IDS 575 Project - Predicting Twitter Follower Counts

Scott Brewer
Ono Gantsog

On the Twitter social media platform, it is important to have many followers. Our hypothesis is that user initiated activities such as posting tweets, liking other users' tweets and retweeting are the main factors in driving followers. We also hypothesize that passive activities such as getting mentions, getting likes, getting retweets and other measures of popularity on the platform and are predictors for number of followers. We initially intended to pursue both hypotheses, but as we'll discuss in Data Gathering we shifted focus to first hypothesis only due to issues gathering passive data.

Data Gathering

Data is collected using R package 'twitteR' to access twitter API. We picked one data science focused account that Ono is familiar with, "BigDataGal", and used the getUser() function to get the basic account information with 17 fields. Next, we used the getFriends() and getFollowers() functions to retrieve the same information for all of the reference account's friends and followers (57k and 94k respectively). During this process, we first encountered a rate limit error from the twitter API, which prevented pulling the full 94k followers and instead stopped the pull at 64k. That said, the combination of the friends and followers with duplicate accounts removed left about 81k accounts, which we decided would be enough to use for analysis.

Next, we tried gathering passive activity information but were not successful. Out approach was to pull the 50 most recent tweets from each user with the userTimeline() function and to then sum up the number of retweets and favorites on those 50 tweets. Unfortunately, the Twitter API has automatic rate limits for activity within a 15 minute window, which limited our accesses to about 1000-2000 accounts (it varied based on amount of other API activity) within that window. While we could have spaced out our accesses to get this data, we decided that it wouldn't be an effective use of time. Additionally, we hoped to use this analysis and resulting models for bulk bot detection, so decided to focus on data that could be more easily accessed in bulk.

It's worth mentioning that these any data project that involved web scraping or api based data collection is likely to encounter similar rate limit issues, so it's important to factor into the project planning phase. Overall Lessons Learned from data collection phase:

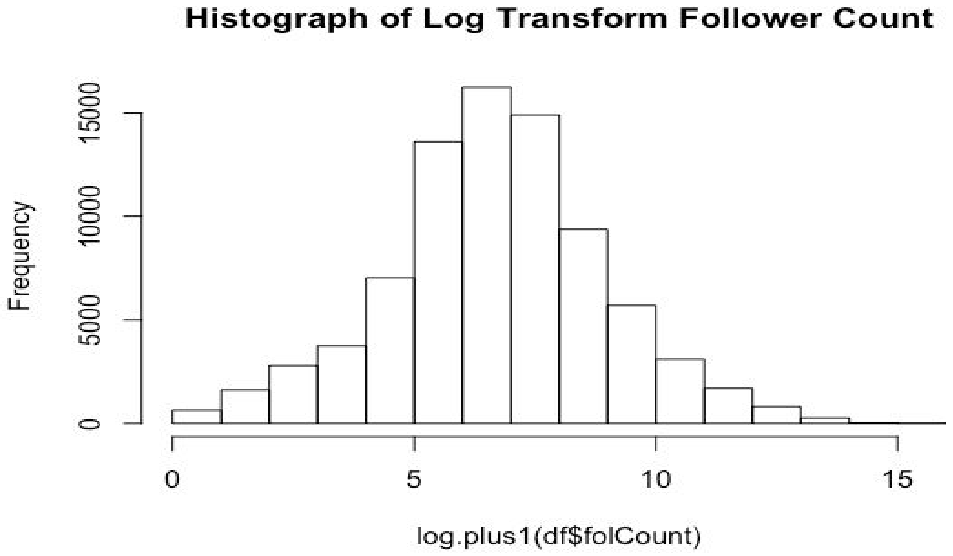
- Must plan for limits since most of online data collection will involve access limit
- Must plan time to pull the data
- Must plan time to clean the data

Initial Exploration

The twitteR getUser() function pulls 17 features by default:

