Help Taylor Swift Identify Anti-fans On Twitter

Yuchen Yang

October 26, 2016

1 Introduction

Twitter nowadays becomes one of the most popular online social networking software which has more than 310 million monthly active users as of 2016 [1]. It enables users to send and read short 140-character messages called "tweets" [1]. Information spreads incredibly fast on Twitter. Every second, on average, around 6,000 tweets are tweeted which corresponds to over 500 million tweets per day [2]. "Twitter trolls" have become a common phenomena worldwide. By Wikipedia's definition, an internet troll is a person who sows discord on the Internet by starting arguments or upsetting people, posting inflammatory extraneous, or off-topic messages in an online community with the deliberate intent of provoking readers into an emotional response [3]. Celebrities especially suffer from trolling or mean comments on Twitter. It is such a common phenomena that a popular TV show *Jimmy Kimmel Live* has a series of short videos about celebrities read mean tweets themselves. Our ultimate goal in this project is to develop a prediction algorithm for identifying and classifying users that are trolling or being mean on Twitter.

However it is difficult to quantitatively define a troll or mean user in general. To make the problem more specific and solvable, in this project, we narrow down the problem to identify anti-fans for Taylor Swift. Taylor Swift, as one of the most popular singer, has more than 80 million followers on Twitter. However she recently has involved into fights with several celebrities, such as Katy Perry, Kim Kardashian and Kanye West [4]. Although she has a huge fandom named themselves "swifties" supporting her, she still has a lot of anti-fans initiate topics such as TaylorSwiftIsOverParty and KimExposedTaylorParty on Twitter.

In this project, we define anti-fans as users who express their dislike to Taylor Swift on Twitter, such as tweet swear words to her. We use mean user and anti-fan interchangeably in the following sections.

2 Methods

We apply text mining approaches to identify anti-fans in this project. The key idea of text mining is that we believe there is a difference of frequency of words used by fans versus anti-fans. We apply machine learning technics to detect this difference for classification. The pipeline of analysis is illustrate in figure 1.

In this project, the "Gold Standard" dataset is obtained by manually tagging whether a user is anti-fan or fan. We believe that to judge if a person expresses dislike to someone is subjective

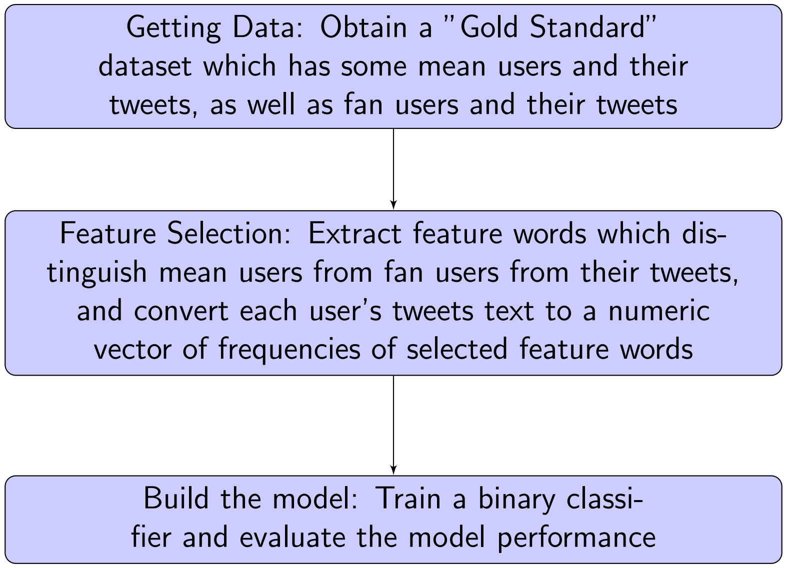

Figure 1: Pipeline of the project

and there is no hard standard for identifying it. So the best estimate we could give is to judge from a human perspective. The details involved in obtaining the data are described in section 2.1. Then we present some exploratory data analysis (EDA) results, by performing sentimental analysis and plot the wordcloud in section 2.2. We introduce the details of feature selection and the statistical model in section 2.3 and section 2.4.

2.1 Data Collection

We use R package "TwitteR" [12] in this project to get data from Twitter. The TwitteR package is intended to provide access to the Twitter Application program interface (API) within R, allowing users to grab interesting subsets of Twitter data for their analyses [5]. OAuth authentication is required for all Twitter transactions. To get the authentication, we need to create a Twitter application from https: //twitter.com/apps/new and follow the instruction [5] to get API key, API secret, Access token and Access secret.

The basic function for TwitteR package is searchTwitter() to search for tweets that contain certain key words. However a limitation of this function is that it will wrap an arbitrary number of actual search calls to provide the number of tweets requested, and the Twitter API could return tweets within several days before the search day. It would not return tweets created too long ago, even a time is specified in the searchTwitter() function. The limitation of this function makes the search procedure not reproducible, since the return is random and even not available when the search request is sent later.

Another function is getUser(), which provides information such as description, number of followers etc for a certain user. To get tweets for each user, we use userTimeline() function to search for recent tweets of a certain user.

Depends on the different requests sent to Twitter API, there are different API rate limit per 15 minutes window. For example, the rate limit for userTimeline() function is 300 requests per 15 minutes window for application authentication search [6]. So during the search process, if the rate limit is reached, we need to pause for 15 minutes to continue another search.

2.1.1 Preliminary Steps

The most straightforward way to identify potential anti-fans is to find users that tweet swear words to Taylor Swift. So we come up with a bad word dictionary, which contains 172 swear words and their internet variations. The bad words are listed in file "mybadword.txt". This dictionary is a modification from [7], which I deleted some uncommon swear words for simplicity and add some bad words particularly for Taylor Swift, such as "snake". For each bad word in the dictionary, we search for 300 tweets that contain the word and mention Taylor Swift. Then we get the screen usernames of those returned tweets as anti-fans. Similarly for fun user we build a good word dictionary. Since there are much more fans than anti-fans, we just use a single word "love" as our good word dictionary. We search for 3000 tweets that contain "love" and mention Taylor Swift, and get the usernames as fans. This search is done on October 1, 2016.

