Twitter Sentiment Classification Analysis: Group COP

Oliver Muellerklein, Chris Lin, Paul Land

April 29, 2016

EXAMPLE CODE!

3.2 $\triangle$TF-IDF

$\hat{y}$ is our estimate.

The $\triangle TF - IDF$ value for a term t in the ith Tweet is $\triangle TF - IDF_{t,i} = TF_{t,i} \times log_2 \left(\frac{P}{P_t} \frac{N_t}{N} \right)$, where $TF_{t,i}$ is the number of times t appears in the ith Tweet, P the total number of positive Tweets, P_t the number of positive Tweets that have t in them, N the total number of negative Tweets, and N_t the number of negative Tweets that have t in them.

Include this in paper:

After finding the max AUC values for the 2 sets of static parameter values and the single set of parameters - from the 5-Folds CV over 300 randomly generated XGBoost model parameters - I decided to attempt a customized approach to the CV and optimization of the boosting model. When I put my parameters from one of static sets into the training algorithm I found that the best mean error for classification was *0.2506* at an iteration index of *127*. Somewhat similarly, I found that when using a custom normalized Gini optimization function, the best score was *0.7059* at the optimal index of *131*.

Due to fluctuations between models with slight parameter tweaks and the low performance of nearly all models on classifiying negative tweets, we decided to incorporate Ensemble Learning by combining sets of our models. We can perform the combination of predictions per model by using either an averaging approach, with mean averaging and geometric averaging options, or a majority rule approach.

EXPLAIN WHAT WE FOUND WHEN WE DID THIS AND HOW WE COULD GO ABOUT THIS IN BETTER WAYS IN THE FUTURE WITH MORE TIME

Although the ensemble approach is very enticing, there may be doubt cast on the validity of blindly picking and combining predictors if proper statistical comparisons are not made. For example, in our next steps we would want to compare the correlation between models using the Pearson's Coefficient algorithm. Then we would want to combine models that are the most uncorrelated to combine. This helps prevent overfitting as well as accounts a complex emergence of combined model behavior due to drastically different model structures. An interesting approach we could try is the use of recursive decision trees applied on the outcome of SVM in various ways.

1 Introduction

1.1 Why Twitter Sentiment?

Social media has rapidly become a substantial mode of communication due to a constant influx of new users, modes of language expression (*e.g. emoticons*), and units of socio-cultural importance. From both a sociological and information theory standpoint due, social media platforms such as Twitter provide an unprecendented depth of knowledge and insight into language and communication in a human-technology infused ecosystem. With an average of 6,000 tweets sent per second and 500 million per day [1], the use of Twitter for self-expression, group-based decision making, and socio-cultural communication is an essential component of modern society. The ability to analyze and predict trends in tweet sentiment may prove extremely useful through: a) financial benefits for industry, b) exploring human behavior within socio-economic, culutural, and linguistic research, and c) discovering novel modes of language in a world where human life and technological advancement are so intimidately related.

1.2 Our Data

We were provided a set of nearly *1.6 million tweets*, as character strings, and corresponding sentiment scores (*0 or 1*) of all but the first 50,000 observations. This first non-sentiment subsample of the data is put aside as the final data to test predictions against (*i.e. compare y_hat to*). There was no metadata provided - for example, we did not obtain the user IDs nor names corresponding to each tweet. Our data consistently of solely the tweet, the source, and sentiment.

Need to actually make the TeX yHat! Also - need some other equations.

