Investigation of Pizza Requests

Cody Stancil, Lulu Ge, Seth Green, Yizhe Ge

July 29, 2016

Parsing Data into R Data.Frame

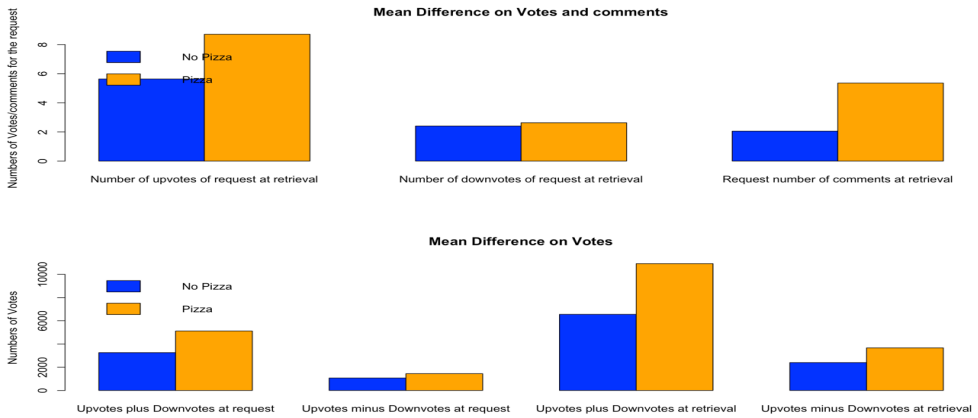
We initially got the data into R by using the `read_lines` function in the {readr} package. Once we had the data in R as a character vector, we began looping through the lines to extract the column names. Throughout this code, we make use of Regular Expressions, fed into various base R functions including `grep`, `grepl`, `gsub`, and `sub`. In particular, the function `gsub('\\"', "", <string>)` is used repeatedly to strip all of the `\"` out of a string, and `grepl('^\\"', <line>)` to check whether a line starts with a quotation mark.

The body of the loop trims whitespace off each line and then splits it at `:`. Next it checks two conditions to filter out %%%% delimiters and subreddit lines. If both of those conditions are met, the first element of the split line is stripped of its `\"` and then fed into a character vector of variable names. Once the loop finishes, an empty data.frame is created with those variable names.

Next, we loop back through the entire raw input file, in order to populate the observations. To summarize, a count is initialized to keep track of which observation we are recording. Then, each line is split on the first colon (because some of the character fields contain a colon). The same conditions as the first loop are checked and then the content is placed in the matching variable and observation. When a %%%% is encountered, the count is incremented by 0.5 (because two lines of %%%% indicate the next observation).

Warm-up Analysis

We investigated seven numeric variables recording feedback of request from internet users. The data shows that people who grant the pizza in the end always have higher numbers of votes and comments on their requests, whatever it is count of upvotes only, or count of absolute upvotes (number of upvotes - number of downvotes), or sum of all votes. So there is a better chance to get pizza if the request is more popular than others.



Mean Difference Analysis

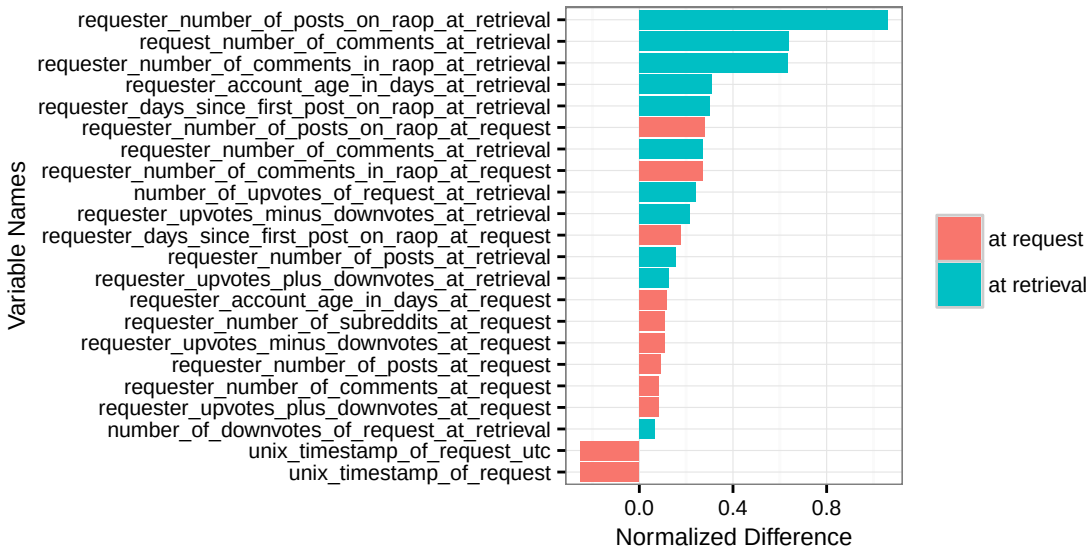
To better understand which variables led to a successful pizza request, we took a normalized difference of means between all of the variables for users who received a pizza and users who did not. Normalization was helpful in making comparisons between the impacts of each variable since they were put on the same scale with mean zero. To normalize the data, we used the following formula:

