COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

Prediction of a Restaurant's Star Rating in Yelp

Subject: GR5293 Topics in DS Applied ML for Financial Modeling

Professor: Michel Léonard

Group # 12

Jose Luis Lopez Torres (jil2239)

Yiming Tan (yt2633)

Nan You (ny2286)

Xuan Jia (ny2286)

Sixuan Li (sl4410)

Semester: Spring 2019

Executive Summary

The ability to drive customers to a venue and have them return is crucial in the restaurant business. Our team developed a project with the motivation of understanding the relationship between customers' comments on a social media network and the performance that a restaurant has on the same platform. We assumed that, all else being equal, new customers would choose a restaurant with a higher star rating when deciding to go out.

This topic has been covered extensively by data scientists, who applied regression techniques on the available business features to predict a restaurant's performance. We believe this paradigm is flawed since, despite the fact that a star rating is a numerical value, a star level better represents a category. We believe that classification techniques can provide a more appropriate approach, so we decided to evaluate if the words in a restaurant's comments, in conjunction with the business features, can provide insights into the marketing topic.

The question is then if Machine Learning techniques using Yelp reviews outperform regression to predict a restaurant's star rating. By method, we expect to provide business insights and help restaurant owners to improve their performance in terms of customer ratings.

To make accurate predictions in this matter, we decided to use Yelp as a source. This social network has over 170 million monthly active users, and it has contained information about different types of businesses across North America. Furthermore, the age of Yelp users is distributed uniformly across different generations (34% of users are aged 18 to 35, 35% are aged 35 to 54, and the other 31% are users over 55 years of age), which means that we would get a representative sample of the population and their opinions.

The available dataset contained information of 10 different metropolitan areas in North America. We decided to use Las Vegas metro as our focus for the project, as it had more than 2,000 restaurants. The restaurant list was split at random into a 75% train set and a 25% test set. We also drew the 283,213 reviews for those 2,000 venues from the dataset, as that was our main interest during the project.

The dataset contained different business features, like whether credit cards were accepted or not, if outdoor seating was available, or the price-tier of the meals at the venue. We analyzed the distributions of each feature to understand if there were any trends that could help us reach a conclusion fast.

In terms of the review portion of the dataset, we used different text analysis techniques to clean the comments and fully take advantage of the vast amount of information. We followed a strategy that involved using a stopword dictionary from Twitter, in order to remove common words and slang. Furthermore, we used lemmentization to group words that shared a common root, thus reducing the number of variables for the model.

In order to make the insights applicable to business, we decided we would separate restaurants into two categories: successful restaurants (those with a star rating of 4 or more) and restaurants under improvement (those with a star rating below 4). The use for restaurant owners becomes obvious, since they will be able to understand what are the factors that will drive their businesses into the former group.

Evaluating whether or not our model is relevant must be the first step of the process. We decided to use a project from *Channapragada, S.and Shivaswarmy, R.* as an external reference, and established their results as our baseline. They used a support vector machine model that achieved an accuracy of 0.627; our goal was to create a model that captured more of the data's variance, and thus set up a goal of 0.70 for the accuracy of the classification technique.

In order to achieve a value-add greater than 10%, we fast-prototyped using different classification algorithms on the dataset. The purpose of this process was to understand if any had a better performance given the distribution of the covariates. The fast-prototyping phase included running the following models: logistic regression, support vector machine, random forest, gradient boosting tree, and neural network. This worked as a first approach to understand which techniques were a better fit, at least at first glance.

We decided to focus on Logistic Regression and Gradient Boosting algorithms, as they provided good accuracy. More specifically, Logistic Regression is one of the most basic classification models and would perform better if the relationship between the covariates and the response were linear. Following a similar logic, Gradient Boosting is an efficient classifier when the aforementioned relationships are non-linear.

The fast prototyping step also helped us define a structure to achieve value-add through different aspects, like normalizing the data and optimizing the hyperparameters of the models. Furthermore, we understood that we were at risk of overfitting the model given the potential amount of features under consideration. To avoid this, we decided to use Principal Component Analysis to capture the data's variance through uncorrelated components. As an additional way of extracting information from features, we applied a Latent Dirichlet Allocation Model to transform the list of words of the comments, and categorize them into similar groups.

Our objective through the machine learning models was to predict the star ratings based on the features of data, while achieving value add over the baseline of previous works based on regression. We split the dataset into train and test groups to try to infer conclusions on the latter based on the model fit of the former.

Having chosen the techniques for the project, we moved on to optimizing their parameters. This process helped us achieve a higher value-add, by improving the predictive power of our models. We used Scikit Learn's Gridsearch function to provide each model with a list of parameters that had the potential of improving accuracy. In the case of the Logistic Regression model, the process showed a gain of 1% in value-add by switching from an 'L2' distance type to 'L1'. For the Gradient Boosting model, the parameter optimization process showed that the value-add increase could be 2% by modifying the learning rate (0.01), number of estimators (1000), and the maximum depth (20).

Our goal through this project was to create a classification model that could be directly applied to restaurants in Yelp, and could thereby help business owners and entrepreneurs run their companies successfully. By using statistical models to draw information from the words contained in the reviews, we were able to effectively develop a model that created a 20% value-add above our baseline.

There was an additional outcome from this project that we realized: being able to include the most important terms in a functional model would work as a guide for restauranteurs to understand the elements of service that were relevant to customers. For this reason, we decided to draw a comparison: what would be the accuracy of either model if we were to change the Principal Components that represent the more than 5,000 words on the dictionary for a list of the most used words?

The results did not exactly fit our expectations, with the four techniques having similar performance and word feature-based models actually getting a better one. Even though we just used 100 words as features, they turned out to classify the ratings properly. All the techniques rendered an accuracy of nearly 0.80, with slight variations based on the types of features (words vs. PCA) and the technique under consideration (Logistic Regression vs. Gradient Boosting). We favored the Logistic Regression model based on words features, since it had high accuracy, as well as good interpretability in terms of knowing which words were positive and which were negative.

Upon reviewing the final versions of the models, we were surprised at how an area that was apparently unrelated to math, such as writing, could be modeled by using statistical models. For instance, the LDA gave us insights into the groups of words that were related by an underlying group within the comments. We were able to create topics (like types of foods, drinks, or even special events) that made sense through an agnostic process, which helped us better understand the Machine Learning technique.

We believed that, although our approach was ambitious, the models based on words successfully predicted the ratings of successful restaurants. Just like in the case of the LDA approach, we were able to see that an agnostic statistical process would include “fresh”, “flavor”, “clean”, “manage” and “friend” as some of the main predictors of a rating. These words carry an important lesson for restaurant managers: in the Las Vegas metropolitan area, customers will give a better rating to venues that offer food that is perceived as fresh and flavourful, that have a friendly ambiance. Also, the surroundings must be clean.

Another important conclusion we drew from our project was the possibility to create a sentiment analysis dictionary for Yelp. Given that we had the coefficients for the word features from the Logistic Regression model, it was possible to derive which words were linked to more successful reviews (above 4 stars) and which were linked to reviews under the threshold. This type of dictionary could provide business owners with a way to not only know that features to pursue, but also which ones to avoid.

Finally, a measure that we believed is valuable from our approach to this problem is the importance of the features for each model: under the Logistic Regression technique, the 25 most important features were words from comments; under the Gradient Boosting technique, 24 were words and a single one was a business feature, which actually had to do with popularity. This further motivates us to pursue alternative methods to obtain and analyze data, as they can hide more important information than what is usually available.

As a potential area of improvement for the next iteration of the project, we would like to develop a dictionary that can help us drop more common words from the model. Despite including a stopword dictionary that is used on Twitter datasets to perform sentiment analysis, words like “be”, “say”, “because”, or “would” were still part of our model. Other terms can help restaurateurs gain a better understanding of where they should focus their efforts, hence our interest in making a meaningful model.

Introduction

Due to the rapid increase in market competition, it is crucial for business owners to adjust their business model in order to take the lead all the time. Determining when and how to make proper business decisions is the most important issue faced by every business owner: What kind of restaurant should be opened? Will offering outdoor seating be a differentiating factor that draws more customers? Is there any significant difference between restaurants that accept credit cards and those that do not? Will a customer consider "take out" option as an important feature while they rate a restaurant? There are several features that can influence the star rating for a restaurant, more than could be listed succinctly; but the more important question is, can we notice the slight difference between how those features impact the result - star rating in our case - and make pertinent decisions immediately?

