Final Project: Classifying Percentage Change in Closing Stock Price From Tweets

Ryan Fischbach
Dr. Khuri

December 4th, 2020

1 The Problem

Social media has allowed for the instant communication of important information, oftentimes before it hits the news or more details are known by the masses. The ability to extract this information from posts, such as tweets on Twitter, can provide a powerful tool to be able to make informed decisions about the future before someone without this knowledge would be able to.

This presents the data mining question: "Can Tweets and other stock information from the current day be used to classify the percent change in a stock's closing price tomorrow?".

2 Gathering the Dataset

To utilize social media and determine its impact on generating actionable insight, Tweets were scraped using the SNScrape library. Tweets from a three-year window (12/1/2017 - 12/1/2020) were pulled on topics surrounding 'Amazon'. Initially, 50,000 tweets were scraped from this query, roughly evenly distributed through each day. The tweets had the following attributes:

Column Index	Attribute
0	Tweet ID
1	Date
2	Tweet

Figure 1: *Attributes of the Tweets scraped from Twitter about 'Amazon'. There are three attributes in total: TweetID as the unique identifier of this Tweet, Date, and the Tweet itself.*

To generate labels for the dataset, Yahoo Finance data was pulled with the ticker 'AMZN', corresponding to Amazon's stock information. An existing

implementation from Gunjan933's GitHub was modified for this problem. This produced Amazon stock information for all trading days starting on 5/15/1997. The stock information had the following attributes:

Column Index	Attribute
0	Date
1	Low
2	Open
3	Volume
4	High
5	Close
6	Adjusted Close

Figure 2: *Attributes of the stock data scraped from Yahoo Finance. The attributes include the Date as the unique key, the low stock price for that day, the opening stock price, the volume (number of trades made), the high price, the closing price, and the adjusted closing price.*

3 Preprocessing Steps

To enable the best classification of the tweets and stock data, several steps were taken.

3.1 Initial Preprocessing Pipeline

1. Duplicate and NA values were removed because the tweets had a unique ID and the stocks had a unique date value.

2. Stocks were filtered to be in the date range of tweets.

3. Labels were created by assigning 1 if the percent change in tomorrow's close from today is greater than 2 percent, -1 if less than 2 percent, and otherwise assigning 0. 2 percent was determined to be a good cutoff as it is higher than the 1 percent average daily volatility of the stock market.

4. Irrelevant attributes were removed.

5. Tweets from the same day were appended together and a count was created to store the number of tweets each day.

6. Tweets from days where the market was not open were removed.

7. Punctuation and numbers were removed, text was made lowercase, the text was tokenized, stop words (common words) were removed because they don't provide meaning, stemming was performed via Porter Stemming, and tokenized words were rejoined to sentences. Additionally, a Tweet specific preprocessing library was applied to remove Twitter-specific characters (@, hashtags, etc).

8. The top X percent and bottom Y percent of words were removed and ngrams were created via TFIDF Vectorizer. The exact values were determined by hyperparameter tuning.

The following steps were taken as a response to difficulty classifying the tweets. More details are below in the "Initial Classification and Resulting Issues" section.

3.2 Additional Preprocessing Steps in Final Pipeline

1. Labels were modified by assigning 1 if the percent change in tomorrow's close from today is greater than 2 percent and otherwise assigning 0. This transitioned the problem from a multi-class classification to a binary classification problem.

2. The majority class was under-sampled to create a 50/50 class distribution.

3. The current day's percentage change in opening stock price and the previous day's volume was included to provide some more contextual attributes to the data.

4. A rolling window of Tweets from n days was created (tweets from today and the previous n-1 days) to provide more contextual information to the algorithm.

5. Sentiment analysis was performed on each day's tweet from TextBlob's NaiveBayesAnalyzer, providing a score from [0,1] specifying how negative to positive the tweet's sentiment was.

4 Exploratory Data Analysis

Before jumping into the classification of the data, several steps were taken to better understand it as well.

