Shop Til' You Drop: Social Sensing to Predict Company Health

Using Twitter Data to Analyze the Consumer Discretionary Sector

Lauren Ferrara and Reilly Kearney
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN

Abstract— **This project focuses on analyzing the health of three public companies in the Consumer Discretionary sector, specifically the Textiles, Apparel, & Luxury Goods industry. These three companies are Burberry (BRBY), Lululemon Athletica (LULU), and Urban Outfitters (URBN). We use three different sentiment analysis tools (Vader, TextBlob, and Twinword) to analyze Twitter posts related to these companies. We also take into consideration the number of tweets about the company on a given day, the number of followers for a user who has written a particular tweet, and the number of retweets and favorites for that tweet. Using Ordinary Least Squares Regression and Random Forest predictive models, we are able to predict the company's stock price.**

Keywords—social sensing; Twitter; stock; company perception; sentiment analysis; Ordinary Least Squares Regression, Random Forest, Consumer Discretionary sector, predictive models

I. Project Overview

This project uses sentiment analysis of Twitter consumer data, information abou the spread of this data via followers, retweets, and favorites, and Twitter presence of the company by number of tweets about that company each day to predict how well their stocks perform. We hope to extend this model to understand how an industry, the Textiles, Apparel, & Luxury Goods industry, is doing overall. In order to do this, we start by analyzing Twitter and stock data for three companies, Lululemon Athletica, Urban Outfitters, and Burberry. These companies span both the low and high end of the fashion industry, allowing us to analyze different target markets.

We built a training data set of tweets for these companies, found based on keyword searches and hashtags, and historical stock closing prices. We then created predictive models using Ordinary Least Squares Regression and Random Forest Regression to predict how the stocks for these companies should perform. These were then compared to the actual stock prices on particular dates to determine how effectively our models predict stocks for these companies.

Please refer to the following model regarding how we formatted our data input and analysis.

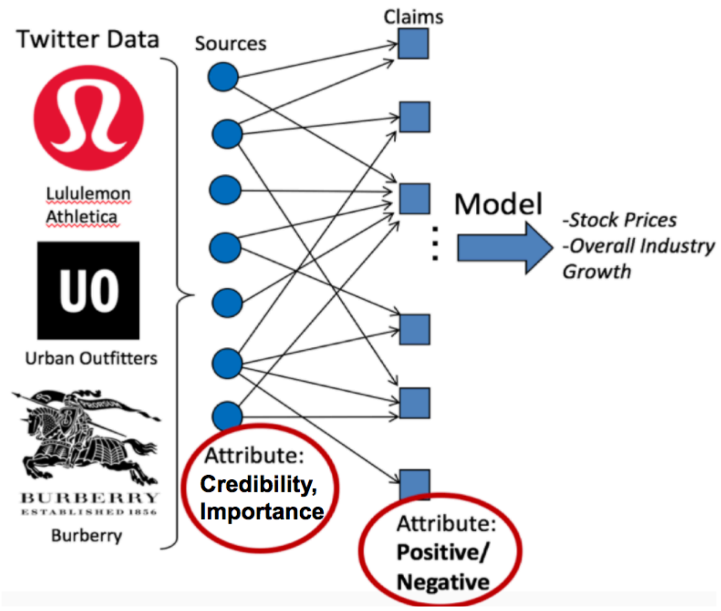


Figure 1. Diagram of Project Overview

II. Review of State-Of-The-Arts

The Efficient Market Hypothesis (EMH) [1], illustrated in Figure 2, tells us that stock prices are more affected by news than by past and present stock prices. This means that news trending on social media would likely have an influence on stocks. Due to this hypothesis, Twitter data has been used to predict stock prices in many other studies and even in professional stock trading.

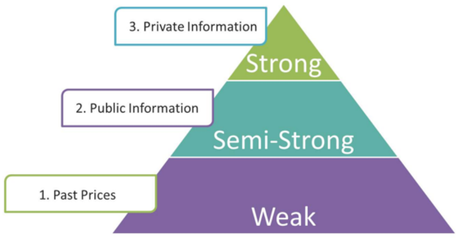


Figure 2. Hierarchy of EMH [2]

The analysis of tweets for stock prediction often relies on sentiment analysis, the processing of natural language to determine how the speaking person feels in some metric. For example, Bollen, Mao, and Zeng [3] used "two mood tracking tools, namely OpinionFinder that measures positive vs. negative mood and Google-Profile of Mood States (GPOMS) that measures mood in terms of 6 dimensions (Calm, Alert, Sure, Vital, Kind, and Happy)" in their study. This stock prediction study and others used a Granger Causality Analysis and Self-Organizing Fuzzy Neural Network [4] to determine whether the mood of tweets has predictive information about stock prices.

These social sensing studies brought to our attention certain factors we should consider to avoid errors in our stock prediction model. For one, it was found that using just Twitter mood states to predict the stock market can be faulty. Instead, the "mood contagion" [5] must also be taken into account, meaning that we must also analyze the number of followers a tweet has as to predict the spread of that mood. Another study pointed out the need to look at what the company itself is tweeting: "consumers' level of exposure to company social media activities precedes perceptions of corporate reputation" [6].

Our research elaborates on these former studies, applying similar methods to the fashion industry in particular.

III. Problem Statement

We embarked on this research endeavor in order to determine whether company-specific Twitter data could be used to predict company stock prices for three fashion companies (Burberry, Lululemon Athletica, and Urban Outfitters) and, if so, with what accuracy. In order to do this, we were faced with a series of problems to address: collecting Twitter data, collecting stock data, analyzing sentiment and "mood contagion," determining correlation, and building and evaluating predictive models.

IV. Solution

A. *Collecting the Twitter Data*

Using Python, the Tweepy module, and Twitter API, we created a data crawler to obtain Twitter data regarding our three companies. Tweets were compiled into folders, one for each company, containing a JSON file of Twitter data corresponding to each day (i.e. Lululemon_data/Lululemon_2017-03-15_tweets.json).