The objective of this study is to use Machine learning techniques to design models which use existing Yelp reviews to outperform regression techniques in the prediction of a restaurant's star rating. By applying this method in reality, we hope it can provide insightful analytics and help restaurant owners improve their performance in terms of customer ratings.

Business Understanding

Unlike previous research which also performed business analysis using Yelp's data set, we believe our approach can improve the accuracy of the model by applying Text Mining methods to the analysis. Natural Language Processing (NLP) is a popular methodology subfield of computer science, information engineering, and artificial intelligence. It is also a methodology in text mining.

In our research, we used several built-in packages in python for NLP like nltk and wordcloud. Further details will be discussed later. From the Yelp customer review dataset we could not only collect customers' ratings for a given restaurant, but also the text reviews for that restaurant. By NLP we could get some "additional" information through these reviews. For example, given a customer's review contains multiple positive words for a specific venue, we believed we can make a prediction saying that his or her rating for this restaurant was very good (say, 4 out of 5 stars). Text mining techniques could provide more detailed information about how a customer actually feels about a given restaurant, since some information could not be entered as a rating feature in the Yelp rating system. Moreover, by analyzing the mood behind each world, we could somehow notice the degree of their preference or dislike for a given independent feature. It was obvious that on average we could conclude that "like" and "love" could not be treated as the same degree of preference, but they might imply the same star rating in Yelp's system.

Compared to the vast amount of information offered by text analysis, the business features originally offered by Yelp's dataset were too roughly defined and limited to describe human feeling. Undoubtedly, the result of a model based on text mining techniques would be more precise and valuable compared to those without this method.

After finishing data pre-processing, data analysis and data visualization, we believed that, since the star rating system varied from 1 to 5 stars, a 4-star rating or better would be strong evidence to define the success of a restaurant. Thus, in our research, we predicted whether the star rating of a given restaurant lied at either side of a threshold at the 4-star level.

There were two baselines that we wanted to overcome: accuracy for a Support Vector Machine (0.627) and the mean square error for a regression model (0.653). Both baseline data came from a previous research paper: the first is from *Channapragada, S.and Shivaswarmy, R.,2015* and the second from *Prediction of rating based on review text of Yelp reviews*. By the end of this project, we hoped our model could increase the accuracy to above 0.70 by using machine learning techniques.

After carefully selecting and optimizing, two models would be used in this research paper to figure out the classification problem: a Logistic Regression model and a Gradient Boosting model. Logistic Regression is a classification model that predicts the likelihood of either of 2 outcomes (in our case, the star rating greater than or equal to 4, or less than 4). This model explained a relationship between one binary dependent variable and several independent variables (nominal, ordinal, interval or ratio-level).

The reason why we chose Logistic Regression was that we assumed all the features and the response were linearly correlated. Furthermore, the technique itself was a basic model for a classification problem. Our hypothesis was that if our prior belief of a linear relationship between features and response was not correct, the performance of the model would be bad and the accuracy would be very low. Through fast prototyping, we were able to determine that the performance of logistic regression was better than our baseline, so we moved on with said technique.

We also tried a non-linear classification model. Gradient Boosting, the second model mentioned above, is a machine learning technique for regression and classification problems. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The reason we chose gradient boosting as our second model was that, since the performance of logistic regression was good enough to beat the baseline, we wanted to choose the most efficient model to compare with it. Boosting models are the fastest models among non-linear classification models, hence our decision.

In order to reduce the risk of overfitting, in each model, we implemented two major different methods of feature engineering: Principal Component Analysis (PCA) and word features based on a Latent Dirichlet Allocation (LDA) model. Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. LDA is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.

Our target was to beat those external baselines by using these two models. Details about how those models worked and the performance will be discussed in later sections.

Data Understanding

For this project, we selected Yelp's dataset. This dataset has a subset of businesses, reviews, photos and user data of 5.79 gigabytes (uncompressed) in JSON format. Given that our project consists of predicting a restaurant's star rating, we decided to focus on a subset of the data: we only considered restaurants in the Las Vegas metropolitan area. The selected dataset formed with business data and review data contains information about 2,013 restaurants and 292,144 reviews, which has a cross-section data structure.

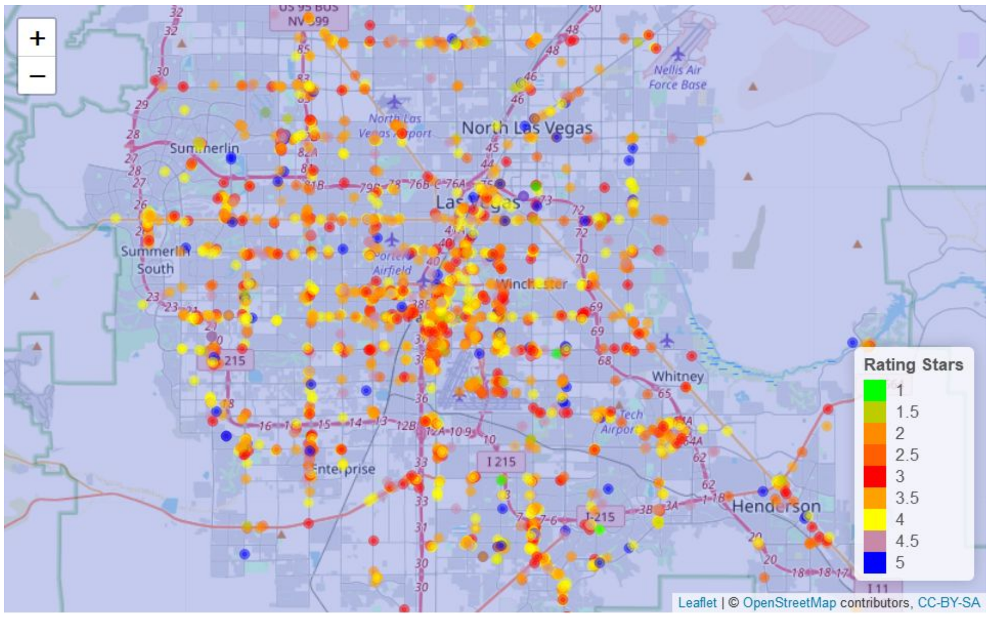
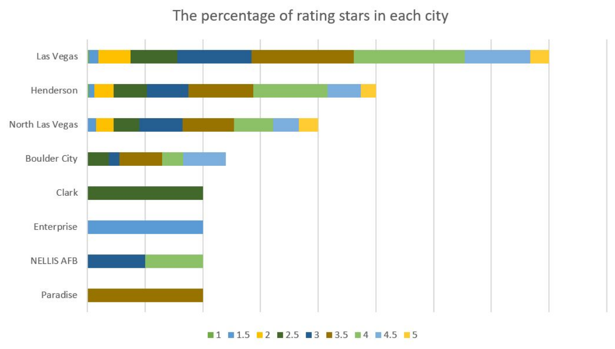


Fig. 1 Star rating of each restaurant

The dependent variable was the star rating of a restaurant, which was quantitative. Figure 1 shows the stars ratings of restaurants in Las Vegas, where warm colors represent the restaurants with 2 to 4 stars while cold colors denote those with 1 or 5 stars which should be paid more attention to. From figure 1, we can see that the colored points are located without any order, which means that the rating of a restaurant may not be influenced by the region of the city where it is located.

For simplicity, we considered splitting the star ratings into two classes: star ratings greater than or equal to 4 and star ratings less than 4. The independent variables were quantitative, including "RestaurantsPriceRange2" and "is_open", and qualitative variables including the 30 PCs, "city", "RestaurantsReservations", "RestaurantsGoodForGroups", "BikeParking", "OutdoorSeating", "RestaurantsTakeOut", "BusinessAcceptsCreditCards", "GoodForKids", "RestaurantsDelivery", "RestaurantsAttire". The PCs were generated by PCA labeled on reviews' term frequency-inverse document frequency (TFIDF) and the rest of the features were drawn from the business dataset.