However we find this is not enough to correctly identify anti-fans and fans. Essentially a user tweet bad words is not equivalent with expressing dislikes, the same for good words. For example, people would tweet "@taylorswift13 is damn pretty" or "why people hate @taylorswift13". So we manually inspect each tweets and decide whether it is a true mean or fan tweet, in order to give the best estimate from a human perspective. We delete tweets that are not expressing dislikes in the anti-fans' set and tweets that are not expressing likes in the fans' set. The remaining tweets are in the file "mean.txt" for anti-fans and "love.txt" for fans. Then we get the unique screen usernames and identify them as anti-fans and fans.

2.1.2 Raw Data

We search for most recent 200 tweets for each individual in the anti-fan and fan set. The search was split to several search requests since the API rate limit is reached by a single search. The search is done on October 8, 2016. Some accounts are private and no search result was returned for those users. After deleting those users, we ended up with 800 users, with 290 anti-fans and 510 fans. We collected 25,915 tweets from those 290 mean users, as well as 41,244 tweets from those 510 fan users. The usernames of anti-fans and fans are listed in the files "usernames_mean" and "usernames_fan". Their tweets are in the files "status_mean_all" and "status_fan_all", with the first line the screen username followed by returned tweets of that user, each tweet a separate line.

2.1.3 Data Cleaning

Tweets contain URL link, hex encoded emoji and other characters that we do not want to include in our analysis. In order to do text mining, we write MyClean() function to get a clean text of all tweets. Cleans text of tweets of anti-fans and fans are in the file "status_mean_clean" and "status_fan_clean". In each file, the first line is the screen username followed by returned tweets of that user, each tweet a separate line. To read in data in R, use readLines() function.

2.2 Exploratory Data Analysis

2.2.1 Count distribution for each person

There are on average 89 tweets and 963 words for mean users and 81 tweets and 858 words for fan users. Anti-fans tend to have more tweets than fans. Number of tweets and words returned for each user is summarized in table 1. The distribution of number of tweets for anti-fans and fans is plotted in Appendix A figure 4.

Statistic	N	Mean	St. Dev.	Min	Max
number of tweets of anti-fans	290	89.417	65.569	1	200
number of tweets of fans	510	80.878	62.038	1	200
number of words of anti-fans	290	962.886	875.256	2	3,988
number of words of fans	510	858.373	834.118	2	4,480

Table 1: Number of tweets and words for anti-fans and fans

2.2.2 Wordcloud

Collapse all tweets of anti-fans as a single text and all tweets of fans as a single text. Use package "tm" [8] [9] to construct the term-document matrix to count the number of occurrence of each word in the two text. Select top 200 frequent words of anti-fans and fans separately. To see difference of word usage of mean and fan users better, we manually delete meaningless words and words of high frequency in both sets. We list the top 30 selected most frequent words of anti-fans and fans in Appendix A table 3.

Use package "wordcloud" [10] to plot the wordcloud of the selected frequent words of anti-fans and fans. See figure 2 and figure 3.

2.2.3 Sentimental Analysis

Use package "sentiment" [14] to do sentimental analysis, and use package "ggplot2" [17] to plot our result. The classify_polarity() function in the package use a naive Bayes approach to classify each text as positive, neutral or negative, returning the posterior probabilities of belonging to each class. The text is classified as the class with the largest posterior probability. We perform 3 levels of sentiment analysis and all suggest that mean users overall tend to be more negative while less positive than fan users. See results in Appendix A.

2.3 Feature Selection

First we randomly choose 80% of the data as training set and 20% as testing set. We select features and fit the model on the training set and evaluate the model performance on the test set.

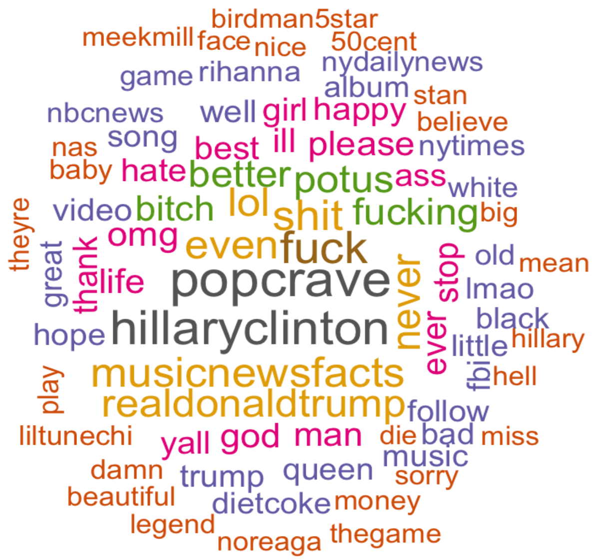

Figure 2: Wordcloud for mean users

Figure 3: Wordcloud for fan users

On the training set, choose top 200 words with the largest frequency difference in mean users' and fan users' tweets. To do this, we first compute the frequency of each appearing word in mean users' and fan users' tweets separately. Collapse all tweets for mean users as a single text and all tweets for fan users as another single text. If word A appears in one text but not the other, set the frequency of A in the other text as 0. Rank all words by descending order of the frequency difference. Choose top 200 words as our feature words. Note that is step is done only using the training data.

For each person, construct a 200 dimensional vector, corresponding to the frequencies of the 200 feature words in this person's tweets. Then use the training set to fit the model and evaluate the model performance on the test set. Use package "dplyr" [18] to manipulate data frames.

2.4 Statistical Modeling

For each person, we have $Y \in \{0,1\}$ is the response variable, with 1 indicate mean user and 0 indicate fun user. Also we have $X = (X_1, ..., X_p)^T, p = 200$ is the 200 dimensional covariate vector. In our training set, we have $N = 640$ individuals, and in the test set we have $N = 160$ individuals. We want to predict Y using X. Denote realization of variable Y of ith individual as $y_i \in \{0,1\}$, realization of variable X of ith individual as $x_i \in R^p = (x_{i1}, ...x_{ip})^T$.