Within the Twitter crawler, data is collected using the Tweepy module Cursor to find tweets corresponding to a given keyword (i.e. Lululemon, urban outfitters, and burberry) and a given time period—a day at a time in this case. The tweets gathered by the cursor were then stored in a list and dumped to the corresponding JSON file. The Twitter crawler collects all attributes, including text, date, user information, retweets, and favorites, corresponding to a given tweet, so that the data set collected is conducive to analyzing how the mood towards a company on Twitter, as well as the mood contagion, corresponds to the stock price for a given day

Using this crawler, we queried tweets related to these fashion companies from March 15, 2017, to April 25, 3017, missing the dates 3/30, 4/15, and 4/16 for all three companies, as well as dates 3/15 and 3/16 for Burberry. These glitches in data collection occurred because of the limits on the Twitter API, which would only allow us to reqeust tweets going up to one week back. In addition, we were limited by the maximum requests allowed for one API key. To aid with this, our Twitter crawler waits when the maximum number of requests is reached until it can continue to collect data.

B. *Collecting the Stock Market Data*

Using Yahoo! Finance [7], [8], [9], we collected stock market data (i.e. closing prices) for each company from March 15, 2017, to May 2, 2017. To collect this information, we used a Python library called yahoo-finance [10] to grab information from the Yahoo! Finance API. With this library, each of the three companies is initialized as an object Share. This object has the method get_historical_data(start_date, end_date), which returns, among other things, the closing price that we use in this data set. We used this data to make different dependent variables corresponding to certain lags in stock prediction, from 1 to 21 days.

C. *Sentiment Analysis*

In order to evalutate the Twitter data, we tried a variety of sentiment analysis tools, settling on the following three: Vader Sentiment [11], TextBlob [12], and Twinword [13].

The Vader API was useful for this project, because it gives a numerical expression of how positive, negative, or neutral a tweet is. The vaderSentiment library in python takes in a string of text, obtained from our tweet data, and calculates what proportion of it is positive, negative, and netural, as well as a compound score (between -1 and 1), by analyzing the words in the tweet. For a visual representation of the Vader scoring, please refer to Figure 3 to see the overall breakdown of postive/negative/neutral for a subset of tweets collected for each of the three companies for March 13-19.

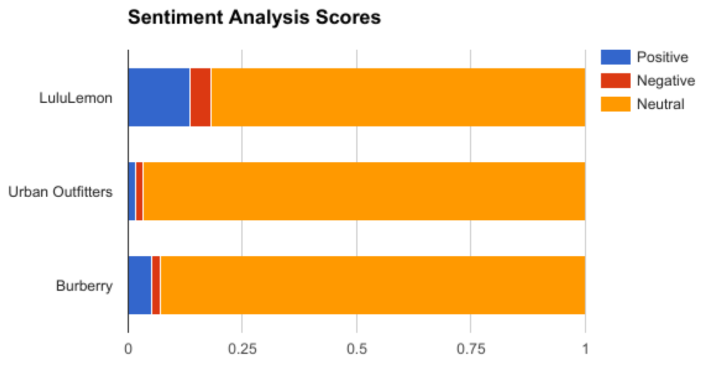


Figure 3. Sentiment Analysis Scores using Vader

Initially, we chose this library for two main reasons: first, it focuses on handling many different Twitter-like language features, such as emoticons, slang, and acronyms, as well as negations, punctuation, and capitalization. The second reason we chose it is in order to use the generated compound score as a one-dimensional metric for direct modeling against stock market prices.

The two other sentiment analysis tools give scores similar to the Vader compund score, also between -1 and 1. The TextBlob Python library works almost exactly the same as the Vader library, giving a score for each tweet based on math done in the package installed on your computer.

Twinword acts differently in that you must send HTTP requests to the Twinword API, hosted on mashape [14] in order to get the sentiment analysis score. Because of limitations on the amount of requests allowed for a given API key, we could not get a score for each tweet. Rather, we sent the API concatenations of the words in tweets on a given day, up to the maximum length of 100 words or 1500 characters. From this we were able to find the average sentiment of all words said about a company on a given day.

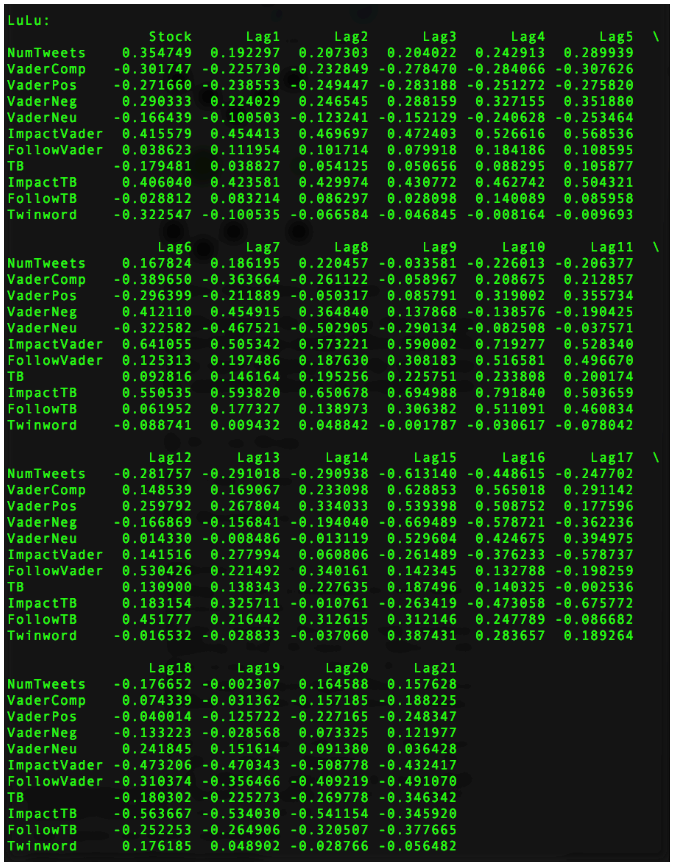
We also incorporated multipliers to get mood contagion, the influence of the sentiment, rather than just pure mood. We had two ways of doing this: what we call Follow and Impact multipliers. For Follow, we took the number of followers for the user who posted a tweet and multiplied that by the overall sentiment score for that tweet. For Impact, we took the number of retweets plus the number of favorites for a given tweets and multiplied that by the overall sentiment score for that tweet. This gave us four new features to consider, follow and impact multipliers for both Vader and TextBlob sentiment scores. We were unable to use this method with Twinword, due to the concatenation of words for scoring rather than individual tweets.

