Machine Learning Engineer Nanodegree

Capstone Project

George Vargas

December 2018

I. Definition

Project Overview

The idea of investing in the U.S. stock market is often met with polarized opinions. On one hand, hedge funds are leveraging data to make enormous amounts of money, but on the other hand, individual investors tend to lose large sums of money in their independent investment efforts. These vastly different experiences from these two equally different groups of investors create perceptions that match the outcome. The negative perceptions of individual investors are such, not because of a lack of confidence in the American business machine, but more so because of the nature of the market itself. Markets are unpredictable, complex, and, for the most part, emotionally driven. Due to these and other factors, the lay-investor often sees the market as a gambling establishment where losing their investment is a guarantee.

Based on these perceptions, the most logical reasoning for anyone attempting to invest in the market would be to follow in the footsteps of the hedge funds and use data to help inform investment decisions. *Renaissance Technologies* is a premier hedge fund known for hiring scientists and mathematicians over investment bankers or brokers. This attempt at shifting the concept of investing in the stock market from a financial focus to a data-driven focus allowed them to generate 98.2 percent return on investment in 2008 as the housing-market crashed and the stock market followed.[1]

Problem Statement

Following the hedge fund reasoning of data-driven investment decisions, the simplest form of investing requires buying low and selling when the price of the commodity is significantly higher than the initial purchase price. In this scenario, determining whether a specific stock will close the day up or down from its current position would aid in making a successful investment decision for said day.

Now, consistently determining the directionality of stock prices is a non-trivial task and can be approached through a multitude of facets. Historically, investors have mainly used three philosophically different approaches in an attempt to predict the directionality of financial instruments: technical analysis, fundamental analysis, and quantitative analysis.

While technical and fundamental analyses have their merits, the philosophical underpinnings of quantitative analysis have been proven to be very lucrative for many hedge funds.

As previously stated, emotions are understood to have a large influence on investment decisions within the financial markets, and as such new headlines are a prime example of the ability to change investment emotions about a particular subject. In understanding this behavior, the problem of predicting market directionality will be approached from a natural language processing perspective. The 25 most user-upvoted headlines of the day from *Reddit's* community *r/WorldNews* will be used to predict the direction in which the *Dow Jones Industrial Average* will close for said day. To keep this simple, the quantities of the directional movements will be ignored and only the directionality, "up or neutral" and "down" will be used as the dependent variable, effectively making this a binary classification problem.

Metrics

The measurement of such an experiment will be vital to understanding the model's performance and any possible enhancements or additional uses outside of the current scope. Since the problem is framed as a classification, the chosen metrics will be accuracy, precision, recall, and F_1 score. While accuracy will provide a generalized understanding of the model's performance, the F_1 score will facilitate a more significant understanding of the performance. Finally, precision and recall will shed further light on the model's performance with respect to false positive and false negative predictions. Precision, recall, and F_1 score were selected for their ability to derive meaningful conclusions from classification solutions.

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$accuracy = \frac{True\ Positives + True\ Negatives}{Positives + Negatives} \quad F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

II. Analysis

Data Exploration

The dataset that will be used to tackle the problem of directional prediction of a financial instrument comes from the online community *Kaggle*. The user *Aaron7sun* has graciously provided the community with a dataset that combines the date and the top 25 user-upvoted news headlines for the day from Reddit's community r/WorldNews.[2] The headlines included approximately number around 49,725. Three of the rows in the dataset have missing values for some of the headline columns, these will be addressed in the data preprocessing phase. The data also includes a label column that utilizes a "0" for a close down and a "1" for a flat close or a close up. The dataset spans approximately eight years from August 2008 to July 2016 and only includes days on which the Dow Jones Industrial Average, or DJIA, openly traded. The provider of the dataset recommends using data from 2008-08-08 to 2014-12-31 for training, and data from 2015-01-02 to 2016-07-01 for testing since this splits the overall dataset in an 80/20 fashion.

The time period proposed for training includes data from the crash of the markets in 2008, otherwise known as the Great Recession. This data is considered to be bearish in nature, or trending down. The data after the market bounced and onward is considered bullish in nature, or trending up. It's generally accepted that the long-term behavior of markets follow a cycle. The data suggested for training captures two, possibly three, different phases in a cycle which could negatively affect the performance of any model. Additionally, it would be interesting to see if a model trained on bullish data could achieve similar performance on bearish data without retraining. For these reasons, the data leading up to April 2009 will be held out for additional testing on the generalizability of the model.

When looking at the class distribution in the dataset Figure 1 below shows that in the bullish dataset it is fairly balanced considering that by its nature a bullish market will have more positive or neutral samples than negative. Similarly in Figure 2, the bearish class distribution can be observed and it seems like an inverse of the bullish class distribution which is also expected considering the nature of a bearish market. Both datasets seem fairly balanced respective to the period in which the data was collected.