Id	id of the twitter account
Screen name	name of the account
Friend count	number of accounts that the account is following
Followers count	number of other accounts that are following the account
Status count	number of the tweets posted by the account
Favorite count	number of likes/favorites made by the account
Created date	number of seconds since 1/1/1970
Verified	if the account is verified or not
Protected	if the account is protected or not
Description	user banner description
URL	user supplied url
Photo	link to user photo
List count	number of lists that include the user
Language	language of the account
Last status	most recent tweet from user
Location	location setting of account
Follow Req	boolean for if follower request has been sent

From those features, we used the numeric and boolean fields: friendCount, followerCount, statusCount, favCount, createdDate, verifiedBool, and protectedBool. We then converted created date is changed to days since 1/1/1970. Our target variable is Followers count. Numeric variables except created date are highly skewed. As example following graph displays a histogram of log transform of followers count.



Since the target variable is numeric, we trained regression models that we learned in the course. Due to the high skew of the data, we performed log(x+1) transformation to normalize the data to better satisfy the assumptions for linear regression. We also used the standard 70/30 split for train/test data. For error comparisons, we used RMSE method:

Models	**RMSE**	**Log+1 transform RMSE**
Ridge	3244.208	0.5916876
Linear regression	3110.264	0.5788409
LASSO	3047.069	0.5710789
Boosting	3312.446	0.5384884
Bagging	3064.723	0.5233222
RandomForest	3144.516	0.5147747

As we observe in the table above, some of the linear regression based models performed well in untransformed space, but once the data was transformed to match linear regression requirements, non-linear tree based and additive models performed better. Due to this observation, we proceeded by using RandomForest regression. As RandomForest regression doesn't require the data to be normalized, we continued our analysis with non-transformed data.

Further investigation of the models showed that due to the wide range of the data (zero to millions) the predictions were decent for data around the center but were poor at the extremes of the data. In the next section we discuss our strategy to address this issue to improve our model.

