MSAN 630: Advanced Machine Learning

# Final Project: Yelp Recommendation Engine

Jeff Baker, Kailey Hoo, Griffin Okamoto

March 15, 2015

Given the prevalence of recommendation engines across so many consumer-facing products (e.g. shopping/commerce sites, or personalized content like Netflix or Pandora), we chose to develop our own recommendation system for our final project. We leveraged readily-available Yelp review data from the *RecSys2013* Kaggle competition. The goal of our system is to predict a user's rating for a particular business that they have not reviewed before, given some information about that user. We leveraged a number of machine learning techniques to accomplish this, including (but not limited to) univariate/bivariate exploratory data analysis, feature engineering, model evaluation, and hyperparameter tuning.

After establishing initial baseline performance among nine machine learning algorithms (without any tuning), we proceeded to fully investigate the best set of features with the top three performing models: linear regression, random forest, and boosting with decision stumps. A substantial part of our time was spent on feature engineering, both to reduce dimensionality and create new predictors. We implemented collaborative filtering, K-means clustering of users, text analytics, and principal component analysis. With feature engineering completed, we tested each combination of features across our three algorithms, measuring success by minimizing root mean-squared error (RMSE). Ultimately, our tuned random forest (with features from category PCA and user clustering) prevailed as our best performing algorithm for predicting user business ratings for this data set, with a RMSE of 0.83 on training set cross-validation and 1.13 on the test set. With more time, we would further our project by implementing our algorithm into an actual recommendation tool for users. Additionally, we would augment our data to decrease sparsity in the features as a way to increase performance.

## Overview & Motivation

Today, recommendation systems are a prevalent feature among many popular online consumer products. Companies such as Amazon, Netflix, and Pandora prominently feature recommended, personalized content for their customers–enhancing their product experience in a significant way. This was the motivation for our chosen machine learning project: each of us were very interested about how these product features worked, and opted to create a product feature that would predict a user's rating of a business, based on certain characteristics that we know about the business and the user.

Our initial plan was to retrieve business and review data for locations within metro San Francisco via Yelp API. However, given the performance level of the API (it's quite slow) and the time constraints on our project (three weeks from assignment to deadline), we chose to leverage existing Yelp data instead. Thus, we turned to an old Kaggle competition, *RecSys2013*, and began our machine learning development process with that data.

## Data Description

In our chosen data set, both a training and a preset test data set were provided. The training data included four files: businesses, check-ins, users, and reviews (see Appendix A for data examples, and Appendix B for further details about each of the data files). After seeing a forum on the competition Kaggle page [1] that detailed how flawed the provided test data was, and given our time constraints, we decided to exclude the test data set from our project completely. As described in more detail below, we instead split the training data 80/20 to hold out for our testing purposes. Additionally, to prevent data leakage, we imputed appropriate values for some columns in our newly formed training data, rather than simply removing those observations.

## Data Preprocessing

Before we could start modeling, we needed to merge all four training set files, subset that into a new training and test set, and remove any records of the test data that may have been in the training data. Each of the files contained JSON objects with their respective attributes. After merging on the user_id and the business_id, we randomly sampled 80% of the data, holding out the remaining 20% to be used as our test data.

### Correcting for Test Data Leakage

After an initial look at the columns in the training set, we suspected some information about the reviews were also in the test data set. For example, if a user had five reviews in the training set and one review in the test set, the user's review count in the training data would say six reviews, and the average stars would be an average of all six reviews. In order to avoid this type of data leakage, we summed the average stars, votes, and review count by user_id, then subtracted those values from any user_id that was in both the training and test sets. After sanitizing our data sets, we began to familiarize ourselves quantitatively with the data.

[1] https://www.kaggle.com/c/yelp-recsys-2013/forums/t/5473/the-dataset-is-critically-flawed

## Exploratory Data Analysis

Our full Exploratory Data Analysis can be found in Appendix B. The primary variable of interest that we will discuss within the report is the target variable: review star rating. The distribution of star ratings is left-skewed; most ratings are either 4 or 5. This contradicts a common conception that users only write reviews when they are either happy or unhappy with a business. The mean rating is 3.77, and the median is 4.


***Figure 1:*** *Target Variable: Review Star Rating*

Other notable findings from our EDA pertain to the sparsity of the data. Although we have about 183K reviews, we also have 41K users and 11K businesses. The distributions of number of reviews per user and reviews per business are extremely right-skewed, and the medians are only 7 and 5 reviews, respectively. This indicates the data is extremely sparse, and finding relationships between users or business using reviews in the data would most likely be difficult.

## Baseline Results: Before Feature Engineering

There are 183,925 examples in our training data and, after creating dummy variables for all of our categorical features, we ended up with 574 features. With such a large matrix of data, we ran a variety of algorithms on our training set–before performing any data manipulation or feature engineering–as a way of narrowing down the best possible models to pursue. Below are the results of seven different models for 5-fold cross-validation. If there were hyper-parameters to tune, we included the best hyper-parameters and their results as well. To stay comparable with the Kaggle competition framework, we measured performance using root mean-squared error (RMSE) throughout our model development process.

| **Model** | **RMSE** | **(Best) Hyper-parameters** | **RMSE** |
|---|---|---|---|
| Linear | 0.9750 | N/A | N/A |
| Random Forest | 1.0265 | Trees: 100; Max Features: All | 0.9864 |
| Boosting (Adaboost) | 1.0407 | Loss: Linear | 1.0426 |
| Logistic | 1.2387 | Penalty; C (Too long) | N/A |
| Majority Classifier | 1.2387 | N/A | N/A |
| K-Nearest Neighbors | 1.3093 | Neighbors: 2000 | 1.2163 |
| Naive Bayes | 1.7141 | N/A | N/A |

In addition to these models, we attempted to implement Support Vector Machines (SVM) and Neural Networks. The SVM model took too long to run and, with the large number of features, the Neural Networks model would not allow us to have more than one hidden unit because there were too many weights. For these reasons, we chose to exclude them from further model experimentation. As seen above, our three best models, based on RMSE, before and after hyper-parameter tuning, were Linear Regression, Random Forest and Boosting. We continued with these three models through the rest of our model experimentation.