1.3 Let's Explore this Data!

1.3.1 First Look

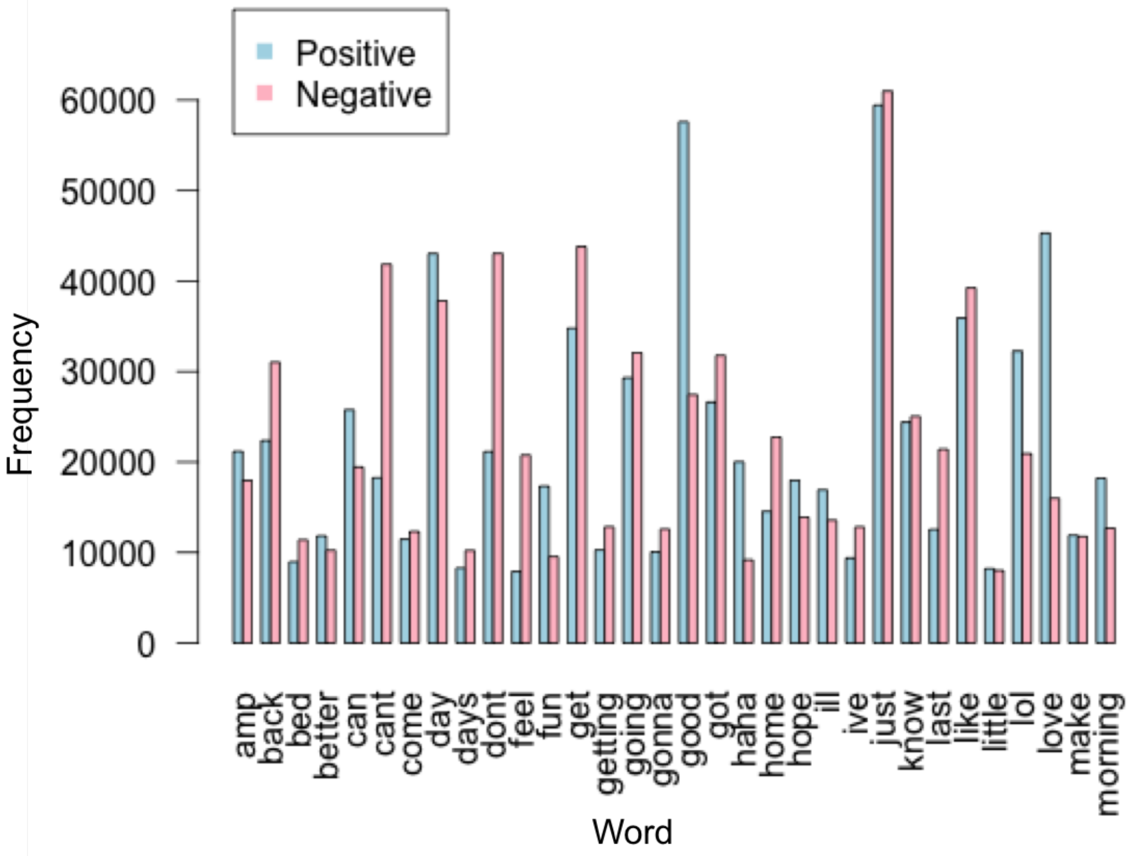
Initially we found that nearly all tweets came from source Sentiment140. However, a small subsample of the tweets were sourced from Kaggle, which we found out was from a Twitter sentiment challenge on Kaggle that was based on movie sentiment [2]. Another attribute of the data that we found (and corrected) was the existence of unmatched double quotation marks within a set of tweets. Using the Regex package in *R* we replaced such quotation marks.

