Machine Learning Capstone Proposal

1. Definition

1.1. Project Overview

This project will evaluate the ability to make stock market predictions based on World Daily News. The original Kaggle project (Daily News for Stock Market Prediction) made by Aaron7sun[i] took the Dow Jones Industrial Average (Dow Jones) and modeled it against Reddit World News data. I have modified Aaron7sun's original dataset by utilizing the dataset from Dominik Gawlik's New York Stock Exchange[ii].

The Dow Jones is a stock market index that takes the value of only 30 stocks whereas the Standard and Poors stock market index factors in 500 stocks. Furthermore, the NYSE dataset is broken down into 11 GICS sectors. Things like Energy and Consumer Discretionary sectors should have more sensitivity to World News in comparison to other sectors.

While world news only has a small correlation to the stock market. If as a stockbroker, I was able to be precise in the stocks I bought and sold based on the world news while minimizing risk, I would do well in making money.

1.2. Problem Statement

Based on Aaron7Sun's Kaggle project, users were unable to have much better than a 50-60% accuracy with the stock market. If you switch to the NYSE data and separate the stock market by GICS sectors, can you make a more accurate and more precise prediction of the stock market?

In the purchasing and selling of stocks you are charged on a per transaction fee by your broker. It is better to purchase less stocks with better quality then many stocks in small quantity and hurt yourself on transaction fees. As a result, it is better to be selective and purchase fewer stocks with the least amount of risk then purchasing more stocks and risk losing money.

So, in this project, we will attempt to solve the problem by providing more data and focusing on cleaning the data to help the model classifier the data. The way we will solve the problem is by multipart strategy:

1. Switch the stock index from Dow Jones Industrial Average to New York Stock Exchange. Also, group the stocks by GICS sector.
2. Change from a regression to classification. Classify the stocks as buy, sell or hold.
3. Label the stocks as a buy/sell if they surpass a threshold. In this case we will use 1 or 2 standard deviations away from the open price of the stock.
4. Perform Data analysis to cleanup the world news. For instance, extract tokens and named entities. Also generate a sentiment score
5. Try multiple time-based split methodologies; single split, expanded window and rolling window
6. Run machine learning algorithms to determine best one to use for this use case
    a. Utilize Decision Tree Classifier and Extreme Gradient Boost classifier as the 2 primary classifiers.
7. Run undersampling to normalize the labeled predictions with the unlabeled predictions.

1.3. Metrics

The key to the success of this project is determining whether we should buy/sell a stock on a given day. So, we need to get as many true positive results while minimizing false positives even if it causes us to have more false negatives. For this reason, the primary scores we will focus on is precision, accuracy and Cohen's kappa.

Precision will compare the true positive vs false positive results. Accuracy will evaluate the overall success and quantity of predictions made. Finally, Cohen's kappa is a mix of precision and accuracy and will let us know whether our predictions are simply random or whether there is a correlation. Cohen's kappa also works well with imbalanced classes and multi-class issues[iii]

Cohen's kappa calculation: (Total Accuracy – Random Accuracy)/(1 – Random Accuracy)

Values:

- < 0 No agreement
- 0-0.20 Slight agreement
- 0.21-0.40 Fair
- 0.41-0.60 Moderate
- 0.61-0.80 Substantial
- 0.81-1.00 Perfect agreement

For comparison sake, we will also generate the score for recall and F1 but they will not be used to determine success/failure.

1.3.1. Buy/Sell Confusion metric

In addition to the standard metrics, I have created a custom metric to evaluate how well the classifier does at predicting a buy/sell stock. While the precision score is good, it does not factor in the differences between buy, sell and hold. For instance, if I predict a buy but the stock goes down is worse then predicting a buy and the stock does not change. Also, predicting a hold on the stock when the stock goes up or down is a bad prediction/false negative but is deemed as ok.

I have broken the predictions into 4 rows and 5 columns. The columns are based on whether the prediction was a buy, sell, hold, miss and total. The rows count how many were good, bad, total and then the percentage of good choices. The miss column is different from the others as it only counts bad misses where prediction was a buy/sell but the actual result was a hold. While it is still a bad prediction, the stock broker only loses out on transaction fees.

- Good predictions
  - Buy – Indicates that the predicted purchase was a good choice
  - Sell – Indicates that the predicted sell was a good choice
  - Hold – Indicates that the predicted hold was a good choice
  - Miss – n/a does not apply
  - Total – Count of all the good choices
- Bad Predictions
  - Buy – Indicates that the predicted purchase was a bad choice resulting in a big loss
  - Sell – Indicates that the predicted sell was a bad choice resulting in missed profit
  - Hold – Indicates that the predicted hold was a bad choice

    - Miss – Indicates that the predicted buy/sell result in an actual hold or no change. Lose transaction fee
    - Total – Count of all the bad choices
- Total Predictions – Totals the good and bad predictions
- True % - Percentage of ratio of good to bad choices
    - Miss – Percentage of missed predictions in comparison with hold predictions.

The reason that I believe my attempt to the problem will work is because certain industries/sectors are more affected by regional changes. For instance, Middle East effects oil and energy and China and East Asia affects imports of consumer goods. While the Dow Jones gives an overall outlook, focusing on industries can help to focus the data on the most influenced by world news.

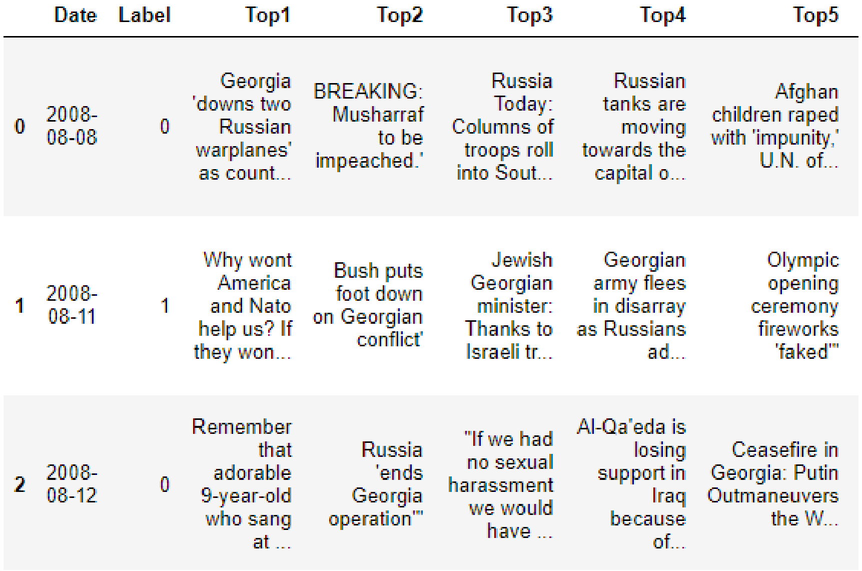
2. Analysis

2.1. Data Exploration

Baseline Dataset: Combined_News_DJIA.csv

Taken from Kaggle using Aaron7Sun's Daily News for Stock Market Prediction dataset. Dataset is an aggregate of the Top 25 World Daily News articles taken from Reddit. Data has already been preprocessed and classified prior to

- Source: https://www.kaggle.com/lseiyjg/use-news-to-predict-stock-markets
- Dimensions: 1989 x 27.
- Columns:
    - Date: date of news article
    - Label: Value was 1 when DJIA Close rose or stayed the same and 0 if it decreased
    - Top1-25: Top world daily new articles for each day.
- Date Range: Dataset was from 8/8/2008-7/1/2016.'
- Original Dataset:



	Date	Label	Top1	Top2	Top3	Top4	Top5
0	2008-08-08	0	Georgia 'downs two Russian warplanes' as count...	BREAKING: Musharraf to be impeached.'	Russia Today: Columns of troops roll into Sout...	Russian tanks are moving towards the capital o...	Afghan children raped with 'impunity,' U.N. of...
1	2008-08-11	1	Why wont America and Nato help us? If they won...	Bush puts foot down on Georgian conflict'	Jewish Georgian minister: Thanks to Israeli tr...	Georgian army flees in disarray as Russians ad...	Olympic opening ceremony fireworks 'faked'"
2	2008-08-12	0	Remember that adorable 9-year-old who sang at ...	Russia 'ends Georgia operation'"	"If we had no sexual harassment we would have ...	Al-Qa'eda is losing support in Iraq because of...	Ceasefire in Georgia: Putin Outmaneuvers the W...