2.4.1 Logistic Regression with L_1 Penalty

A logistic regression model is

$$Y \sim Bin(1,p)$$
$$logit(p) = \beta^T X$$

The β is estimated by maximizing the log-likelihood

$$\ell(\beta) = \sum_{i=1}^{N} y_i logit(p_i) + log(1-p_i) = \sum_{i=1}^{N} y_i x_i^T \beta - \sum_{i=1}^{N} log(1 + e^{x_i^T \beta})$$

To preform dimension reduction, we add penalty to the objective function $\ell(\beta)$ to force some coefficients β to 0. We want to minimize the objective function

$$-\frac{\ell(\beta)}{N} + \lambda \times \text{penalty}$$

where

$$\text{penalty} = \sum_{j=1}^{p} \frac{(1-\alpha)}{2} ||\beta_j||_2^2 + \alpha||\beta_j||_2.$$

Implement and results for penalized logistic regression are described in Appendix B.1

2.4.2 Tree-based Methods

Tree-based methods, which are non-likelihood approaches, are wildly used in prediction and classification. Essentially a classification tree split the feature space into small "rectangles". At each internal node, we split the training data into 2 parts by a binary split of a predictor. At each terminal node, we predict the class of each observation as the class most observations belong to in that node. We evaluate the performance of the tree by either misclassification rate or Gini index. Since it is impossible to consider all splits of the feature space, a top-down,greedy approach, known as recursive binary splitting is performed in constructing the tree[13]. For each internal node, fix the previous split of the tree, consider all possible new split which gives the smallest misclassification rate or Gini index.

We use full classification tree, pruned classification tree, bagging and random forest. Implement and results for tree-based methods are described in Appendix B.2

3 Result

In summary we consider 5 models:

- Logistic regression with L_1 penalty
- Classification Tree
- Pruned Classification Tree
- Bagging
- Random Forest.

We summarize the training and test accuracy of the above methods in table 2. In summary, random forest performs the best, with test accuracy 0.831, which beats bagging in terms of both training and test accuracy. Classification tree and pruned classification tree are both tend to overfit the training data. Penalized logistic regression also tends to overfit the data.

Non-zero coefficients in penalized regression, internal nodes for classification trees and variable importance plots for bagging and random forest all provide information about how much each word (covariate) influence the model, see details in Appendix B.

We choose random forest as our final prediction model since it gives the smallest test error. It has a test accuracy 0.831, with 0.79 sensitivity and 0.85 specificity. From the variable importance plot we find that "love" and "snake" are influential for the model. Also bad words such as "fuck", "bitch","ugly","dick","black","pussy" are of great importance of the model. The frequency of mention Taylor Swift such as "taylorswift13", "taylorswift","taylor","swift","tay" are also important.

	training accuracy	test accuracy
glmnet	0.853	0.725
tree	0.909	0.712
tree_prune	0.883	0.725
bagging	0.822	0.806
random_forest	0.834	0.831

Table 2: Comparison of model performance

4 Discussion

4.1 Reproducibility

We manually tag a user as anti-fan or fan, since we believe there is no good hard standard for it and the best we could do is from human perspective. This step is not reproducible since different people may have different judgement about emotions. Also given the list of usernames of anti-fans and fans, we search the most recent tweets for each user with respect to the date the search was done. The returned tweets would be different if the search is done a different day.

If given a "Gold Standard" dataset, with anti-fans and fans and their tweets, the rest of analysis is reproducible. In algorithms involved random sampling, such as cross validation, bagging and random forest, we use set.seed() function to make the subsamples generated each time the same.

4.2 Evaluate model performance

In this project we use misclassification error rate, both as the objective function being minimized in fitting the model and as a measure of model performance. However, other measures may be desirable to serve these two purposes. This depends on the goal one tries to achieve. In the tree-based methods, Gini-index, as measure of purity of the node, is considered as an alternative to misclassification error rate. Sensitivity, specificity, precision and recall are alternatives to misclassification error rate in terms of selecting the best model. For example, if the goal is to identify as many anti-fans as possible but care less about correctly classify a fan, then the model should maximize sensitivity, instead of classification accuracy. Receiver operating characteristic (ROC) curve could serve as a tool to select model with a specified sensitivity and specificity.

References

[1] `https://en.wikipedia.org/wiki/Twitter`.

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

[3] `https://en.wikipedia.org/wiki/Internet_troll`.

[4] `http://www.elle.com/culture/celebrities/a37995/taylor-swift-anti-squad-members/`.

[5] `http://geoffjentry.hexdump.org/twitteR.pdf`.

[6] `https://dev.twitter.com/rest/public/rate-limits`.

[7] `https://github.com/mattkcole/Data_Science_Project`.

[8] Ingo Feinerer and Kurt Hornik. *tm: Text Mining Package*, 2015. R package version 0.6-2.

[9] Ingo Feinerer, Kurt Hornik, and David Meyer. Text mining infrastructure in r. *Journal of Statistical Software*, 25(5):1–54, March 2008.

[10] Ian Fellows. *wordcloud: Word Clouds*, 2014. R package version 2.5.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[12] Jeff Gentry. *twitteR: R Based Twitter Client*, 2015. R package version 1.1.9.

[13] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 6. Springer, 2013.

[14] Timothy P. Jurka. *sentiment: Tools for Sentiment Analysis*, 2012. R package version 0.1.

[15] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[16] Brian Ripley. *tree: Classification and Regression Trees*, 2016. R package version 1.0-37.

[17] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

[18] Hadley Wickham and Romain Francois. *dplyr: A Grammar of Data Manipulation*, 2016. R package version 0.5.0.