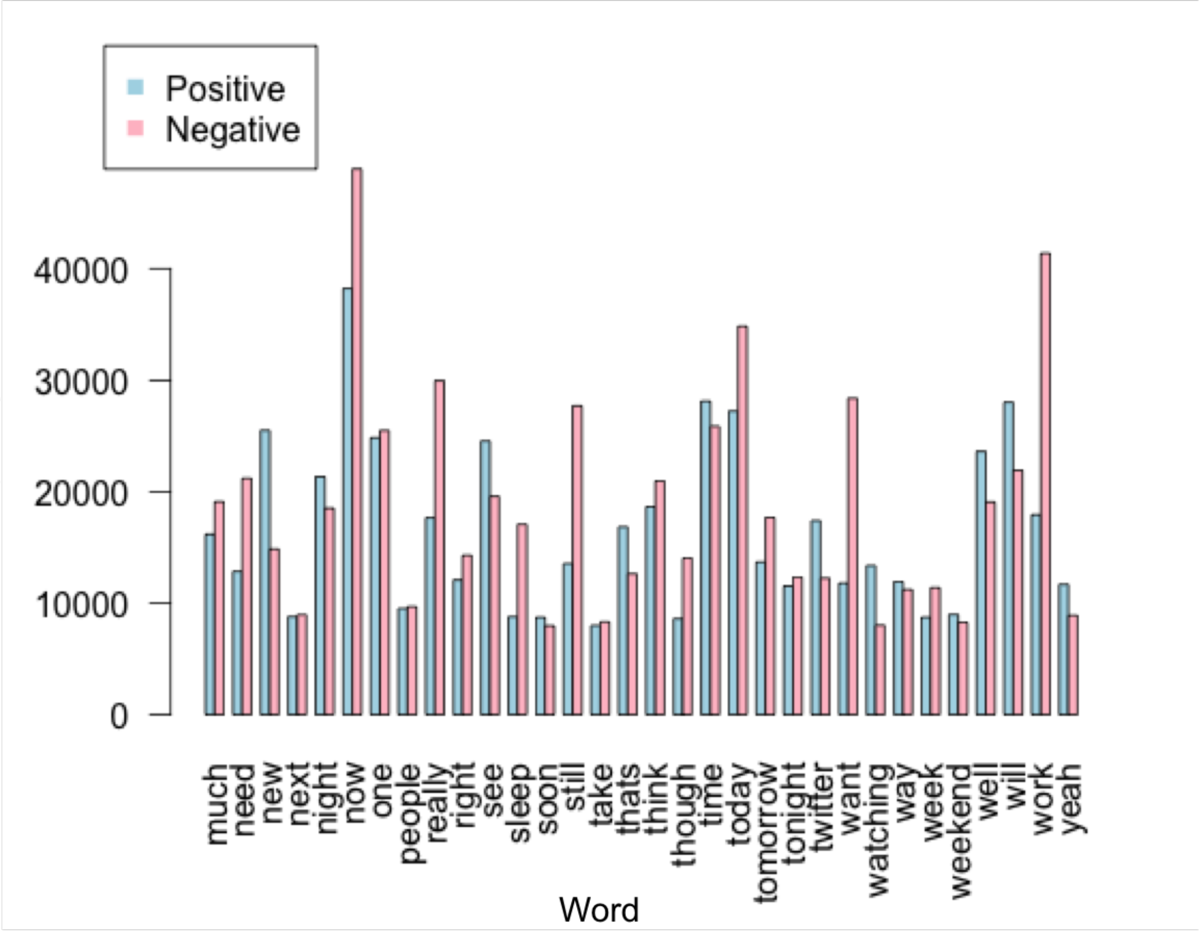
1.3.2 Data Breakdown

The raw data contains 4 variables: `ItemID`, `Sentiment`, `SentimentSource`, and `SentimentText`. There are 1578627 total tweets. There are 765127 positive tweets, 763500 negative tweets, and 50000 unlabeled tweets. Furthermore, we used numerous `regex` patterns to explore the tweets.

1.3.3 Positive and Negative Word Distributions

As shown in figure 1 (below), there were differences between the word occurrences of positive and negative tweets.


Positive
Negative
60000
50000
40000
30000
20000
10000
0
Frequency
amp
back
bed
better
can
cant
come
day
days
dont
feel
fun
get
getting
going
gonna
good
got
haha
home
hope
ill
ive
just
know
last
like
little
lol
love
make
morning
Word



2 Feature Engineering

2.1 Username/Hashtag Positive/Negative Score Features

After we loaded the raw data into `R` and explored the tweets, we began the feature engineering process. We focused on using the strings of tweets to extract useful information and construct features. We did not use the `ItemID` and `SentimentSource` variables in the raw data. The first set of features we constructed were based on the frequency of usernames and hashtags in the tweets. We wanted to explore the possibility that certain usernames/hashtags tend to appear in more positive or more negative tweets. To build those features, we first divided the data set into a set of positive tweets and negative tweets. For each set, we calculated the frequencies of each unique username and hashtag. For example, `@mileycyrus` appeared `2833` times in the positive tweets. With this information, we gave each username and hashtag a positive and negative score. The scores were calculated by dividing each username's/hashtag's frequency by the postive/negative set's highest username/hashtag frequency. Finally, if a particular username/hashtag appeared in a tweet, we assigned the tweet a corresponding username/hashtag score. Ultimately, this process created `4` features: positive username score, positive hashtag score, negative username score, and negative hashtag score.