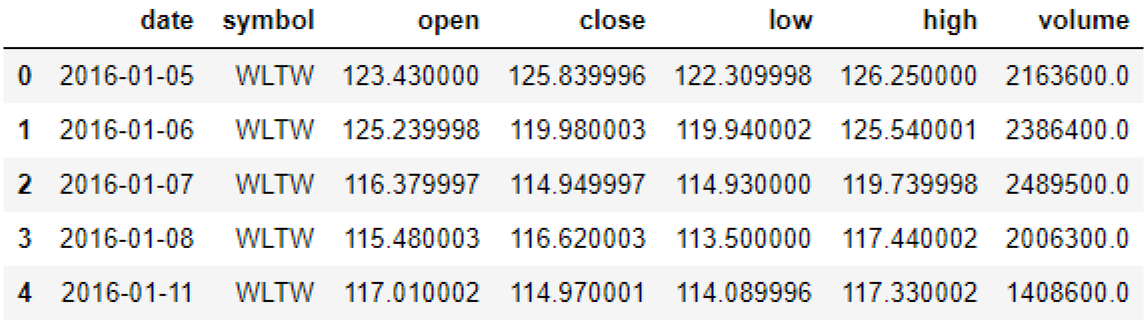
- Grouping: Combine the top 25 news articles by date to enable you to classify the articles by day.

- Other: Only abnormality is that some of the data is utf-8 and other parts are byte encoding but saved as string. Outliers were leveled by the classification of the stocks.

News Dataset: prices-split-adjusted.csv

Taken from Dominik Gawlik's New York Stock Exchange on Kaggle. Prices were fetched from Yahoo Finance. The prices-split-adjusted takes the raw daily prices for companies on the NYSE and adjusts for the 140 splits during that time period.

- Source: https://www.kaggle.com/dgawlik/nyse
- Dimensions: 851264 x 7.
- Columns:
  - Date: price of stock on a given day
  - Symbol: stock symbol as listed on NYSE
  - Open: stock price at opening bell
  - Close: stock price at closing bell
  - Low: lowest price for stock symbol on that day
  - High: highest price for stock symbol on that day
  - Volume: Number of transactions on a given day
- Date Range: 1/4/2010-12/30/2016
- Original Dataset:



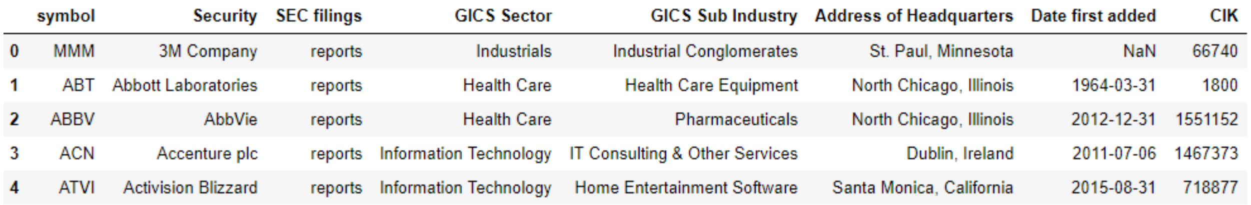
	date	symbol	open	close	low	high	volume
0	2016-01-05	WLTW	123.430000	125.839996	122.309998	126.250000	2163600.0
1	2016-01-06	WLTW	125.239998	119.980003	119.940002	125.540001	2386400.0
2	2016-01-07	WLTW	116.379997	114.949997	114.930000	119.739998	2489500.0
3	2016-01-08	WLTW	115.480003	116.620003	113.500000	117.440002	2006300.0
4	2016-01-11	WLTW	117.010002	114.970001	114.089996	117.330002	1408600.0

- Calculated columns: In order to account for abnormalities and determine a buy, sell, hold of stock groups, we modify the data to a classification problem.
- To label the data we will give a value of 1 if the price is 1 standard deviation away from the mean and 0 if the price is less than 1 standard deviation away.
  - Daily Return: difference in stock price from open to close
  - Label1Sig: Value set to 1 if price went up 1 standard deviation away from mean, -1 if price went down by 1 standard deviation away, otherwise 0
  - Label2Sig: Value set to 1 if price went up 2 standard deviation away from mean, -1 if price went down by 2 standard deviation away, otherwise 0
- Issue: The issue with using 1 standard deviation (1 sigma) is that as more data comes in the mean and standard deviation will change.

News Dataset: securities.csv

Taken from Dominik Gawlik's New York Stock Exchange on Kaggle. Description of each stock symbol in the dataset and it is categorized by GICS sectors.

- Source: https://www.kaggle.com/dgawlik/nyse
- Dimensions: 505 x 8.
- Columns:
  - Ticker Symbol: Stock symbol as listed on stock exchange
  - Security: Name of company for stock symbol
  - SEC filings: How it is reported to SEC
  - GICS Sector: Category that the stock symbol is grouped in
  - GICS Sub Industry: Sub-category that the stock symbol is grouped in
  - Address of Headquarters: Company headquarters
  - Date first added: Date that the company was added to the exchange
- Original dataset:



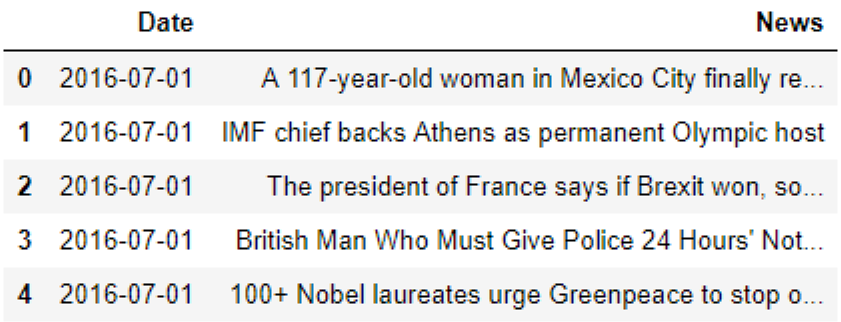
	symbol	Security	SEC filings	GICS Sector	GICS Sub Industry	Address of Headquarters	Date first added	CIK
0	MMM	3M Company	reports	Industrials	Industrial Conglomerates	St. Paul, Minnesota	NaN	66740
1	ABT	Abbott Laboratories	reports	Health Care	Health Care Equipment	North Chicago, Illinois	1964-03-31	1800
2	ABBV	AbbVie	reports	Health Care	Pharmaceuticals	North Chicago, Illinois	2012-12-31	1551152
3	ACN	Accenture plc	reports	Information Technology	IT Consulting & Other Services	Dublin, Ireland	2011-07-06	1467373
4	ATVI	Activision Blizzard	reports	Information Technology	Home Entertainment Software	Santa Monica, California	2015-08-31	718877

- Usage: Table will be joined the price split table to group the stocks by GICS sector and determine 1 or 2 standard deviations away from the mean.
- Issue: GICS Sector and GICS Sub Industry are categorical and will be used to help classify the data. Some of the stocks were not in the NYSE until after 2010.