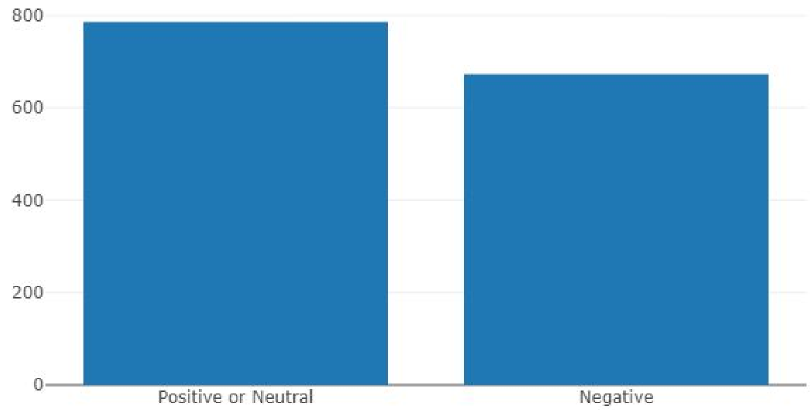

Figure 1 - Bullish set class distribution

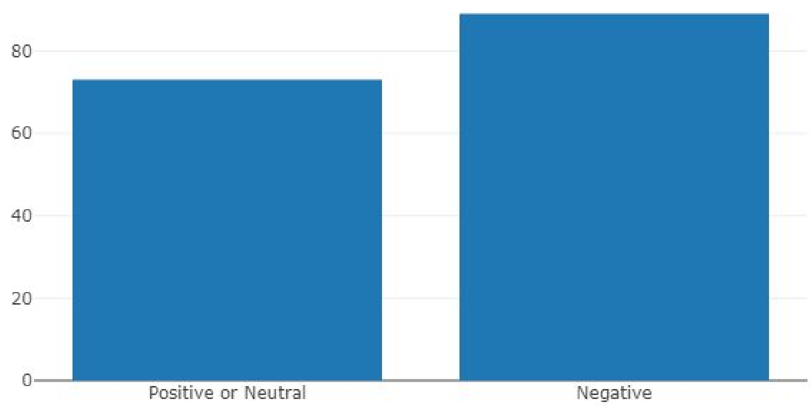

Figure 2 - Bearish holdout set class distribution

Further analysis of the distributions of headline lengths, in figures 3 and 4, yielded interesting information about the headlines and the phases from which they were captured. The distributions depicted in figure 4 clearly show that, regardless of the market phase or movement direction, news headline lengths tend to predominantly be within 40 to 80 words. Digging deeper into the headline length distributions by phase revealed that in the bearish data the average word length of a headline was 62 putting it in the middle of the previously stated range, however, the average of the bullish data headline length was 87 which sat just outside this range. Additionally, it can be noted that the longtail on the bullish headline length distribution is much fuller than the bearish longtail. Furthermore, the standard deviation of the bearish data was much smaller when compared to the bullish data, sitting at 33 and 51 respectively. Taken alone these facts could mean very little but when considered as part of a whole it seems to lend to the idea that market cycle phases and news headlines may have some relationship.

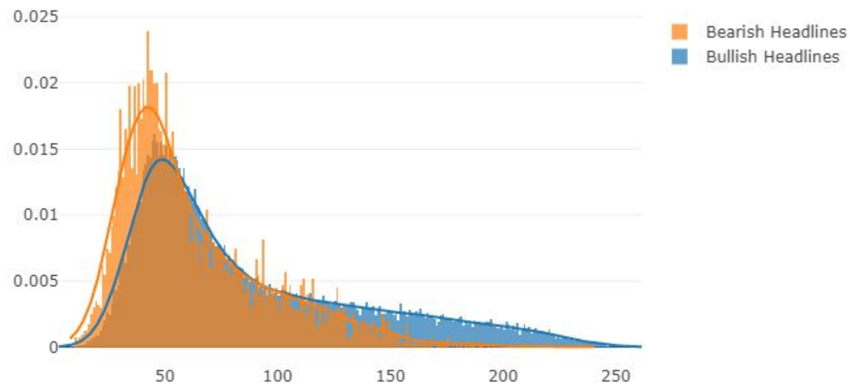

Figure 3 - Stopwords exclusive bullish and bearish headline length distribution

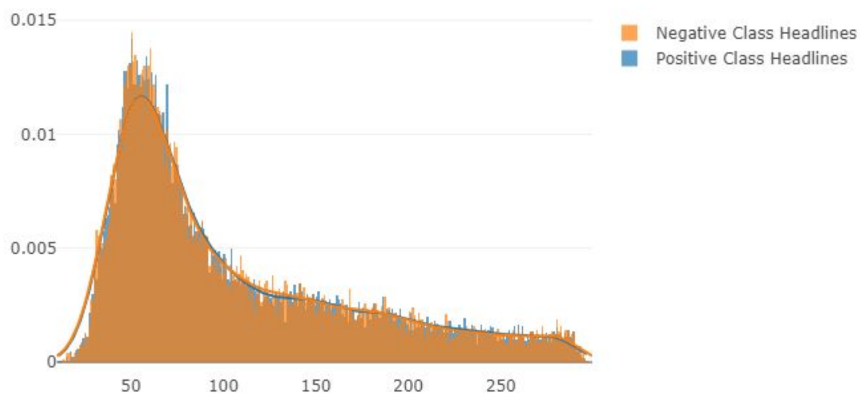

Figure 4 - Stopwords exclusive positive and negative class headline length distribution