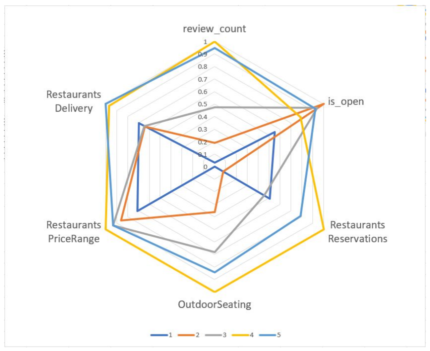


Fig. 2 Percentage of star rating by city

Fig. 3 Radar chart of attributes vs stars

From figure 2, we can see that Las Vegas had the largest number of restaurants among the cities in the area. The number of restaurants with ratings above 4 was lower than those with lower stars in Las Vegas, Henderson, North Las Vegas, and Boulder City, while the other cities only had a limited number of venues.

Figure 2 shows the average score of the attributes of restaurants at each star level. The average attributes varied strongly by star rating. For example, the attribute “OutdoorSeating” was a variable for whether the restaurant had a seating area outside, which might be a positive variable from the perspective that outdoor seating could be a sought-after trait. We transformed the variable by assigning 1 to restaurants whose “OutdoorSeating” is TRUE, and 0 otherwise. The score of “OutdoorSeating” of each star level was calculated by averaging the values of those restaurants. The scores of “OutdoorSeating” from star 1 to 5 were 0, 0.4, 0.7, 0.8, and 1, which meant that the restaurants with higher star ratings were more likely to have outdoor seating.

In addition, for each star level, we could tell that the restaurants with low stars somehow performed badly in some attributes compared with positively rated restaurants. For example, the restaurants with stars below 3 had low scores of “OurdoorSeating”, “Review_count”, “RestaurantDelivery” and “RestaurantReservation” compared with those with more than 4 stars. Figure 2 indicates the attributes of a restaurant had an important influence on the stars it received from customers, and we believed we can give suggestions to improve a restaurant’s rating star through these features.

The reviews dataset was matched to the business dataset through the use of the “business_ids” variable. Table 1 shows a summary of the response variable distribution. We see that 41.8% of the restaurants had stars above 4 while the reviews to those restaurants were 57.6% of all reviews, which indicated that the restaurants with high star ratings seemed to receive more reviews than those with lower star ratings.

The number of reviews of each restaurant had a wide range, where the maximum was 6708 and the minimum was 3. From the histogram in figure 4 we can see that 71% of the restaurants had less than 100 reviews, and 11% of the restaurants had between 100 and 200 reviews. Only 28 restaurants have more than 1152 reviews. Since the small number of reviews might not provide enough information, we needed to conduct review selection in text modeling.

Star	Restaurants	Reviews
above4	842	163139
below4	1171	120074
Total	2013	283213

Table 1 Number of restaurants and reviews

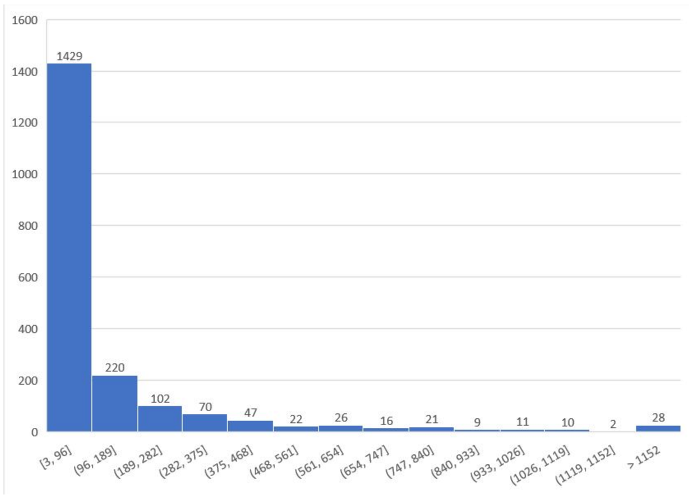


Fig. 4 Histogram for number of reviews

As the dataset had a Cross Section data structure, the dataset was randomly split into the training dataset (75%) and test dataset (25%).

Data Manipulation

Since the original datasets of business information and customers' reviews were separated into two files and had different structures, we had to manipulate them separately before merging them.

As for the business information, the original data format was json and the data were semi-structured at first. We used json library in Python to read the data and did random sampling on the business data to get 2013 restaurants in total. They were distributed in greater Las Vegas Area. There were totally 59 columns, including business id, name, address, locations, review_count, WiFi, Restaurant Reservations and a lot of other variables. However, some of the independent variables in the dataset had a large number of missing values since the original dataset was semi-structured, so we could just use a part of the variable that had an acceptable proportion of missing value.

We dropped all the independent variables whose missing value rates were more than 35% to ensure the quality of the variables we chose. After selection, there were only 13 independent variables remained. For these variables, most of which were qualitative data. We used the mode of each column to impute the missing values. After data manipulation on the business information data, we got a dataset with no missing values.

Data manipulation for the reviews was more complicated because we had to generate meaningful information from the text. The structure of the review data was different from business information features because each column represented a review text instead of a business. The 2013 restaurants had 292,144 reviews in total, which might be tricky to deal with. So we did sampling on the restaurants that had more than 50 reviews to make sure that each restaurant had no more than 50 reviews.

After doing this, we got 64,229 reviews, which was an acceptable number to handle. To turn the reviews into a variable we could use in the model, we first tokenized each review into words and then removed stopwords. Previously, a stopwords dictionary with a size of around 200 was used, but we found that there were still a large number of meaningless words remained. Another stopwords dictionary with a size of more than 1000 was used then. The word cloud of the frequently used words is as followed.

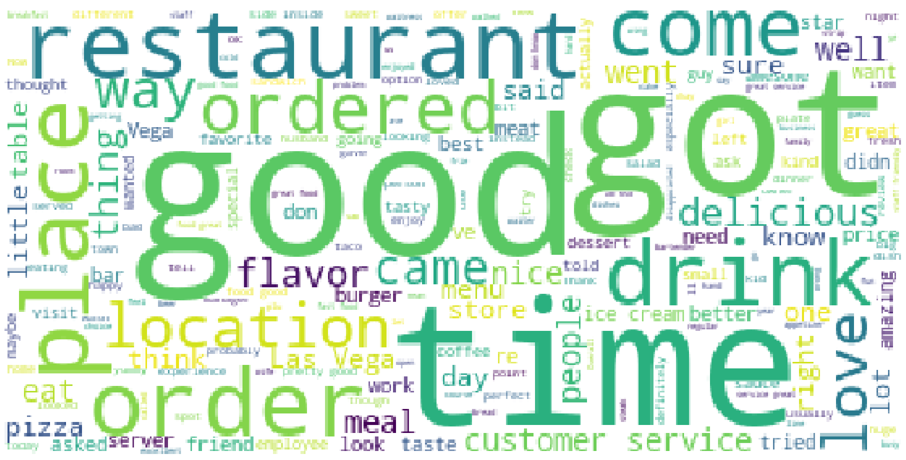
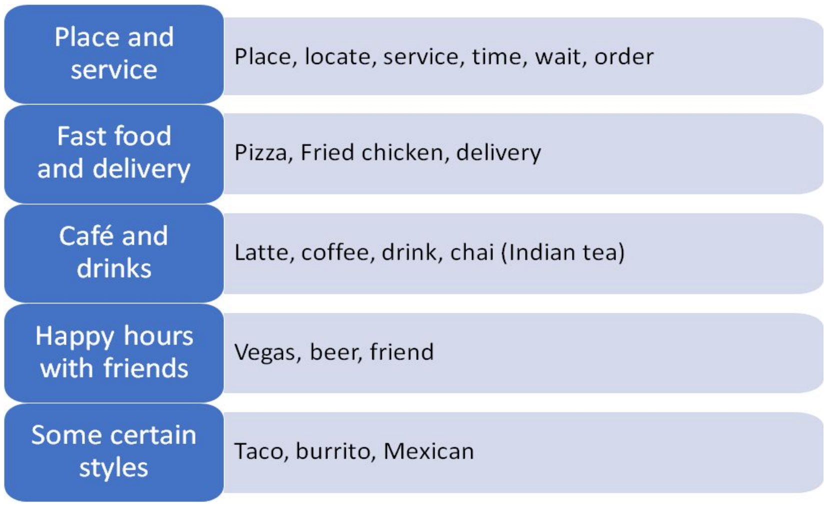


Fig. 5 Word cloud of frequently used words in reviews

After removing stopwords, we lemmatized the words so that similar words could be regarded as the same. The topic model, LDA (Latent Dirichlet Allocation) was used to generate the topics and corresponding keywords. The result of LDA is as follows. It gave us some aspects that the customers cared about and could be points that business should focus on if they wanted to improve the management.