D. *Determining Correlation*

From the data described above, we concatenated csv files to get an overall data set containing all possible features (Number of Tweets, Vader Compound Score, Vader Positive Score, Vader Negative Score, Vader Neutral Score, Vader Compound * Impact, Vader Compound * Follow, TextBlob Score, TextBlob * Impact, TextBlob * Follow, and Twinword) and all possible lags in stock data (0-21) and put this into new csv files. From these csv files, we loaded the data in Pandas DataFrames using the Pandas read_csv function.

We then used the Pandas correlation function (DataFrame.corr()) to determine the correlation between the features and the possible dependent variables (stock lags). Correlation, commonly written as r^2, is a value between -1 to 1. The closer it is to 0, the less correlation present. -1 is a strong negative correlation, and +1 is a strong positive correlation. An example output for the Lululemon data set can be seen in Figure 4. From this output, we chose which lag to use as our dependent variable for each stock and which combination of features.

For Burberry, we chose Lag 14, meaning that we would predict the stock price 14 days into the future, as the dependent variable, as well as the feature set [Vader Negative, Number of Tweets, Vader * Follow, and TextBlob] when needed. For Lululemon Athletica, we chose Lag 10 and feature set [Vader * Follow, TextBlob * Impact, and Twinword] when needed. For Urban Outfitters, we chose Lag 4 and feature set [Vader, TextBlob, and Twinword] when needed.



```
LuLu:
                Stock      Lag1      Lag2      Lag3      Lag4      Lag5  \
NumTweets    0.354749  0.192297  0.207303  0.204022  0.242913  0.289939
VaderComp   -0.301747 -0.225730 -0.232849 -0.278470 -0.284066 -0.307626
VaderPos    -0.271660 -0.238553 -0.249447 -0.283188 -0.251272 -0.275820
VaderNeg     0.290333  0.224029  0.246545  0.288159  0.327155  0.351880
VaderNeu    -0.166439 -0.100503 -0.123241 -0.152129 -0.240628 -0.253464
ImpactVader  0.415579  0.454413  0.469697  0.472403  0.526616  0.568536
FollowVader  0.038623  0.111954  0.101714  0.079918  0.184186  0.108595
TB          -0.179481  0.038827  0.054125  0.050656  0.088295  0.105877
ImpactTB     0.406040  0.423581  0.429974  0.430772  0.462742  0.504321
FollowTB    -0.028812  0.083214  0.086297  0.028098  0.140089  0.085958
Twinword    -0.322547 -0.100535 -0.066584 -0.046845 -0.008164 -0.009693

                 Lag6      Lag7      Lag8      Lag9     Lag10     Lag11  \
NumTweets    0.167824  0.186195  0.220457 -0.033581 -0.226013 -0.206377
VaderComp   -0.389650 -0.363664 -0.261122 -0.058967  0.208675  0.212857
VaderPos    -0.296399 -0.211889 -0.050317  0.085791  0.319002  0.355734
VaderNeg     0.412110  0.454915  0.364840  0.137868 -0.138576 -0.190425
VaderNeu    -0.322582 -0.467521 -0.502905 -0.290134 -0.082508 -0.037571
ImpactVader  0.641055  0.505342  0.573221  0.590002  0.719277  0.528340
FollowVader  0.125313  0.197486  0.187630  0.308183  0.516581  0.496670
TB           0.092816  0.146164  0.195256  0.225751  0.233808  0.200174
ImpactTB     0.550535  0.593820  0.650678  0.694988  0.791840  0.503659
FollowTB     0.061952  0.177327  0.138973  0.306382  0.511091  0.460834
Twinword    -0.088741  0.009432  0.048842 -0.001787 -0.030617 -0.078042

                Lag12     Lag13     Lag14     Lag15     Lag16     Lag17  \
NumTweets   -0.281757 -0.291018 -0.290938 -0.613140 -0.448615 -0.247702
VaderComp    0.148539  0.169067  0.233098  0.628853  0.565018  0.291142
VaderPos     0.259792  0.267804  0.334033  0.539398  0.508752  0.177596
VaderNeg    -0.166869 -0.156841 -0.194040 -0.669489 -0.578721 -0.362236
VaderNeu     0.014330 -0.008486 -0.013119  0.529604  0.424675  0.394975
ImpactVader  0.141516  0.277994  0.060806 -0.261489 -0.376233 -0.578737
FollowVader  0.530426  0.221492  0.340161  0.142345  0.132788 -0.198259
TB           0.130900  0.138343  0.227635  0.187496  0.140325 -0.002536
ImpactTB     0.183154  0.325711 -0.010761 -0.263419 -0.473058 -0.675772
FollowTB     0.451777  0.216442  0.312615  0.312146  0.247789 -0.086682
Twinword    -0.016532 -0.028833 -0.037060  0.387431  0.283657  0.189264

                Lag18     Lag19     Lag20     Lag21
NumTweets   -0.176652 -0.002307  0.164588  0.157628
VaderComp    0.074339 -0.031362 -0.157185 -0.188225
VaderPos    -0.040014 -0.125722 -0.227165 -0.248347
VaderNeg    -0.133223 -0.028568  0.073325  0.121977
VaderNeu     0.241845  0.151614  0.091380  0.036428
ImpactVader -0.473206 -0.470343 -0.508778 -0.432417
FollowVader -0.310374 -0.356466 -0.409219 -0.491070
TB          -0.180302 -0.225273 -0.269778 -0.346342
ImpactTB    -0.563667 -0.534030 -0.541154 -0.345920
FollowTB    -0.252253 -0.264906 -0.320507 -0.377665
Twinword     0.176185  0.048902 -0.028766 -0.056482
```

Figure 4. Lululemon Data Set Correlation

It should be noted that another method was attempted to choose features for our models, sklearn.feature_selection's f_regression function [15]. Using this function with the chosen lags, we got the following outputs, seen in Figures 5-7, expressing the importance of features for each company. However, when these results were used to create the predictive models, they produced worse results than the features chosen purely from correlation values. Based on this, f_regression was not used for creating the models explained in the next section.