Min	3
Max	262
Mean	87.3248245614035
Standard Deviation	51.17215810473163

Figure 5 - Stopwords exclusive bullish headline word count statistics

Min	8
Max	241
Mean	62.501234567901236
Standard Deviation	33.032126581140695

Figure 6 - Stopwords exclusive bearish headline word count statistics

Exploratory Visualization

From a cursory visualization of the bullish data set without the stopwords it can be seen that words such as "people", "world", "police", "new", "says", and "government" dominate in frequency (figure 7). Finding these words in these frequencies in the bullish dataset could lead to a basic hypothesis which considers these words as "prosperous" words in the media. Considering that "says" is the most frequent word in the bullish data it can be assumed that quoting people is very popular in a time of economic accumulation. Performing the same analysis on the bearish dataset yields an interesting initial discovery, namely, the words between the two datasets are quite similar but differ in frequency.

The bearish data has a high frequency of words such as, "us", "gaza", "israel", "israeli", "b'the" (use of the word "the" at the beginning of headlines), "says", and "war" (figure 9). The word "says" is once again in the highest frequency words but it's a bit further down the ranking than in the bullish data. The words in either set are similar but differ in frequency which may speak to the nature of the data and the sentiment in the market at that point in time. The analysis performed on the bearish data could lead to the simple postulation that words associated with war and conflict, especially at a global scale have a negative impact on the financial markets. Considering the timeframe from which the data was collected and the fact that it is significantly smaller in volume than the bullish data, it would be wise to test the previous postulation further before assuming its validity.

The least common word frequency visualizations as seen in figures 8 and 10 didn’t yield an easily recognizable pattern.