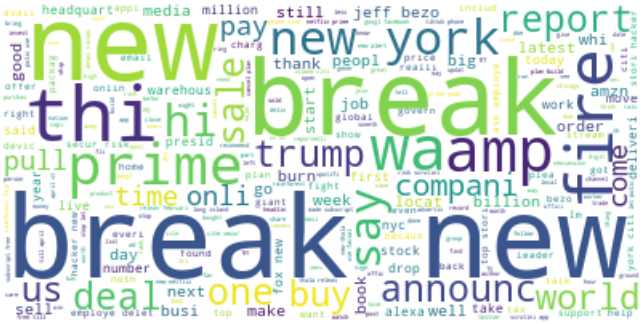

Figure 3: *Visualization of the most prevalent words from the tweets being analyzed. The larger words signify a higher frequency of occurrence, with smaller words not occurring as much. New York, Prime, Trump, New, and Break are the most commonly occurring words.*

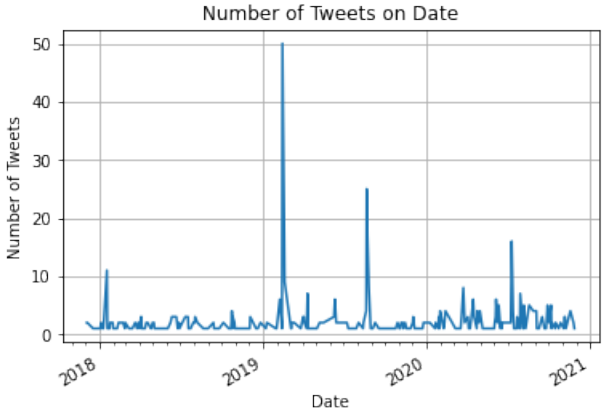

Figure 4: *Visualization of the Number of Tweets on the Date. The number of tweets seems to be consistently around 1, with a few spikes over the number of dates.*

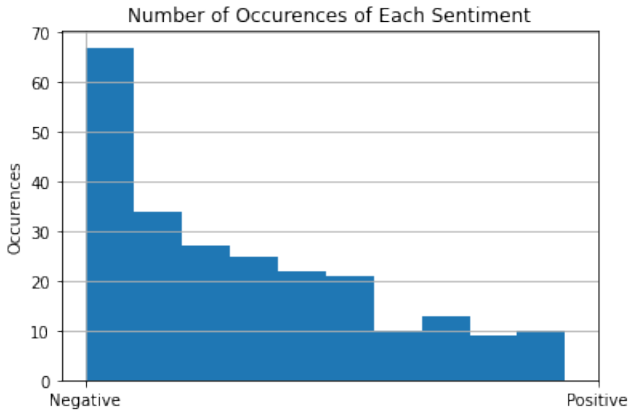

Figure 5: *Visualization of the Number of Occurrences of Each Sentiment. The sentiment is overwhelmingly negative across tweets, with there being over 60 occurrences of tweets where the probability of being negative is over 90 percent.*

5 Text Classification

5.1 Initial Classification and Resulting Issues

To classify this text, pipelines were created including a TFIDF vectorizer to translate the tweets into a matrix representation followed by a classifier. The classifiers tested were Naive Bayes, Random Forest, Linear SVC, LogisticRegression, and SGDClassifier (all from SKLearn). These pipelines were analyzed using SKLearn's GridCV to determine the optimal hyperparameters for TFIDF and the classifier via a brute force approach. Once the optimal hyperparameters were determined, classification was performed 5 times on the test dataset and performance metrics were averaged.

Additionally, temporal cross-validation was applied to evaluate model performance while preventing temporal information leaks. SKLearn's TimeSeriesSplit was used and performance metrics were once again averaged.

A naive classifier would assign the predominant class label, and in this case, with a balanced class distribution, be right 50 percent of the time. The models all struggled to break that 50 percent accuracy with most around 40 percent, even after hyperparameter tuning. This suggests that there was not enough information in tweets containing 'Amazon' to give any indication about the stock market's percent change in close the next day.

5.2 Mitigation and Resulting Efforts

To combat these issues, tweets containing 'Amazon Breaking News' were acquired from the same three-year range to replace 'Amazon' tweets. This dataset

was much smaller (1,280 tweets), but would hopefully encode more meaning about the company.

Additionally, more preprocessing steps were added as highlighted in the "Additional Preprocessing Steps in the Final Pipeline" section above. One major step was to perform sentiment analysis as well as include opening and volume information from the previous day. This facilitated these numeric attributes being used in traditional similarity-based classifiers such as K Nearest Neighbors. Additionally, this encoded the tweets differently for classification. Below are the results from classification using the optimal hyperparameters for each TFIDF vectorizer and model performed 5 times.