$$\text{Given variable X: } (x_i - \bar{x})/s$$

After the observations were normalized, we took the mean of each variable for successful and unsuccessful requests. The differences between the means of the respective variables in the two categories were then taken. The following figure displays these normalized differences.

The data can be categorized by when the data trail stopped i.e. time of request or time of retrieval. We noticed the normalized differences of the retrieval data is always greater than its respective variable that stopped at the point of request. This makes sense because the retrieval data includes all of the request data plus all of the information past the request up to the retrieval date. Moreover, one might expect the most predictive power to be the users activity directly after the pizza request was made; this further explains why retrieval data is always higher than the request data.

From the analysis, we were able to better understand what led to a successful pizza request. The top three ranked variables included: requester number of posts on "raop" at retrieval, requester number of comments at retrieval, and requester number of comments in "raop" at retrieval. On average, the mean requester number of posts on "raop" at retrieval for successful pizza requesters was more than 1 standard deviation higher than the number of comments in "roap" made by unsuccessful requesters. In addition, two of the top three ranked variables were interactions in "roap"; this suggests that to be more successful one should spend more time interacting with "roap". Lastly, from the following Figure, the time variables seem to be predictors of the outcome of a pizza request. These variables display that requesters were more likely to be successful if they made the request closer to when "roap" began. This suggests the "roap" thread was more of a fad that was more successful at first and has died out over time.



Looking at Word Frequencies

We decided to see if certain words appeared frequently in successful pizza requests that did not appear in unsuccessful pizza requests. We subset out the `request_title` and `request_text` variables and then divided them into TRUE and FALSE sets based on whether or not they received pizza.

To process the text, we used the {stylo} library to get an ordered table of word frequencies for each subset. After a bit of exploratory analysis, we settled on some thresholds that made sense to compare: for the title, we compared the Top 100 most frequent words and for the request text we looked at the Top 200 most frequent words. Finally we filtered down to the list of words that appeared in the Top 100 for titles of successful requests, but did not appear in the Top 100 for unsuccessful requests. We did the same for text, but with the Top 200.

This analysis wasn't terribly illuminating, but we did notice some interesting words. For instance, "ramen" and "unemployed" were featured in a lot of successful titles, but were much less frequent in unsuccessful titles. For the request text, similarly notable words were "paycheck" and "bills." Apparently sympathy is a strong motivator. The full tables are below:

Unique Words in Successful Text

	words in text	Freq
1	imgur	158
2	paycheck	129
3	ago	126
4	bills	125
5	couple	120
6	nice	118
7	recently	115
8	come	113
9	soon	113
10	another	112
11	again	111