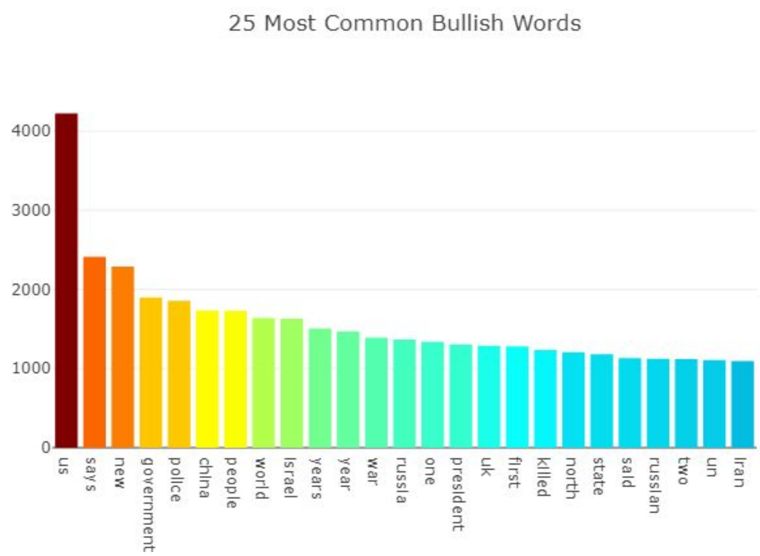

Figure 7 - Stopwords exclusive bullish most common word frequencies

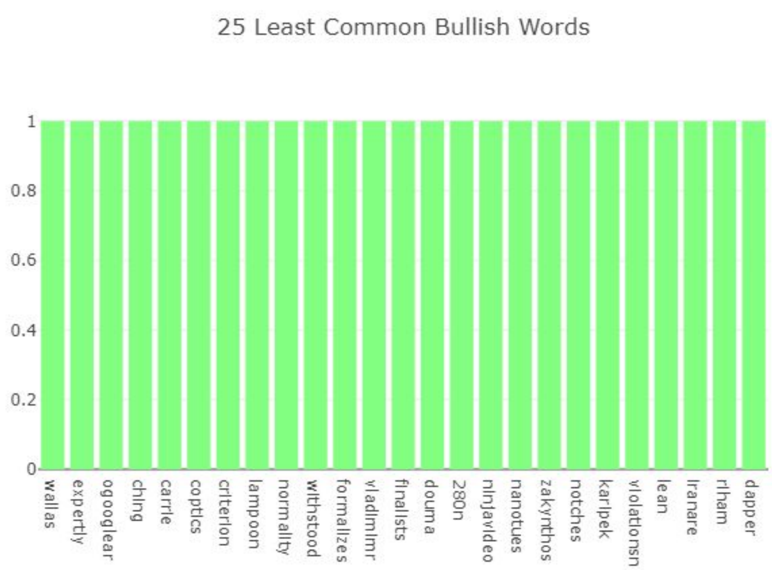

Figure 8 - Stopwords exclusive bullish least common word frequencies

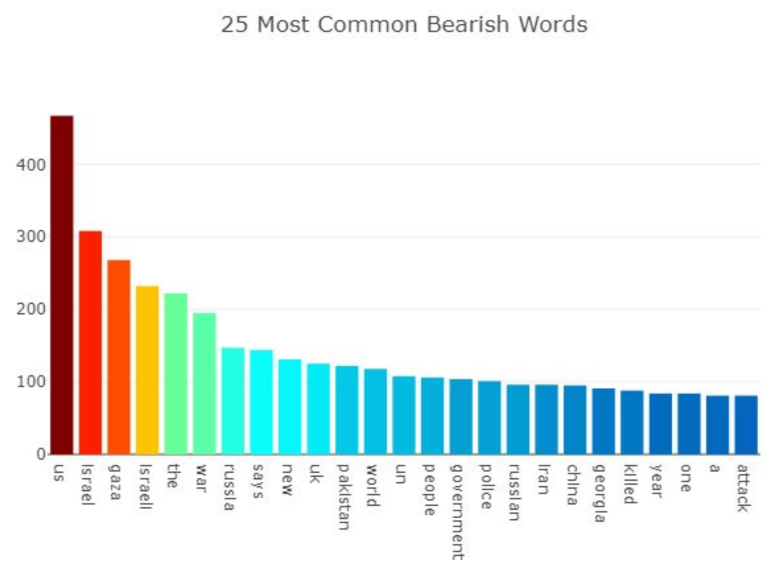

Figure 9 - Stopwords exclusive bearish most common word frequencies

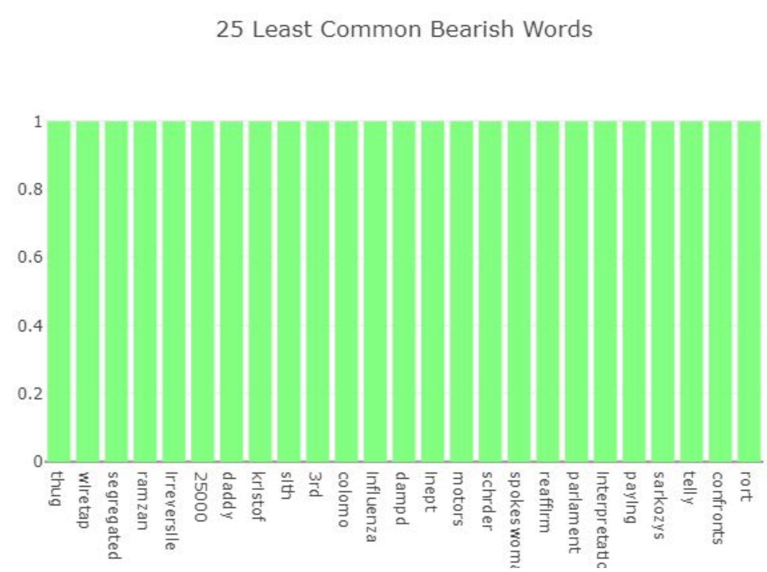

Figure 10 - Stopwords exclusive bearish least common word frequencies

Algorithms and Techniques

Considering the data contains headlines ranked by popularity in text format, the most conventional approach would be a natural language processing model. The algorithm chosen for this model is a Support Vector Machine. Support Vector Machines work by attempting to find a hyperplane that will best separate the data provided. This hyperplane is optimized by maximizing the margin, the distance between the nearest correctly classified data point of each class and the hyperplane itself. The Support Vector Machine was chosen because the data shows similarities in words across both classes which may make a linear boundary difficult to find and for which the *kernel trick*, mapping the data into a higher dimensional using dot products, may prove useful.

Based on preliminary research, the most effective method currently used for natural language processing is the conversion of words and sentences to their vector representations before training a model.[3] While there are several libraries that are popular and provide good performance for the vectorization task, the one chosen to be utilized is the *InferSent* library for the sentence vectorization with the *fastText* library used for the underlying word vectorization task.[4][5]

InferSent is a library created by Facebook's Artificial Intelligence Research team to create sentence embeddings, also known as sentence vectors. These embeddings provide a semantic sentence representation which allows the embeddings to be used in the construction of models that learn patterns based on semantic information.

The fastText library is similar to InferSent in that it is used for embeddings or vectorizations, however, the difference is that fastText works one semantic level lower and focuses on words.