News Dataset: RedditNews.csv

Taken from Aaron7Sun's Daily News for Stock Market Prediction dataset on Kaggle. Taken from Reddit WorldNews Channel (/r/worldnews), the top 25 news articles for each date were extracted.

- Source: https://www.kaggle.com/lseiyjg/use-news-to-predict-stock-markets
- Dimensions: 73608 x 2.
- Columns:
  - Date: date of news article
  - News: extracted news article from Reddit WorldNews channel
- Issue: Some of the extracted news data is in string format and other dates are in byte code but saved as string.

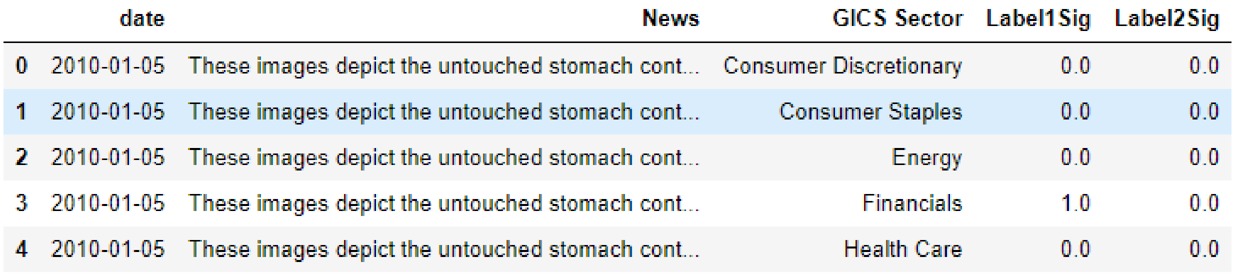


	Date	News
0	2016-07-01	A 117-year-old woman in Mexico City finally re...
1	2016-07-01	IMF chief backs Athens as permanent Olympic host
2	2016-07-01	The president of France says if Brexit won, so...
3	2016-07-01	British Man Who Must Give Police 24 Hours' Not...
4	2016-07-01	100+ Nobel laureates urge Greenpeace to stop o...

Final Dataset: cache/data.csv

Combined Aaron7Sun's Daily News for Stock Market Prediction and Dominik Gawlik's New York Stock Exchange dataset on Kaggle. Based on a given date, aggregates the news into one column and then groups the stocks based on GICS sector and labels the data based on 1 or 2 standard deviations from the mean.

- **Dimensions**: 17985 x 5.
- Columns:
  - Date: date of news and stock data
  - News: aggregation of the top 25 news by date
  - GICS Sector: Sector that stocks were grouped by
  - Label 1 Sigma: Classifies stock sector if 1 standard deviation away from mean
  - Label 2 Sigma: Classifies stock sector if 2 standard deviation away from mean
- Dates: (1/5/2010-7/1/2016). Data taken daily.
- Dataset:



	date	News	GICS Sector	Label1Sig	Label2Sig
0	2010-01-05	These images depict the untouched stomach cont...	Consumer Discretionary	0.0	0.0
1	2010-01-05	These images depict the untouched stomach cont...	Consumer Staples	0.0	0.0
2	2010-01-05	These images depict the untouched stomach cont...	Energy	0.0	0.0
3	2010-01-05	These images depict the untouched stomach cont...	Financials	1.0	0.0
4	2010-01-05	These images depict the untouched stomach cont...	Health Care	0.0	0.0

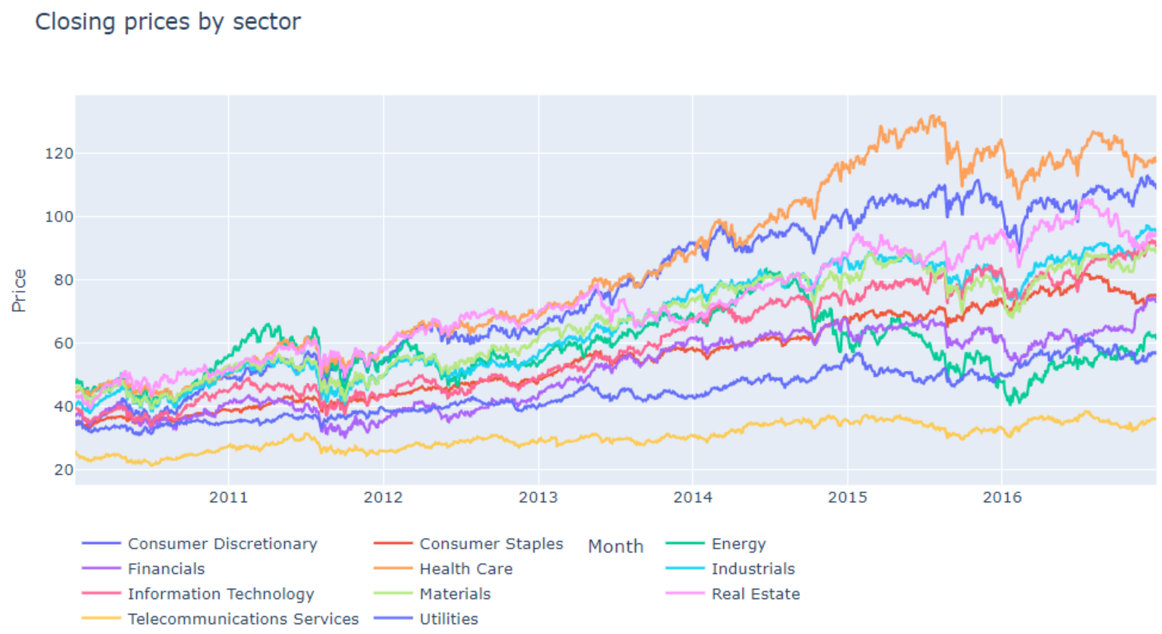
- Descriptive Statistical Analysis:
  - Label 1 Sigma Counts: Sell (-1): 13.3%, Buy (1): 12.7%, Hold (74.0%)
  - Label 2 Sigma Counts: Sell (-1): 3.31%, Buy (1) 2.12%, Hold (94.56%)
- Issues:
  - The classification of the stock data using standard deviation does not completely consider the variability of the stock over time. While 2010-2016 might have only slight volatility whereas 2017 might see an influx of volatility. This will result in more or less values being classified as 1 or 2 standard deviations away.

- Data is imbalanced. 1 sigma is only classified 26% of the time, the rest is held. 2 sigma is even more imbalanced with only 5% of the data being classified.