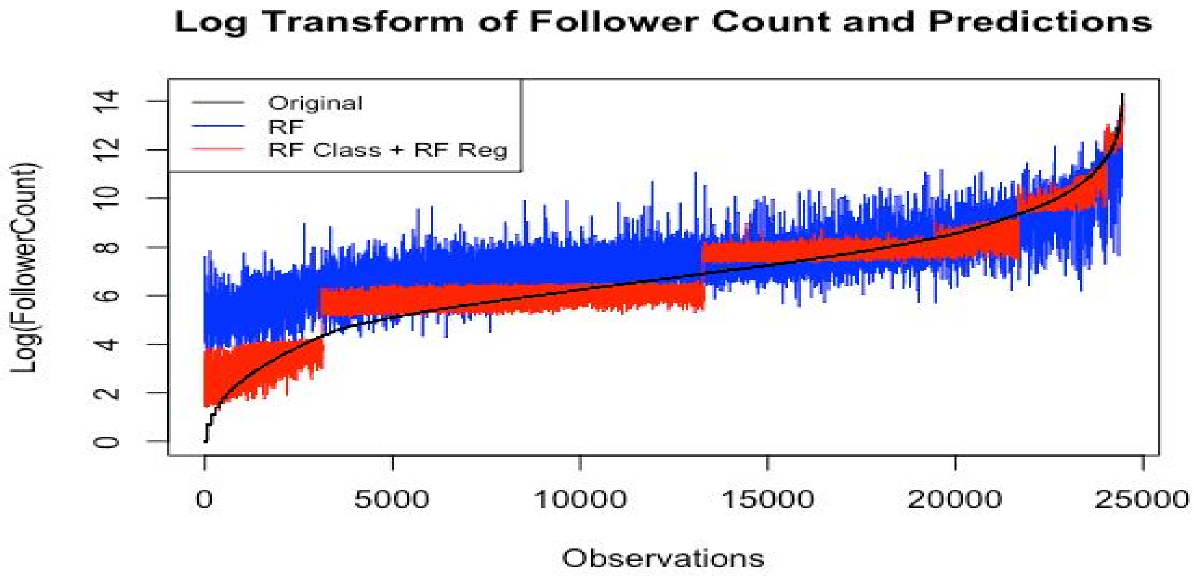
Combined Classification and Regression

As we weren't satisfied with the accuracy we were achieving, we tried combining models to improve our predictions. Our strategy was to divide the data into classes and then train regression models on each of the classes. First, we divided the train data into different classes based on follower count. Next we developed RandomForest regression models (as they had superior performance in our initial investigation) on training data for each class. We then evaluated the chained models by predicting classes for test data and then predicted numeric values with the using the previously trained regression models to predict numeric values within each class.

First Approach: Subjective Split Labels

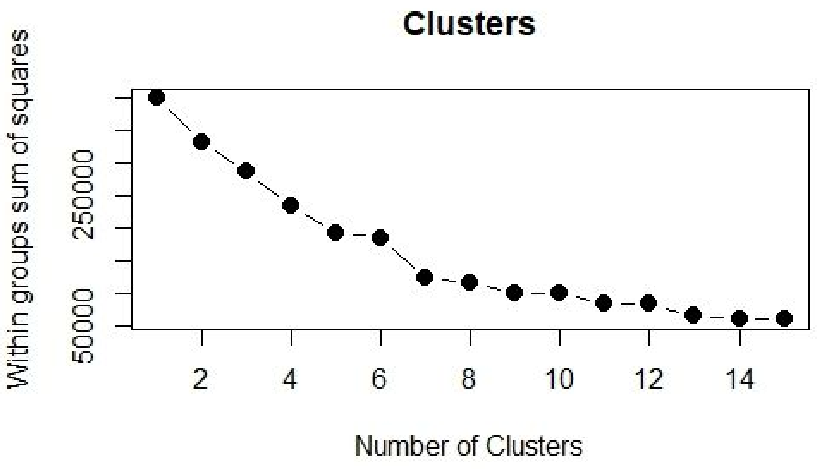
Split labels were defined by splitting follower count into the following bins: 0-100, 100-1000, 1000-10k, 10k-100k, and >100k followers. RandomForest classification model was trained with these labels as the target. Random Forest Regression models were trained within each class with folCount as the target. The following graph shows the performance of this method vs

RandomForest regression over entire data set:

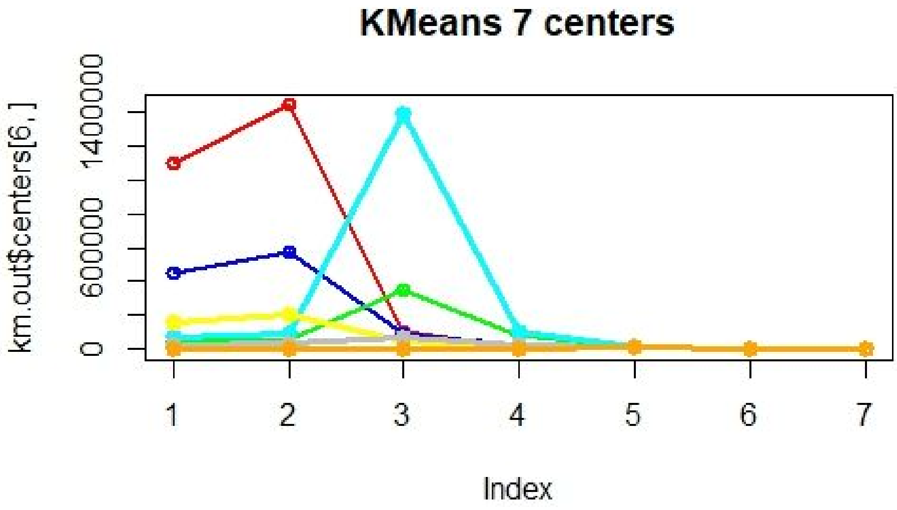


Second Approach: K-Means Clustered Split Labels

For the clustering, we used Kmeans clustering. At first we determined cluster numbers by dividing the data upto 15 clusters and compared their within cluster distance as displayed in following graph. At 7 clusters, within clusters distance is dropping significantly.

