Twitter Can Help Predict Crash

Mingxuan Wu (mingxuaw) - Pittsburgh

Shuying Wu (shuyingw) - Pittsburgh

Yumeng Zhang (yumengz) - Pittsburgh

Yuyao Zhang (yuyaoz) – Pittsburgh

I. Introduction

Social media has become a source of "big data" due to the sheer amount of user-generated content. Each generation spends more time on social media than their predecessors, leading to the significant role of social media data in understanding opinions, trends and public sentiment. Analyzing these rich datasets coming from social media can help us understand the overall market including stock and business performance.

In our project, we used Twitter as our social media platform, investigated the correlation between the users' sentiment and the stock market. The goal is to analyze and specify words that can predict a potential crash of the stock market.

II. Data Collection and Analysis

1. Build tweets library

   The raw tweets data were collected from Twitter API. One of the limitation was that Twitter only allowed us to retrieve the most recent tweets (about 3,200 pieces) per user account. As a result, we have more data in 2018, fewer in 2017, even fewer in 2016 and 2015. Our final tweets library comes from 40 twitter accounts consisting of 3 main groups: prestigious investors, medias and financial institutions. You can refer to our appendix for a list of twitter accounts.

2. Construct bag-of-words

   Once we built our tweet library, the next step was to construct a bag-of-words. We first captured all words with more than 2 characters using regular expression and filtered out stop words using the built-in dictionary in Python Natural Language Toolkit (nltk). We also implemented stemming using the Porter Algorithm in the nltk library, and one important reason is that the Porter algorithm has a non-random result, which can guarantee that our test processing file has the same result on different laptops and at different time. There were around 65,192 words in total, but most of them appeared less than 100 times, not helpful for our prediction. Therefore, we chose the top 1000 words to build our bag-of-words and transformed it into the documentation matrix. Although we still had many features, we decided to implement further feature selection within each model later.

3. Label market drops

   In order to measure the stock market performance, we calculated the weekly log return of S&P 500 index and labeled the top 10% worst drops as "1", and "0"

otherwise. Therefore, our binary dependent variable consists 1 and 0 with a base rate around 10%.

4. Split into training and test sets

We split the data into a training set (67%) and a test set (33%) with random state = 1.

5. Build our model

Our model is a "week to week" prediction for 172 weeks from 2015/1/1 to 2018/4/25. We studied whether the tweets in one week lag could predict the drop conditions happened in the next week. Although day to day prediction might be interesting, we could not find enough data for the dates in 2016 and 2015. Therefore, using weekly data can help average the outliers.

III. Model Development

1. Model Selection

We implemented the following models from both supervised and unsupervised learning:

Supervised learning	Logistic regression	Ordinary logistic
		Logistic GAM
		Lasso logistic
	Random forest	
	Boosting	Gradient descent boosting
	Naïve Bayes	
	SVM	Linear and rbf kernel
	KNN	
Unsupervised learning	Clustering	K-means
		Hierarchical clustering

2. Scaling

As mentioned above, due to the limited access, we have more data in 2018, fewer in 2017, and even fewer in 2016 and 2015. To fix this problem and ensure the equal means for every rows, we used the built-in scale function in sklearn library. For reference and comparison purpose, we also fitted the models without scaling. In most cases, scaling led to a better prediction result.

3. Feature Selection

   Our data consists of 1000 features but only 172 observations, so feature selection is necessary. Some models including lasso logistic regression are able to handle feature selection themselves. However, for models without the function of feature selection, PCA was implemented. Among the 1000 features, PCA showed that 35 dimensions were able to capture 90% of the information. We also compared each model to see whether PCA helped our prediction. However, PCA had less impact on the model performance than scaling.

4. Parameter Tuning

   We tried many parameter combinations in each model including lambda, class weight and kernels. Please refer to the appendix to see how each one performed. However, because of time limitation, we could not try all possibilities, especially in the gradient boosting algorithm. We tried several simple tunings there.

IV. Model Evaluation

We looked at 3 metrics to measure the performance of our models:

1. Misclassification rate

   Given the unbalanced data and a base rate of 10%, the misclassification rate is not very helpful and important in our project. But we still treated it as a very-first judgement.

2. Top 10/20 predictions

   We identified the top observations that were most likely to score `yes' (that is the observations from the test set with the highest predicted probability of Y = 1). We then calculated the probability that a market drop actually happened, as we would like to see whether the model predicted accurately if a model highly believed that a drop would happen. Due to the sample size of our test data (52 weeks, 8 were labeled 1), we only calculated the top 10/20 predictions rather than the top 1000 predictions as in the homework.

3. AUC score

   AUC stands for the area under the ROC curve, which is used in classification analysis to determine which of the used models predicts the classes best. We relied a lot on this measurement to compare models.

Here is a summary of how each model performed:

Model	Misclassification Rate	Top 10/20 predictions	AUC score
Logistic Regression	0.1731	predict 5 in top 20	0.5909
Logistic GAM	0.1154	predict 4 in top 10	0.642
Lasso Logistic	0.1538	Predict 2 in top 10	0.651
Random Forest	0.1538	predict 3 in top 10	0.723
Gradient Boosting	0.1538	predict 4 in top 20	0.6023
Naïve Bayes	0.1538	predict 2 in top 10	0.5
K-means Clustering	0.1676	-	-
SVM	0.1538	Predict 6 in top 20	0.7798
KNN	0.1538	predict 3 in top 10	0.6478

According to the table above, Logistic GAM enjoyed the lowest misclassification rate. SVM was the best in capturing potential crashes. Random forest also had a decent outcome. Note that if a model has multiple parameters available to choose (for example, SVM with linear kernel or "rbf" kernel), only the best outcome was provided in the table above.