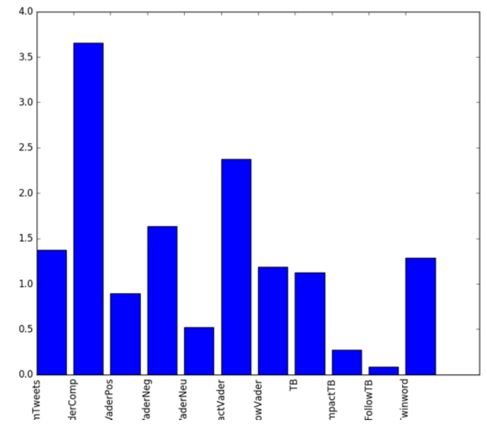


Figure 5. Burberry f_regression Feature Importance

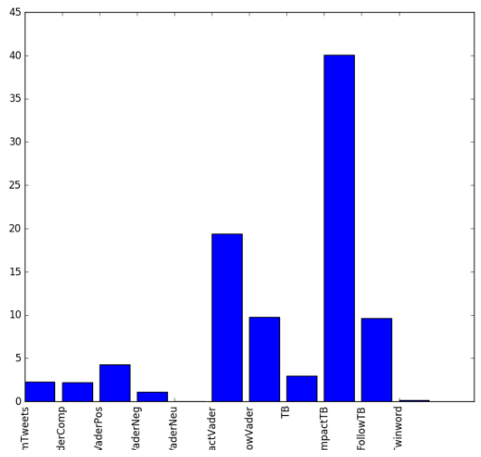


Figure 6. Lululemon f_regression Feature Importance

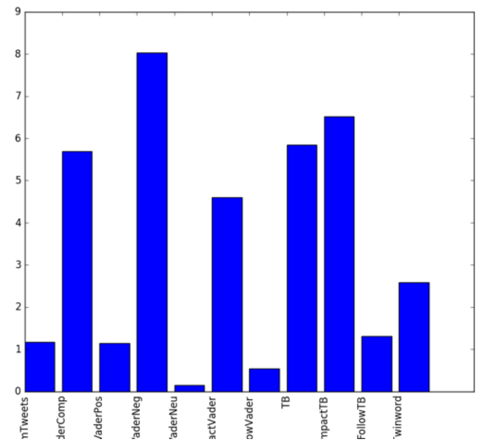


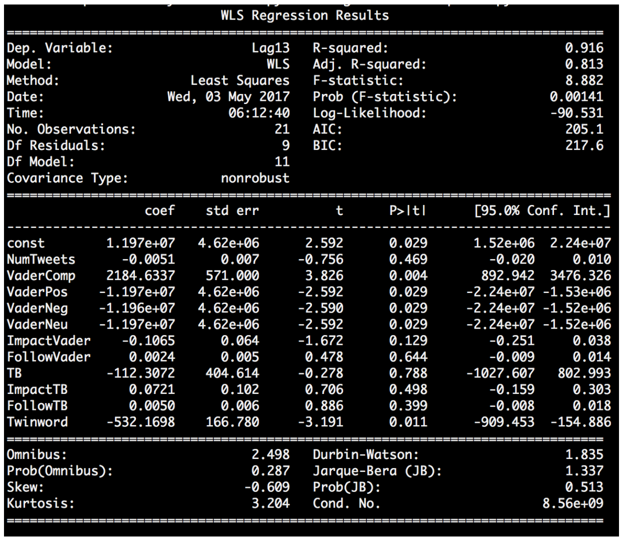
Figure 7. Urban Outfitters f_regression Feature Importance

E. *Predictive Models*

We first used Ordinary Least Squares (OLS) regression to form predictive models for each of the companies using the lags and feature sets determined by the correlation above. OLS [16] is a linear regression approach that gives coefficients to independent variables in order to reduce the sum of squares of the difference between the the actual values of the dependent variable and those predicted by the linear model. In order to take into account the starting price of the stock before our model was made, we added y-intercepts to all OLS models. This regression was done using the OLS function in the statsmodels python library.

Ordinary Least Squares regression assumes the regressors, the features included, are linearly independent. However, for our data set, this is nearly impossible as a number of our features come from the sentiment of the same tweets only with different sentiment analysis tools. Because of this, the condition number, indicating strong multicollinearity [17] is very high for two companies' models.

Another common problem with OLS regression is overfitting the model to the training data. We tried to avoid this by choosing features based on both the correlation values found using Pandas and also the logical relations between variable. For example, it would make sense for the Vader negtive score to be negatively correlated with the stock price. An example of overfitting the model can be seen for a Burberry OLS model in Figure 8. This model using all possible features and results in a strong overall correlation of 0.916. However, it has a large condition number, 8.56e9, indicating high multicollinearity. In Figure 9, the predictions, plotted as the dashed red line, are compared to the actual results, plotted as blue squares. The dates March 17th to April 6th were included in the training data, whereas April 7th to April 15th were held out as testing data. The model fits the training data extremely well, but is not effective in predicting the testing data, demonstrating the problem of overfitting.



```
                            WLS Regression Results
==============================================================================
Dep. Variable:                  Lag13   R-squared:                       0.916
Model:                            WLS   Adj. R-squared:                  0.813
Method:                 Least Squares   F-statistic:                     8.882
Date:                Wed, 03 May 2017   Prob (F-statistic):            0.00141
Time:                        06:12:40   Log-Likelihood:                -90.531
No. Observations:                  21   AIC:                             205.1
Df Residuals:                       9   BIC:                             217.6
Df Model:                          11
Covariance Type:            nonrobust
================================================================================
                  coef    std err          t      P>|t|     [95.0% Conf. Int.]
--------------------------------------------------------------------------------
const        1.197e+07   4.62e+06      2.592      0.029      1.52e+06  2.24e+07
NumTweets      -0.0051      0.007     -0.756      0.469        -0.020     0.010
VaderComp    2184.6337    571.000      3.826      0.004       892.942  3476.326
VaderPos    -1.197e+07   4.62e+06     -2.592      0.029     -2.24e+07 -1.53e+06
VaderNeg    -1.196e+07   4.62e+06     -2.590      0.029     -2.24e+07 -1.52e+06
VaderNeu    -1.197e+07   4.62e+06     -2.592      0.029     -2.24e+07 -1.52e+06
ImpactVader    -0.1065      0.064     -1.672      0.129        -0.251     0.038
FollowVader     0.0024      0.005      0.478      0.644        -0.009     0.014
TB           -112.3072    404.614     -0.278      0.788     -1027.607   802.993
ImpactTB        0.0721      0.102      0.706      0.498        -0.159     0.303
FollowTB        0.0050      0.006      0.886      0.399        -0.008     0.018
Twinword     -532.1698    166.780     -3.191      0.011      -909.453  -154.886
==============================================================================
Omnibus:                        2.498   Durbin-Watson:                   1.835
Prob(Omnibus):                  0.287   Jarque-Bera (JB):                1.337
Skew:                          -0.609   Prob(JB):                        0.513
Kurtosis:                       3.204   Cond. No.                     8.56e+09
==============================================================================
```