The top 5 usernames found in the positive tweets:

Positive Username	Frequency	Negative Username	Frequency
@mileycyrus	2833	@tommcfly	1480
@tommcfly	2073	@mileycyrus	1339

Positive Username	Frequency	Negative Username	Frequency
@ddlovato	2001	@ddlovato	1255
@DavidArchie	902	@DonnieWahlberg	504
@Jonasbrothers	860	@mitchelmusso	416

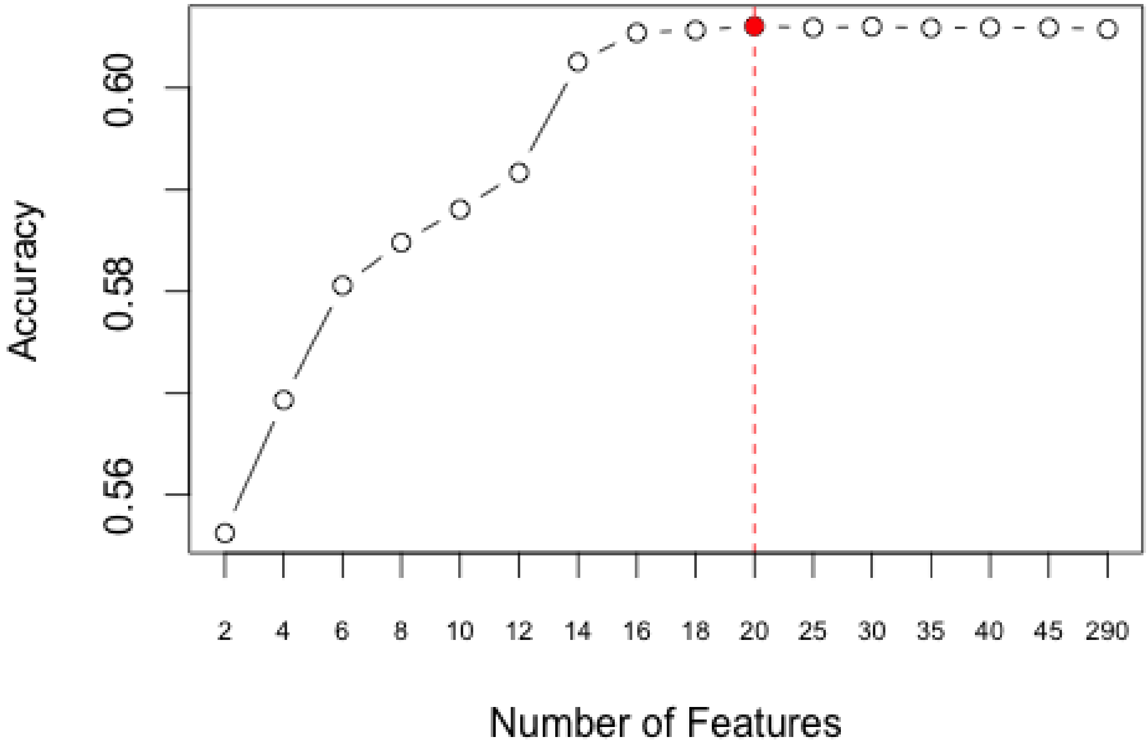
The top 5 usernames found in the negative tweets:

2.2 Bag of Words Features

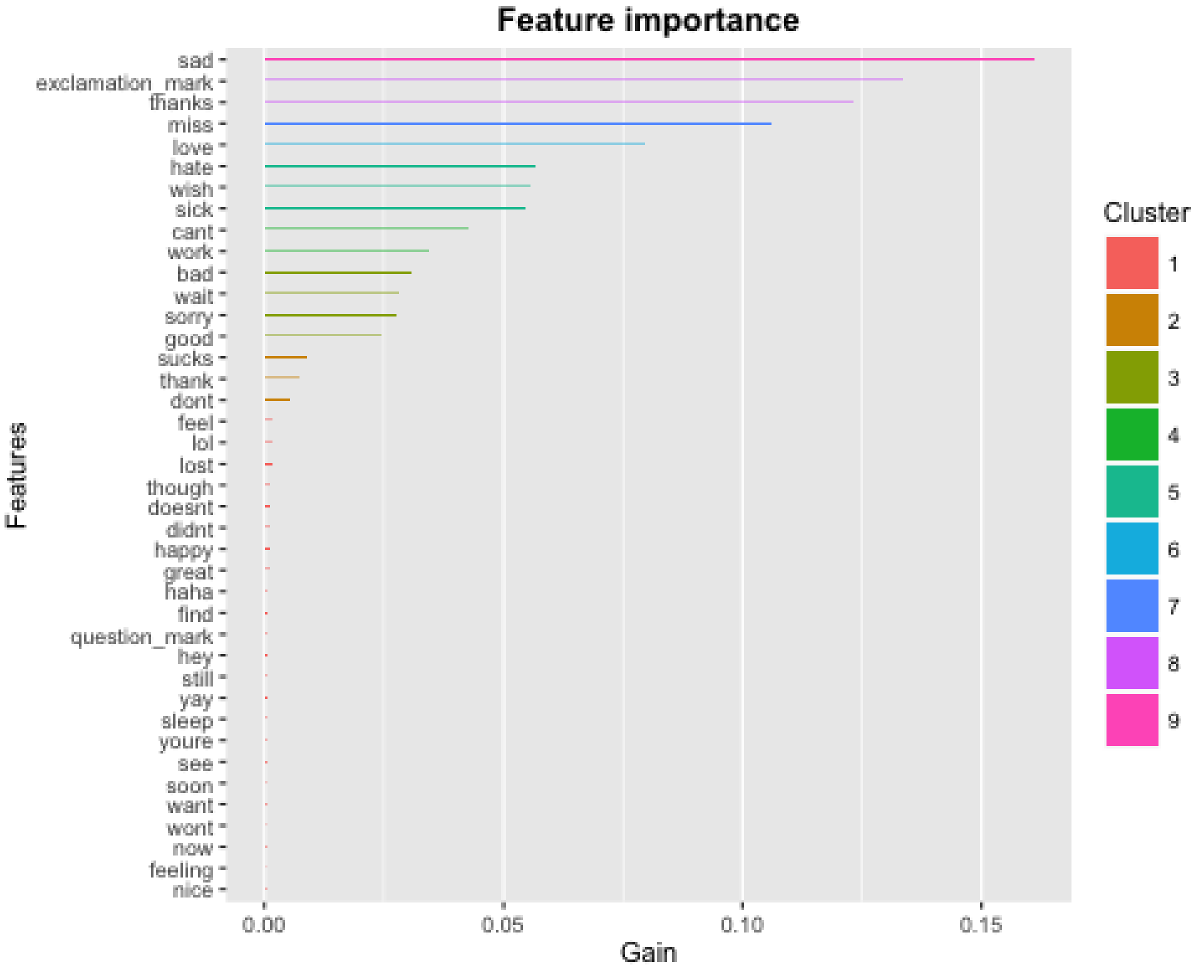
We also incorporated a **Bag of Words (BOW)** approach with **delta Term Frequency - Inverse Document Frequency** (delta TF-IDF, explained below) to construct a corpus of positive, negative, and relative term frequencies. These terms were used to generate a large sparse matrix with direction relative term frequencies (positive - negative tweets). Due to the practical limitations of computational cost, we generated our corpus and corresponding frequencies based on all words that appeared at least 1% of the time. This limited our word (term) choice to **118**. However, we also decided to incorporate key special terms and character usage that had high correlation and frequencies with sentiment. For example, we incorporated exclamation mark and *LOL* usage as seperate features. In all, we generated a 290 feature sparse matrix of delta TF-IDF based frequencies, 4 additional score features based on frequencies of usernames and hashtags, and, eventually, a 20 feature subsample based on feature selection methods (described below).