Appendices

A Exploratory Data Analysis

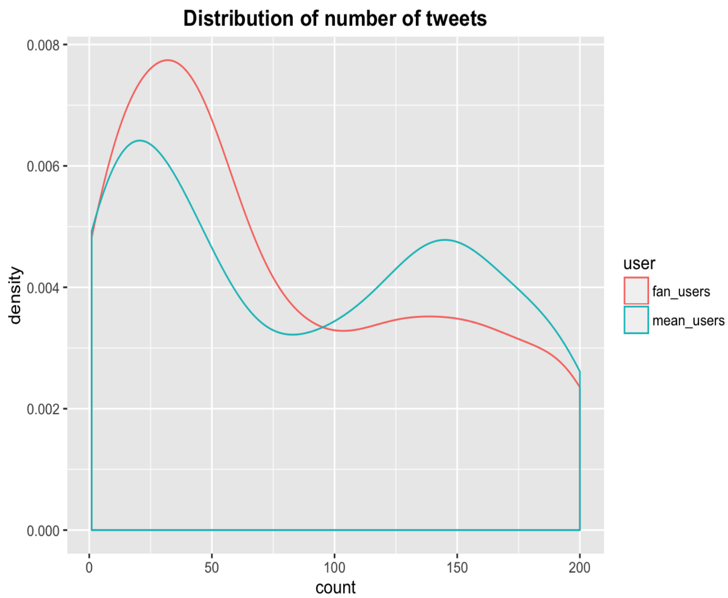

Figure 4: Distribution of number of tweets of each user

Sentimental analysis for each tweet Analyze sentiment of each tweet. Among all mean users' tweets, 22% are negative, and 67% positive. Among all fan users' tweets, 15% are negative and 76% are positive. Mean users overall tend to be more negative while less positive than fan users. The results are illustrated in figure 5.

Sentimental analysis for each person Analyze sentiment of each person. Integrate all tweets from each user as a single text and analyze the sentiment of this text. Among all mean users, 41% are negative and 18% are positive. While among fan users, 11% are negative and 47% are positive. The results are illustrated in figure 6. Again mean users overall tend to be more negative while less positive than fan users, and the difference is greater than tweet level sentimental analysis.

Sentimental percentage distribution for each person A user cannot be positive or negative all the time. To analyze how often a user being positive or negative, analyze sentiment of each tweets of that user, calculate how many percentage of positive and negative tweets among all tweets of that user. Plot the distribution of percentage positive and percentage negative in figure 7. Mean users overall tend to be more negative while less positive than fan users.

	words for anti-fans
1	popcrave
2	hillaryclinton
3	fuck
4	shit
5	musicnewsfacts
6	even
7	lol
8	realdonaldtrump
9	never
10	potus
11	better
12	fucking
13	bitch
14	omg
15	life
16	please
17	god
18	ill
19	man
20	ever
21	stop
22	best
23	girl
24	happy
25	hate
26	thank
27	yall
28	ass
29	well
30	video

	words for fans
1	happy
2	will
3	please
4	always
5	life
6	great
7	hope
8	back
9	best
10	thank
11	taylornation13
12	beautiful
13	music
14	never
15	follow
16	lol
17	miss
18	song
19	ever
20	birthday
21	thanks
22	right
23	night
24	girl
25	well
26	even
27	wait
28	taylorswift
29	ive
30	video

Table 3: Top 30 frequent words used by anti-fans and fans

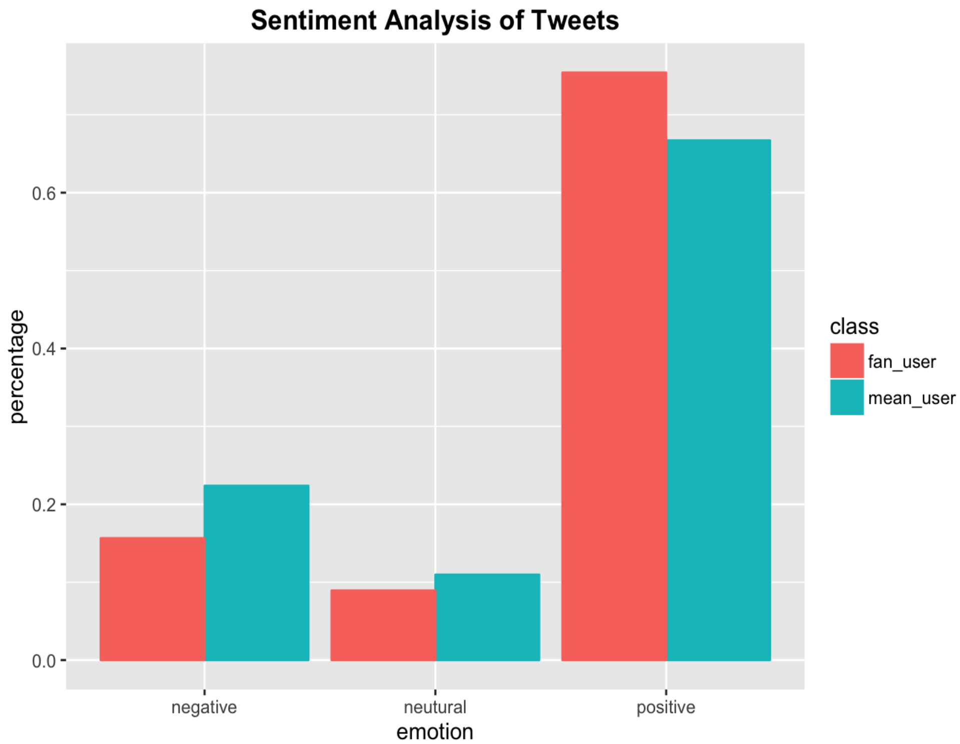

Figure 5: Sentimental analysis of tweet, with 25,915 tweets from 290 mean users, and 41,244 tweets from 510 fan users.