# Feature Engineering

## Missing Data and Data Leakage

Before testing our models with constructed features/reduced dimensions, we found that some of the users in the "reviews" data set (one of the four original data files) did not have any user information in the "user" data file (such as total review count, number of votes, average star rating, etc.). Previously, we replaced all missing values with the mean of the column/feature. We found an opportunity to potentially increase our accuracy by imputing these missing values with the user's information found in our constructed training set instead. For example, user A could have reviewed 10 businesses total; however, if she is not in the user set (where this feature is), we couldn't know this. However, seeing that she has 4 reviews in the review set, we would put 4 as our "best guess."

Additionally, we ensured all of the users in the test set were excluded from our training data. However, we realized that some users' star ratings were included in the user's and business's average star ratings. To remedy this, we updated all columns that include the current review's information. This included the review counts and average star ratings for both business and user, and the count of votes for "funny," "cool," and "useful." Once we re-processed our data with these updates, we re-ran our top models and found the following results.

| **Model** | **Previous RMSE** | **New RMSE** |
|---|---|---|
| Linear | 0.9750 | 1.1157 |
| Random Forest | 1.0265 | 0.8671 |
| Boosting (Adaboost) | 1.0407 | 1.1183 |

After addressing the data leakage, both Linear and Boosting had higher RMSE's, as we expected. However, the RMSE for Random Forest improved. We attributed this to the updates of the missing user information.

## Category Reduction

The training data contains 508 different business categories, of which businesses can fall into more than one (examples include "Restaurant," "Italian," or "Beauty & Spa"). For our initial data preprocessing, we created 508 dummy variables to represent them. Since most businesses fall into more than one category (and a few fall into none), we did not have to eliminate one feature to avoid perfect multicollinearity. However, these features are fairly sparse, and we wanted to decrease the number of features needed to capture this information. Thus, we employed principal component analysis (PCA) to reduce dimensionality.

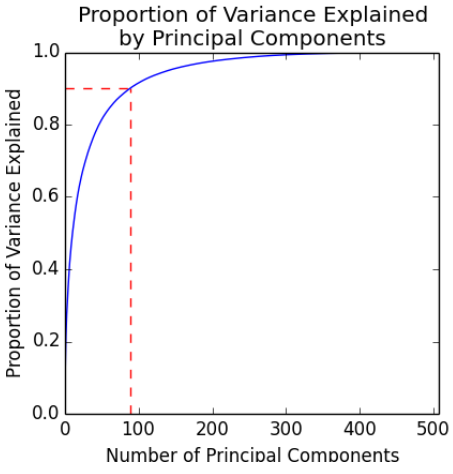



***Figure 2:*** *Selecting Principal Components*

After performing PCA on our entire training data, we narrowed down our principal components to the number that cumulatively explain 90% of the total variance present in the

category features. Figure 2 shows the proportion of variance explained against the number of principal components. As indicated by the red dotted line, fewer than 100 of the principal components explained 90% of the variance. We chose to round up to 100 components. Thus, the process of applying this feature engineering comprised of separating out the category dummy variables in the training data, fitting PCA on them, and inserting the first 100 back into the data. The components for the test data are simply calculated using the same linear combinations determined by the training data.

## User Clustering


***Figure 3:*** *K-Means Scree Plot*

In our user data, we have information about how many reviews a user has written, what their average star rating is, and how many people voted their reviews as "funny," "cool," or "useful." We clustered these users based on similarity to other users (using K-means) as a way predict their ratings. Since we saw high correlation between the three different votes columns, we excluded two of them from our analysis since they would be accounted for in the third one. Before we could run K-means on the data, however, we needed to determine the number of clusters. In order to do this, we created a Scree plot of cluster sizes, from 1 to 15, and plotted the variance of the sum of squares. We found that the best number of clusters to use was five. We performed K-means on our training set; however, we created the clusters based on unique users.

## Text Features

In order to leverage users' business reviews in our model, we generated text features from among user review text within the training data. After cleaning the text of stopwords and punctuation (highlighted earlier in the EDA section), n-gram features were created using a "bag of words" approach, where we counted the frequency of bigrams and trigrams across documents. To do this, review text was aggregated to the user level (all of a user's review "strings" were combined and processed for n-grams). We then calculated a modified term frequency inverse-document frequency (tf-idf) as $\frac{1}{r}\sum_{i=1}^{N} tf(t,d)\cdot\log\left(\frac{N}{df(t,N)}\right)$, where a document here is a user. As shown, the tf-idf ratio is offset by the number of reviews by that user ($\frac{1}{r}$). These scores show relative n-gram

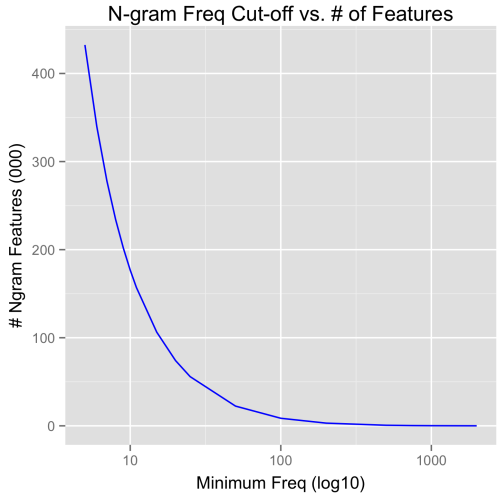



***Figure 4:*** *N-gram Features vs. Frequency Cut-offs*

importance across the users in our training data.

Finding the nonzero tf-idf values for our n-gram features (from within our "corpus" of reviews) resulted in over 15 million bigram and trigram features. Clearly, this was problematic, since the feature space was very sparse, and not all features were of equal importance. As a result, we chose to set a minimum frequency across users for n-grams to occur. Figure 4 shows how the number of features changes as the threshold (in log scale) increases. As a result of this exercise, we set the frequency minimum to 1000 reviews, allowing for 163 n-gram features for use in our final training set. We felt this to be a sufficient compromise between reducing feature space and retaining predictive power.