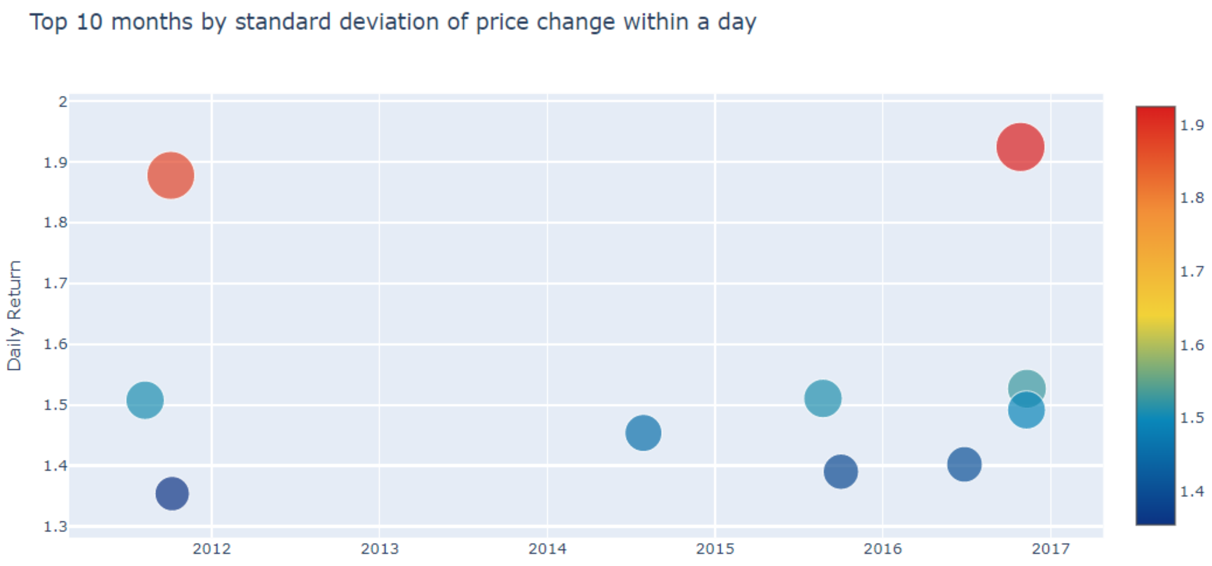
2.2. Exploratory Visualization

I visualized the difference in the closing stock prices by sector and date. It helps me compare the variance of the stock market by date. When visualizing the stock sectors, it shows there is a strong correlation of the sectors as the increase and decrease follow the same path. When you break it down by 10 energy stocks you can see the variance even more.

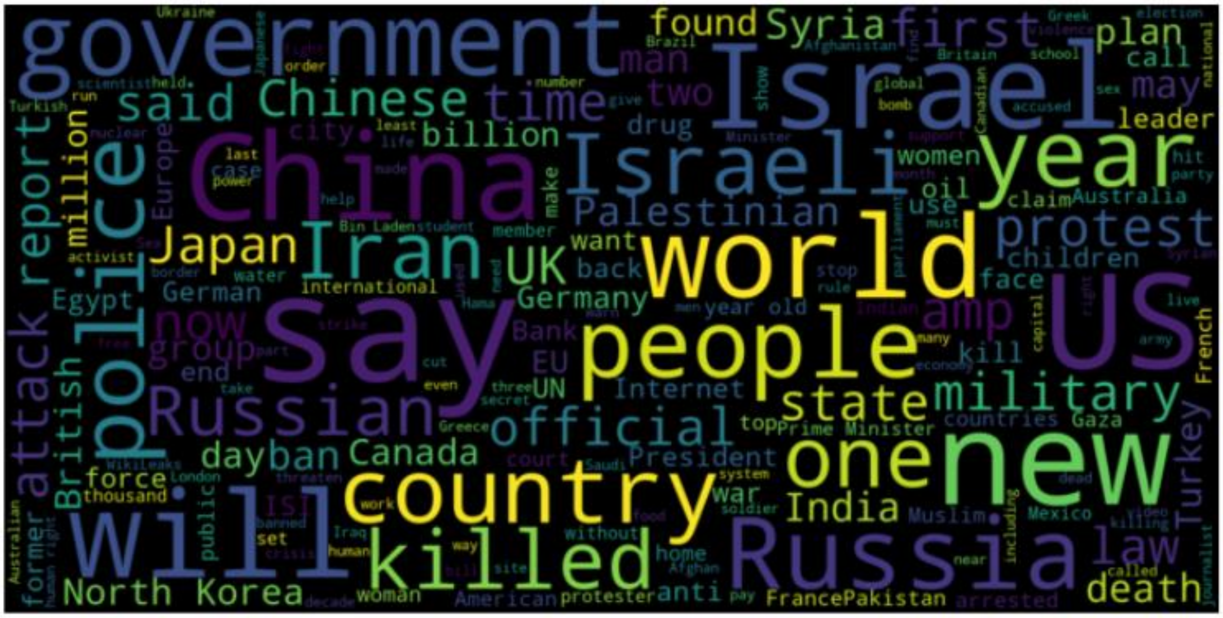
Exploring the stock sectors help us see how the variance changes over time.



I also used a heat map to look at the top 10 months where the standard deviation was highest. This shows where the changes are strongest. In visualizing the standard deviation, it helps to show where the greatest variation in price occurs. It shows me that the greatest variance occurred in late 2011 and late 2016. It worries me as the time-based split of the training set might miss the late changes of standard deviation in 2016 and 2017.



Use a word cloud to see if there are any words that have a strong correlation to stock sectors going up or down. A word cloud helps to see words that have the strongest correlation to the labeled data. Here you can see articles in regards to US, China and Israel had the greatest affect on the stock market.



2.3. Algorithms and Techniques

2.3.1. Bag of words (CountVectorizer) and term frequency (TF-IDF)

The first technique we will use is to fit the news articles into a bag of words and term frequency. These techniques help to convert the articles into a numerical value. It helps to classify the words that have are common in a small group of documents but is not frequent in the articles overall. This helps to extract the words that cause the stock market to go up or down.

2.3.2. Undersampling

Since we are only looking at values that are 1 standard deviation away from the mean, we will likely see less than 35% (1 std dev) of the data being classified. Therefore, 65% of the data will not be classified as buy or sell. By undersampling, we should see that the buy, sell and hold of the stock prices are evenly classified. The issue with undersampling is that it will result in more classifications but possibly less precision.

For the baseline, I ran multiple undersampling methods including Nearest Neighbor, Tomek Links, Near Miss, Cluster Centroids and Random. None of the methods performed well, but Cluster Centroids and Random Under Sampling performed the best.

2.3.3. Time Series Split

Since we are making prediction of future stock prices based on historical prices, we cannot use a random split as that will not tell us how we might perform in the future. So we will use three different methods to test the time based split. First, we will use a cutoff date that will basically split the training and test dataset into a 70/30 ratio. Second, we will use an expanded window which will increase the dataset and date range with each iteration. Finally, we will use a rolling window which will keep the size of the dataset constant but will move the date range by a set step with each iteration.

2.3.4. Model Classifiers

Based on the baseline dataset, Decision trees seemed to be the most effective at classifying our training data. In this project, we will be comparing two specific decision tree classifiers; Decision Tree and Gradient Boosting Machine.

Decision Tree classifier builds a decision tree that prioritizes based on the purity of the node/leaf. A gini score of zero means that the node is pure and only 1 class of samples exist. Whereas a score greater than zero means that samples contain more than 1 class. The strength of Decision Tree is it is must easier to tune and harder to overfit[iv].

On the other hand, gradient Boosting builds tree one at a time where each tree helps to correct errors made by previously trained trees. Gradient Boost use loss functions to get to determine nodes/leafs. Boosting does well with highly unbalanced datasets like we have in this case with stocks. The weakness in Gradient Boosting is that it is more sensitive to overfitting due to noise. Since we are grouping 25 articles worth of words per day. There is a lot of noise, maybe only 1 article affects the stock sector on a given day[v].