3 Cross Validation and Model Formation

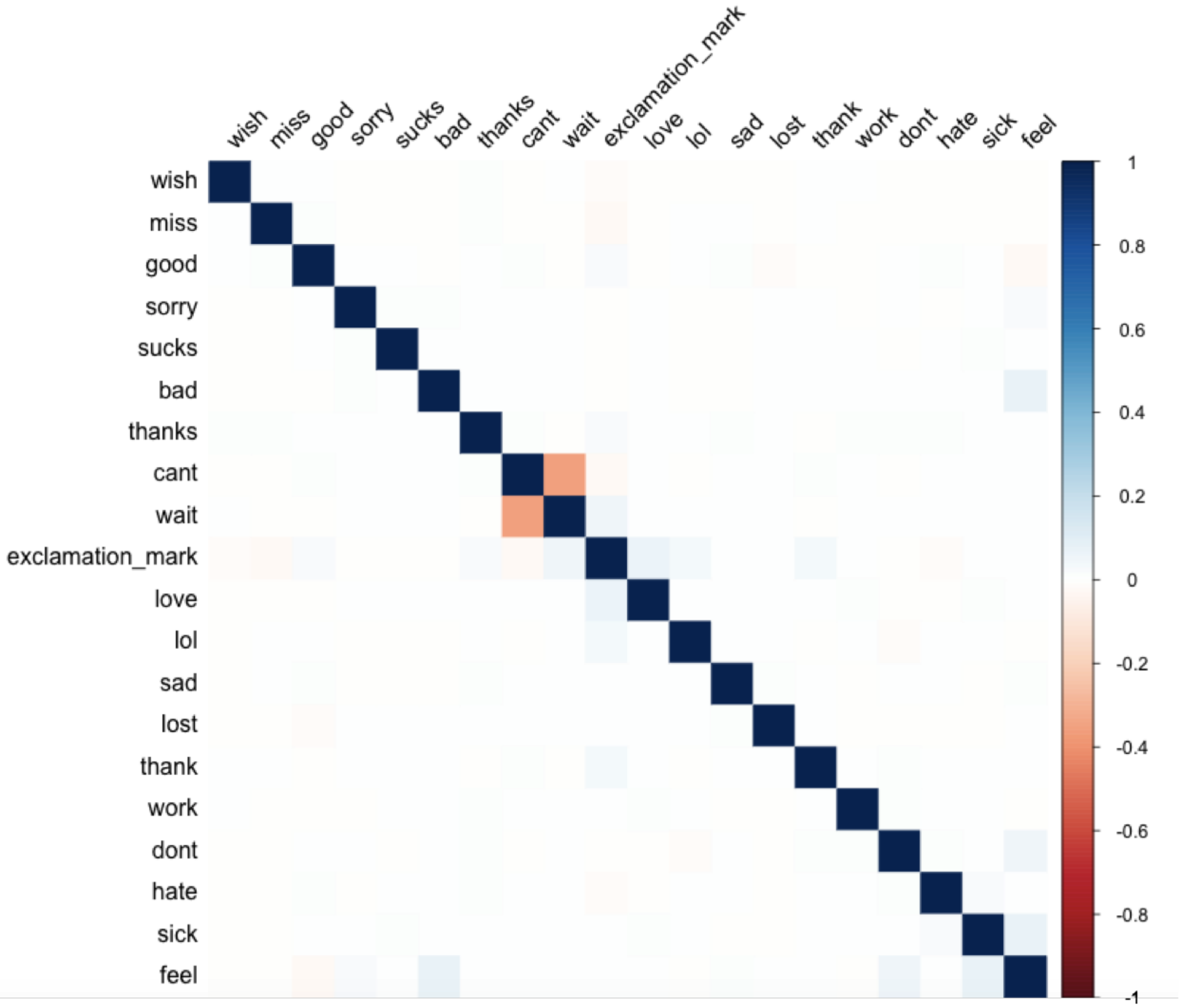
3.1 Feature Selection Overview



Through various feature importance and selection methods, we found that a majority of the loss reduction within random forest and boosting methods (via XGBoost) was captured by a small subset of our total features. For example, we approached the problem initial with 50 features from the respective word frequencies via Bag-of-Words and four normalized features corresponding to usernames and hashtags that occurred most frequently with both positive and negative tweets. Feature selection in both random forest and gradient boosting (XGBoost) dramatically selected for the four normalized username and hashtag predictors. Furthermore, the same pattern existed when increasing our Bag-of-Words term frequency usage from 50 to 118 words. We had also found that our initial models - including logistic regression with regularization, random forest, and gradient boosting tree ensemble (XGBoost) - generally obtained significantly greater accuracy and decreased misclassification for positive than negative tweets. For example, one method produced an accuracy above 80% for positive and just over 50% for negative tweets. As expected, when we compare the negative prediction accuracy and misclassification rate to random guessing we see minimal to no benefit.



One possible explanation for the discrepancy in sentiment prediction may be due to feature engineering bias. The total number of positive and negative tweets was nearly equal in the original dataset. However, our Bag-of-Words and Term Frequency - Inverse Document Frequency methods may not have captured the vast array of word choices for negative tweets as we simply took words that occurred in at least 1% of all tweet words. This rests on the assumption that single word use and frequency either 1) give a general, accurate account for tweet sentiment differences and 2) the underlining distributions of positive and negative word counts and frequencies are equal. The first point could be explained by claiming that some n-gains or set of words would give stronger features. The second point comes from the statistical conceptualization of finding signal within noise of data. To elaborate, in order to use the word frequencies and differences between positive and negative tweets from the words that appeared in at least 1% of tweet words, we must assume that such words are a valid subsample. Thus, we are assuming that the 118 words that occurred in at least 1% of the total tweet words are an accurate representation of both the actual word frequencies (for all words) in all tweets and the differences in usage between positive and negative sentiment.



On the other hand, we used normalized username and hashtag occurrence scores for positive and negative features. These four features were far less sparse than the Bag-of-Words and TF-IDF generated features - i.e. far less observations with 0. This can bias the importance of those features in the decrease in misclassification (loss, similar principle can apply for other forms of optimization) for each node split.

3.2 Delta TF-IDF

3.2.1 Standard TF-IDF

We can apply the TF-IDF principle to the top occurrences of both usernames and hashtags to get the corresponding sparse features in the same format as the 118 word frequencies. We determined the number of important usernames and hashtags by analyzing the total number of unique values, e.g. unique usernames in all tweets, and finding the total number of occurrences per element. From this data exploration method we found that the username @mileycirus appeared as the second most frequented tweeted username in all negative tweets. The top three negatively tweeted usernames appeared more than 1,000 times but dropped significantly by the fourth or fifth most population negative username. We were able to pick the top 50 usernames based on occurrences in positive, negative, or all tweets. Similarly, we found the top 50 hashtags. These top users and hashtags were fed into the TF-IDF algorithms we created as terms and their corresponding sparse predictors were augmented to the data as features.

3.2.2 To Delta

We augmented our 118 words with an addition set of high occurrence *usernames*, *hashtags*, and *special terms* - such as special characters that do not necessarily appear often but are consistent with either positive or negative tweets - thus creating a sparse matrix of 290 predictors. This was fed into Random Choice, Logistic Regression, Support Vector Machines (SVM), and Gradient Boosted Tree Ensembles (via XGBoost). However, after initial model runs with poor performance we decided to explore the use of *delta TF-IDF* to create vectorizations of our sparse matrix features. The directionality is calculated by assigning the features, as described above, with positive or negative signs depending on the relative frequency and relationship of the corresponding term in positive and negative tweets.

3.3 Incorporating Normalized Scoring

Surprisingly, we obtained significant increases in the quality of our accuracy, AUC, and (decreased) train and test mean error through the incorporation of 4 normalized username and hashtag features. These 4 extra predictors were obtained by calculating the relative frequency of positive and negative usernames and hashtags for all unique instances of each.