Place and service	Place, locate, service, time, wait, order
Fast food and delivery	Pizza, Fried chicken, delivery
Café and drinks	Latte, coffee, drink, chai (Indian tea)
Happy hours with friends	Vegas, beer, friend
Some certain styles	Taco, burrito, Mexican

Fig. 6 Topics and associated keywords from the topic model - LDA

We used TFIDF to get features from the keywords, which was more efficient and more effective than just using a bag-of-words approach. TFIDF, short for term frequency-inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. TFIDF of the 1000 most frequent words were chosen as features to be used. Since each row represented a review, we aggregated the data by business ID. Since there were still 1000 words, which meant we had 1000 columns, we used Principal Component Analysis to reduce the dimension. 30 principal components were extracted and we joined the preprocessed review data with the business information data.

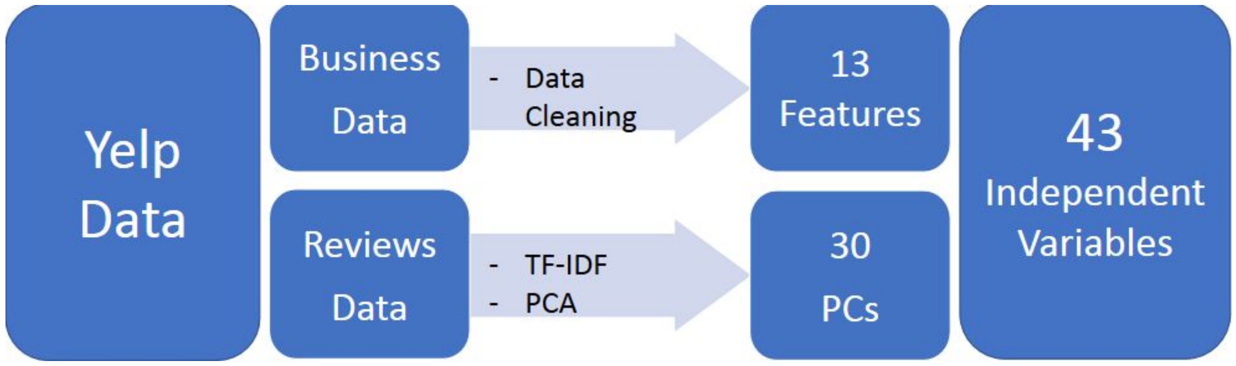


Fig. 7 Data preprocessing procedure

It is true that PCA was a good way to reduce the dimension, but another approach might be choosing fewer words when we were extracting features. Therefore, after the presentation, in addition to principal components of 1000 words, we also tried TFIDF of 100 words without dimension reduction. The reason why we did so was that we could reduce dimension, and at the same time make our models and results more meaningful by directly using the readable words. This might make us lose some information from the text, since only 100 words were chosen, but we still needed to check the model performance to get a clear conclusion on this problem. The results were also presented in the model performance section.

EDA and Descriptive Statistics

In this portion, we are going to discuss some exploratory data analysis and descriptive statistics that can help us get a brief understanding of the relationships between the independent variables and the dependent variable.

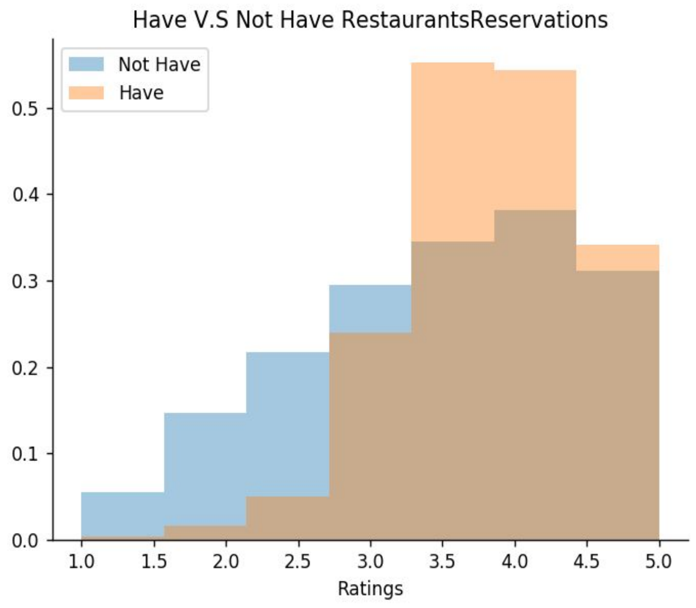


Fig. 8 Distributions of ratings against reservations

Figure 8 shows how the distribution of ratings varied from whether the restaurant supported reservations. For restaurants having reservations, the ratings were more concentrated between 3.5 and 4.5, while restaurants without reservations had a lower distribution.

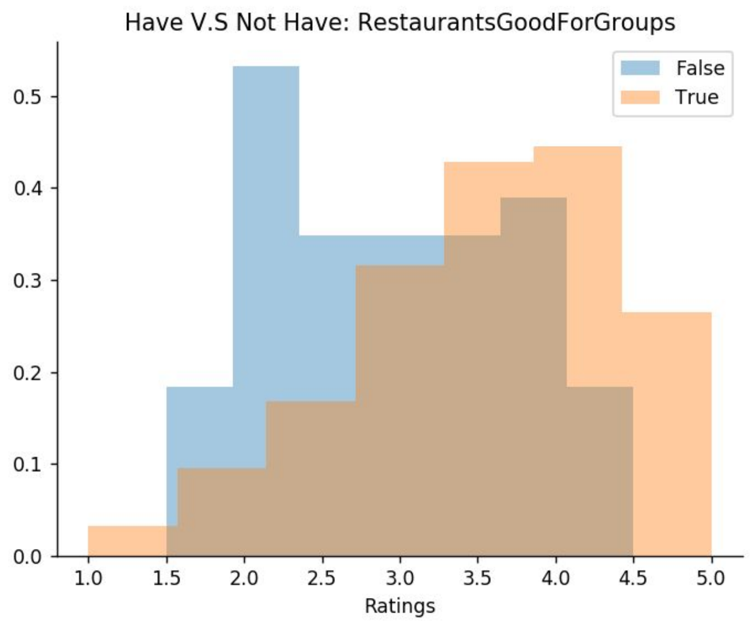


Fig. 9 Distributions of ratings against GoodForGroups

Figure 9 shows how the distribution of ratings varied from whether the restaurant was good for groups. For restaurants good for groups, the ratings were more concentrated between 3.5 and 4.5, while restaurants not good for groups had a lower distribution and their ratings had a peak at 2.0.

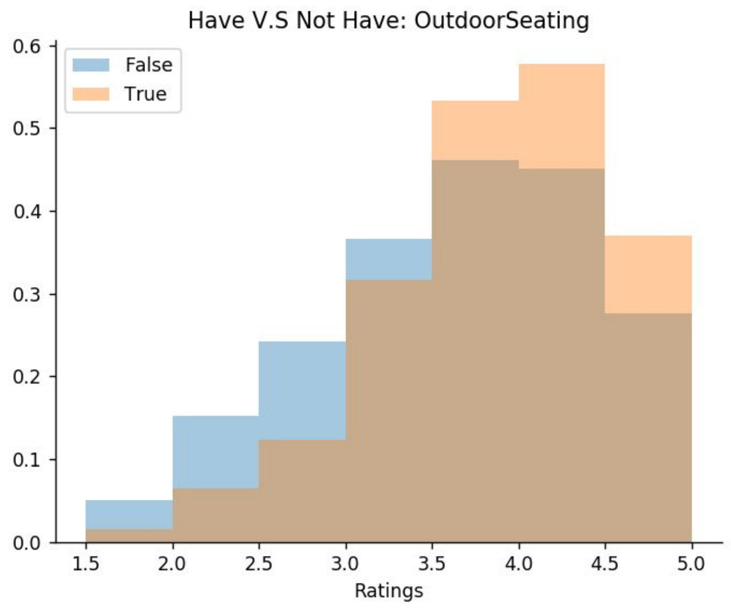


Fig. 10 Distributions of ratings against OutdoorSeating

Figure 10 shows how the distribution of ratings varied from whether the restaurant had outdoor seatings. For restaurants having outdoor seatings, the ratings were more concentrated between 3.5 and 4.5, while restaurants without outdoor seatings had a lower distribution.