We will use several ScikitLearn algorithms and XGBoost to see which ones had the best ability to classify the training data. In this case we will try a variety of classifiers such as Logistic Regression, Linear Discriminant Analysis, Decision Tree Classifier, Multinomial Naïve Bayes, LGBM Classifier, Support Vector Machine, Random Forest and Extreme Gradient Boost.

Based on the Baseline, Decision Tree Classifier and Extreme Gradient Boost did the best in classifying the baseline data. They will be used for analyzing the data.

2.3.5. NLP methods

Use Natural Language Process tools like NLTK and Spacy to preprocess the text. This will include stripping html tags, special characters, remove stop words, stem the words and lemmatize them. Use spacy to extract named entities and parts of speech tokens.

2.3.6. Preprocessed Sentiment

Use prebuilt algorithms that have a generated sentiment score to see if that will better classify. Use Afinn and TextBlob to generate sentiment, polarity and subjectivity.

2.4. Benchmark

The benchmark we will be using for this project will be the Dow Jones score from the original dataset combined with a rolling window time series split and an Extreme Gradient Boost classifier. Similar to the calculated dataset we will also generate a 1 and 2 standard deviation from mean (1/2 sigma) label to determine whether a stock should be bought, sold or held

3. Methodology

3.1. Data Preprocessing

3.1.1. Categorize stock prices

Rather than run the stocks as a regression problem we will change it to a classification problem. Should the stockbroker buy, sell or hold the stock. To normalize the data, we will do one of two things. First, we will transform the open and close values to a Daily Return which will be the difference between the open and close. A positive value means the stock went up and a negative value means the stock went down.

Next, we will determine if the change in the stock price surpassed our control threshold that day to be a good stock to buy or sell. The control threshold will be 1 or 2 standard deviations away from the mean. 1 standard deviation away (1 sigma) means that 68% of the labels will be 0 and that 32% will outside the threshold with 16% with a label of 1 and 16% with a label of -1. 2 standard deviations (2 sigma) will result in 95% of the labels being 0.

3.1.2. Group by GICS sector

Utilize DataFrame to categorize the stocks by GICS sector. To factor for the differences in stock values and the number of stocks by sector, we will use the average of all the stocks in a given sector.

3.1.3. Combine News articles

There are 25 World Daily News articles by date. Rather than determining whether an article is relevant or not to the stocks we will combine/transform all the news articles by date into 1 big article.

3.1.4. Text Preprocessing

Some of the articles are in string format and others are in byte string format. We will preprocess the text to remove the byte string.

3.1.5. Bag of words

Extract text features from news articles by generating a bag of words. Utilize both Count Vectorizer which counts the n-grams by document and Term Frequency – Inverse Document Frequency to weighting the terms based on the fewest words in the entire document corpus.

3.1.6. NLP methods

Cleanup the text and normalize it so only the most relevant words are used in the classification. This will remove unimportant words, symbols, accents and stop words that have no relevance to the sentiment of an article. We first strip html tags, accented characters, special characters and stop words. Then we find the stem of the word so Russia and Russian have the same root word.

3.1.7. Sentiment Analysis

Use lexicons to extract sentiment scores rather than determining by news articles. We will use 2 lexicons; AFINN and TextBlob. AFINN is an easy and simple to use lexicon that helps determine whether the news article is positive, negative or neutral. To normalize the sentiment we will classify the overall sentiment rather than keeping a numerical value. TextBlob is similar to AFINN but it gives you both polarity and subjectivity. Polarity gives a positive/negative score and subjectivity which compares objectivity vs subjectivity.

3.2. Implementation

3.2.1. Methods

Methods I used to help simplify my modeling and text classification.

3.2.1.1. Vectorize Text

Receives a vectorizer such as Count Vectorizer and TfidfVectorizer and transforms both the training and test set. The 1 flaw with this method is it did not factor the fit into the transform which skewed many of my tests as it was using the fit from the previous method.

3.2.1.2. Train model

Uses a given classifier and fits the training data to the model. We then take that model and run predictions on the training set. For the original run, we will use default metrics from the classifier.

3.2.1.3. Test Score metrics

The score test taken in a name/label and then runs a couple standard Sklearn scoring metrics against the predictions for comparison. The one metric that is atypical is Cohen's kappa which measures total accuracy vs random accuracy. Cohen's kappa was a good way to have a single score that took both accuracy and precision into its equation. It also is effective for an imbalanced and multiclass dataset.

As I was running through my implementation, I realized that the classification was only good at determining if all values were accurate. However, I did not really care if we predicted hold and it should have been bought or sold. In the same token, I was not worried about stocks that were bought or sold and only resulted in a hold/no change. The only loss is the transaction fee that day. Finally, I should have created an init method to initialize the DataFrame rather than typing it in.

3.2.1.4. Stock scoring metrics

So instead I created my own accuracy/precision score based on whether the prediction would have resulted in a good or bad transaction. I first classified the scores based on buy, sell or hold. If I predicted a purchase and it resulted in a profit it was a good buy. On the other hand, if I predicted a purchase and it resulted in a drop in stock then that was a bad buy.

The difference between Sklearns classification and mine was that if I predicted a purchase and it resulted in no profit or loss then I categorized it as a Bad miss. Also, if I predicted a hold or do nothing and the stock sector made a profit or loss then I classified it as a Bad hold. While bad holds are a lost opportunity to make or save money, I would rather more false negatives then false positives. I would rather have 5 extremely good buys then 100 average buys that I have no confidence in.

There were many complications with this method. The confusion matrix assumed that each actual and prediction had 3 labels (-1,0, & 1) but that was not always the case. I was also getting an Undefined Metric warning because 1 of the classifications was missing. My original selection retrieved the values by index but that resulted in index out of range errors. So I had to use the get method to retrieve a value and return a default value if key does not exist. I also had to account for a divide by zero problem

Good:
   Buy - Indicates predicted purchase was a good choice and would have made a profit
   Sell - Indicates predicted sell was a good choice and would have saved money
   Hold - Indicates that holding the stock or not purchasing the stock was a good choice
   Miss - 0 value. Only applies to bad section.
   Total - Total good choices
Bad:
   Buy - Indicates predicted purchase was a bad choice and would have resulted in a loss
   Sell - Indicates predicted sell was a bad choice and would have resulted in loss of profit
   Hold - Indicates that predicted hold was a bad choice and would have resulted in missed profit or loss of money
   Miss - Indicates that predicted buy/sell transaction resulted in no change in stock price
   Total - Total bad choices
Total - totals of the good and bad choices for each column
True% - ratio of good to bad choices

3.2.1.5. Time train test split

I built a helper method for time based split algorithms. It took three values that helped extract a training and test dataset. The training set was between the start date and split date and the test set was between the split date and stop date. I was forced to create this split by date because I could not split by index. Each date could have up to 11 labels based on the GICS sectors and it would not be good to have it split in between the GICS sectors. Therefore, I did the time-based splits based on the unique dates and then used those dates to split the training and test set.

3.2.1.6. Sector helper methods

Once I determined the best time-based split and algorithms I would like to use. I created a helper method where all I had to do was pass in the GICS sector and it would output the scoring metrics. I would then print the scores. In hindsight, I could have made it more extensible but for clarity sake I just wanted to make it easier to read. I probably should have printed a helper for printing out the scores instead of having 3 or 4 lines for each score.

3.2.2. Classifiers

3.2.2.1. Baseline

Baseline uses the Dow Jones data cross-referenced against Extreme Gradient Boost and a rolling window. No undersampling was done.