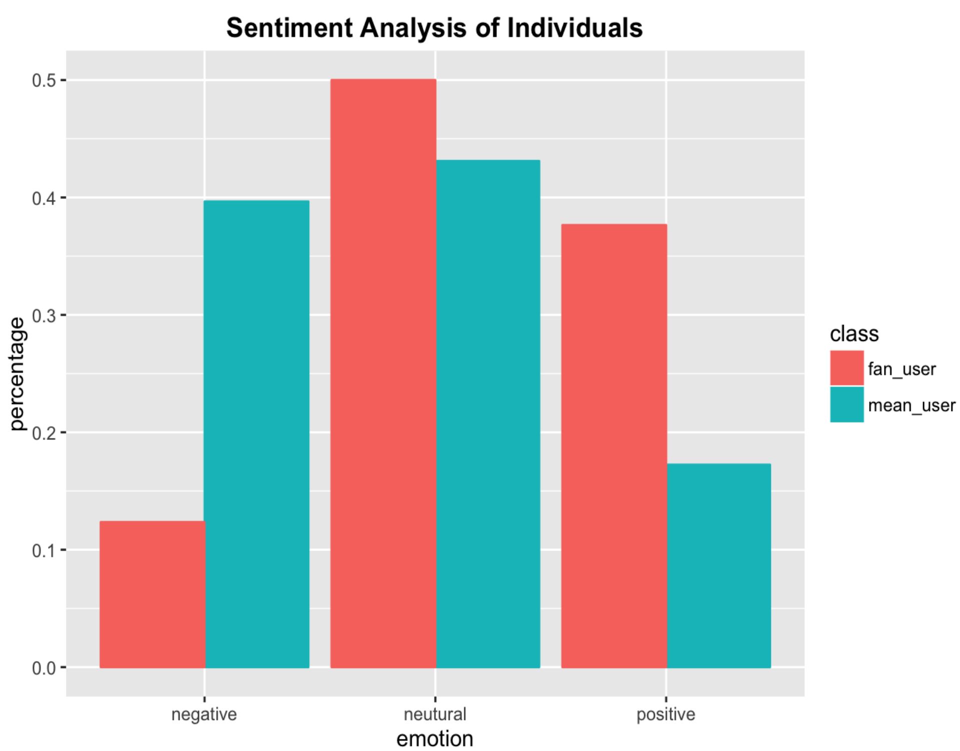

Figure 6: Sentimental analysis of Individuals, with 290 mean users and 510 fan users.

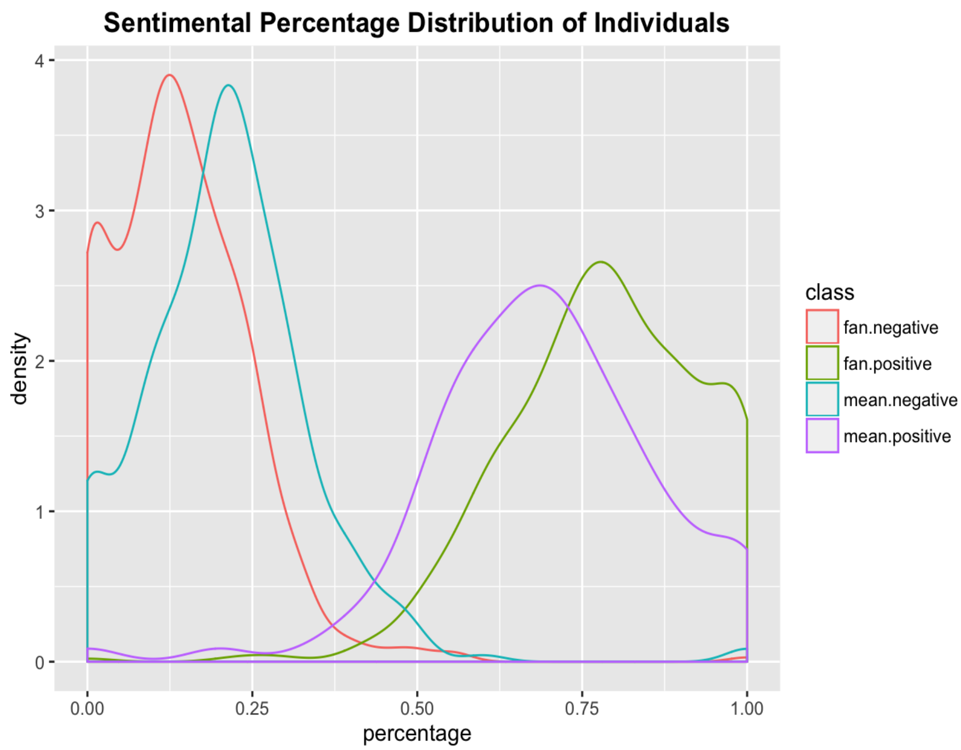

Figure 7: Sentimental percentage distribution of Individuals, with 290 mean users and 510 fan users.

B Statistical Implement and Results

B.1 Penalized Logistic Regression

Use the R package "glmnet" [11]. We set $\alpha = 1$ which is a L_1 penalty. To select best parameter λ, we use "cv.glmnet()" function to perform 10 fold cross validation on the training set. The optimal parameter is $\lambda = 0.02$. Words with non-zero coefficients are listed in Appendix table 4, rank by the absolute value of the coefficients in 4. The interpretation of the coefficient is that a unit change of the frequency of the corresponding word, the coefficient unit change of the logit probability of belonging to the mean user class. The positive coefficients indicate more frequently this word is used, more likely the user is an anti-fan. And the negative coefficients indicate more frequently this word is used, more likely the user is a fan. The larger the absolute value of the coefficient, the more influence of the corresponding word on the model.

B.2 Tree-based Methods

B.2.1 Classification Tree

Use R package "tree" [16]. We fit a classification tree with all 200 predictors on the training set. There are 25 variables used in building the tree with 28 terminal nodes. The misclassification rate for the training data is 0.09062.

The training error rate is very small, that is because classification tree tends to overfit the

(a) Words for mean users

	word	coef
1	black	147.188
2	free	143.895
3	sex	106.671
4	white	106.149
5	legend	105.455
6	shit	104.613
7	dick	101.984
8	ugly	98.706
9	bitch	90.163
10	jesus	87.066
11	snake	78.021
12	flop	72.958
13	beyonce	53.050
14	said	47.932
15	yall	35.644
16	know	24.973
17	ended	24.726
18	better	21.235
19	hillaryclinton	18.336
20	popcrave	8.551
21	lmao	8.293
22	fuck	5.541
23	pussy	4.701
24	take	4.260
25	bangyourankles	2.540
26	pay	1.341
27	fat	1.150
28	fun	0.961
29	like	0.760
30	queen	0.577
31	name	0.559
32	hot	0.471
33	hell	0.307
34	dont	0.208
35	one	0.059

(b) Words for fan users

	word	coef
1	proud	-107.394
2	friend	-54.694
3	ts6	-43.962
4	person	-42.679
5	love	-36.106
6	day	-32.580
7	smile	-32.048
8	lang	-31.866
9	haha	-24.621
10	awesome	-20.701
11	miss	-19.529
12	beautiful	-12.145
13	naman	-11.721
14	see	-9.968
15	tay	-9.962
16	amazing	-5.230
17	today	-3.773
18	yung	-3.257
19	live	-2.320
20	hope	-1.902

Table 4: Words selected by Penalized logistic Regression

data. We prune the tree by adding a penalty for the tree size. We use 10 fold cross validation to choose the best tree size. The best tree size is 14. The error rate on the training set is 0.1172.