Model Name	Mean Accuracy	Std Deviation
GaussianNB	0.549020	0.000000
KNeighborsClassifier	0.588235	0.000000
LinearSVC	0.501961	0.010125
LogisticRegression	0.627451	0.000000
GaussianNB	0.627451	0.000000
RandomForestClassifier	0.456209	0.042228

Figure 6: *Results of classification from models averaged over 5 iterations. The Naive Bayes and Logistic Regression classifiers performed the best, yielding an average accuracy of 62.74 percent. The K Nearest Neighbors classifier, using only the sentiment analysis of the tweets as well as some basic stock information had an average accuracy of 58.82 percent. The other classifiers broke 50 percent accuracy, barely beating the baseline, except for the Random Forest Classifier.*

Model Name	Mean Accuracy	Std Deviation
GaussianNB	0.435897	0.079031
LinearSVC	0.482051	0.107875
LogisticRegression	0.482051	0.104784
RandomForestClassifier	0.466667	0.091377
SGDClassifier	0.502564	0.103521

Figure 7: *Results of classification of models using temporal cross validation with 5 splits. The SGD Classifier performed the best, with a mean accuracy of 50.025 percent. All other classifiers did worse than baseline, suggesting that there is not a problem with the classifiers but rather the data being input into the classifiers.*

6 Analysis

6.1 Limitations of this Method

From this analysis, it can be concluded that the tweets analyzed do not encode enough meaning to make a powerful classification of percent change in closing stock price. The tweets alone could not facilitate any model beating baseline accuracy. This can be seen in Exploratory Data analysis as well, with many of the most frequently occurring words not providing much information regarding Amazon or items that would impact Amazon's stock price. Additionally, the number of tweets with the query 'Amazon Breaking News' was very low. Lastly, the sentiment of the tweets was overwhelmingly negative, suggesting there are not many positive tweets to indicate an increase in closing stock price.

When other attributes such as sentiment, the percent change in opening stock price, and volume are included, KNN can reach nearly identical performance as a fully optimized classifier using TFIDF. Text classification is significantly higher dimensional, having the opportunity to encode a lot more meaning. Because of this small discrepancy between the two methods, this point is further proof that the Tweets scraped cannot be used in this classification task.

6.2 Potential Improvements of this Method

To improve this model's performance, several steps can be taken. The first, and most important, would be to scrape higher quality tweets that include meaning related to Amazon's stock price. Some possible ideas would be to pull verified user's tweets, to filter tweets by a certain like or retweet number, or to pull in tweets from various news stations that are on an approved list. Currently, no scraping library for Python allows for this kind of functionality that I discovered. However, these kinds of suggestions would increase the likelihood of acquiring data the encodes meaning related to stock price information.

Alternatively, another data source could be pulled in to assist or replace tweets with the classification task. As long as this source follows the model of being readily available and "piplineable" to provide actionable insight in the future, some more diversified data would likely help.

Lastly, the data mining question could be rephrased slightly to provide more data. If the question was changed to: "Can social media posts and other stock information from the current day be used to classify the percent change in the SP500's close tomorrow?", significantly more data could be used and hopefully, tweets could encode more meaning to be used for classification.

While this is not the desired result of this project, this is an important lesson to aspiring and current data professionals on the impact of choosing the right question, gathering high-quality data, and using a robust analysis process on achieving the desired results.

7 Citations

[1] SNScrape. 2020. https://github.com/JustAnotherArchivist/snscrape
[2] Tan et al. 2005. Introduction to Data Minings.
[3] Numpy. 2020. https://numpy.org
[4] Pandas. 2020. https://pandas.pydata.org
[5] Matplotlib. 2020. https://matplotlib.org
[6] Seaborn. 2020. https://seaborn.pydata.org
[7] Scikit learn. 2020. https://scikit-learn.org/stable/
[8] Natalia Khuri. 2020. Lectures.
[9] NTLK. 2020. https://www.nltk.org
[10] StockMarketScraper. 2020. https://github.com/Gunjan933/stock-market-scraper
[11] WordCloud. 2020. http://amueller.github.io
[12] TextBlob. 2020. https://textblob.readthedocs.io/en/dev/
[13] EfficientTweetPreprocessor. 2020. https://www.kaggle.com/sreejiths0/efficient-tweet-preprocessing
[14] How to Preprocess Text Data in Python. 2020. https://pythonhealthcare.org/2018/12/14/101-pre-processing-data-tokenization-stemming-and-removal-of-stop-words/