Accuracy: 73.7%, Precision: 9.2%, and Cappa: -12.6%

3.2.2.2. Run Decision Tree Classifier with Randomized split

I split the stocks at 33% split of test to training set. I then tried both CountVectorizer and TfidfVectorizer to determine which bag of words had the best result and TF-IDF had a better precision and Cappa score. I then used Decision Tree Classifier to classify the data.

Accuracy: 81.1%, Precision: 64.5%, and Cappa: 96.8%

The Decision Tree performed extremely well seeing that the precision and accuracy were both high resulting in a good Cappa score. The only issue is that the precision was low and it did not factor time based splits.

3.2.2.3. Run XGBClassifier with Randomized split

Again, split the stocks at 33% split of test to training set. I used the TfidfVectorizer to extract features from the news articles and Extreme Gradient Boost decision tree classifier. I spent most of my time using the randomize split which hurt my original observations. I should have used a time based split from the beginning and that would have made it easier to come to a classification.

Accuracy: 77.1%, Precision: 81.0%, and Cappa: 98.6%

The Gradient Boost performed very well here with the precision being higher than the accuracy which helped push up the Cappa score. In terms of stocks being more precise is better and so XGBoost looks like the best classifier to use based on random split.

3.2.2.4. Split stocks by sector

I used a for loop to iterate through all the GICS sectors. I then split, transform and classified the dataset. I did however not account for the time-based split.

Name	Accuracy	Precision	Cappa
Materials	74.6%	-	-
Health Care	71.9%	-	-
Consumer Staples	73.5%	16.6%	-
Industrials	74.3%	-	-
Financials	74.6%	-	NaN
Real Estate	73.9%	9.2%	NaN
Information Technology	69.8%	-	NaN
Consumer Discretionary	74.4%	49.3%	NaN
Energy	74.6%	-	NaN
Telecommunications Services	72.4%	15.7%	NaN
Utilities	73.5%	-	NaN

In this case, you can see that the random split had a hard time calculating precision and cappa and this may be because there was not enough data to make a determination.

3.2.2.5. Undersampling

I attempted 2 different undersampling strategies; RandomUnderSampler and ClusterCentroids. Both were essentially the same although Random had some variability. I ran into issues with ClusterCentroids as it did not like values in the Dataframe format. I had to transform the training set to an array to make it work.

Accuracy: 61.5%, Precision: 34.9%, and Cappa: 83.0%

Undersampling was able to improve recall but it dropped precision to 35% which makes undersampling not a good use case for this scenario. Undersampling reduces overfitting but it also increases classified labels at the expense of false negatives. For the purchase of a stock it is better to not classify instead of misclassify and lose money.

3.2.2.6. 70/30 Split training/test set by cutoff date

Split the training and test set sequentially instead of randomly. Have it where 70% of the dataset is in training and 30% is in the test set. The issue I found is that the split did not factor in that I had 11 categorized sectors so I had to create a helper method. Found the unique set of dates and then extracted the 70th percent of the date length and got the index of that cutoff date.

Another issue that came up was that XGBoost had problems classifying the dataset with the time series split. It only was able to classify two of the labels (hold, buy).

Accuracy: 73.2%, Precision: 44.7%, and Cappa: 0.0%

Here the XGBoost was able to perform the split and kept accuracy high but unfortunately the precision was too low to make a classification.

3.2.2.7. 70/30 Split training/test set with Decision Tree by cutoff date

Split the training and test set sequentially instead of randomly. Have it where 70% of the dataset is in training and 30% is in the test set. I ran it with the Decision Tree classifier. The Decision Tree was able to classify all three labels but unfortunately the True Buys/Sells were equal to the False Buys/Sells which made it no better.

Accuracy: 56.1%, Precision: 12.1%, and Cappa: -0.07%

The Decision Tree did much worse than Gradient Boost here with precision dropping to 12% and Accuracy to 56%.

3.2.2.8. XGBoost with expanded window

Split the training and test set sequentially using TimeSeriesSplit. As we iterate through the split we continually add additional data to the training set. While the expanded window did better it did take a lot longer to run and the ending test set had a poor precision and cappa score.

Name	Accuracy	Precision	Cappa
10/28/2010	73.4%	25.6%	-13.0%
8/19/2011	66.4%	64.5%	67.3%
6/12/2012	80.1%	0%	NaN
4/8/2013	78.9%	6.5%	0

1/28/2014	78.1%	8.7%	NaN
11/17/2014	75.6%	0%	NaN
9/10/2015	70.9%	51.8%	NaN

Expanded window with XGBoost did very well towards the end with over 50% precision and 71% accuracy but the data could not determine a Cappa score.

3.2.2.9. DecisionTree with expanded window

Split the training and test set sequentially using TimeSeriesSplit. As we iterate through the split we continually add additional data to the training set.

Name	Accuracy	Precision	Cappa
10/28/2010	55.9%	17.6%	30.0%
8/19/2011	52.6%	25.0%	25.4%
6/12/2012	57.9%	7.9%	-3.3%
4/8/2013	63.8%	12.0%	6.7%
1/28/2014	55.0%	8.2%	-14.8%
11/17/2014	57.7%	10.2%	-6.7%
9/10/2015	52.6%	15.9%	14.9%

We tried Decision Tree with an expanded window and we were able to get a Cappa score but the precision to 16% and 52% accuracy.

3.2.2.10. XGBoost with rolling window

Split the training and test set sequentially using a rolling window. I setup the rolling window training set as 90 days of stocks and the test dataset as 30 days after. I ran the algorithm a couple times going up to 15 months and going down to 30 days for the training set and 90 days seemed to have the best results. The issue I ran into was that XGBoost with rolling window took a long time to run and it had to be run overnight.

Name	Accuracy	Precision	Cappa
1/5/2010	38.2%	19.9%	NaN
5/14/2010	79.4%	12.2%	-14.3%
9/22/2010	71.8%	0.0%	NaN
1/31/2011	70.6%	0.0%	NaN
6/9/2011	56.7%	12.4%	-51.0%
10/17/2011	90.0%	0.0%	NaN
2/27/2012	79.7%	0.0%	0.0%
7/5/2012	82.1%	0.0%	NaN
11/13/2012	77.0%	0.0%	NaN
3/26/2013	84.2%	0.0%	0.0%
8/2/2013	84.2%	0.0%	NaN

12/10/2013	84.2%	0.0%	NaN
4/22/2014	69.4%	8.6%	NaN
8/28/2014	64.5%	0.0%	NaN
1/7/2015	82.4%	0.0%	0.0%
5/18/2015	66.7%	6.5%	0.0%
9/24/2015	62.7%	17.8%	NaN

Here the rolling window once again failed with the accuracy being high but no values for the Cappa score as a result of limited predictions.

3.2.2.11. DecisionTree with rolling window

Split the training and test set sequentially using a rolling window. I setup the rolling window training set as 90 days of stocks and the test dataset as 30 days after.

Name	Accuracy	Precision	Cappa
1/5/2010	29.1%	18.1%	-38.9%
5/14/2010	47.0%	5.9%	-5.6%
9/22/2010	59.4%	5.8%	-22.7%
1/31/2011	60.6%	19.2%	11.9%
6/9/2011	50.9%	13.1%	-41.3%
10/17/2011	68.5%	9.1%	66.7%
2/27/2012	68.8%	9.9%	-19.4%
7/5/2012	52.1%	15.0%	43.1%
11/13/2012	70.6%	4.0%	0.0%
3/26/2013	67.9%	1.4%	-66.7%
8/2/2013	79.4%	21.7%	58.8%
12/10/2013	73.9%	5.7%	20.0%
4/22/2014	62.1%	17.4%	37.0%
8/28/2014	60.3%	40.1%	79.5%
1/7/2015	52.1%	5.5%	35.3%
5/18/2015	59.1%	23.7%	-10.5%
9/24/2015	51.2%	24.9%	26.2%

Decision Tree did a lot better than Gradient Boost here with many of the rolling dates getting good classification. However, the percentages were too erractic to determine whether the 90 day window was a good decision.