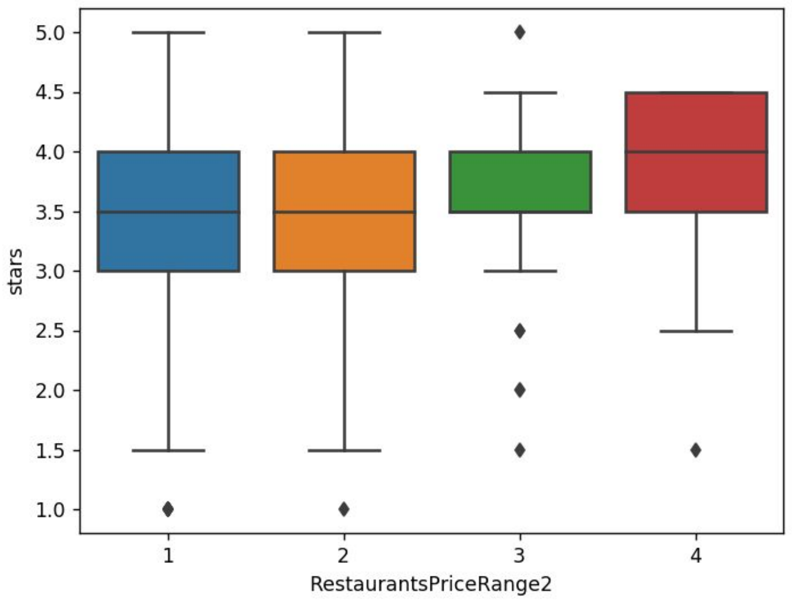


Fig. 11 Boxplot of ratings against pricing ranges

The boxplot in Figure 11 shows how the distribution of ratings varied against the restaurants' price ranges. The restaurants having the highest pricing level tended to get a higher rating, which makes sense because people are willing to pay a premium for better restaurants. We could not see obvious differences between the other 3 price ranges.

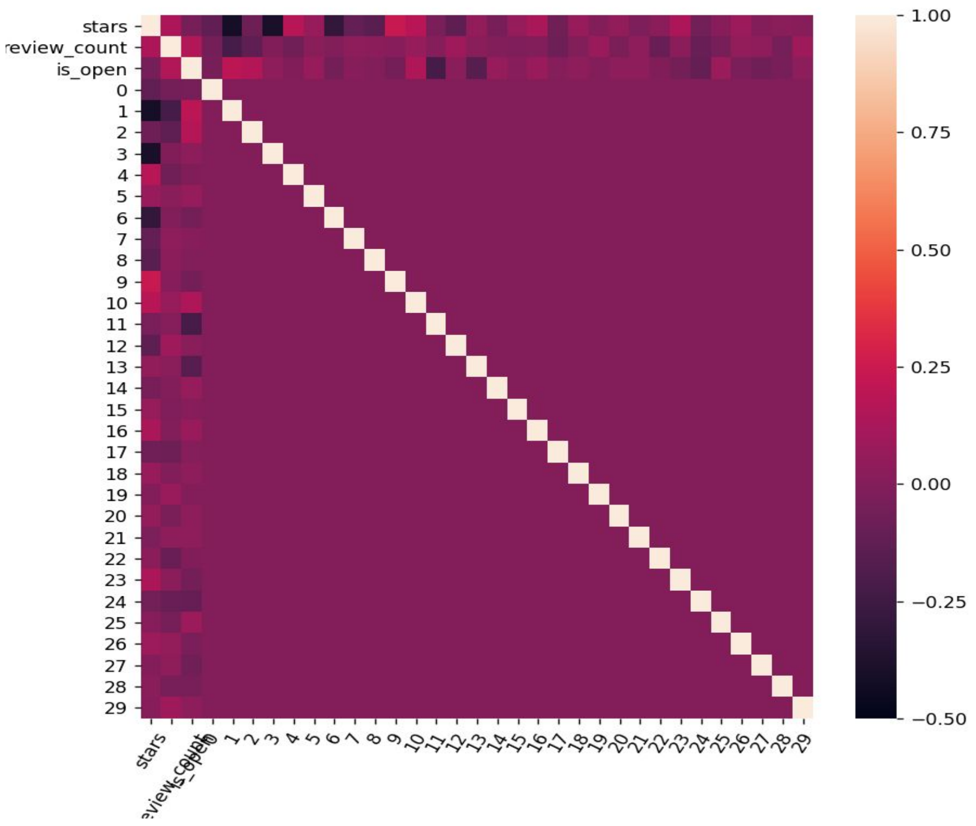


Fig. 12 Correlation matrix heatmap

The correlation heatmap in figure 12 shows the linear correlation between all the quantitative variables. Stars had a linear relationship with quite a few independent variables like the 1st, 3rd and 6th principal components. There was a large purple area at the bottom right part, which makes sense because they are principal components and should be independent with each other.

Model Understanding

Our goal for the models was to predict the star ratings based on the features. The data was all in the past and what we were predicting was also based on the business features and reviews that we already had, so we were not forecasting the future data. Rather than that, we split our dataset into train and test groups, to try to infer conclusions on the latter group based on the model fit to the former.

Model Selection

Since the dependent variable (star rating) was categorized into two classes: equal or above 4 stars and lower than 4 stars, which was a discrete variable, we needed models for binary classification. We first selected five models for binary classification – logistic regression, random forest, gradient boosting tree, support vector machine, and neural network, and implemented them in Microsoft Azure Machine Learning Studio. The diagram for modeling is shown in figure 13.

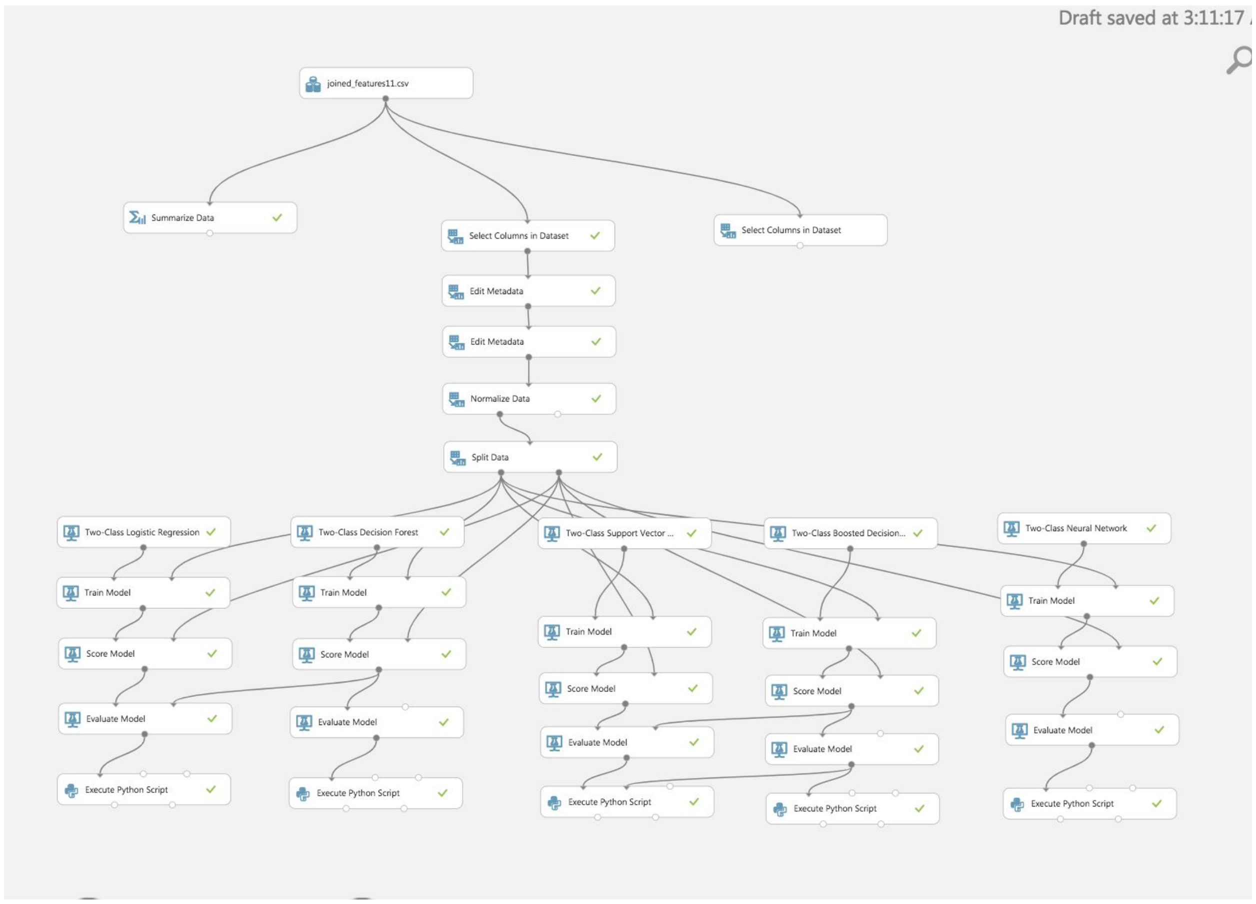


Fig. 13 Diagram for modeling in Azure

Our goal for this step was to compare the performance of these models under default parameter settings to figure out one or two outstanding models. The measurements for model performance would be ROC-AUC score and accuracy. As the data set was not balanced with around 60% of the target variable was equal or above 4 stars and the other 40% was below 4 stars, ROC-AUC score would be the best metric for this imbalanced binary classification. This resulted in that ROC-AUC helped us not to overfit to a single class given the skewed sample distribution. Visualizing the ROC curve also let us see the tradeoff between sensitivity and specificity for all possible thresholds rather than just the one that was chosen by the modeling technique. Additionally, accuracy was an easily interpretable measurement, so we also tool accuracy into account for the logic to model selection. We would compare the ROC-AUC scores and the accuracies of these five models in order to select out one or two best-performing models. Then we would tune the parameters of these selected models for better performance in terms of ROC-AUC score and accuracy. Compared with the external baseline accuracy of 0.63 for SVM, our models had more than 0.1 value-add. The summary of the metrics of the five models is shown below in Table 2.

Model	Accuracy	Precision	Recall	F1-score	ROC AUC
Logistic Reg.	0.800	0.806	0.793	0.805	0.809
Random Forest	0.796	0.801	0.705	0.781	0.780
SVM	0.752	0.769	0.722	0.753	0.715
Boosted DT	0.781	0.804	0.776	0.784	0.820
Neural Network	0.817	0.794	0.761	0.776	0.800

Table 2 Model evaluation metrics

Descriptive Statistics

There were 2013 rows of data and 44 features including 14 business features and 30 principal components from the reviews. Among the 14 business features, excluding the “business_id”, only the “review_count” was a numeric variable with an extremely right-skewed distribution. The mean of “review_count” was 140.692, the median was 36, the mode was 3, and the standard deviation was 336.9634. Its min was 3 and max was 6708, which range was very large, so that it needed standardization to zero mean and unit variance before being put into the models. Among the other 12 categorical business features, the “city” feature had 14 unique values: Las Vegas had the dominant frequency, and Henderson had the second highest, followed by North Las Vegas. The distribution was shown as figure 14 below.

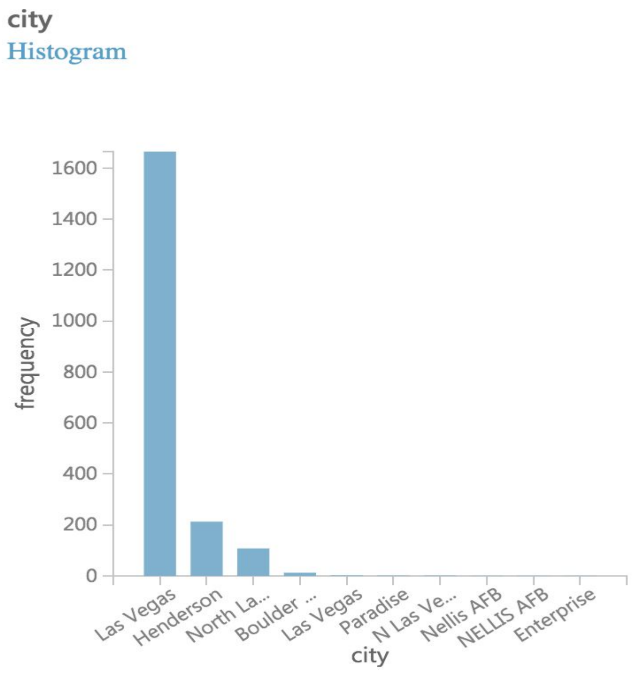


Fig. 14 Distribution of city frequencies

The other business features were all binary, including is_open, "RestaurantsReservations", "RestaurantsGoodForGroups", "BikeParking", "OutdoorSeating", "RestaurantsTakeOut", "BusinessAcceptsCreditCards", "GoodForKids", "RestaurantsDelivery", and "RestaurantsAttire". "RestaurantsPriceRange" had 4 distinct values from 1 to 4, so was also treated as a categorical feature.

The 30 principal components were all numeric variables with different means, medians, standard deviations, and ranges. The distributions of most of them were nearly normal, and the others were skewed. For instance, as figures 15 and 16 show below, the 23rd PC was nearly normal distributed with mean 0.0001, median -0.001, and standard deviation 0.0175, whereas the first PC had a skewed distribution with mean -0.0021, median -0.0145, and standard deviation 0.0635. Therefore, these features also needed to standardize to zero mean and unit variance before modeling, since many models were sensitive to scaling.

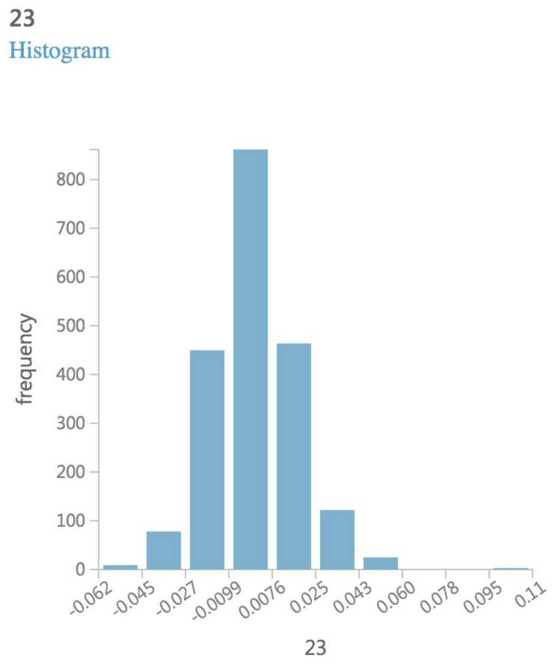


Fig. 15 Distribution of the 23rd PC

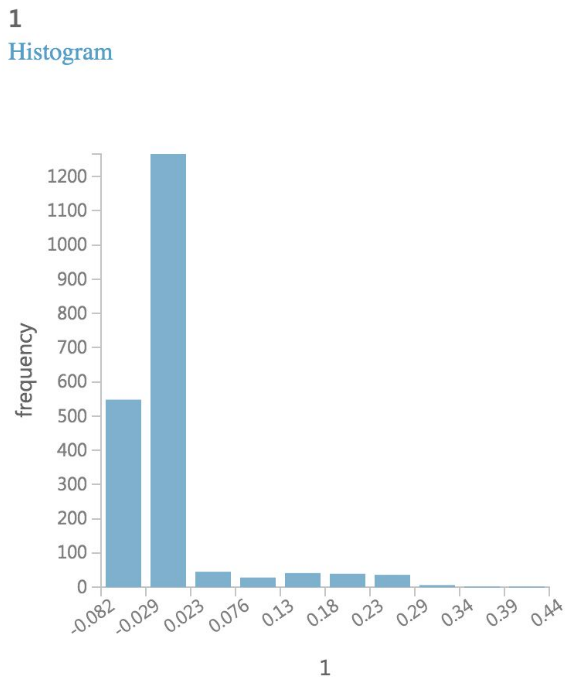


Fig. 16 Distribution of the 1st PC

Model Implementation

The data were randomly split into 75% for training and 25% for testing, stratified on the dependent variable. To train our classification prediction models, we used five supervised learning algorithms, including logistic regression, random forest, gradient boosting tree, support vector machine, and neural network.

In logistic regression, it models each y_i as independently generated, with $P(y_i = +1|x_i, w)$ = sigma(x^T_i w), where sigma(x_i; w) is the sigmoid function which maps x to P(y = +1|x). Here, y_i was each rating, and x_i was the feature vector. y = +1 meant the rating was equal or above 4 stars, and y = -1 stood for a below-4 rating. This is a discriminative classifier because x was not directly modeled. The loss for logistic regression is log loss. When predicting a new star rating, if $x^Tw > 0$, then sigma(x^T w) > ½ and then predicted y = +1, which was >= 4 stars, and vice versa. In Azure, the logistic regression was penalized by both L1 and L2 regularizations by default. The default weights of the regularizations were both 1. As the regularization weight increases, the weight on loss decreases, and the model fits faster since it becomes simpler. After training the logistic regression model, the top 16 features with the largest weights were all principal component features, followed by review_count, then another four principal components.

Support vector machine (SVM) aims to find a hyperplane (the decision boundary) such that its distance to the closest point in each class is maximized. Linear SVM has a solution only when the two classes are linearly separable. If the classes are not linearly separable, a soft-margin SVM permits training data to be on the wrong side of the hyperplane, but at a cost. Different from logistic regression, the loss for SVM is hinge loss. The only parameter for SVM is C. As C decreases, the margin becomes wider, less emphasis would be on data fitting, more points would be included inside the margin, and more points would influence the solution. The weight for L1 regularization was set to 0.001 by default in Azure to avoid overfitting the model to the training dataset. After training the SVM model, the top 17 features with the largest weights were all principal component features, followed by BusinessAcceptsCreditCards, then another two principal components.

The idea for decision trees is to loop through all possible thresholds for all possible feature to find the best "questions" for each split in order to minimize impurity. Two of the drawbacks of decision trees are that trees were unstable and they tend to overfit as going deeper or growing larger. One solution to this is ensemble models, which combines multiple trees and averages over this collection of classifiers. One way of ensemble is bagging that stood for bootstrap aggregation, the idea of which is sampling 66% of the data with replacement each time. Bagging reduces the variance of the model so that the model becomes more stable. Random forest uses the technique of bagging and is randomized in two ways: for each tree, it picks a bootstrap sample of data; for each split, it picks a random sample of features without replacement. Bagging and sampling help to reduce the correlation between decision trees and avoid overfitting. Random forest is also more insensitive to outliers compared with decision trees. The parameters for a random forest are the number of decision trees, the maximum depth of the trees, the number of random splits per node, and the minimum number of samples per leaf node. The default parameters were set to be 8 decision trees, maximum 32 levels for the trees, 12 random splits per node, and minimum 1 sample per leaf node. After training the random forest model, 8 decision trees were built, one of which was visualized as below in figure 17.

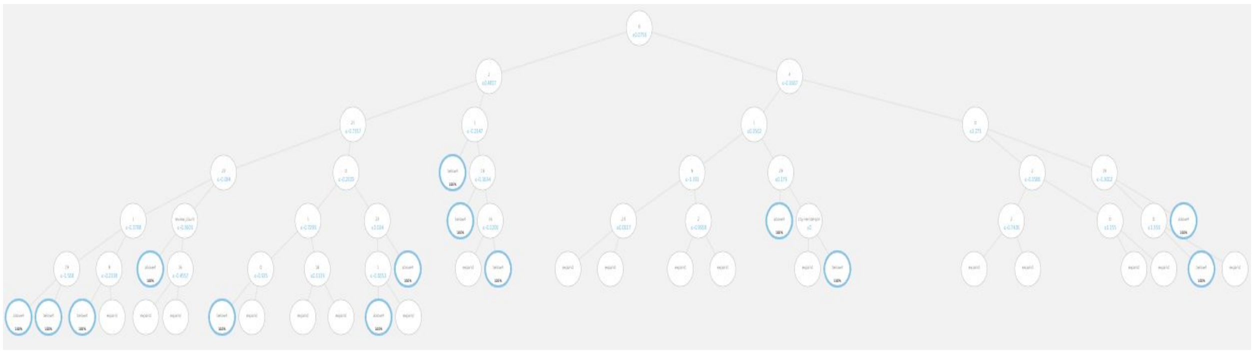


Fig. 17 Visualization for the 1st tree in the Random Forest model

Another ensemble technique is boosting – weighted vote with a collection of classifiers. The gradient boosting tree algorithm also aims to keep constructing trees with low bias. Unlike random forest that builds multiple trees simultaneously, boosting constructs trees sequentially, each tree depending on the previous one. In particular, gradient boosting algorithm uses gradient descent method to optimize the loss function to correct the residual errors of the existing sequence of trees. For example, set the first function to be $f_1(x) = y$, then $f_2(x) = y – \text{gamma } f_1(x)$, and $f_3(x) = y – \text{gamma } f_1(x) – \text{gamma } f_2(x)$ where gamma is the learning rate. Each f(x) is a weak learner to make predictions, and in the end, an additive model adds the weak learners to minimize the loss function. Since the gradient boosting algorithm tends to overfit a training dataset quickly, the trees are constrained in multiple ways in order to improve the performance of the algorithm by reducing overfitting. In Azure by default, these constraint parameters were set to be maximum 20 leaves per tree, minimum 10 training instances required to form a leaf, and 100 trees to construct in total. Also, the default learning rate was set at 0.2. a learning rate of less than 1.0 has the effect of making fewer corrections for each tree added to the model, which results in that more trees must be added. There are many advantages of gradient boosting. Firstly, although it is slower to train than random forest, it would be much faster to predict due to not as deep and not as many trees as in random forest. Additionally, gradient boosting is typically more accurate than random forest. Also, the model size is small. After training the gradient boosting tree model, 100 trees were built, one of which was visualized as below figure 18. The size of this tree appeared to be much smaller than that of the tree of random forest in figure 17. The feature importances will be discussed in the later Model Performance section.

