Using News to Predict Market Behavior

Martin Blatz

# Overview

This project aims to classify the financial market behavior of a day as either "up" or "down" using only the news headlines available on the day prior. The purpose of this is to simulate as closely as possible the concept that an investor will only have knowledge of the past when attempting to make trading or investing decisions for the future. The data required for this work is sourced from two locations: news headlines are retrieved from the New York Times API (<https://developer.nytimes.com/apis>) and stock market data for the S&P 500 index are scraped from Yahoo Finance (<https://finance.yahoo.com/>) using the pynytimes Python package, which can be downloaded at <https://github.com/michadenheijer/pynytimes>. Headline text data was processed using count vector and TF-IDF algorithms, and various combinations of random forest and SVM classifier models were created in conjunction with a custom trained Word2Vec embedding, a pretrained Google Word2Vec embedding, and a pretrained Twitter GloVe embedding. The most promising model generated involved the use of a random forest classifier and the pretrained Twitter GloVe embedding layer. While the average test accuracy of the resulting model was greater than 55%, repeated iterations on randomly selected train/test sets did not yield an accuracy which was a statistically significant improvement over 55%.

# Summary

The goal of this project is to predict the directionality of the S&P 500 index using only data harvested from daily news headlines. The data required to perform this study has been collected, cleaned and joined prior to the application of various NLP processing, embedding layer weightings, and machine learning models .

# Project Description

## Motivation and Aims

Predicting the behavior of financial markets is an underlying theme to all trading and investing strategies. Being able to predict the directionality of the stock market with even moderate accuracy will allow an analyst to maximize trading profitability through informed positioning. Using the daily news headlines and previous S&P market performance information, can the direction of the stock market for the following day be predicted with greater than 55% accuracy? This is a classification machine learning question which looks to predict if each day belongs to category A (a down day) or category B (an up day).

I initially decided to use all data from the year 2022 as a test set. I have since changed that to using a randomized train/test split, which has yielded better results. A down year, market performance in 2022 was down more days than it was up, while the opposite is true for the other years in the dataset. This resulted in many models overpredicting "up" days and yielding poor predictive performance. I considered including past market performance measures such as the simple

## Dataset

The New York Times maintains a documented and freely available interface to query their archived and current content. Additional information can be found at <https://developer.nytimes.com/apis>. The query written for this project retrieves a maximum of 30 headline results per day, sources articles from the New York Times, Associated Press, and Reuters, and filters for the following news desks:

* Business Day
* Business
* Financial
* National
* Personal Investing
* Politics
* U.S.
* World

All searches were completed between 2/23/2023 and 2/26/2024. A total of 5 years of data was collected, starting on January 1, 2018 and continuing through December 31, 2022. The resulting New York Times data set is 1825 entries with 1 date column and 30 individual headline columns. Each entry contains at least 2 headlines, and only 90 entries contain fewer than 6 headlines for the day.

The pynytimes python package was used to build a separate Google Colab notebook to obtain basic S&P 500 index performance data over the same time frame as the New York Times headline data was collected. Pynytimes documentation can be found at [https://github.com/michadenheijer/pynytimes.](https://github.com/michadenheijer/pynytimes.%20) The package extracts data from [Yahoo finance](http://www.finance.yahoo.com), but only allows specific time frames to be selected when requesting information. A 10 year time frame is the only available option which encompasses the entirety of our target data set, so the data will need to be trimmed to match the 5 year span collected from the NYT. The data fields returned include the date, open, daily high, daily low, and close values in dollars, the daily trading volume, and a dividends and stock splits field.

For this project, the predictor variables are the text based news headlines for each day. The response variable is a boolean value, with 0 representing a "down" day in the market and 1 representing an "up" day as determined by comparing the closing value on a day to the previous day’s value.