The words used in construction the full and the pruned tree are listed in table 5. The full and the pruned tree are plotted in figure 8 and figure 9 with cutoffs of each node.

B.2.2 Bagging and Random Forest

Although we prune the tree to reduce over-fitting, the above tree based method still suffer from high variance, especially we have a larger number of predictors. So we further consider technics to reduce variance of our model. The most commonly used approach is bootstrap aggregation, or bagging. The idea is to take B bootstrap samples from the training set, fit a model on each bootstrap sample, then average over B samples to reduce variance. Denote the fitted model for the bth bootstrap sample as $\hat{f}^b(x)$, then the average bagging prediction is

$$\hat{f}(x) = \frac{1}{B} \sum_{i=1}^{B} \hat{f}^b(x)$$

In the classification case, we take the majority vote for the B predictions as our final prediction for the observation x. The tree constructed by each bootstrap sample is not pruned.

Random forest further improves the performance of bagging by decorrelating the trees. Instead of consider all possible splits for all predictors at each internal node, random forest randomly select m predictors to split on. Then taking the majority vote for B bootstrap trees gives the final prediction.

Use R package "randomForest" [15]. For bagging, we take 500 bootstrap samples and fit 500 trees, with number of variables tried at each split is m = p = 200. For random forest, we take 500 bootstrap samples and fit 500 trees, with number of variables tried at each split is $m = \sqrt{p} = 14$.

Averaging over multiple trees would be difficult to interpret the classification rules. Here we use varImpPlot() function to plot the importance predictors. The importance is evaluated essentially by the total amount that the misclassification rate or Gini index decreased due to splits over a given predictor, averaged over all B trees [13]. Variable importance plots are shown in figure 10 and figure 11.

(a) Words used in full classification tree

	Words
1	love
2	fuck
3	just
4	great
5	music
6	bitch
7	now
8	days
9	via
10	can
11	cute
12	take
13	que
14	follow
15	come
16	fan
17	taylorswift13
18	black
19	world
20	shawnmendes
21	name
22	snake
23	new
24	heart
25	like

(b) Words used in pruned classification tree

	words
1	love
2	fuck
3	bitch
4	take
5	que
6	follow
7	taylorswift13
8	black
9	world
10	days
11	shawnmendes
12	snake

Table 5: Words used in full and pruned trees

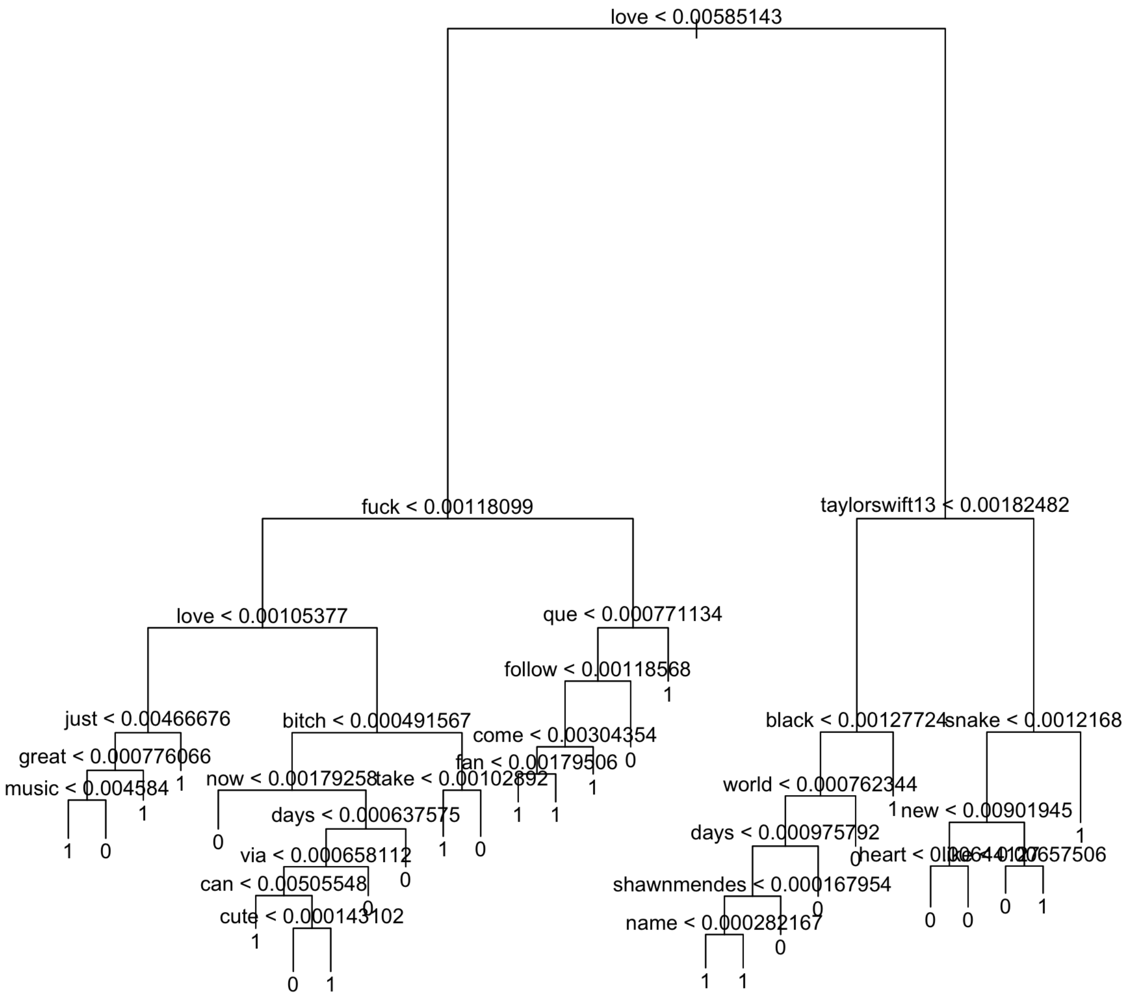

Figure 8: Full classification Tree

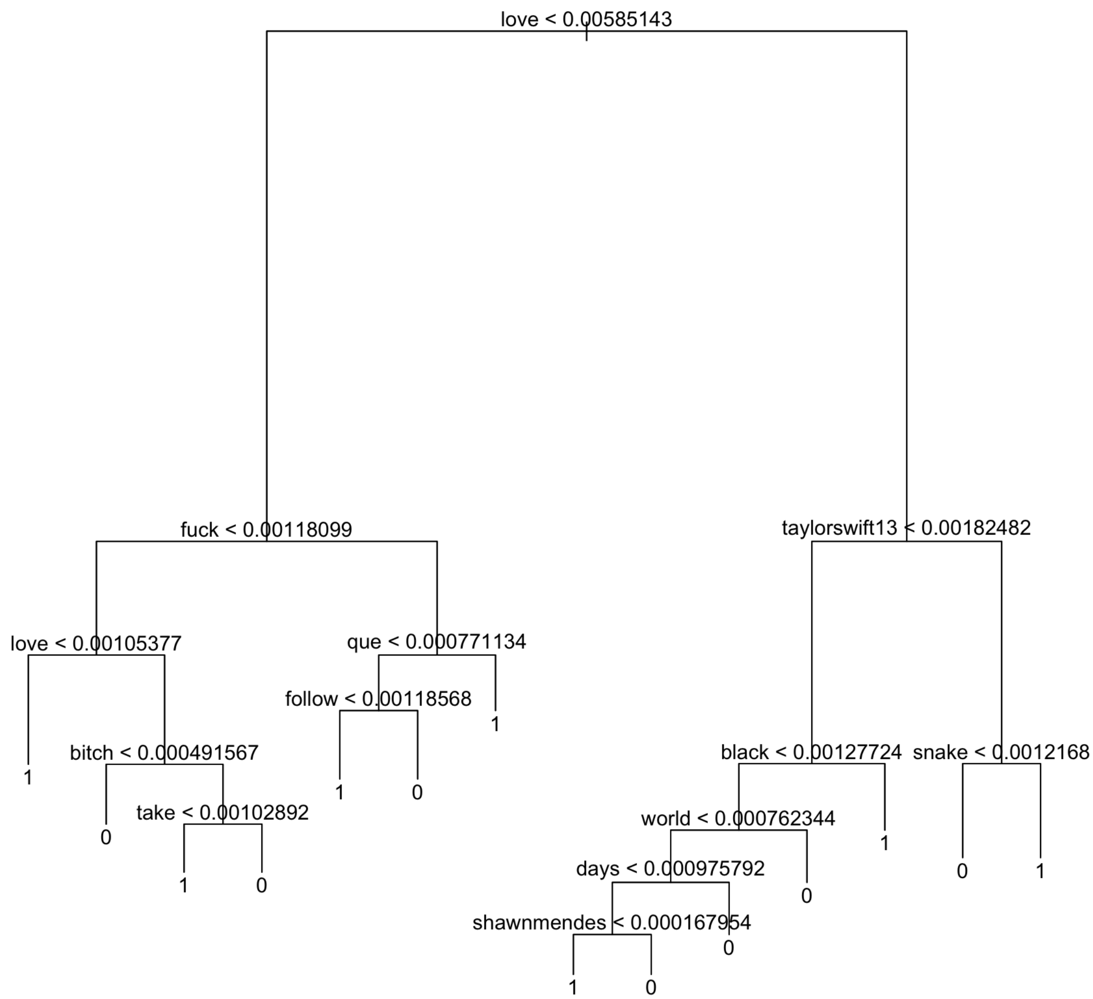

Figure 9: Pruned classification Tree

Variable importance plot for Bagging