3.2.2.12. DecisionTree rolling window by sector

I extracted only the dataset by sector and then split it into the rolling training and test sets. The features were then ran through the TF-IDF transformation and then fit against the DecisionTree model.

Name	Accuracy	Precision	Cappa
Materials	58.2%	8.1%	-6.9%

Health Care	52.5%	8.9%	-11.0%
Consumer Staples	57.5%	18.6%	0.0%
Industrials	62.0%	15.7%	1.2%
Financials	63.5%	14.2%	30.7%
Real Estate	68.4%	12.3%	3.5%
Information Technology	51.0%	13.4%	-0.5%
Consumer Discretionary	54.5%	13.5%	24.2%
Energy	58.4%	6.4%	-20.0%
Telecommunications Services	55.1%	10.4%	-16.1%
Utilities	63.1%	14.4%	-14.3%

Since we could not get any valid values from XGBoost, we decided to run the rolling window by sector using Decision Tree. In this case you can see 2 sectors that had a slight correlation to the world news and both make sense. Financials is governed by the health of the international market as well as Consumer Discretionary as these items are usually imported in and are an indicator of positive global trade.

3.3. Refinement

3.3.1. Reprocess the news article data to generate sentiment scores

With the weakness of the time-based scores as well as the visualization of the word cloud for positive and negative words, and the display of the feature extraction. I noticed that word clouds for positive and negative words were essentially the same.

The same countries were extracted and created the most volatility; China, North Korea, Saudi Arabia, USA. The problem was that it affected both positive and negative scores. So my theory was the articles were the right ones but we did not factor the polarity of the adjectives in the article with these countries.

The other issue I saw was that there were a lot of words that became features that should not have been as they were stop words that could have been filtered out. So I wanted to refine the data by extracting meaningful tokens (nouns, adjectives and verbs) that when grouped may help with the association.

3.3.1.1. NLP methods

Use natural language processing to normalize the corpus of the articles and only extract the most significant words from the article. The issue I saw was that the byte strings were not getting cleaned so I had to remove them prior to the corpus.

3.3.1.2. Afinn lexicon

Afinn lexicon is one of the simpler sentiment analysis classifiers. It generates a summed numerical value based on the words in the document. However, the issue I found is that a highly positive or negative article was skewing the average for the 25 articles in that same day. So instead, I decided to classify each of the 25 articles as positive, negative and neutral. I then averaged the sentiment from the 25 articles to generate the sentiment score from that day and generated an average for each day of positive, negative and neutral articles.

3.3.1.3. TextBlob lexicon

TextBlob lexicon has a sentiment score similar to Afinn (polarity) but it also generates a subjectivity score.

3.3.1.4. Extract named entities

I used Spacy's named entity extraction to break down articles into entity types. I was then able to eliminate any entities that did not factor into our classification. So I removed all numerical and date/time data. I then put these entities into a column. The problem I have with the entities is the same with the features I saw in the word cloud. Using this feature alone will not be able to identify the positivity or negativity of a stock.

3.3.1.5. POS Taggers

I used Spacy's POS taggers to extract the words that I hoped may be more associated. I also extract verbs and adverbs because those words may be correlation to the positive and negative sentiment. The issue I realized with Spacy's POS tagger is that a sentence has too many fluff words in between the main idea in a topic. Also, those words such as adjectives give the sentiment rather than the word itself.

4. Conclusion

4.1. Model Evaluation and Validation

The final model I selected was the DecisionTreeClassifier because XGBoost was failing to make enough positive and negative predictions. On the other hand, Decision Tree was able to make some predictions with a rolling window.

During the baseline test, there was a higher accuracy rate but the precision was in single digits which resulted in a Negative Cappa score showing only random accuracy. The baseline Decision Tree scores were as follows: 73% accuracy, 7.6% precision and -12.6% Cappa. When running the rolling window with Decision Tree on our modified dataset, you could not pick out the variances when looking at accuracy and prediction alone. But when you added the Cappa score it became apparent that 2 sectors stood out; Financials and Consumer Discretionary.

Both Financials and Consumer Discretionary had the highest Cappa scores (>20%), which would make sense as they are both governed by global trade. Financials are affected by the news of the world markets and Consumer Discretionary is usually paid overseas and brought to the US and when the market is low consumers reduce their spending. In the same sense the majority of the negative markets (<-10%) were mainly domestic sectors that had little influence on the world market such as Telecommunications, Utilities, and Health Care.

The single outlier in the sectors is Energy. One would have thought that the Energy stock market would be more affected by World News but unfortunately it looks like it had the opposite effect. As changes in the world news did not reflect changes in the energy stocks.

In addition to the Energy sector, we also saw a decrease in the accuracy of the stock classification. Further analysis would be warranted to see why energy did not correlate and why the results were less accurate at the expense of precision.

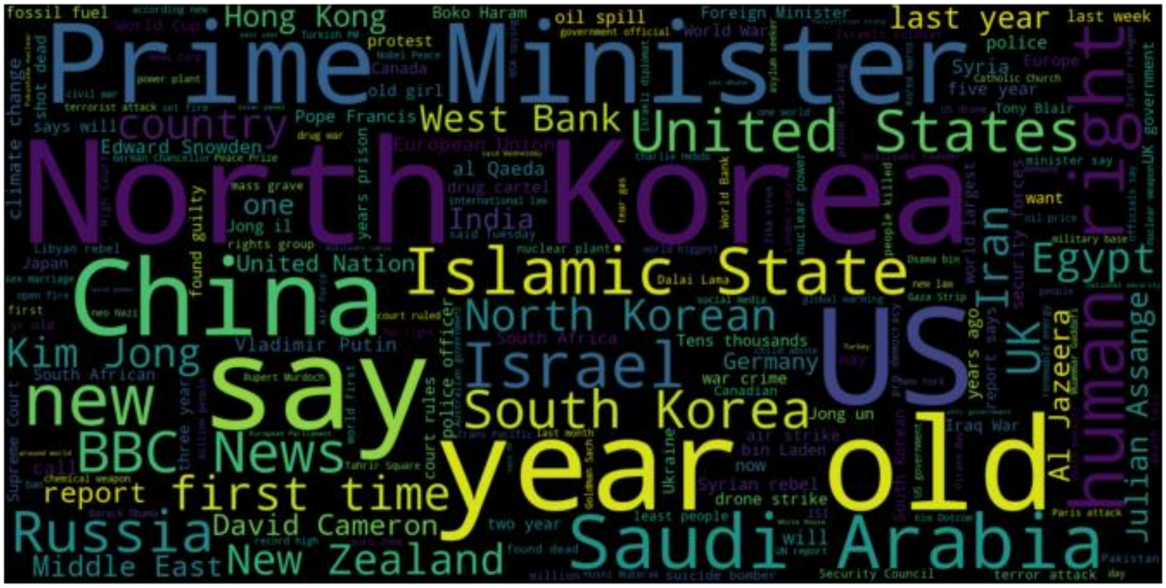
4.2. Reflection

Overall, the accuracy and precision were never high enough that a stock broker would make an educated decision to buy/sell a stock based on the World News sentiment. It looks like the daily news needs other features to balance out in order to get a better determination.