### COLLABORATIVE FILTERING

In order to take advantage of the deeper relationships between users (rather than relying simply on average stars, review count, etc.), we chose to implement a user-user collaborative filtering algorithm. The predictions that result from this process would serve as a new feature for the algorithm. For any given user and business, we looked at the top five users who were most similar to the user who had rated the business. Then, we took a weighted average of their ratings of the business. In many cases, there were no more than five users who had also rated the business, so the algorithm took a weighted average of up to five users depending on what information was available. For this reason, we found common distance metrics (such as Euclidean and cosine) to be a poor fit for this sparse data. Instead, we opted to use Jaccard distance, which simply takes into account how many businesses two users have both reviewed. As an added benefit, using Jaccard distance was also more computationally efficient. However, despite these adjustments, the data was simply too sparse: about 90% of user-business combinations did not have sufficient information about users who rated the same business to make a prediction, leading to the "cold start" problem. Imputing these missing values with the mean led to a very low-variance feature overall.

### MODEL EVALUATION

Upon completion of the above feature engineering, we tested the performance of each of our chosen three algorithms on every combination of the new features. Ideally, for this portion of the modeling process, we would perform cross-validation on the full training data. However, due to time constraints, we only performed 5-fold cross validation on 50% of the data. In the below results, we refer to each of the features by a number (as listed in the first chart).

***Figure 5:*** *RMSEs of Different Model-Feature Combinations*

| Category Reduction | User Clustering | Text Features | Collaborative Filtering |
|---|---|---|---|
| 1 | 2 | 3 | 4 |

| | Model | | |
|---|---|---|---|
| **Features** | **Linear** | **Random Forest** | **Boosting (Adaboost)** |
| 1, 2, 3, 4 | 338.1979 | 1.3312 | 1.1197 |
| 1, 2, 3 | 1.1277 | 1.0119 | 1.1139 |
| 1, 2, 4 | 60.2772 | 1.0656 | 1.1161 |
| 1, 3, 4 | 89.7992 | 1.1632 | 1.1227 |
| 2, 3, 4 | 1.1301 | 1.0905 | 1.1184 |
| 1, 2 | 4.2760 | 0.9416 | 1.1156 |
| 1, 3 | 2.8759 | 1.0101 | 1.1185 |
| 1, 4 | 3.0231 | 1.0534 | 1.1227 |
| 2, 3 | 1.1264 | 1.0901 | 1.1203 |
| 2, 4 | 1.1185 | 1.0457 | 1.1166 |
| 3, 4 | 1.1276 | 1.0866 | 1.1215 |
| 1 | 7.4225 | 0.9494 | 1.1184 |
| 2 | 1.1221 | 0.9837 | 1.1138 |
| 3 | 1.1342 | 1.0771 | 1.1199 |
| 4 | 1.1781 | 1.0395 | 1.1147 |

There were substantial increases in RMSE for many of the linear regression models. In particular, it appeared the addition of the category reduction features affected RMSE considerably. We suspect this was due to the original business category dummy variables correlating strongly with the review rating, but the new principal component features were not. PCA only considers the features when fitting, not the target variable. Thus, we immediately eliminated linear regression as our final model.

Overall, the Random Forest model had the best RMSE scores with all different combinations of our feature engineering. Adaboost with decision trees performed similarly, though slightly worse. However, implementing all new features did not result in the best model. We suspected that the sparsity of the data prevented the collaborative filtering and text features from being as effective as they could be. Many users and businesses had few reviews, so it was difficult to compare users to each other based on review text or ratings. Therefore, we chose our final model by the best RMSE that applied category reduction and user clustering. This resulted in a total of 578 predictors.

## FINAL MODEL SELECTION: HYPER-PARAMETER TUNING

In our final step of the model development process, we tested different combinations of hyperparameters to see which resulted in the lowest RMSE. The hyperparameters we tested were 1) the number of trees, and 2) the number of features considered at each split. The other hyperparameters are those that follow from decision trees: purity criterion, maximum tree depth, minimum examples to perform a split, minimum examples in a leaf, and maximum number of leaves. In sklearn, the latter three hyperparameters default to be less restrictive, allowing trees to overgrow as much as possible. Since ensemble methods benefit from more variability among the individual estimators, this is typically the optimal choice for these hyperparameters. We considered the following values for each of the two parameters we tested:

| Feature | Parameter Space |
|---|---|
| Number of trees | 10, 50, 100, 200, 500, 1000 |
| Number of features | $p = 578,\ \sqrt{p} = 9,\ log_2(p) = 24$ |

After performing a grid search over these values, we found the lowest 5-fold cross validation RMSE (0.8265) resulted from the model with 1000 trees and the full set of $\sqrt{p}$ features. This hyperparameter tuning lowered our RMSE by roughly 0.12–a significant improvement. These hyperparameter values are very consistent with typical values used in practice.

## Final Evaluation & Results

Using our final algorithm, predicting on the test set resulted in an RMSE of 1.1319. This is worse than our average cross-validation score of 0.8265 on the training set, indicating we have likely overfit a little to our training set. For reference, random guessing would have resulted in an RMSE of about 3.4274 and guessing the training set mean (zeroR) would have resulted in an RMSE of 1.2183. Therefore, our algorithm performed much better than random guessing, and slightly better than zeroR. Given more time to investigate, we would have liked to explore the possible reasons behind this leap in error.

## Discussion & Conclusion

In this project, our main challenges arose from correcting data leakage and creating new features that were helpful for our predictions. When we realized that our training data was "tainted" by our test data, it took careful consideration on how best to correct for it. Even after we determined a way to correct features such average star rating and total number of reviews for users and businesses, data leakage was something we had to consider throughout our feature engineering process as well. During our feature engineering, the sparsity of the data also posed a major challenge. The text features and collaborative filtering depend on having a reasonable number of reviews per user, but this was often not the case. However, despite these feature engineering challenges, we managed to create two useful feature sets by using PCA to reduce business category features and K-means to cluster users.

Our final model performed well in cross-validation on the training set, with an RMSE of 0.8265. However, it performed reasonably worse on the test set with an RMSE of 1.1319, indicating possible issues with overfitting. Despite the jump in RMSE, our algorithm still performs better than random guessing and zeroR, but not to the extent we would like. This is certainly an area for further improvement in the future.

Our project has clear business applications, as recommendation systems already serve as a key feature for products, while substantially enhancing user experience. However, we see several areas for improvement and further exploration. Since sparsity created such a problem for us, we could try and supplement our data with the Yelp API and web scraping. This would potentially improve the positive impact of the text features and collaborative filtering on our model. Secondly, we would like to take our project to the final step: creating a tool that provides a list of recommended businesses for a specific user based upon their predicted star rating of that business.