The reason for utilizing fastText over *word2vec* or *gLoVe* is that fastText creates word representations from character n-grams (as opposed to skip-grams or a continuous bag of words) in a convolution-like manner instead of creating the representations from word proximity or sentences. This helps piece together a much more specific representation for each word, and also facilitates the construction of vectors for words that are outside of the training corpus which will be useful when generalizing against the portion of the dataset that was held out for testing generalization on the recession of 2008.

example = <ex, exa, xam, amp, mpl, ple, le>

Example of character n-grams for the word "example" where n = 3.

Benchmark

The benchmark models used are scikit-learn dummy classifiers utilizing the stratified and constant strategies. The stratified strategy, "generates predictions based on the training set's class distribution."[6] On the other hand, the constant strategy respects its namesake by predicting the same user defined label over and over without any other intelligence. This predictive behavior enables the ability to avoid creating models for stopword exclusivity and inclusivity since it will have no bearing on the predictions.

The reasoning for following the dummy classifier route was because of the nature of the bullish data that the model's will be trained on. Most successful individual investors will have what is referred to as a *position sizing strategy*. This strategy allows for minimized risk and the ability to profit from their trades even if only 51% of their trades are winning trades. Since the training data clearly shows a higher presence of positive classes we will use the aforementioned benchmark models in a simple attempt to exploit this characteristic.

III. Methodology

Data Preprocessing

To be able to feed the text from the dataset to a classification model, some feature transformations were required. Since the headline features in the data are in a text format they had to be transformed into vector representations, also known as word embeddings, in order to be used with any model.

FastText was used to pull the embeddings for the 100,000 most frequent words in the *Stanford Natural Language Inference/MultiGenre Natural Language Inference* datasets and load them into the InferSent class. These embeddings are pre-trained and are provided with fastText by the facebook artificial intelligence research team. Then, the entire dataset was loaded from a file and the three rows containing NaN values were dropped. The date and label columns were also removed from the data. While the label was removed for obvious training and testing purposes the label was removed to avoid the potential for the model to pick up on a sequential pattern that is found in the dates correlation with the labels.

As previously noted, the timeline of the data is a very important factor. We know that up until April 2009 the Dow Jones Industrial Average was in a downward trend as a direct result of the Great Recession. Training on data that spans two, arguably three, phases within a market cycle could lead to poor performance on unseen data from any of the phases. To address this, the data was split based on the two most apparent phases; the bear phase and the bull phase. Before splitting, all the data was changed to lowercase including the column headers, this was done for ease when utilizing keys for columns and building generic functions. After splitting, both sets were passed through a function to generate derivative datasets; one dataset contained stopwords and the other did not. The stopwords cross-referenced for this generation of derivative datasets were the English stopwords from the *Natural Language Toolkit* or *nltk* for short.[9] At this point there were four datasets: bullish with stopwords, bullish without stopwords, bearish with stopwords and bearish without stopwords.

Upon completion of an analysis of the exploratory visualization, the data was to be converted from the headline text features into the means of their vector representations. An embedding function was built to be able to repeat the same encoding process for all four datasets. This embedding function essentially ingested a dataset in the form of a *pandas* dataframe and returned a copy of the dataframe where each cell's data was replaced by the mean of the vector provided by InferSent's encode function for that headline. From here on the focus shifted to only using the bullish dataset and it was split into training and testing portions so as to avoid information leak when encoding.

The datasets were independently passed to the embedding function where InferSent performed the vectorization task. InferSent utilized fastText, and together they generated a vector representation of the entire headline based on the previously loaded word vectors on which fastText was pre-trained.

The Support Vector Classifier that was trained with the bullish data required an array of values, and if those values happened to be arrays themselves, was the case, the arrays had to all be of equal length. The data presented for this problem, when transformed into embeddings, produces vectors of many different lengths. Two ways of normalizing this data for acceptable ingestion could have been to either pad the vectors with zeros, or to derive the average of each vector. Understandably, averaging the vectors removes a possibly important latent feature of the data; the length of the given headline, but for the sake of simplicity on the initial run, averaging the vectors was used to normalize the embeddings for training.[7] In the evaluation, length as a latent feature may be examined to determine its correlation to the predictions.

Implementation

With the data preprocessed, the following phase entailed defining two separate models; one for the bullish dataset without stopwords and one for the bullish dataset which included the

stopwords. These models were both defined with the scikit-learn default parameters which were: *C=1.0, kernel='rbf', degree=3, gamma=1/n_features, coef0=0.0, shrinking=True, probability=False, tol=1e-3, max_iter=-1, decision_function_shape='ovr',random_state=None*
The two models were then trained on their respective datasets and asked to predict on their respective test sets.

The predictions were then used to calculate the accuracy, precision, recall, and F_1 score of the models. A calculation function was also built for this task in order to easily repeat it with future models. The function ingested the true values and the predicted values and output a table with the aforementioned metrics.

Once metrics were obtained for both the bullish stopwords inclusive model and the bullish stopwords exclusive model, the benchmark models were defined and included a stratified dummy classifier and a constant dummy classifier both from scikit-learn with the random_state parameter set to 42.