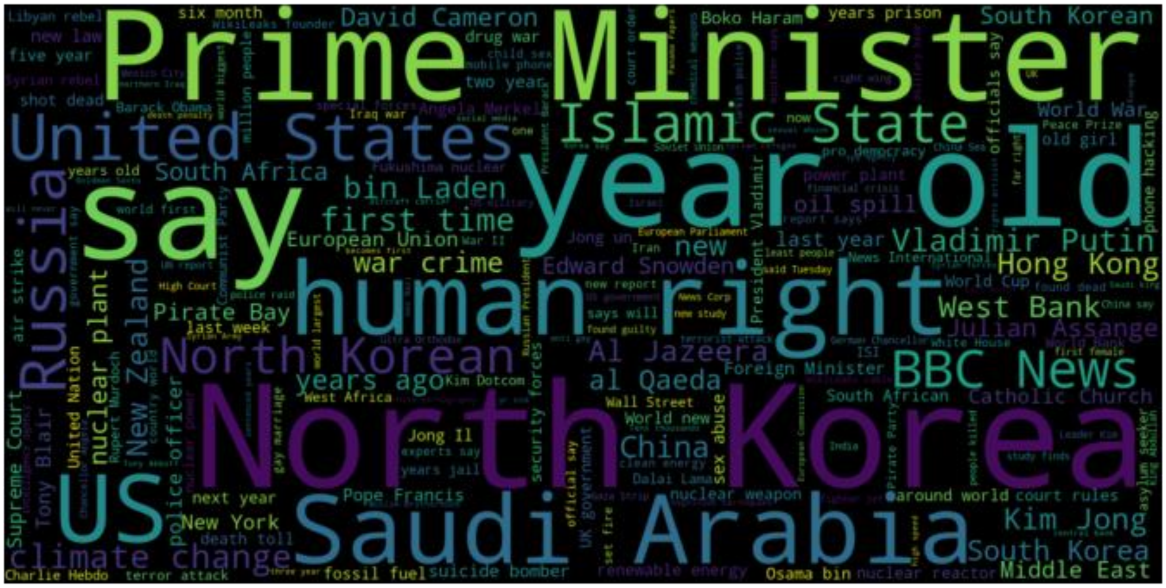
What I found interesting about this project was how much more difficult it is to predict something in the future as opposed to predicting a random prediction. 1 of the fatal flaws is the mean and standard deviation changes over time. The standard deviation for 2010 will not be the same as 2019 as the prices and the volatility changes by year.

I also found it interesting that the several named entities that showed up both in the positive and negative sentiment. This goes to show that certain countries have a great influence on the market but the influence could be both positive and negative. The problem I had is combining these entities with the sentiment to determine if the stock price would go up or down.

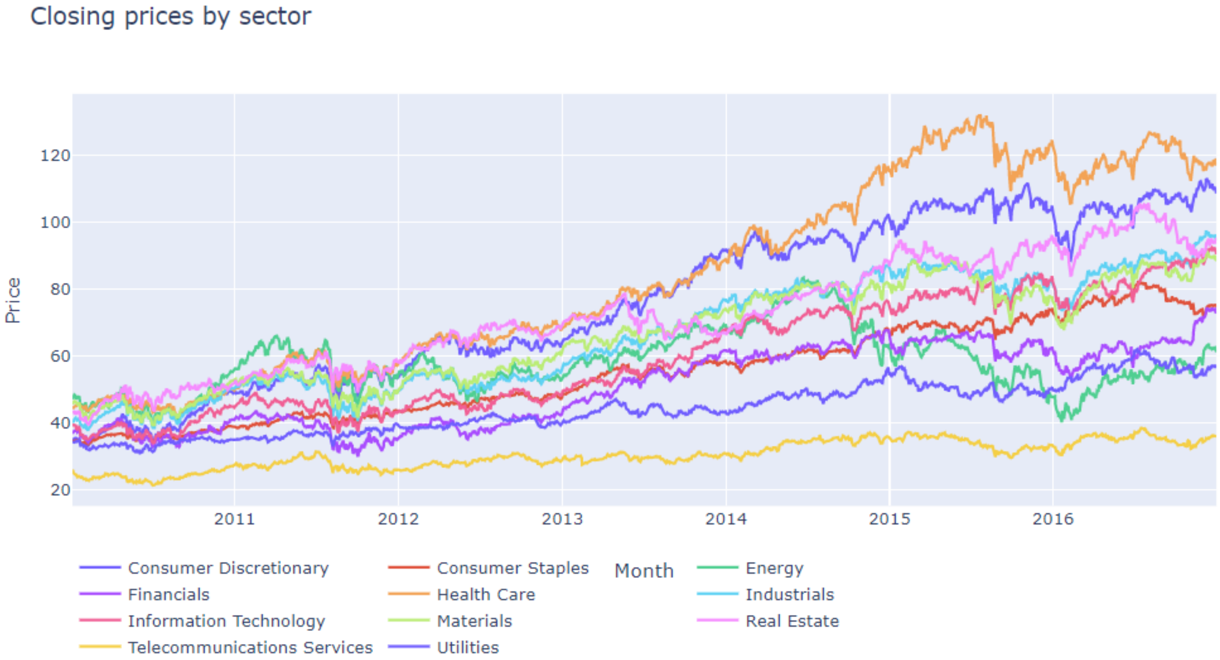
Positive text



Negative Text



While I thought that certain sectors would be able to extract events that showed a correlation to the stock price going up or down, the sectors never diverted from each other to show a true trend. This could be seen in the visualization of the 11 GICS sectors. The market does not like uncertainty and when uncertainty occurs it affects all stocks instead of only impacting one sector



4.3. Improvement

The one improvement I would do is find a better way to extract the text features and the sentiment score to help the model better classify the data. If I knew how to run the features through a neural network where the first stage extracted the entities and then used those entities were to determined which article to use and pull the sentiment from those article I think you might see a greater impact of World News to stocks. I think if I knew more about Keras or another Deep Learning library it would have helped to develop a multi-part approach to the factors that affect the stock market.

I also did not factor the price of the stock over time; 1, 5, and 10 days after the news article that day. Sometimes the effects of news 1 day will have lasting effects. I also was unable to give weight to the articles by views or relevance or break down the articles by category.

Since I was unable to come to any real solution, I think there is room to benchmark my solution and build upon it. There were multiple paths that someone can go by to try and associate the News articles with the stock market.

[i] Sun, J. (2016, August). Daily News for Stock Market Prediction, Version 1. Retrieved 7/29/2019 from https://www.kaggle.com/aaron7sun/stocknews

[ii] Gawlik, D. (2017, February). New York Stock Exchange. Retrieved 7/29/2019 from https://www.kaggle.com/dgawlik/nyse

[iii] Kampakis, S., Dr. (2016, May 8). Performance Measures: Cohen's Kappa Statistic. Retrieved 7/29/2019 from https://thedatascientist.com/performance-measures-cohens-kappa-statistic/

[iv] Ceballos, F. (2019, February). Scikit-Learn Decision Trees Explained. Retrieved 9/2/2019 from https://towardsdatascience.com/scikit-learn-decision-trees-explained-803f3812290d

[v] Ravanshad, A. (2018, April). Gradient Boosting vs Random Forest. Retrieved 9/2/2019 from https://medium.com/@aravanshad/gradient-boosting-versus-random-forest-cfa3fa8f0d80