Figure 8. OLS Model Demonstrating Overfitting

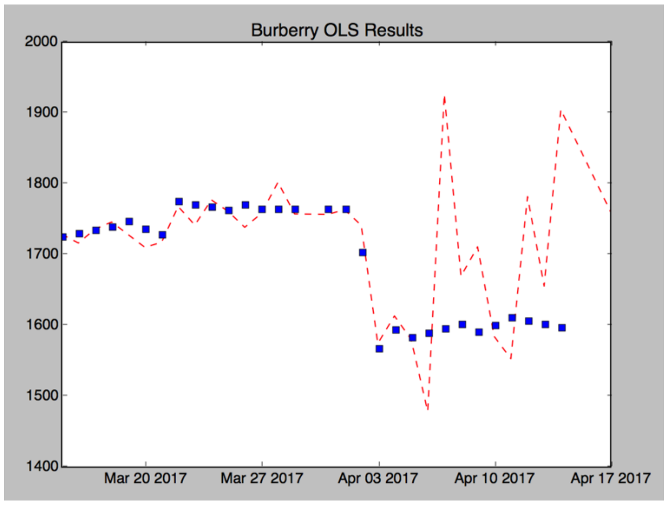
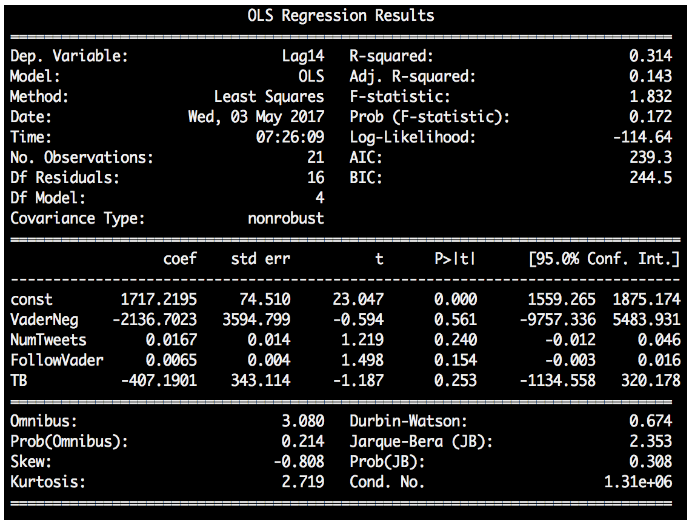