Refinement

Upon following the aforementioned implementation, it became apparent that there was an issue with the way that the Support Vector Classifier was predicting. When compared to the constant benchmark model the issue became clear. The model would always predict the positive class. This caused the metrics to report an artificial recall of one. This was a trivial behavior that mimicked the constant dummy classifier. This behavior would produce poor results in generalization sets such as the bearish holdout dataset therefore it was imperative to figure out the cause of this behavior and a possible remedy.

First, the distribution of classes in the bullish training dataset was reviewed to determine if there was a significant enough imbalance that could have accounted for the trivial predictive behavior of the model. As can be seen in figure 1, no distinct evidence of a severe imbalance was present in the bullish training dataset so this hypothesis was ruled out.

Next, the algorithm for the chosen model was reviewed to determine if there were any intricacies in its performance. It was found that the libSVM implementation in scikit-learn is not scale invariant and therefore you must scale data before training and testing. SVMs in general perform better on scaled data so the preprocessing phase was revisited and the scaling was performed just ahead of training using the scikit-learn StandardScaler class with default parameters. This transformed the performance of the model into a non-trivial behavior.

Finally, the model's behavior was no longer trivial but the performance was still in a suboptimal state. It was at this point that the focus shifted to the hyperparameters for improvement in the model's predictive power. The grid search cross-validation technique provided by scikit-learn was used to find the optimal values for the *kernel*, *C*, and *gamma* parameters. The scoring used for the grid search cross validation was the F_1 score and the parameter grid used can be seen in figure 11. Interestingly enough, this led to the hyperparameter values *C=1, kernel=rbf, gamma=0.001* for the estimator with the best F_1 score. This once again caused the same trivial predictive behavior that the model was displaying before the data was scaled.

```
{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4], 'C': [1, 10, 100, 1000]}
{'kernel': ['linear'], 'gamma': [1e-3, 1e-4], 'C': [1, 10, 100, 1000]}
{'kernel': ['poly'], 'gamma': [1e-3, 1e-4], 'C': [1, 10, 100, 1000]}
{'kernel': ['sigmoid'], 'gamma': [1e-3, 1e-4],'C': [1, 10, 100, 1000]}
```

Figure 11 - Cross-validation parameter search space

IV. Results

Model Evaluation and Validation

After preprocessing, implementation and refinement of the model and its data were performed a final model was chosen for the task of predicting DJIA movement based on the top 25 user-upvoted headlines of the day. This decision was contingent upon a number of factors, namely: comparison to the benchmark models, testing set evaluation, generalization to unseen data, model behavior, and the type of data ingested. The model chosen as the final model was the stopwords exclusive model as it exhibited the highest performance in all the aforementioned areas.

One of the first observations made was that stopword removal produced a clear improvement across the board for the metrics on which the models were evaluated. Figure 12 shows the metrics for the stopwords inclusive model and figure 13 shows the metrics for the stopwords exclusive model. The results derived from the stopwords exclusive model, as seen in figure 13, outperform the stratified benchmark model (figure 14) on every metric calculated. However, for the constant benchmark model the comparison is a bit more complicated (figure 15). While the stopwords exclusive model outperformed the constant benchmark model it just barely did so and was outperformed in every other metric. However, the constant benchmark model's metrics are a bit misleading. The accuracy is higher because it was always predicting the positive class and naturally there were more positive samples in the bullish dataset than negative samples. This caused an increased accuracy. Furthermore, the recall score for the constant benchmark model was perfect but this was expected since the model never predicted the negative class leading to this artificially perfect recall score. Additionally, the F_1 score was also higher but this was due to the aforementioned artificial perfection of the recall score since the F_1 score is simply the harmonic mean of the recall and precision scores.

Accuracy	0.5150684931506849
Precision	0.552
Recall	0.6798029556650246
F1	0.6092715231788081

Figure 12 - Stopwords inclusive model metrics on bullish data

Accuracy	0.5287671232876713
Precision	0.5617529880478087
Recall	0.6945812807881774
F1	0.6211453744493391

Figure 13 - Stopwords exclusive model metrics on bullish data

Accuracy	0.5013698630136987
Precision	0.5538461538461539
Recall	0.5320197044334976
F1	0.542713567839196

Figure 14 - Stratified benchmark model metrics on bullish data

Accuracy	0.5561643835616439
Precision	0.5561643835616439
Recall	1
F1	0.7147887323943662

Figure 15 - Constant benchmark model metrics on bullish data

The second observation made was in relation to the ability of the model to generalize to unseen data. The stopwords exclusive model was asked to predict utilizing the stopwords exclusive bearish holdout set and the results can be seen in figure 16 below. These results show that the model doesn't perform too poorly considering it was trained on bullish market data. In fact, though the precision decreased significantly, the recall score actually increased significantly. The F_1 score and the accuracy only slightly decreased. Furthermore, when the model's generalization is compared to the constant benchmark's generalization, as seen in figure 17, generally the benchmark model performed worse in all metrics when run on the generalization set and continued reporting the artificially perfect recall that inflated the F_1 score.

Accuracy	0.5123456790123457
Precision	0.475
Recall	0.7808219178082192
F1	0.5906735751295336

Figure 16 - Stopwords exclusive model metrics on bearish data

Accuracy	0.4506172839506173
Precision	0.4506172839506173
Recall	1
F1	0.6212765957446809

Figure 17 - Constant benchmark model metrics on bearish data

Justification

The stopwords exclusive model was not found to perform significantly enough to consider having solved the problem even though it did perform better than the benchmarks and its sister, the stopwords inclusive model. Even though there seems to be decent generalizability to the unseen data, results from this model should not be trusted as it has been proven to be unreliable with the predictive behavior quickly reverting to trivial upon the tweaking of any hyper-parameters.

One thing to note, is that the recall score of 0.78 for the stopwords exclusive model on the bearish data reveals an interesting possibility. Due to its focus on false negatives, a good recall score would allow for the creation of an investment strategy that contained either *short-selling* in bearish markets or the avoidance of *going long*, buying low and selling high, in bullish markets.

V. Conclusion

Free-Form Visualization

After performing predictions on both the bullish and the bearish datasets and achieving sub-optimal results, visualizations were performed in an attempt to further the understanding of the model behavior. The visualizations represented in figure 18 and 19 depict normalized distributions for the daily average headline length by correctly and incorrectly predicted subsets on either the bearish or bullish set of data. Basically, after predict on the bullish test set the predictions and labels were reintroduced into the test set. The word counts for the 25 headline features were averaged by row, added as a new feature to the test set, and then used to plot the distribution by correct predictions and incorrect predictions separately. These distributions were normalized in order to aid in deriving insights from the visualizations.

In examining these visualization distributions a very slight difference is noted and only appears in the bearish dataset where the model performed arguably better. The difference is in the height of the normalized curves. In the bullish predictions the curves are very close together for the correct and incorrect predictions but in the bearish predictions the correct predictions distribution is significantly shorter and spread out than the incorrect predictions. This observation showed that as the model performed better it was able to do so in spite of the average word count for the headlines of any given day not because of it.

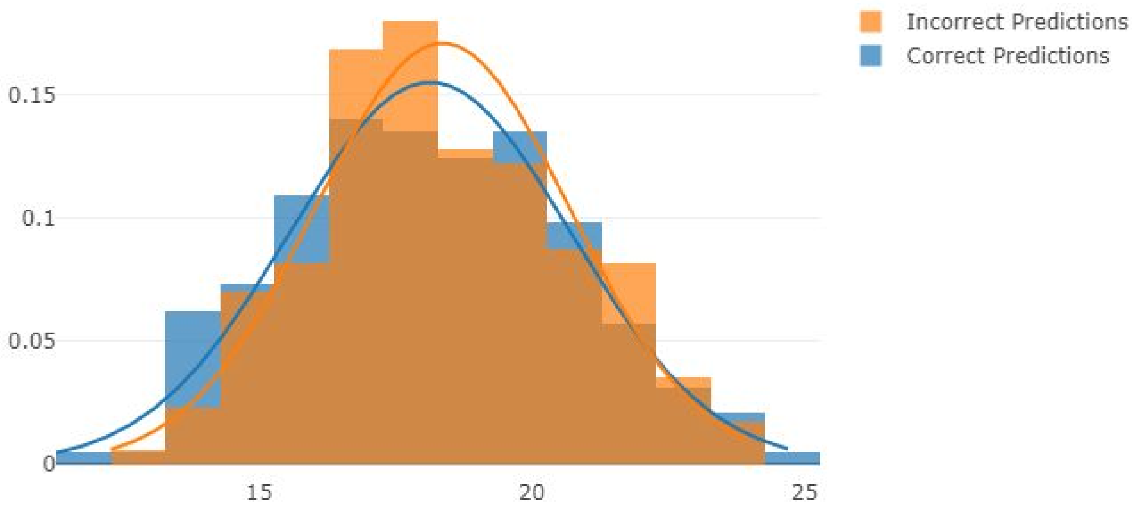

Figure 18 - Daily average headline length distributions for bullish predictions

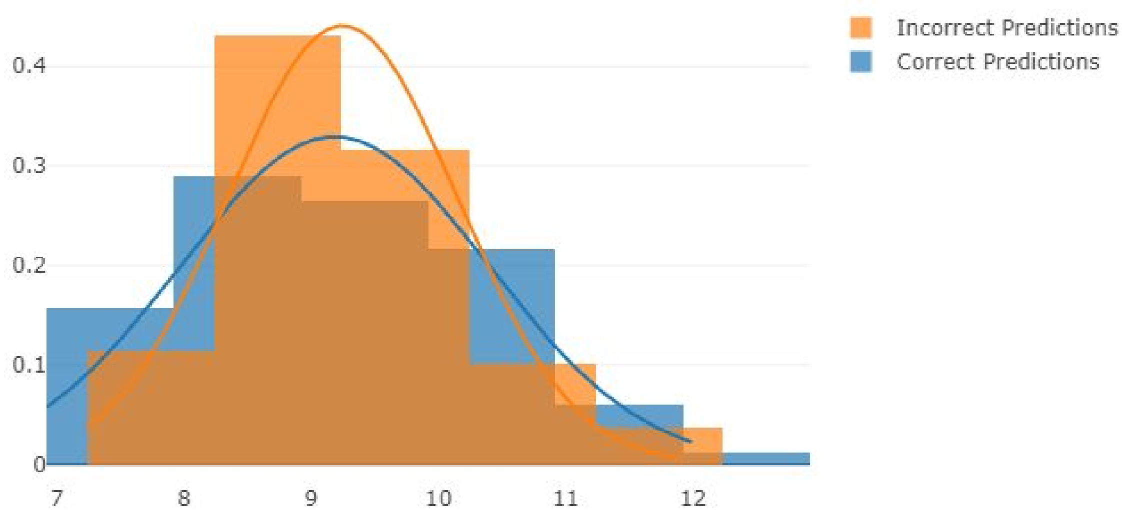

Figure 19 - Daily average headline length distributions for bearish predictions

Reflection

Following in the footsteps of successful hedge funds, an attempt was made to utilize alternative data in an effort to predict the general direction of Dow Jones Industrial Average. The data used to run the experiment was a combination of date, general direction (up/neutral or down), and the top 25 user upvoted headlines from Reddit's r/WorldNews thread for the date. The solution was approached as a natural language processing task utilizing a classification algorithm. InferSent underpinned by fastText, a relatively new preprocessing technique, was used to help convert all the text data into embeddings that would be used to train the classification model. While one of the benchmark models seemed more successful than the chosen model, it could not generalize to a holdout dataset.

With the mild effectiveness of the model and the impressive earnings presented by some of these successful hedge funds, it is safe to say that the model did not meet expectations and probably should not be used in similar contexts to attempt financial instrument movement predictions. In any case, it has proven that financial market prediction is a complex task.

Improvement

Considering the model's mild predictive success there is no doubt that improvements could be made to multiple parts of this experiment.

In further research, whilst testing the model it became apparent that InferSent is no longer state-of-the-art despite its optimal performance for the sentence embeddings task. To improve upon the existing results the use of Google's Universal Sentence Encoder or the Montreal Institute of Learning Algorithms/Microsoft Research Montreal's sentence encoder to generate the headline embeddings could improve performance.[10][11]

An even easier target for performance improvement may be the preprocessing steps. The vectorization of the data is very important to training and Support Vector Machines are good for data that may need heavy transformation so it may be possible to boost performance by padding the vectors with zeros to the largest vector rather than deriving the average of each vector thereby allowing the algorithm to better perform its central function, the kernel trick. Scaling also had a significant effect on the model's behavior and if more time was taken to carefully tune the scaling process for the data used, the model's performance could increase. However, care must be taken so as not to scale too specifically to this dataset and cause an issue in generalization.

Finally, the best area for a performance increase is the Support Vector Classifier itself. Support Vector Machines are notoriously temperamental and specific, it would behoove the reader to attempt training with other machine learning algorithms such as Random Forest or even Deep Neural Networks.

References

[1] R. Rubin and M. Collins (2015) "How an Exclusive Hedge Fund Turbocharged Its Retirement Plan"

[2] Aaron7sun (2016) *Dataset: Daily News for Stock Market Prediction*

[3] Li, Yang & Yang, Tao. (2017). "Word Embedding for Understanding Natural Language: A Survey." 10.1007/978-3-319-53817-4.

[4] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, "Supervised Learning of Universal Sentence Representations from Natural Language Inference Data"

[5] A. Joulin, E. Grave, P. Bojanowski, T. Mikolov (2016) "Bag of Tricks for Efficient Text Classification"

[6] scikit-learn developers (2007-2018) scikit-learn.dummy.DummyClassifier

[7] R. Socher (Lecture 2, March 2016) "Deep Learning for Natural Language Processing" Minute 46

[8] H. Saif, M. Fernandez, Y. He, H. Alani (2014) "On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter"

[9] Bird, Steven, Edward Loper and Ewan Klein (2009), "Natural Language Processing with Python." O'Reilly Media Inc.

[10] Cer D, Yang Y, Kong S-y, Hua N, Limtiaco N, John RS, et al. (2018) "Universal Sentence Encoder." arXiv preprint arXiv:1803.11175.

[11] S. Subramanian, A. Trischler, Y. Bengio, and C. J. Pal. (2018) "Learning general purpose distributed sentence representations via large scale multi-task learning." arXiv preprint arXiv:1804.00079.