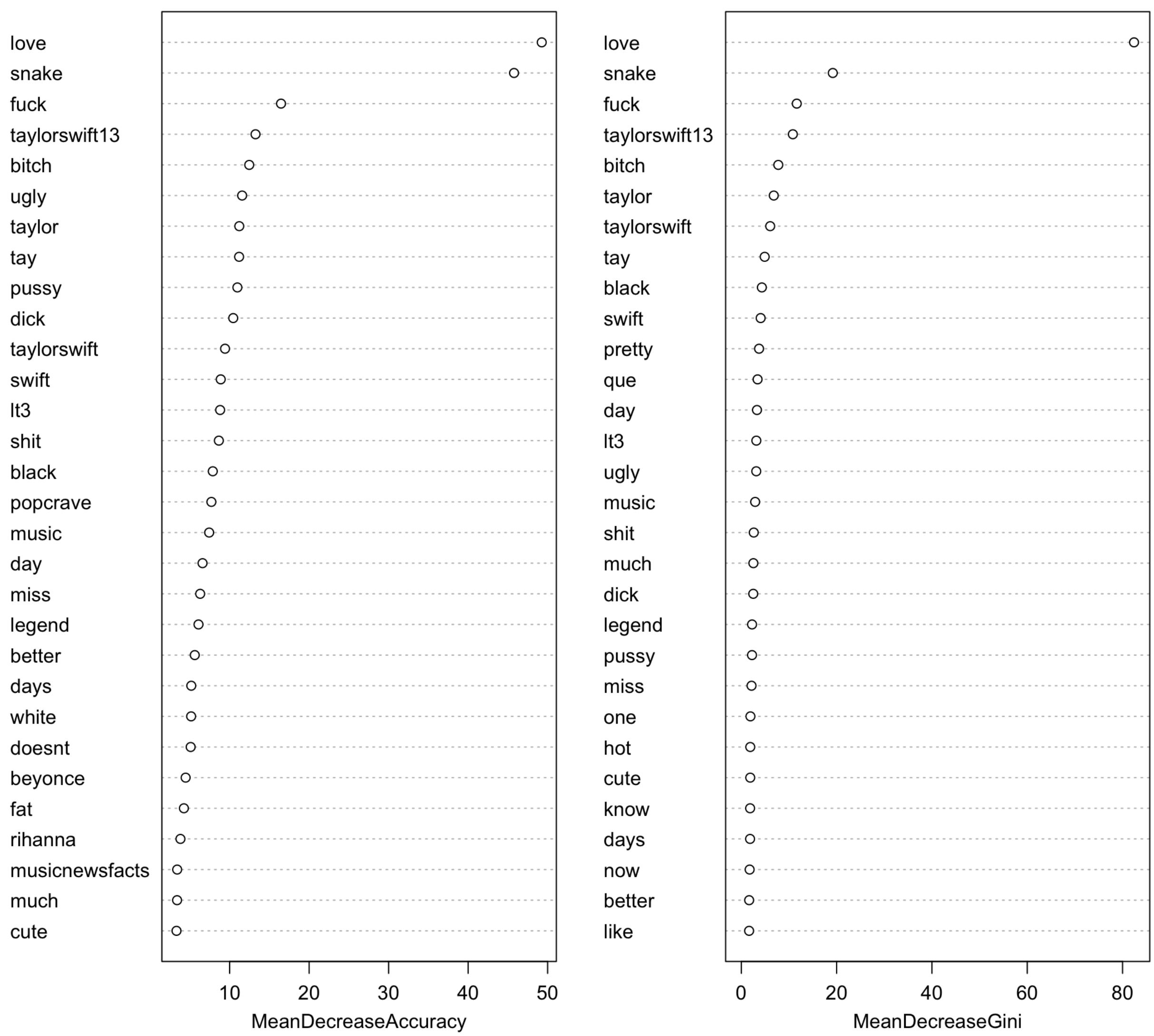

Figure 10: Variable Importance Plot for Bagging

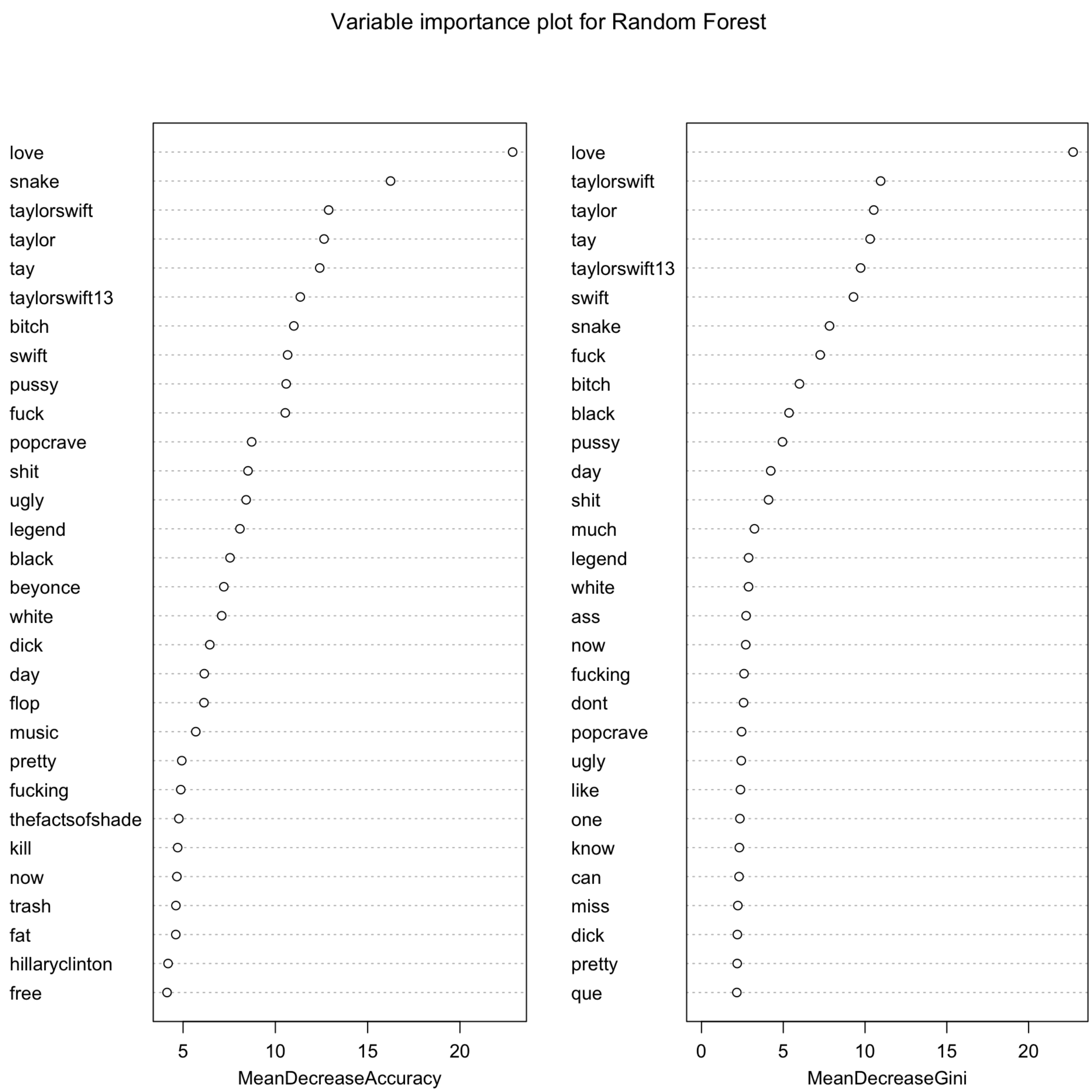

Figure 11: Variable Importance Plot for Random Forest

List of Figures

List of Tables