Figure 9. OLS Predictions Demonstrating Overfitting



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Lag14   R-squared:                       0.314
Model:                            OLS   Adj. R-squared:                  0.143
Method:                 Least Squares   F-statistic:                     1.832
Date:                Wed, 03 May 2017   Prob (F-statistic):              0.172
Time:                        07:26:09   Log-Likelihood:                -114.64
No. Observations:                  21   AIC:                             239.3
Df Residuals:                      16   BIC:                             244.5
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const       1717.2195     74.510     23.047      0.000      1559.265  1875.174
VaderNeg   -2136.7023   3594.799     -0.594      0.561     -9757.336  5483.931
NumTweets      0.0167      0.014      1.219      0.240        -0.012     0.046
FollowVader    0.0065      0.004      1.498      0.154        -0.003     0.016
TB          -407.1901    343.114     -1.187      0.253     -1134.558   320.178
==============================================================================
Omnibus:                        3.080   Durbin-Watson:                   0.674
Prob(Omnibus):                  0.214   Jarque-Bera (JB):                2.353
Skew:                          -0.808   Prob(JB):                        0.308
Kurtosis:                       2.719   Cond. No.                     1.31e+06
==============================================================================
```

Figure 10. Burberry OLS Model Details

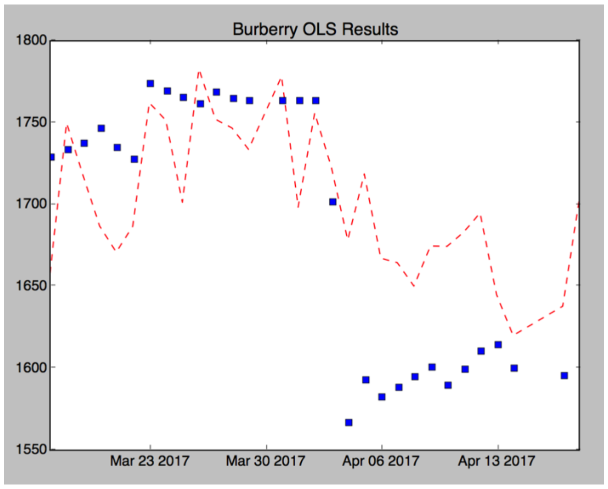


Figure 11. Burberry OLS Predictions vs Actual

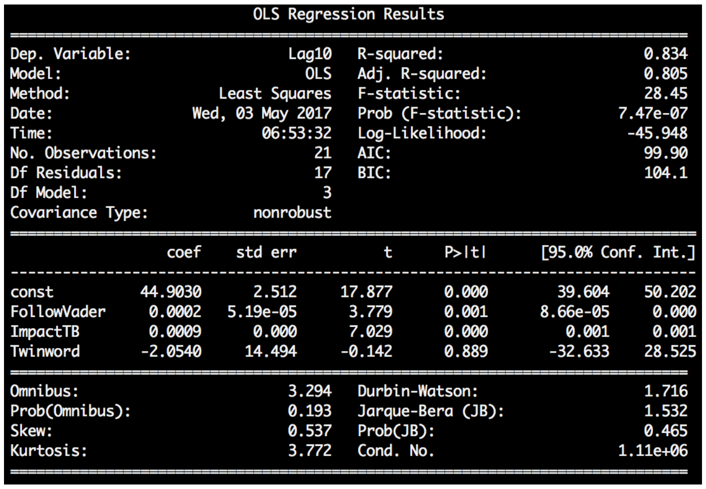
Due to the problems of multicollinearity and overfitting, we chose Random Forest (RF) regression as our second predictive model. Random forests [18] use multiple learning algorithms together in order to obtain a better predictive model. It produces many decision trees at the time of training and then outputs the mean regression of those trees as the final result. Random forests can rank the importance of variables in a regression, so for this model we did not need to give it the features we chose from the correlation values. Rather, we give it all predictors and it choose which to use and with what weights in order to form the outputted regression the model. The fact that this method constructs many different trees based on different features and aggregates their results minimizes issues of overfitting.

V. Evaluation

A. *Ordinary Least Squares Regression*

The following visuals are given in order to demonstrate the predictive accuracy of the OLS models we created for Burberry, Lululemon Athletica, and Urban Outfitters respectively. In the graphs, the dashed red line represents the predicticted stock price and the blue squares represent the actual stock price. The dates March 17th to April 6th were included in the training data, whereas April 7th to April 17th were held out as testing data.

The Burberry OLS model resulted in an overall correlation of 0.314 and a mean squared error, which measures the average difference between the prediction and the actual value, of 3495.030. The mean squared error (MSE) is best used to compare multiple models of the same data. It is not as useful in seeing how accurate a model is on its own.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Lag10   R-squared:                       0.834
Model:                            OLS   Adj. R-squared:                  0.805
Method:                 Least Squares   F-statistic:                     28.45
Date:                Wed, 03 May 2017   Prob (F-statistic):           7.47e-07
Time:                        06:53:32   Log-Likelihood:                -45.948
No. Observations:                  21   AIC:                             99.90
Df Residuals:                      17   BIC:                             104.1
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const         44.9030      2.512     17.877      0.000        39.604    50.202
FollowVader    0.0002   5.19e-05      3.779      0.001      8.66e-05     0.000
ImpactTB       0.0009      0.000      7.029      0.000         0.001     0.001
Twinword      -2.0540     14.494     -0.142      0.889       -32.633    28.525
==============================================================================
Omnibus:                        3.294   Durbin-Watson:                   1.716
Prob(Omnibus):                  0.193   Jarque-Bera (JB):                1.532
Skew:                           0.537   Prob(JB):                        0.465
Kurtosis:                       3.772   Cond. No.                     1.11e+06
==============================================================================
```

Figure 12. Lululemon OLS Model Details

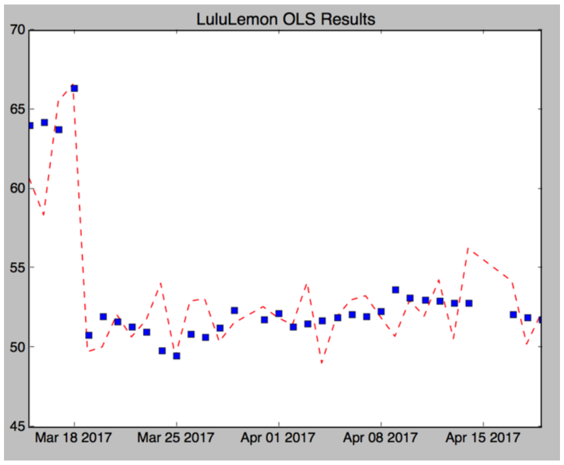
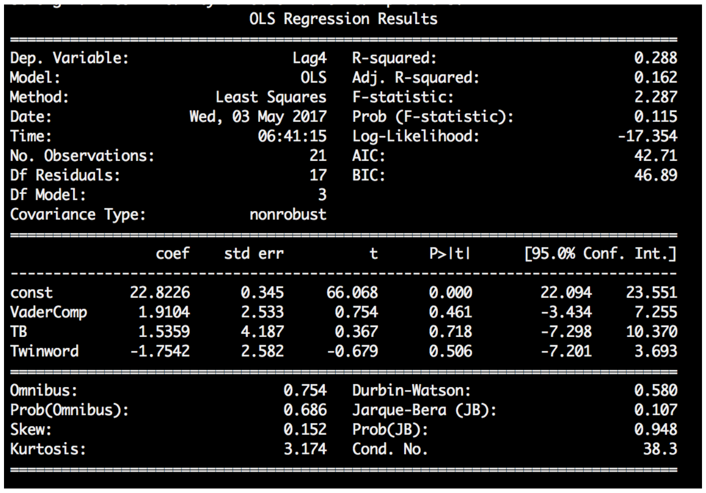