# APPENDIX A: EXAMPLE DATA (KAGGLE TRAINING FILES)

## BUSINESSES


```
{
    "business_id": "o3ehs4ZEdsizbJyB9_j7uQ",
    "full_address": "6938 E 1st St\nScottsdale, AZ 85251",
    "open": true,
    "categories": ["Art Schools", "Shopping", "Jewelry", "Accessories",
                   "Fashion", "Education"],
    "city": "Scottsdale",
    "review_count": 14,
    "name": "Jam",
    "latitude": 33.492195000000002,
    "longitude": -111.9312992,
    "state": "AZ",
    "stars": 5.0,
    "type": "business"
}
```


## CHECK-INS


```
{
    "business_id": "RUZvUPOn90ScX60eETwcCw",
    "checkin_info": {
        "16-0": 3,
        "21-1": 1,
        "13-5": 7,
        "8-5": 1,
        "8-0": 1,
        "17-2": 2,
        "21-2": 1,
        "11-0": 1
    },
    "type": "checkin"
}
```

### REVIEWS


```
{
    "votes": {
        "funny": 0,
        "useful": 1,
        "cool": 1
    },
    "business_id": "Li5LOL873Ep8HoPRML18sw",
    "user_id": "Ak2jlINPRk9dWWazWUW2hA",
    "review_id": "Cmkb7SDJWYtWZXgAfOwJkQ",
    "stars": 5,
    "date": "2008-12-05",
    "text": "This place is great! I am a huge fan of any car wash
             that doesn't use brushes. The people here are incredibly friendly [...]"
    "type": "review"
}
```


### USERS

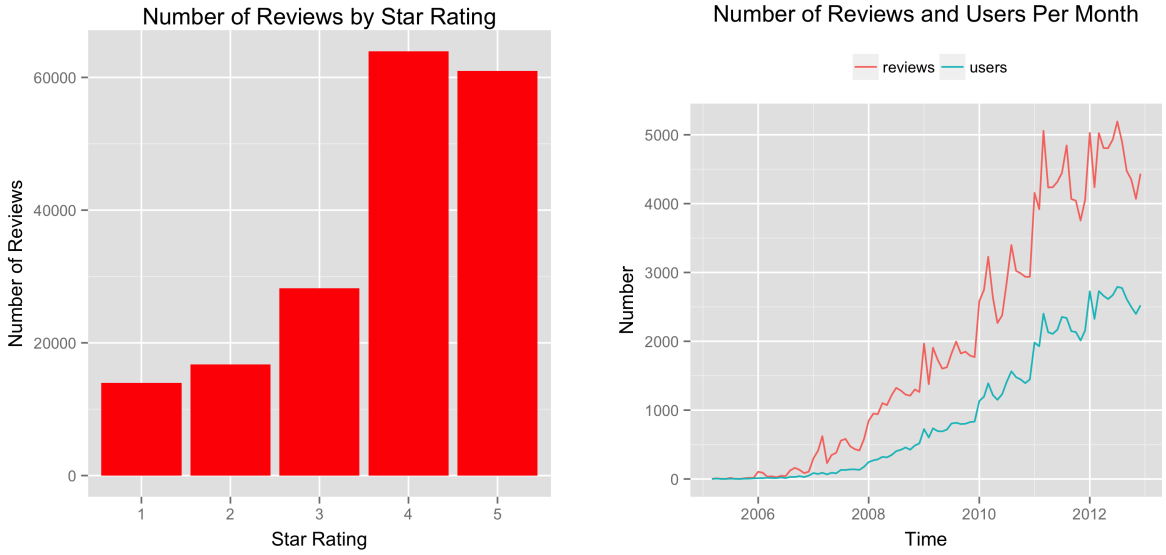

```
{
    "votes": {
        "funny": 13,
        "useful": 21,
        "cool": 5
    },
    "user_id": "qMCT8OblN3finTvXcba-oA",
    "name": "Elizabeth",
    "average_stars": 3.6200000000000001,
    "review_count": 16,
    "type": "user"
}
```

# APPENDIX B: EXPLORATORY DATA ANALYSIS

## REVIEW DATA

### VISUALIZATIONS

The training reviews data set has 183,925 Yelp reviews, including the user id, business id, date posted, star rating, review text, and votes for the review as being useful, funny, or cool. The review star rating for a given user id and business id is the target variable we are trying to predict. Star ratings can only be an integer from 1 to 5. Thus, we investigated the distribution of this variable, shown in Figure 6a. The distribution of star ratings is left-skewed; most ratings are either 4 or 5. This contradicts a common conception that users only write reviews when they are either very happy or unhappy with the business. The mean rating is 3.7674, and the median is 4.




***(a)*** ***(b)***

***Figure 6:*** *Reviews*

In addition, Figure 6b shows how the number of monthly reviews and users has changed over time. Both have increased at about the same rate, indicating the number of reviews per user has remained relatively constant.

### TEXT ANALYSIS

Of the ~184K reviews in our training data set, we found four empty (null) reviews. Of the remaining observations with text, we found the following summary statistics:

| | |
|---|---|
| Total tokens | 23,744,120 |
| ·After stopword/ punctuation removal | 12,163,797 |
| Unique unigrams | 296,263 |
| Unique bigrams | 4,556,832 |
| Unique trigrams | 10,432,333 |

Looking deeper into the text, we identified the most frequent n-gram word combinations (after stopword & punctuation removal) from our data set:

| UNIGRAMS | | BIGRAMS | | TRIGRAMS | |
|---|---|---|---|---|---|
| (place) | 118,837 | (happy, hour) | 10,377 | (sweet, potato, fries) | 1,579 |
| (good) | 118,572 | (go, back) | 9,973 | (wait, go, back) | 1,231 |
| (food) | 108,432 | (pretty, good) | 7,292 | (cant, wait, go) | 1,194 |
| (like) | 91,558 | (really, good) | 7,051 | (cant, go, wrong) | 1,046 |
| (great) | 89,562 | (first, time) | 6,982 | (would, go, back) | 1,023 |
| (one) | 73,182 | (great, place) | 5,990 | (definitely, go, back) | 890 |
| (get) | 70,404 | (food, good) | 5,453 | (go, back, try) | 859 |
| (time) | 62,419 | (next, time) | 5,318 | (great, food, great) | 735 |
| (go) | 62,121 | (dont, know) | 5,156 | (love, love, love) | 699 |
| (really) | 62,078 | (ice, cream) | 5,107 | (best, ive, ever) | 685 |

## BUSINESS DATA

The business data set contains information about 11,508 businesses. It includes the business id, name, star rating, applicable business categories, and various geographic information (latitude, longitude, address) about each business. It is to be noted that the star rating and number of reviews are calculated from all of the business' reviews, not just the subset of reviews included in the review data set. For all intents and purposes we will reference these as the "true" values. Conveniently, all of the businesses in the business and review data sets had matches in the other. However, some businesses are missing values for certain variables; 2,912 business are missing star rating and total number of reviews. These correspond to 12,559 reviews in the data. For this reason, it was difficult to compare how many of the reviews for the business were represented in the review data. However, we were able to compare how the average star ratings for each business differed between all reviews and what was included in the review data. The resulting plot is below.