I think the beginning of this section (3) needs to be cut and put here after explaining the delta-TF-IDF. Also, the first and second paragraphs of this section (3) have stuff about future direction that we need to put in that section (at the end).

4 Model Formations and Results

All models were analyzed using **5-Fold CV** to optimize mean error and / or AUC:

Models	Tuning	Training	Predicting
Random Forest	Features	20% of data	All data
XGBoost	Features & Parameters	20% of data	All data
Linear SVM	Grid-search Cost	0.05%-20% of data	?
Gaussian KSVM	Grid-search Cost & Sigma	0.05%-20% of data	?

4.1 Random Forest

We used random forest approaches to perform initial model runs, as well as feature importance, with our 290 generated delta TF-IDF predictors. Feature importance analysis with random forest and XGBoost approaches involved examining the *gain* of each feature as an average of net total over all node splits. Gain is measured as the decrease in misclassifications - or recipricol of increase in correct classification.

4.2 Compare XGBoost

We used XGBoost with optimal parameter tuning on 300 randomly generated values within specified bounds (parameter-specific). Through these randomly generated parameter sets, we ran 5-fold Cross-Validation with a 20% subsample of the total labeled data. Out of this 20% subsample we used a 90% / 10% split for training and testing in-group, respectively. Our 300 random parameter searches, via grid search methods from the Caret package in R, were performed in sets of 50 so we could ultimately rank and analyze 6 different sets of optimal parameters. The best set, in terms of maximum AUC values, was used to generate our most accurate model by total accuracy.

The results and comparison of several models are summarized in the table below:

XGBoost	Test	Score	Data	Method
	CV Accuracy	0.7724891	290 BOW + 4 Scores	5-Fold CV
	CV AUC	0.8706949		

```
CV Accuracy   0.7429594   290 BOW              5-Fold CV
CV AUC        0.8410817
```

4.3 Linear and Gaussian K-SVM

Not sure if we are including this section really? Or I guess at the least we can say we ran them etc.

5 Model Analysis and Fine Tuning

5.1 Feature Selection and Tuning

We performed several optimization and parameter tuning tricks, including:

1. various train / test splits of the labeled (sentiment of 0 or 1) data from 50% / 50% to 90% / 10%
2. early stopping (XGBoost) when loss is not improved for 10-50 iterations of boosting - this is an optimal method to extracting optimal iteration number while preventing overfitting
3. K-means clustering on relative feature-based gain (or loss) - this is particularly useful for *post facto* feature selection, augmentation, and further ensemble-level engineering
4. K-folds Cross-Validation for feature and model selection - this is performed by randomly extracting the training dataset into K number of subsets, training the model on that subset, and then testing on the remaining training dataset

6 Conclusion and Future Ideas

Other future ideas:

- one-hot-encode: for XGBoost features - known to do well for sparse matrix classification Check this out...
- ensemble combination of predictions with: average, geometric average, majority rule

There are several steps and ideas we had for future direction with this project. The two most obvious are to i) redefine our term use in the TF-IDF algorithm and ii) further our feature and model selection and tuning methods towards an exhaustive search of both. The former idea could be done by adjusting our idea of *words* and redefining the terms we use to be some set of characters or relative word groupings, perhaps through an n-gram approach. The latter idea could be done through the use of the *Caret* library in *R* to perform an exhaustive grid search. Another approach could be to perform adaptive solvers either *post facto* or during iteration boosting - such as particle swarm optimization or other forms of evolutionary algorithms. These approaches may be able to adaptively solve for the optimization problem of misclassification or accuracy gain / loss in a non-parametric fashion. I.e. such adaptive solvers in optimization problems do not assume distributions or relationships between parameters and can be used to search through the feature dimensional space for non-linear solutions. This could of course be performed in a different route with kernel based approaches of dimensional reduction, such as kernel-PCA and kernel-SVM (NOTE: some of these approaches were touched on or implemented fully in our report above).

Appendix

[1] http://www.internetlivestats.com/twitter-statistics/

[2] https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/data

[3] XGBoost: https://github.com/dmlc/xgboost

END