Figure 13. Lululemon OLS Predictions vs Actual

The Lululemon OLS model resulted in an overall correlation of 0.834 and a mean squared error of 4.140.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   Lag4   R-squared:                       0.288
Model:                            OLS   Adj. R-squared:                  0.162
Method:                 Least Squares   F-statistic:                     2.287
Date:                Wed, 03 May 2017   Prob (F-statistic):              0.115
Time:                        06:41:15   Log-Likelihood:                -17.354
No. Observations:                  21   AIC:                             42.71
Df Residuals:                      17   BIC:                             46.89
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const         22.8226      0.345     66.068      0.000        22.094    23.551
VaderComp      1.9104      2.533      0.754      0.461        -3.434     7.255
TB             1.5359      4.187      0.367      0.718        -7.298    10.370
Twinword      -1.7542      2.582     -0.679      0.506        -7.201     3.693
==============================================================================
Omnibus:                        0.754   Durbin-Watson:                   0.580
Prob(Omnibus):                  0.686   Jarque-Bera (JB):                0.107
Skew:                           0.152   Prob(JB):                        0.948
Kurtosis:                       3.174   Cond. No.                         38.3
==============================================================================
```

Figure 14. Urban Outfitters OLS Model Details

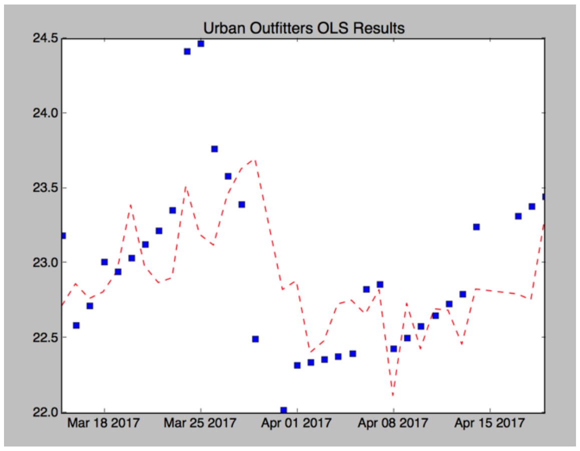


Figure 15. Urban Outfitters OLS Predictions vs Actual

The Urban Outfitters OLS model resulted in an overall correlation of 0.288 and a mean squared error of 0.229.

B. *Random Forest*

Likewise, the following visuals are given in order to demonstrate the predictive accuracy of the Random Forest regression models for Burberry, Lululemon Athletica, and Urban Outfitters respectively. The first graph shows the importance that the aggregate regression forest gave to each of the features. In the graphs, the dashed red line represents the predicticted stock price and the blue squares represent the actual stock price. The dates March 17th to April 6th were included in the training data, whereas April 7th to April 17th were held out as testing data. It is important to note that, on each run of our random forest regression script, a slightly different forest will be decided for the regression model, as the trees are constructed in a randomized manner each time.

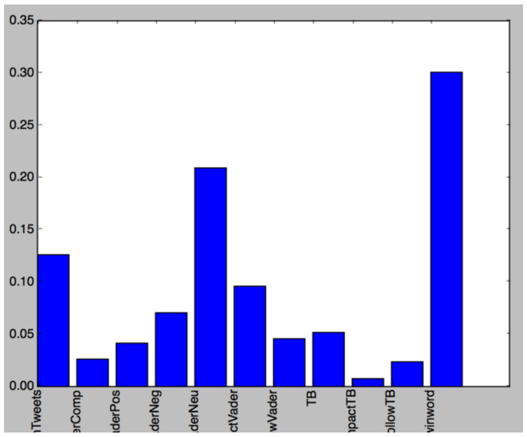


Figure 16. Burberry RF Feature Importance

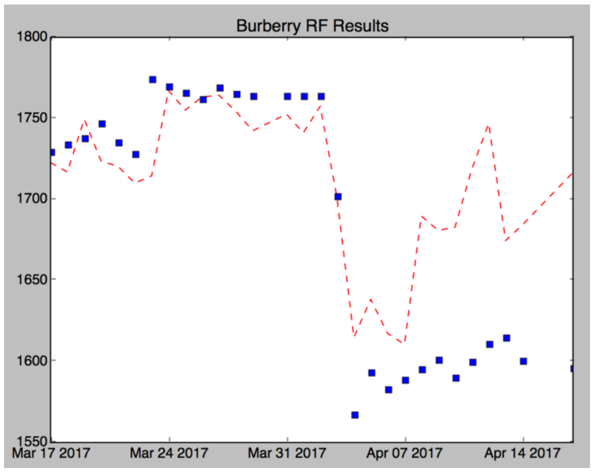


Figure 17. Burberry RF Predictions vs Actual

The Burberry RF model resulted in an overall correlation of 0.470 and a mean squared error of 3296.252.

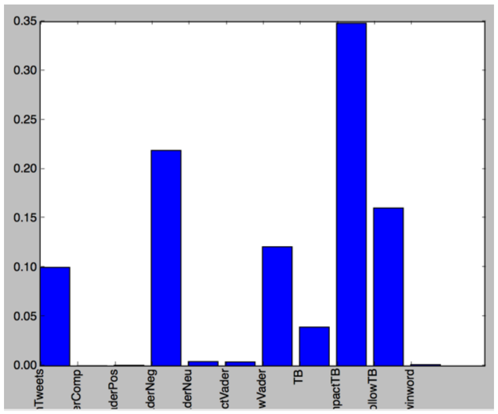


Figure 18. Lululemon RF Feature Importance

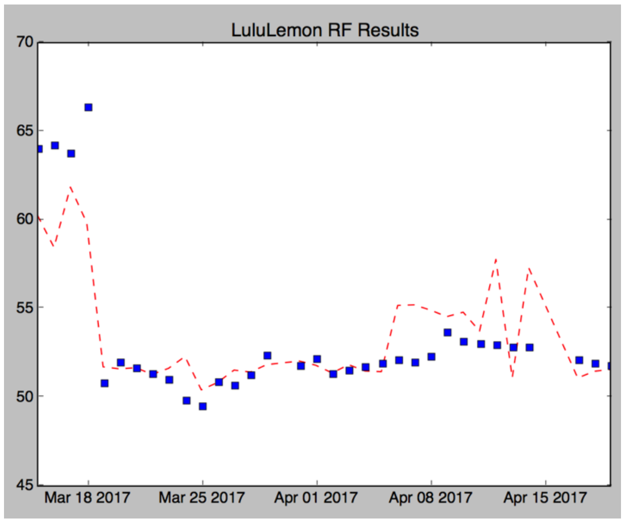


Figure 19. Lululemon RF Predictions vs Actual

The Lululemon Athletica OLS model resulted in an overall correlation of 0.698 and a mean squared error of 5.534.

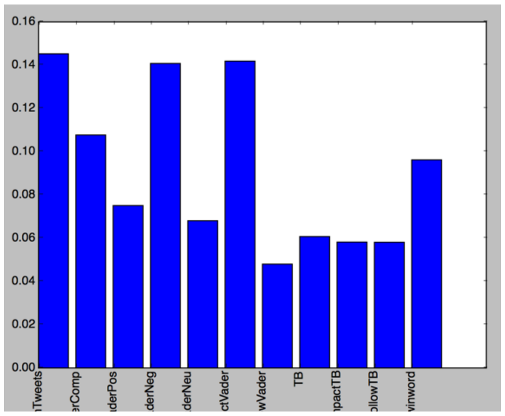


Figure 20. Urban Outfitters RF Feature Importance

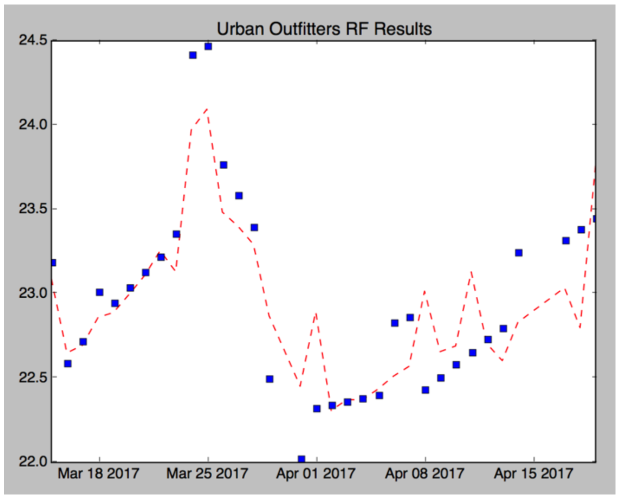


Figure 21. Urban Outfitters RF Predictions vs Actual

The Urban Outfitters OLS model resulted in an overall correlation of 0.741 and a mean squared error of 0.083.

C. *Discussion*

Both regression models seem to predict the overal trend in stock prices fairly accurately. That being said, Random Forest performs better for Burberry and Urban Outfitters predictions as seen by the comparison of correlation and MSE values. As a reminder, we would like a correlation (r^2) value closer to +1 and an MSE value closer to 0. The OLS model for Lululemon, however, performs slightly better than its Random Forest regression. Overall though, random forest performs well for all three companies, with less of a difference in performance between companies than OLS. Therefore we conclude that our Random Forest regression models would be good predictors of stock price for these companies, especially once given more training data.

VI. Limitations and Future Work