The cleaned version of the two collected data sets can be found at <https://github.com/martinblatz/SP500_And_Headlines/tree/main/data/clean>. The Colab notebooks written to gather the datasets can be found at <https://github.com/martinblatz/SP500_And_Headlines/tree/main/src>. Note that the New York Times API requires the user to sign up for an API key. The script includes a variable which must be filled in with the user’s own API key before it will work.

# Methods

Data returned from the Yahoo Finance web scraper functions are very clean. When I explored the yfinance package, several function calls and capabilities documented by the developer don't appear to be working, but the basic market data I require for this task was working and available. The 10 year daily S&P 500 market data was simply downloaded using the yfinance package and saved.

The NYT data was a more complicated process. Each search request returns a tiered JSON object with a significant amount of metadata. Rather than storing all data, I extracted only the headline from each article returned to the query and saved them to a Pandas DataFrame with 1 'Date' column and 30 'News [n]' columns. Not all days will have 30 news articles which meet the search criteria, so I expect a significant number of individual data in the "News [n]" columns to be empty. Since every day will have a positive, non zero number of articles, I don't expect this characteristic to adversely affect the aggregated text analysis in later steps.

The only additional data cleaning necessary was to standardize the "Date" column to ensure a clean join between the two sets of data. Then, the NYT data was left joined with the financial data. This approach left weekend news data without matching financial data, but automatically trimmed the excess financial data. After the join, entries where no financial data is present are dropped. A label column is calculated to represent the dependent variable: 1 - the market went up or 0 - the market went down or stayed even.

The resulting cleaned and joined data set was reviewed using a grid of scatter and hysteresis plots. As expected, the open, low, high, and close values were highly correlated since they are all temporal variations of each day’s market price action. The text processing step involved joining each headline together in a single column, tokenizing the words, converting everything to lowercase, removing numbers, stop words, punctuation, and finally lemmatizing the remaining words. Additional exploratory visualizations were created using word clouds, but many of the same key words were present in both classes of the dataset.

A random forest classifier model was fit to the vectorized results of both a TF-IDF and count vector algorithm with poor to moderate results near or below 50% accuracy rates on the test data set. Then, the tokenized data was used to build a custom Word2Vec embedding layer using both the skip gram and bag of words approaches. The custom Word2Vec weightings were then applied to the tokenized headline data and the results used as input vectors for both a random forest classifier and SVM classifier. I began a wide parameter grid search and manually reduced the parameter options until the model was no longer obviously overfitting the data. This process was repeated for each combination of embedding layer (custom, Google Word2Vec, Twitter GloVe) and classification model (SVM, random forest). For each of the 6 models, accuracy metrics were printed and a confusion matrix displayed for reference. The dataset was split into random training and test sets 30 times and the most promising modeling approach was implemented for each iteration to obtain a mean accuracy performance for the model and ensure that the results are statistically significant rather than a fluke event.

# Conclusion

The most effective classifier model created during the model revision process described was a random forest classifier using the Twitter pretrained GloVe embedding layer weightings. The hyperparameters used were as follows:

* Selection criterion: gini
* Number of estimators: 5
* Max features: square root
* Max depth: 3
* Max leaf nodes: 3

The model consistently yields classifier accuracy on the test data set over 55%, which was the goal of this project. The confusion matrix below represents an instantiation of the model which resulted in an accuracy score of 58.7%. However, we are not able to reject the null hypothesis that news headline data can't be used to predict S&P directionality with an accuracy greater than 55% since a one sided t-test on 30 iterations of the model results in a p value of greater than 0.05.

Chart, treemap chart

Description automatically generated

Despite the inability to reject the null hypothesis, the model created during this study still shows promise. 53.9% of the days in the dataset are “up” days in the market. A one sided t-test comparing model accuracy to 53.9% results in a statistically significant, though modest, improvement using the model. It is likely that model performance could be further improved by incorporating historical price action of the market. Other future work towards incremental improvements could include the implementation of LSTM and an approach that uses data from the past N days to predict the market performance on day N+1.