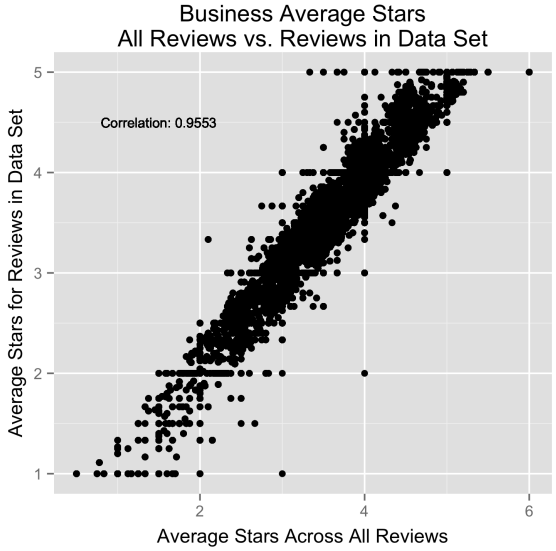
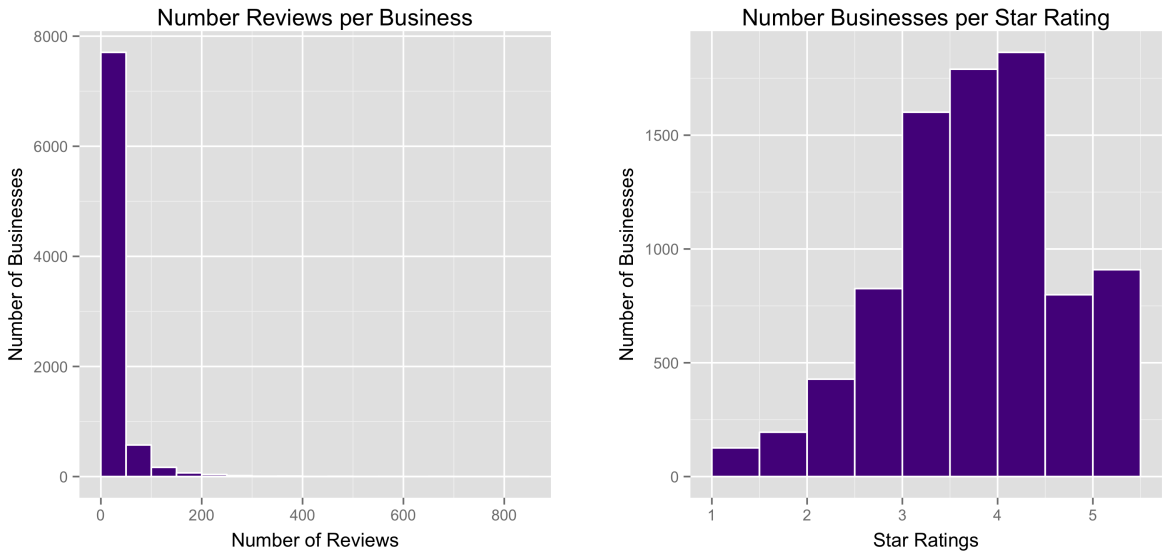



***Figure 7***

There is a very strong correlation (0.9553) between the two, indicating that the reviews included in the review data set are representative of the business' reviews overall. Next, we

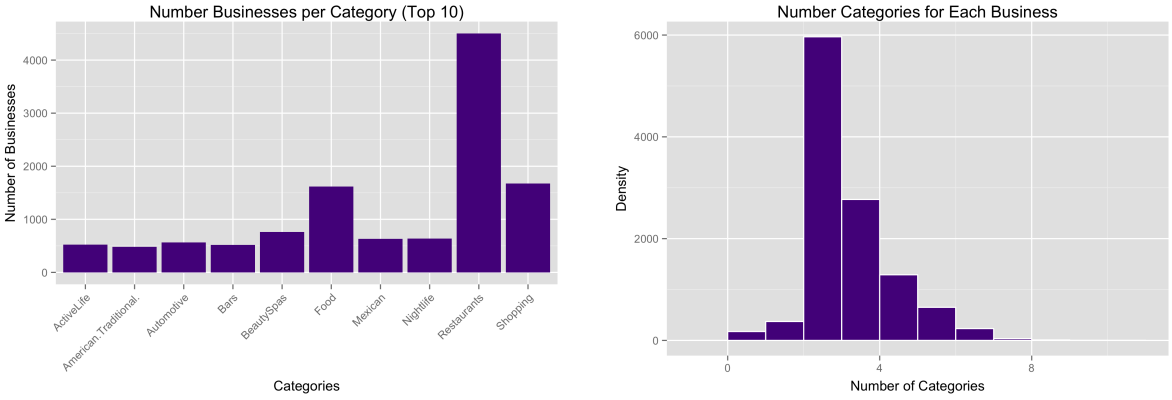
looked at the attributes of the reviews of the businesses.




***(a)*** ***(b)***

***Figure 8:*** *Reviews of Businesses*

The number of reviews per business has a very right-skewed distribution, with the vast majority of businesses having less than 100 reviews. The distribution of star ratings is not quite symmetric. Most businesses have a rating of 3, 3.5, or 4 (all business ratings are rounded to the nearest 0.5). Low ratings are very rare, and high ratings are slightly less so. Next, we explored the business categories, which posed a difficulty for us, as there are 508 categories present in the training data. In addition, businesses can have more than one category. Below are the number of businesses per category for the top 10 most frequent categories, and the number of categories per business.




***(a)*** ***(b)***

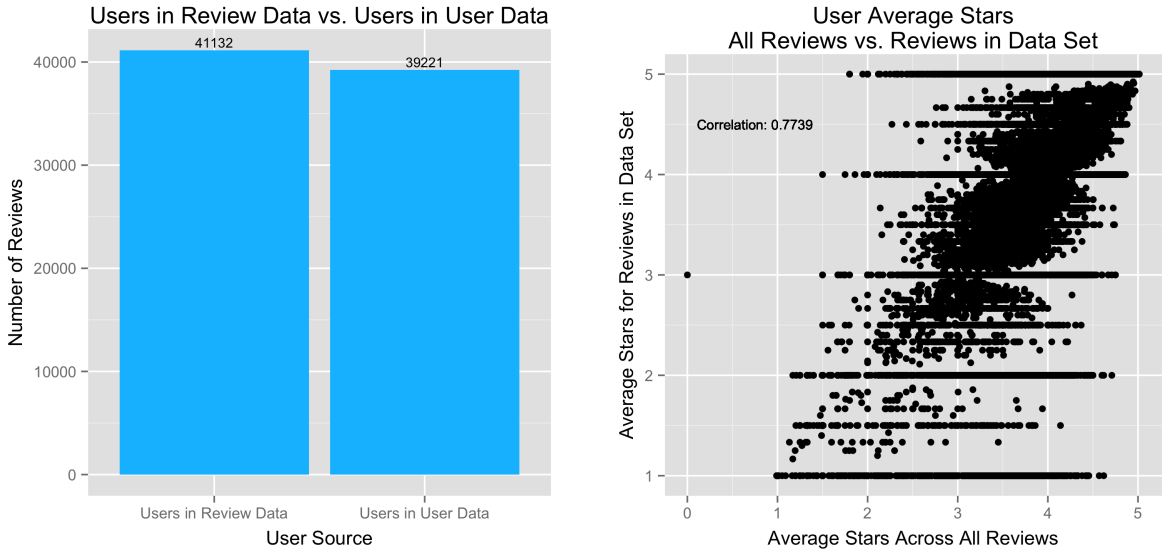
***Figure 9:*** *Business Categories*

Not surprisingly, the most common category is “Restaurants.” Similarly, “American Traditional” (food), “Food,” and “Mexican” make the top 10. The rest of the top categories are relatively diverse, such as “Automotive,” “Beauty/Spas,” and “Shopping.” The number of

categories listed for a business ranges from 0 to 10. Thus, creating dummy variables for these categories would create a set of very sparse features.

## USER DATA

The user training data includes the user id, average rating, number of reviews, and total votes received (funny, cool, useful) for 41,132 Yelp users. It is to be noted that the latter three attributes are calculated from all of the users' reviews, not just the subset of reviews included in the review data set. For all intents and purposes we will reference these as the "true" values. This brought us to first investigate whether the reviews in our available data set are representative of the users' overall review behavior. The plots below show how many users there are in the review data vs. user data, and how the user average stars compare between the data sets.

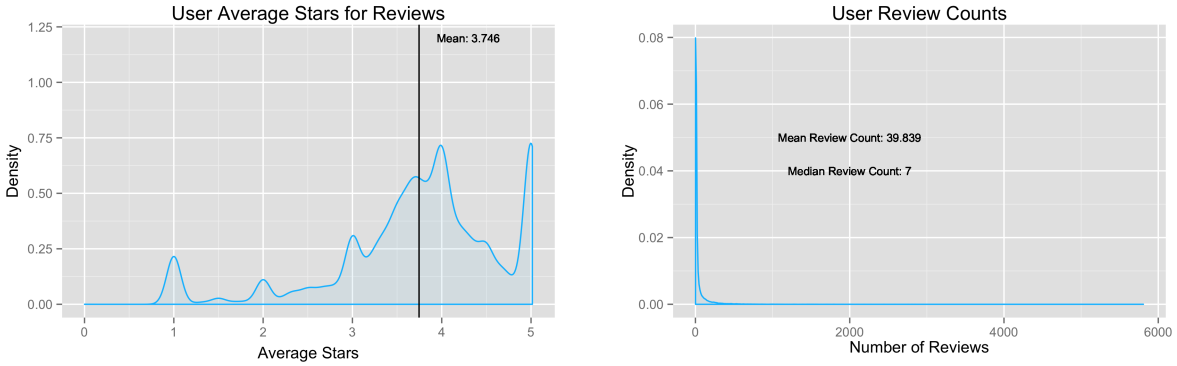



***(a)*** ***(b)***

***Figure 10:*** *Users in Review Data Set vs. User Data Set*

There are 1,911 users in the review data that are not in the user data, so these users have missing values for (true) average rating, number of reviews, and votes received. There is a strong correlation (0.7739) between the user average stars in the user data and review data, indicating the reviews in our review data are generally representative of the users' true behavior. However, the user average stars in the review set appear to be less precise, which is evident from the large number of averages that are exactly 1, 2, 3, 4, and 5.

Next, we looked at the univariate distributions of the true user average stars and number of reviews per user.

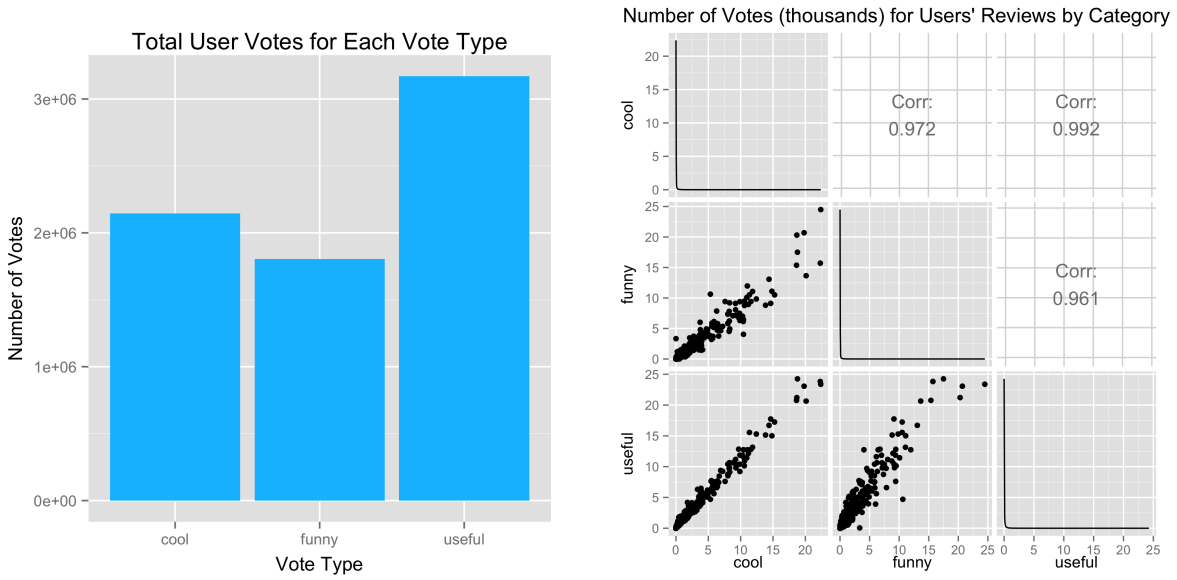



***(a)*** ***(b)***

***Figure 11:*** *User Average Stars and Review Counts*

The distribution of user average stars is complex, with modes at the integer values 1-5. This is possibly due to many users who have only written a few reviews, which is supported by the heavily skewed distribution in Figure 11b. However, we see that the majority of non-integer averages fall between 3 and 5, and the mean of user average stars is 3.746.

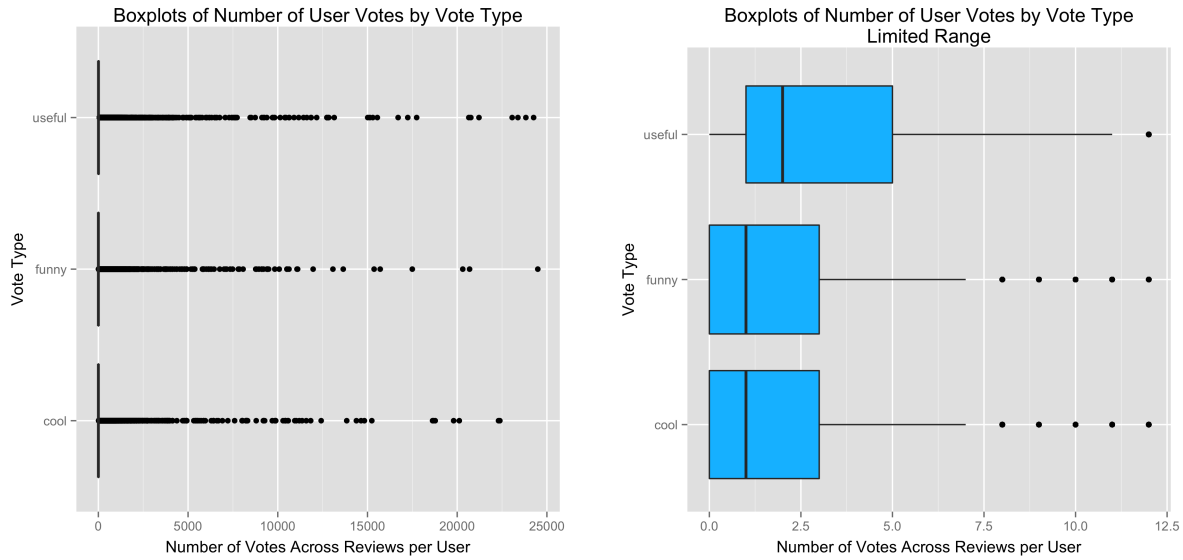
Lastly, we explored the number of votes users received in the funny, cool, and useful categories. The plots below show the sum of all votes in each category and the pairs plot for the categories.




***(a)*** ***(b)***

***Figure 12:*** *Votes Received for Users' Reviews*

Here we see that there is a tendency for users to vote reviews as useful rather than funny or cool. In addition, there is strong multicollinearity between the categories of votes; all three correlations between pairs of categories are above 0.95. Thus, we were better off eliminating or combining these variables rather than risk multicollinearity affecting models sensitive to it. We observed the univariate distributions for each vote category in with the boxplots below.




***(a)*** ***(b)***

***Figure 13:*** *Boxplots of Users' Votes Received*

The distributions are all heavily right-skewed. Figure 13b shows the same boxplots with a smaller range on the x-axis. It is clear that the variability on the number of useful votes a user receives is larger than for the other two categories. Thus, this feature may be more useful than the other two.

### CHECK-IN DATA

The check-in data contains information about how many times users "checked in" to each business, and in what hour interval the check-in occurred. We chose to aggregate this data to find the total number of check-ins each business has overall. We immediately noticed that many businesses have no check-ins, as shown in Figure 14a. Figure 14b shows how the number of check-ins compares to the number of reviews a business has, excluding those with no check-ins and a single outlier with 22,977 check-ins.


*(a) Whether a Business has Any Check-Ins*

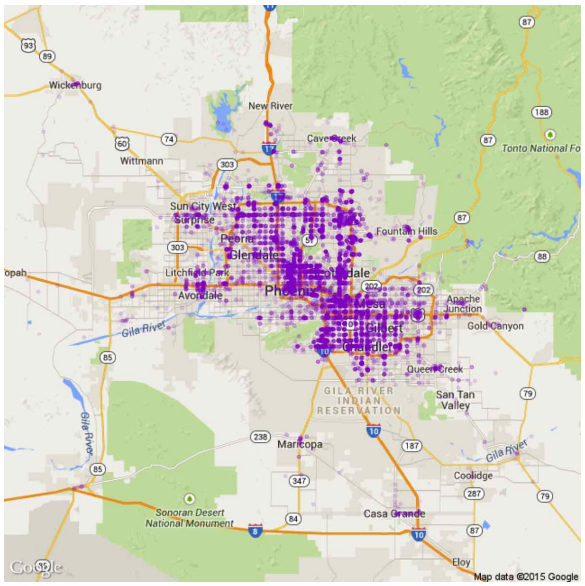

*(b) Excludes Businesses with 0 Check-Ins and Single Outlier*

***Figure 14:*** *Check-Ins for Businesses*

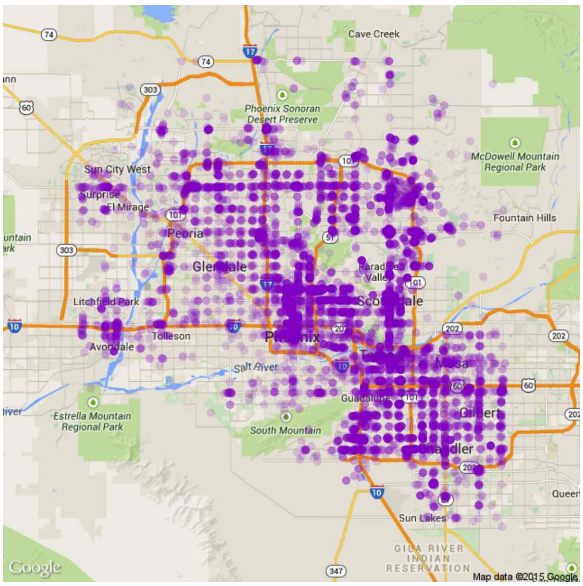
Surprisingly, for businesses that do have check-ins, they seem to receive more check-ins than reviews. The relationship between these two features is moderately strong, with a correlation of 0.5552.

## MAPS

Lastly, we wanted to explore the geographic aspects of our data. We plotted the density of businesses, as well as reviews and check-ins using the latitude and longitude of the corresponding business. The first maps below show the businesses.



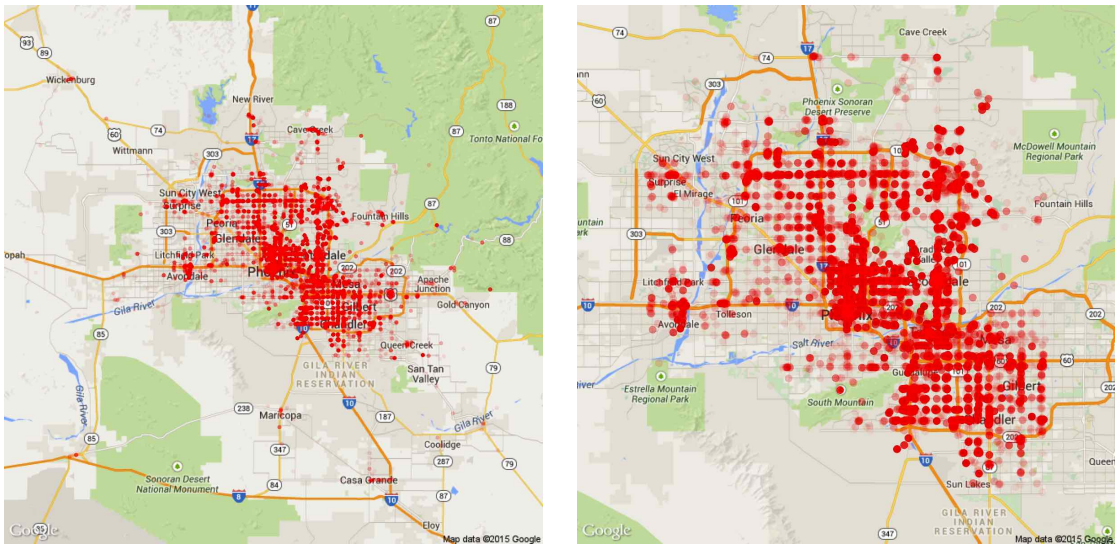

*(a) Full Map*




*(b) Zoomed Map*

***Figure 15:*** *Maps of Businesses*

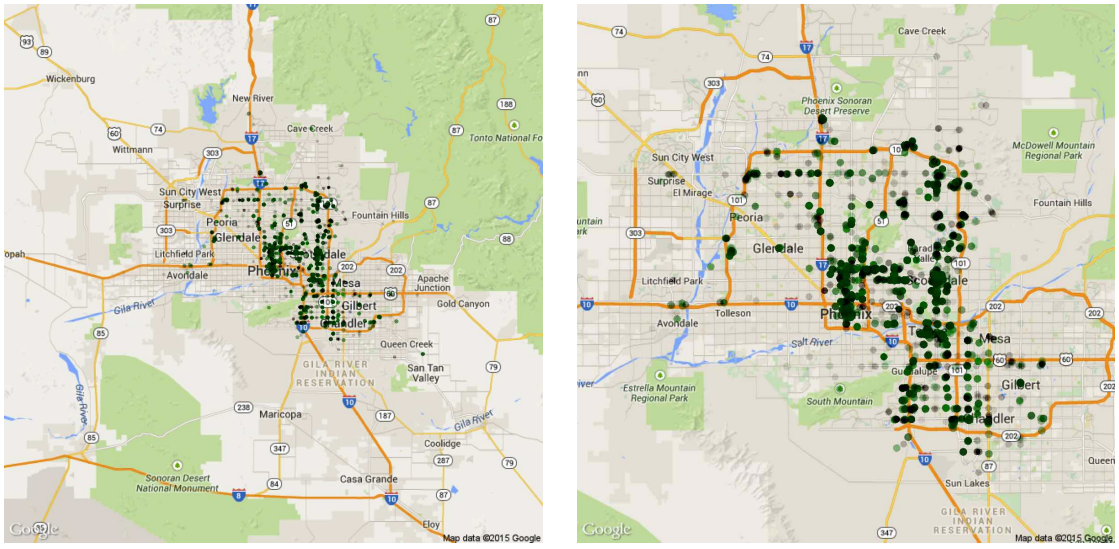
Most of the businesses are concentrated in Phoenix and the cities nearest to it. There is a particularly dense group of businesses over the center of Phoenix. There are very few businesses in the data that are located in the outskirts of the greater Phoenix area. The maps of reviews below show a very similar pattern.




***(a)** Full Map* ***(b)** Zoomed Map*

***Figure 16:** Maps of Reviews*

Thus, the geographic distribution of the reviews closely mimics the distribution of the businesses. There do not appear to be any particular areas where businesses are much more likely to get reviews. Finally, we see the check-ins on the maps below.




***(a)** Full Map* ***(b)** Zoomed Map*

***Figure 17:** Maps of Check-Ins*

The check-ins are significantly more geographically sparse than the reviews or businesses. The cluster of businesses in Phoenix is even more emphasized here, indicating users mostly check-in to businesses in popular areas.