Our predictive models, particularly the Ordinary Least Squares regression, was negatively impacted by our limited selection of features. It would help to incorporate more predictors in future iterations of this work. For example, we would like to include different categories and rankings of tweets by certain users, such as celebrities, the company itself, and the ordinary customer. This would allow us to take into consideration special tweets like celebrity endorsements or sales posted by the company. It would also allow us to see if a company does better when they engage with their customers more on social media. We would also try to separate tweets about the company stock itself from tweets about company products, giving two separate categories for analysis. Another interesting feature we could add to the model would be one to measure overall financial health. Since other studies have found that Twitter data can predict trends for the stock market in general, we could use this to account for swings in company stock that are caused by unrelated world events.

We were also limited by a lack of data. As we move forward, we need to have larger sets of training and testing data. We also think it would be helpful to include more companies for further sets of data. This may help us understand certain aspects of our models. For example, we may be able to conclude that more highly valued companies, like Burberry, take longer to react to sentiment on social media, resulting in the need to predict stocks further out. We questioned this based on the lags chosen for this iteration of research, but without more companies for comparison, it is impossible to make any conclusive deductions. Additionally, we could get more data by looking at other forms of social media besides Twitter. This would be particularly interesting given we are analyzing companies in the fashion industry, which has a large presence on visual platforms, such as Instagram.

Finally, we were limited by the number of models we had time to form and analyze. We would like to build other types of models, such as neural networks, and see if these can predict stock prices even bettter than Ordinary Least Squares or Random Forest regression.

VII. Conclusions

In regards to our problem statement, we believe, based on the results of this research, it can be concluded that company-related Twitter data can predict trends in that company's stock for companies in the Consumer Discretionary Sector, specifically Burberry, Lululemon Atheltica, and Urban Outfitters. We find it important to emphasize that our research showed the ability to predict trends, not individual stock prices for a given day. Additionally, we concluded that these trend prectictions are more accurate using Random Forest regression than using Ordinary Least Squares regression.

We believe this work will be helpful to both companies and investors. If companies could use social media to give them advanced notice of falling stock prices, they may be able to change their marketing strategies to remedy this situation before it even happens. On the other hand, investors could use this information to gain a more robust understanding of when to buy or sell a particular company's stock.

VIII. References

[1] E. Fama, The Journal of Business (1965) 38, 34–105.

[2] http://knowledgegrab.com/glossary/efficient-market-hypothesis-emh-8/

[3] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science* 2.1 (2011): 1-8. Web.

[4] L. Bing, K. Chan, C. Ou, and S. Ruifeng, "Discovering Public Sentiment in Social Media for Predicting Stock Movement of Publicly Listed Companies," in *Information Systems*, Elsevier, February 2017.

[5] M. Nofer, "Using Twitter to Predict the Stock Market: Where Is the Mood Effect?" *Value of Social Media for Predicting Stock Returns*. Germany: Springer Vieweg, 2015.

[6] C. Dijkmans, P. Kerkhof, A. Buyukcan-Tetik, and C. Beukeboom, "Online Conversation and Corporate Reputation: A Two-Wave Longitudinal Study on the Effects of Exposure to the Social Media Activities of a Highly Interactive Company."*Journal of Computer-Mediated Communication* 20.6 (2015): 632-48.

[7] https://finance.yahoo.com/quote/LULU/?p=LULU

[8] https://finance.yahoo.com/quote/BRBY.L?p=BRBY.L

[9] https://finance.yahoo.com/quote/URBN/?p=URBN

[10] https://github.com/lukaszbanasiak/yahoo-finance

[11] https://github.com/cjhutto/vaderSentiment

[12] https://textblob.readthedocs.io/en/dev/

[13] https://www.twinword.com/api/sentiment-analysis.php

[14] https://market.mashape.com/twinword/sentiment-analysis

[15] http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html

[16] http://setosa.io/ev/ordinary-least-squares-regression/

[17] https://www3.nd.edu/~rwilliam/stats2/l11.pdf

[18] https://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf