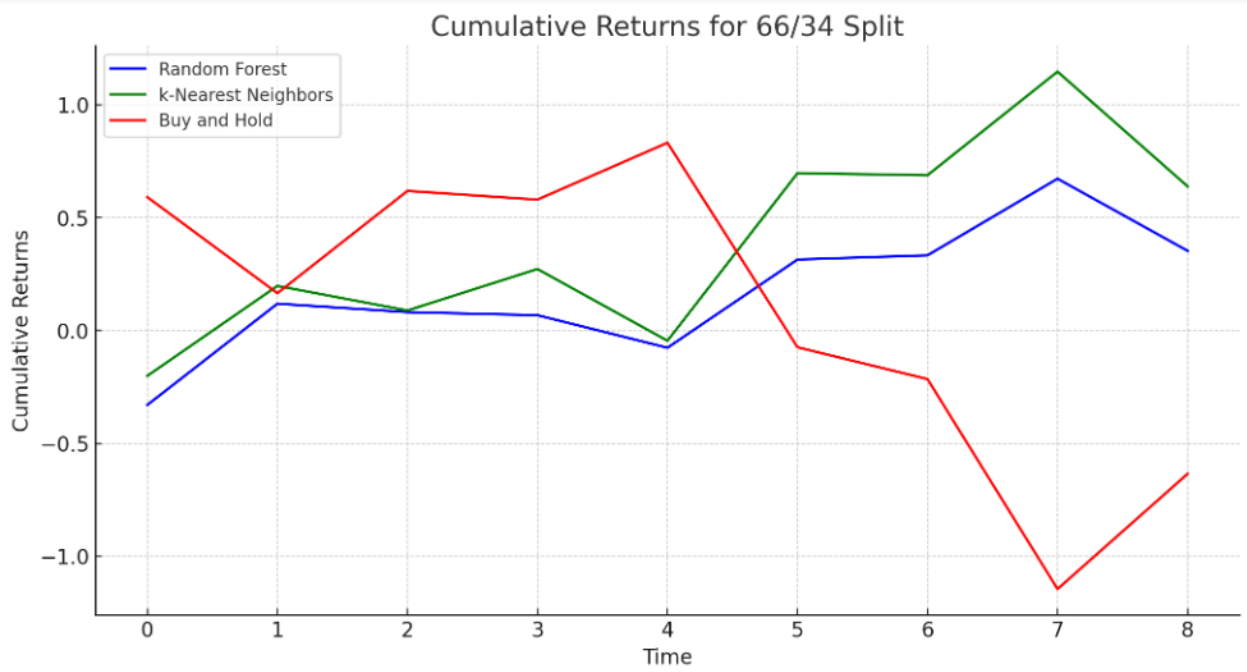

Figure 5.4 Cumulative returns over time for the different ML classifiers vs. buy and hold for 66/34 split using 3 features.

With using 3 features and 66/34 data split, KNN does an even better job with accuracy and precision than RF. Also, both classifiers still did a better job than the buy and hold which means that they were both still able to outperform the market. RF and KNN do an excellent job in detecting negative returns as they perform very well when the market goes down.

Discussion

When using only 2 features, the accuracies of both machine learning classifiers were relatively low (Both lower than 50%), however, both classifiers in both different data splits have managed to outperform the market in terms of cumulative returns. On average, across the 104 stocks, the 3 months of tweet data (676,000 tweets), have resulted in higher returns.

When using 3 features (Sentiment_Weighted, Sentiment_MA, and Tweets Volume) both classifiers increased in average accuracy, precision, and cumulative returns across both the different data splits.

The KNN Classifier with 3 features achieved the highest accuracy, precision, and cumulative returns. The KNN classifier achieved an accuracy of 51.0% with 3 features in the first data split (34/66) and rose a little with the second data split (66/34). These accuracies are lower than some of the models that already exist out there such as the SLOT model [6], which has an average accuracy of 56.46% across three different datasets.

Despite both classifiers having a somewhat average accuracy, both models were still able to outperform a simple buy and hold strategy across the 104 different tickers. Where a buy and hold strategy resulted in the case of only using 2 features in an average return of -1.32% while the RF classifier achieved a return of 0.685% across both data splits which comes to a % difference. Meanwhile the KNN classifier had an average higher return of 0.915% which is a 2.24% difference between the KNN classifier and a regular buy and hold strategy. This showcases the predictive power of machine learning using tweet data as input.

When using 3 features for prediction, both classifiers did even better, where the difference between the average RF classifier return and a buy and hold strategy was 2.1% while the difference between the KNN classifier returns and the buy and hold strategy was 3.332% which is 1% higher than with 2 features.

Although the models' accuracies did not exceed current state of the art model (SLOT), it still performed well in terms of overall returns.

Chapter 6 Conclusion and Future Work

This project considered stock market prediction by analyzing the sentiment of tweets using natural language processing then using machine learning to classify the next day returns based on sentiment and tweets frequency. Different Natural language processing methods were first explored including VADER, Naive Bayes, and BERT. The best natural language processing model (BERT) was used to analyze the sentiment of the full tweet data which consisted of 104 datasets of tweets of stocks that are part of the (NASDAQ). Machine learning was then used to predict the next day returns of individual stocks based on its corresponding dataset. The results show that the KNN classifier achieved the highest accuracy in predicting individual stocks.

The results also show that all classifiers in both cases have outperformed a simple buy-and-hold strategy. When predicting individual stocks across the 104 stocks, on average, the KNN classifier (although did not have an accuracy of higher than 50%) was able to achieve a 2.01% return across, which is 3.33% higher than a simple buy and hold strategy.

Possible areas of future work would be gathering a larger dataset. The datasets for each stock in this experiment contained 3 months of data and around 6,500 rows or tweets. A dataset containing 1 year of data could be useful to further prove this thesis. Furthermore, more advanced natural language processing methods such as using OpenAI's new GPT can possibly lead to a better sentiment accuracy and perhaps a better overall return and accuracy in prediction. Moreover, incorporating a methodology to filter out spam tweets can perhaps also increase accuracy in predictions.

References:

[1] J. Chen, “What is the stock market, what does it do, and how does it work?” Online], Mar. 12, Investopedia, https://www.investopedia.com/terms/s/stockmarket.asp (accessed Jul. 9, 2023).

[2] I. Hwang, “A Brief History of the Stock Market,” *SoFi*, [Online], Mar. 25, 2020. https://www.sofi.com/learn/content/history-of-the-stock-market/ (accessed Jul. 9, 2023).

[3] C. Majaski, “The Difference Between Fundamental vs. Technical Analysis?,” *Investopedia*, 2019. [Online], https://www.investopedia.com/ask/answers/difference-between-fundamental-and-technical-analysis/ (accessed Jul. 9, 2023).

[4] D. E. Allen, M. McAleer, and A. K. Singh, “Daily market news sentiment and stock prices,” *Applied Economics*, vol. 51, no. 30, pp. 3212–3235, Feb. 2019, doi: https://doi.org/10.1080/00036846.2018.1564115.

[5] I. Ivanova, “Stock market’s fall has wiped out $3 trillion in retirement savings this year - CBS News,” *www.cbsnews.com*, Jun. 17, 2022. [Online], https://www.cbsnews.com/news/stocks-drop-recession-retirement-savings-401k-ira-3-trillion-2022/#:~:text=Nearly%20three%2Dquarters%20of%20all (accessed Jul. 29, 2023).

[6] A. Lopez-Lira and Y. Tang, “Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models,” *SSRN Electronic Journal*, 2023, doi: https://doi.org/10.2139/ssrn.4412788.

[7] [A. L. Hansen and S. Kazinnik, “Can ChatGPT Decipher Fedspeak?,” *SSRN Electronic Journal*, Mar 24, 2023, doi: https://doi.org/10.2139/ssrn.4399406.

[8] Q. Xie, W. Han, Y. Lai, M. Peng, and J. Huang, “The Wall Street Neophyte: A Zero-Shot Analysis of ChatGPT Over MultiModal Stock Movement Prediction Challenges,” Apr. 2023, doi: https://doi.org/10.48550/arxiv.2304.05351.

[9] Y. Soun, J. Yoo, M. Cho, J. Jeon and U. Kang, "Accurate Stock Movement Prediction with Self-supervised Learning from Sparse Noisy Tweets," 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 2022, pp. 1691-1700, doi:10.1109/BigData55660.2022.10020720.

[10] [1]“Advances in Natural Language Processing,” *Nature*, Oct. 23, 2023. [Online], https://www.nature.com/collections/fbehficiij (accessed Jul. 9, 2023).

[11] J. Mahreen, “Sentiment Analysis Using VADER,” *Analytics Vidhya*, Oct. 02, 2022. [Online], https://www.analyticsvidhya.com/blog/2022/10/sentiment-analysis-using-vader/ (accessed Jul. 9, 2023).

[12] S. Ray, “6 Easy Steps to Learn Naive Bayes Algorithm (with code in Python),” *Analytics Vidhya*, Sep. 03, 2019, [Online], https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/ (accessed Jul. 9, 2023).

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” 2018, Article, Google AI Language, doi:10.48550/arxiv.1810.04805.

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need., Article, 31st Conference on Neural Information Processing Systems, https://doi.org/10.48550/ARXIV.1706.03762

[15] R. Meltzer, “What is Random Forest?” *careerfoundry.com*, Oct. 21, 2020. [Online], https://careerfoundry.com/en/blog/data-analytics/what-is-random-forest (accessed Jul. 9, 2023).

[16] “Scikit Learn - Introduction,” *www.tutorialspoint.com*, [Online], [https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm#:~:text=Scikit%2Dlearn%20(Sklearn)%20is (accessed Jul. 18, 2023).

[17] “What is the K-nearest neighbors’ algorithm?,” IBM, [Online], https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point (accessed Jul. 9, 2023).

[18] Y. Chaudhary, “Stock-market sentiment dataset,” Kaggle, [Online], https://www.kaggle.com/datasets/yash612/stockmarket-sentiment-dataset (accessed Jul. 9, 2023).

[19] “NASDAQ 100 tweets - dataset by Kike,” data.world, https://data.world/kike/nasdaq-100-tweets (accessed Jul. 9, 2023).

[20] “Stock market, Finance & Business News | Yahoo Finance Canada,” Yahoo! Finance, 2000, [Online] https://ca.finance.yahoo.com/ (accessed Jul. 9, 2023).