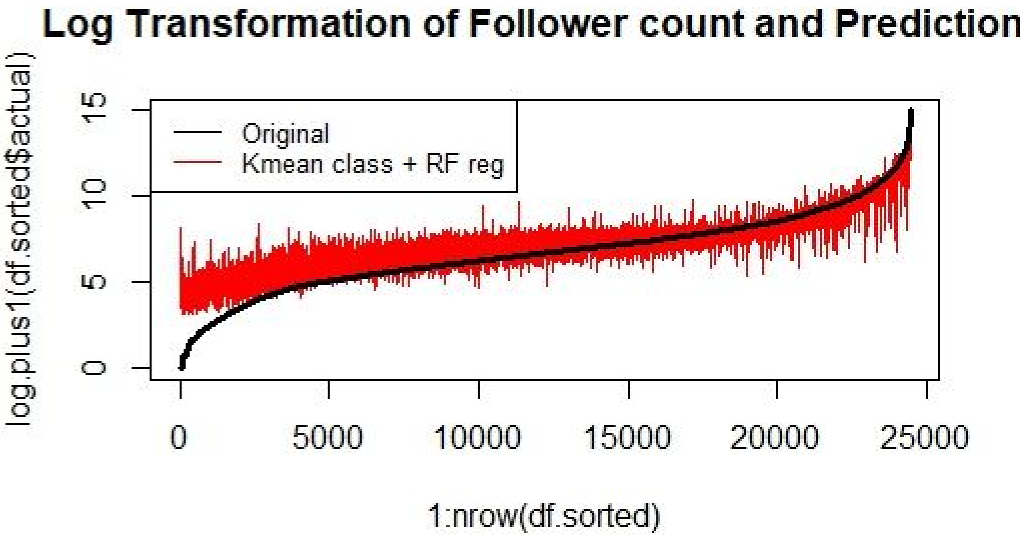


After dividing data into seven clusters, we displayed each cluster centers on the following graph. The X axis shows variable index in the data (1-friend count, 2-follower count, 3-status count, 4-fav count, 5-created date, 6-verified, 7-protected) and the Y axis shows values for the center on respective features.



After dividing into clusters, we combined cluster numbers into the dataset as factor variable. Then we trained RandomForest classification model having Cluster as target variable and then we trained RandomForest regression model on each cluster data.

After training classification model and regression models, we tested our models on the test data. From test data, we removed folCount variable and got cluster numbers as classes. Then for each cluster class, we ran regression model. In the following graph, we displayed log transformation of actual follower count and log transformation of predicted follower count after sorting the data frame according to actual follower counts.



Overall the two approaches reduced RMSE from 3144 using RandomForest over entire data range to 2076 using Subjective Splits and 2912 from K-Means Splits.

Further Improvement

Better Sampling

When we were collecting data from Twitter API, account information of friends and followers of BigDataGal account was used, which we thought would be a good representation of different accounts, but after conducting this analysis we suspect that that many of the accounts has significant algorithmic aspects (ie while maybe not purely bots, the accounts may have automated following and likes). As an alternative, we could try using curated lists of accounts or try to pull random accounts with the twitter API.

Right Predictors

Certainly, only user initiated activity information such as tweets, likes and friend numbers are not the good predictors for follower count. Popularity of an account (like a celebrity in real life) should be a good predictor along with the number of mentions coming from other accounts. Also, other predictors such as hashtags in the tweets and number of retweets could add more predictive power to the model.

Eliminating Wrong Data

While working on the data we realised that some accounts could be bot accounts, which would certainly affect our learning models. As displayed in the kmeans center graph, there is one cluster that has accounts with many tweets but has average number of friends and followers, those accounts could be bots. While this wasn’t exactly an issue for this project, we had hoped to extend the project into a bot detection algorithm, but that would require labeled data or some proxy for label.