V. Model Validation and Interpretation

After fitting different models, we wanted to see what words were the main indicators for a crash and whether they actually made sense. In this report, we picked lasso logistic regression and random forest to delve deeper into the analysis. These two models were chosen because they were the best in model performance as well as result interpretation simultaneously.

1. Lasso logistic regression

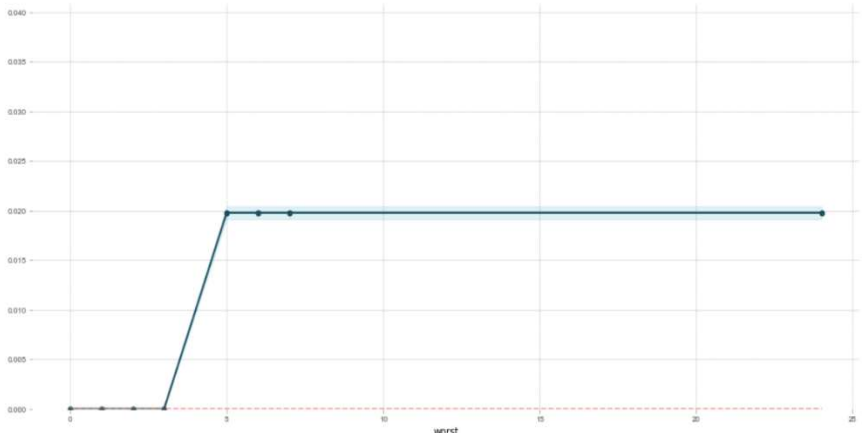
   Lasso is able to select features that are important in prediction. Our lasso picked 15 words, which can be divided into 2 groups: twitter account names and descriptive words. Below are some selected words and their coefficients:

Words	Coefficient
GDP	-0.00311922
remain	-0.0372262
highlight	-0.02579761
soft	-0.06621725
morganstanley	0.08584726
eagle	0.00410814
crash	0.03951969

Words with negative sentiments usually have positive coefficients. For example, when the tweets were filled with the word “crash”, a real crash/market drop is more likely to happen. Safe/positive words, such as “soft”, “remain”, “highlight”, typically indicate a prosperous macroeconomic environment, and if we see those words in tweets, it is less likely that a crash will happen. We also noticed that Morgan Stanley has a positive coefficient and would like to investigate that further in the future.

2. Random forest

Random forest is capable to describe the importance of each feature. Our random forest model picked 2 kinds of words: high-frequency words (such as “co” and “http”, the two most frequent words in our word-of-bag) and descriptive words. The words with the largest feature importance are: “co”, “http”, “market”, “continue”, “remark”, “worst”, “hk”, “morganstanley”, “read” and “today”. We will take a close look at the word “worst”.



The partial dependence plot above illustrates that when the word "worst" appears more frequently in tweet, it is more likely that a market drop will happen.

VI. Conclusion

Our model shows a decent capability of predicting a potential drop using Twitter data. After carefully studying each model, we believe that there can be a high correlation between investor emotions and market performance. Specifically, if investors are expecting or worrying about a crash, they will post their opinions and feelings on Twitter, and by searching words such as "worst" and "crash", our models are able to capture these emotions. When the panic gradually accumulates to some extent, investors tend to sell their assets, resulting in a market drop.

VII. Improvement

1. More data

   Ideally, we would like to obtain more tweets to collect data for 2015, 2016, and even earlier years. We hope to have a larger twitter account list, or perhaps get some funding to apply for a premium user account in Twitter which is able to get earlier data from Twitter API.

2. More tuning

   If time allows, we would like to spend more time adjusting parameters for certain models such as gradient descent boosting, which possibly generate better results. Moreover, we would like to try different time frames other than the "week to week" prediction currently used.

3. Better text representation

   We used bag-of-words to represent text. However, there were some single words that are difficult to interpret. We plan to use other representations that can preserve more context for further analysis.

VIII. Appendix

Our code contains 4 parts: code to fetch tweets from a certain account in Twitter; code to build and update our document matrix; code to fetch weekly return of SP500 and label drops; code to fit different models on our training and test data.

1. "fetch_by_username.pdf"

   Because of the policy of Twitter, we are not permitted to share tweets from other users. To get tweets from a user, see our "fetch_by_username.pdf". It is able to fetch tweets by username, and store the result in a csv file for future use. Note that you have to apply at Twitter API website to get your own token and secret.

2. "text_processing.pdf" and "Update X_raw.pdf"

   These 2 files were used to build the initial word bag and update the document matrix from new tweets. After getting a csv file from step 1, you can call update_X function to feed your matrix.

3. "Sector Index.pdf", "SP500 Index.pdf"

   We downloaded:

   MSCI USA IMI Sector Index for 11 sectors from
   https://app2.msci.com/eqb/ussector/indexperf/dailyperf.html
   S&P 500 index from
   https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC .

   Then we calculated the weekly log return and labeled the top 10% drop as 1.

4. "index_analysis.pdf"

   A thorough test of different models on our data. In the end of the document, a ROC curve of our 3 top models are provided.