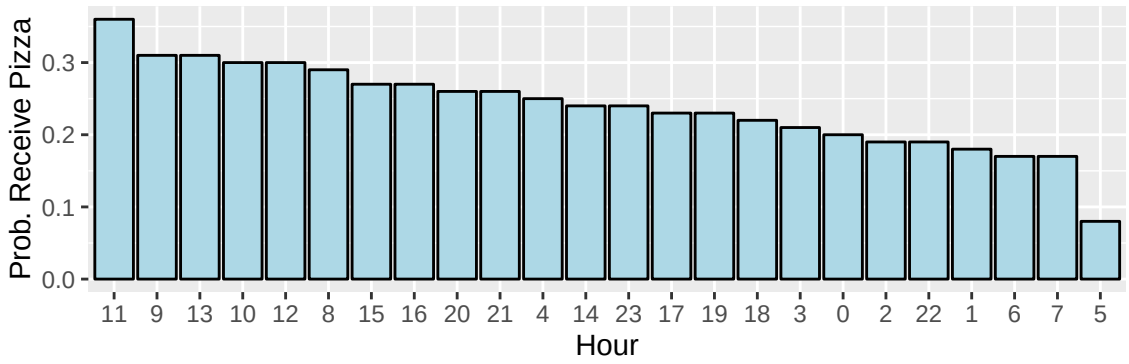
Unique Words in Successful Title

	words in title	Freq
1	ramen	35
2	tx	34
3	next	33
4	little	31
5	first	30
6	unemployed	30
7	forward	29
8	if	28
9	couple	27
10	having	27
11	here	27

When Should You Request For a Pizza? (Time Analysis)

We first tried to do a time series analysis, but we could not find anything interesting. So we decided to see if there is anything interesting by just analyzing the hours. We did this by converting the variable `unix_timestamp_of_request` to `POSIXct` format. Then, we used the `lubridate` library to process the time and get the hours. The `requester_received_pizza` was then grouped based on hours. We calculated the probability of receiving a pizza by dividing the number of pizza received by the total number of requests for each group.

We found out that 8am-1pm requests have a higher probability of being accepted compared to other request times. The following plot shows the visualization of our results:



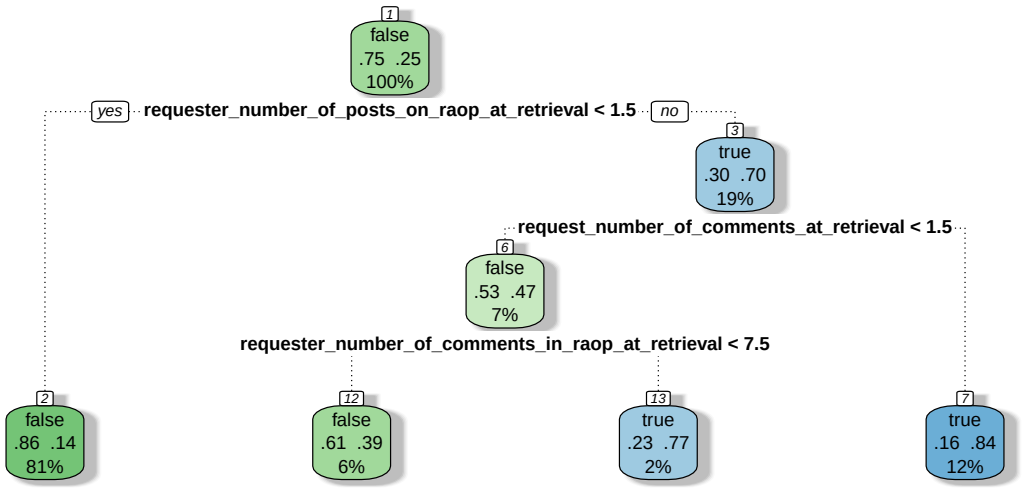
Predicting using Decision Trees

We tried to predict whether the requesters actually received pizza by using a `Decision Tree Model`. Specifically, we combined the use of `rpart` and `caret` packages to do the regression. We first filtered out the variables that could not be easily used to do the regression. For example, `request_title` and `request_text`. In order to validate our prediction after we built the model, we separated the whole dataset into two parts - training dataset and testing dataset. We then transformed the predictor variables into appropriate formats, and applied the `rpart` function to them. The `rattle` package is used at the end for visualizing our predictive model.

We got a prediction result that had a 97.43% sensitivity but only a 41.05% specificity. This means that our prediction model can successfully predict the false case 97.43% of the time, but we can only correctly predict the true case 41.05% of the time. Because most of the receivers did not get the pizza at the end, we got an accuracy of 83.54%.

	false	true
false	1249	247
true	33	172

The following plot visualizes our `Decision Tree Model`:



Appendix

1. How many requests resulted in granting pizza, and how many did not?
   Answer:
   Granted: 1397
   Not Granted: 4274

2. What is the average number of subReddits subscribed to by those who did not get a pizza?
   Answer: 17.37833

3. What is the average number of down votes at retrieval for those requests that were successful?
   Answer: 2.631353