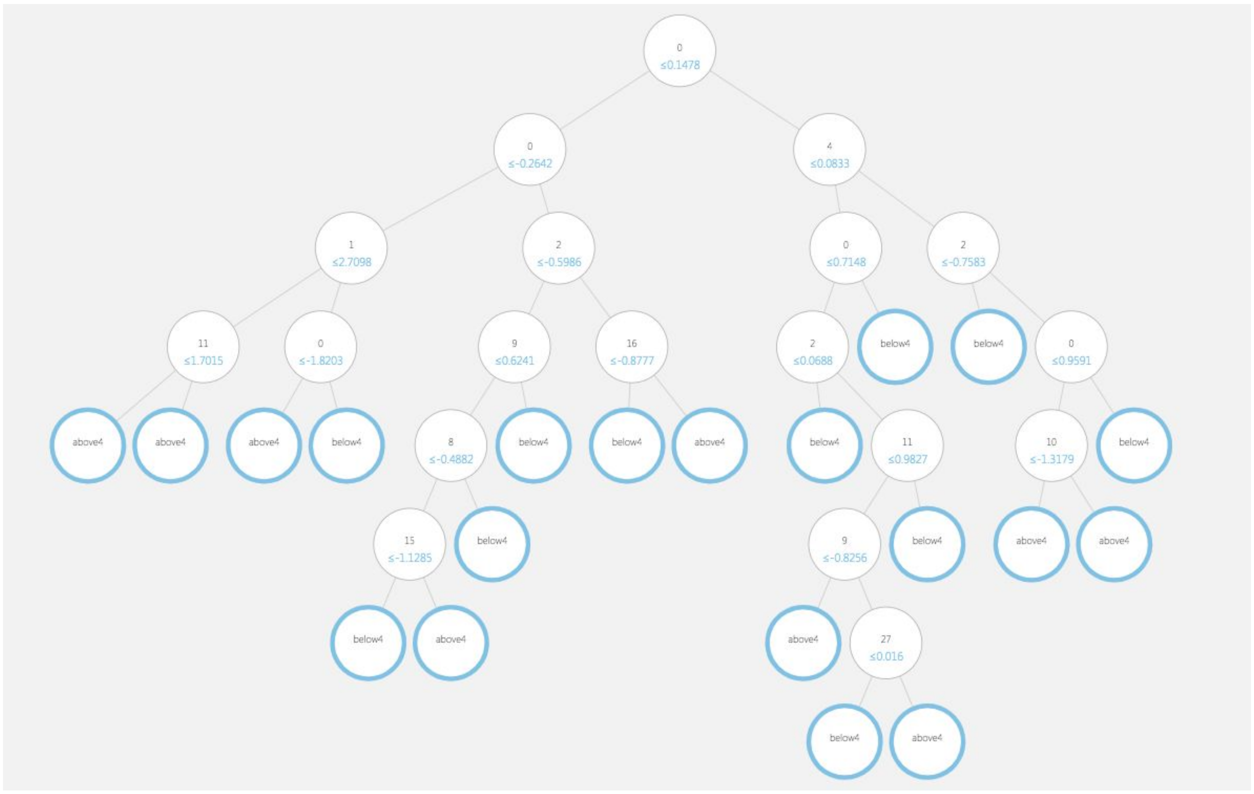


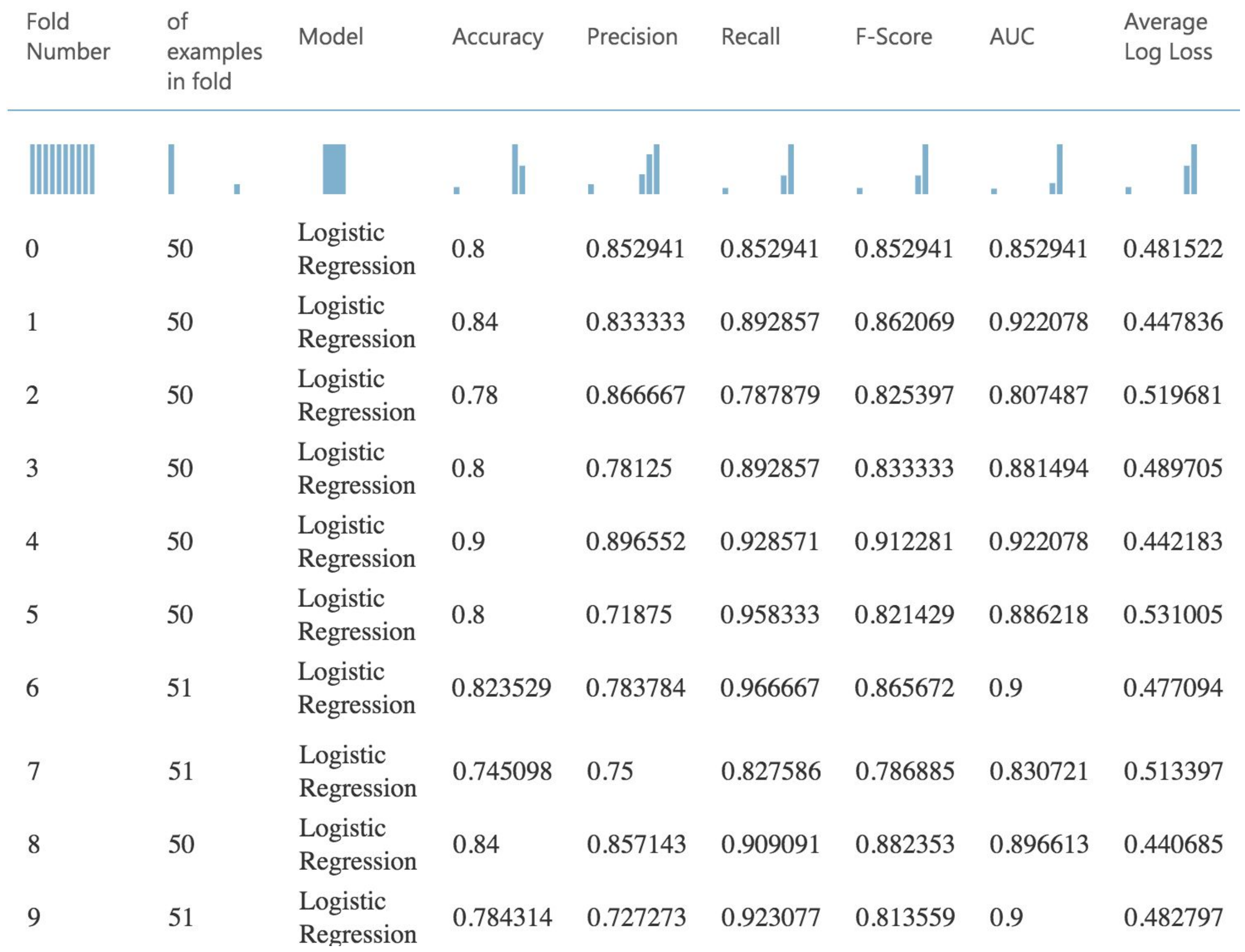
Fig. 18 Visualization for the 1st tree in the GBDT model

The last algorithm we used was a supervised neural network. Neural networks are non-linear model for classification and work well for very large datasets. The structure of a vanilla neural network consists of an input layer, one or more hidden layers, each with a nonlinear activation function, and an output layer that outputs decisions with softmax function. Similar to SVM or linear models, neural networks requires data preprocessing as they are sensitive to scaling. There are many variants of neural networks nowadays, including convolutional nets, residual nets, and recursive nets. We used a naïve neural network with fully-connected hidden layers for this project. In Azure, there were 100 hidden nodes for each hidden layer, the learning rate was 0.1, there were 100 learning iterations, and the initial learning weights diameter was 0.1 by default.

More hidden units in each hidden layer would result in a smoother line segment, i.e. decision boundary. These parameters could be tuned to achieve a high training score. However, one drawback of neural networks was that it was hard to interpret and visualize.

Cross-validation

In training the models, since we did not want to overfit to the training dataset, we utilized the technique of k-fold cross-validation. It partitioned the training data into k folds (k = 10 by default in Azure). In each iteration, it trained the model using the data of the k-1 folds, and then validated on the remaining validation fold. During testing of the model for each iteration, multiple accuracy statistics were evaluated. One of the advantages of cross-validation over simply splitting data into training and testing data was that we were able to get more metrics, 10 scores in this scenario, so the variance of the resulting estimate was reduced. After cross-validation, we could see the predicted value for each row the estimated probability of the value for each row. We also visualized the evaluation results by fold as the followed Fig. 19. It shows the evaluation metrics for each fold.



Fold Number	of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss
0	50	Logistic Regression	0.8	0.852941	0.852941	0.852941	0.852941	0.481522
1	50	Logistic Regression	0.84	0.833333	0.892857	0.862069	0.922078	0.447836
2	50	Logistic Regression	0.78	0.866667	0.787879	0.825397	0.807487	0.519681
3	50	Logistic Regression	0.8	0.78125	0.892857	0.833333	0.881494	0.489705
4	50	Logistic Regression	0.9	0.896552	0.928571	0.912281	0.922078	0.442183
5	50	Logistic Regression	0.8	0.71875	0.958333	0.821429	0.886218	0.531005
6	51	Logistic Regression	0.823529	0.783784	0.966667	0.865672	0.9	0.477094
7	51	Logistic Regression	0.745098	0.75	0.827586	0.786885	0.830721	0.513397
8	50	Logistic Regression	0.84	0.857143	0.909091	0.882353	0.896613	0.440685
9	51	Logistic Regression	0.784314	0.727273	0.923077	0.813559	0.9	0.482797

Fig. 19 Cross-validation evaluation metrics for Logistic Regression model

Model Performance

Our goal through this project was to create a classification model that could be directly applied to restaurants in Yelp, and could thereby help business owners and entrepreneurs run their companies successfully. By using statistical models to draw information from the words contained in the reviews, we were able to effectively develop a model that created a 20% value add above our baseline.

There was an additional outcome from this project that we noticed could be achieved: namely, being able to include the most important terms in a functional model would work as a guide for restauranteurs to understand the elements of service that were relevant to customers.

For this reason, we decided to draw a comparison: what would be the accuracy of either model if we were to change the Principal Components that represent the more than 5,000 words on the dictionary for a list of the most used words? The results of the comparison were resumed as follows:

	PCA	Words
Logistic Regression	0.799	0.806
Gradient Boosting	0.809	0.804

Table 2 Comparison of accuracies for models under different feature methods

Table 2 shows the comparisons between the accuracies of models under different features. Logistic Regression has an accuracy of 79.9% under principal components, and a bit higher (80.6%) using words features. Gradient Boosting achieves 80.9% on principal components, and a bit lower at 80.4% using words features. The results are quite similar to each other, while Gradient Boosting under principal components achieves the highest accuracy among the four. There is no large difference between using principal components or words features to build the model, and logistic regression and gradient boosting also achieve similar results. 100 words appear to provide similar information as the principal components.

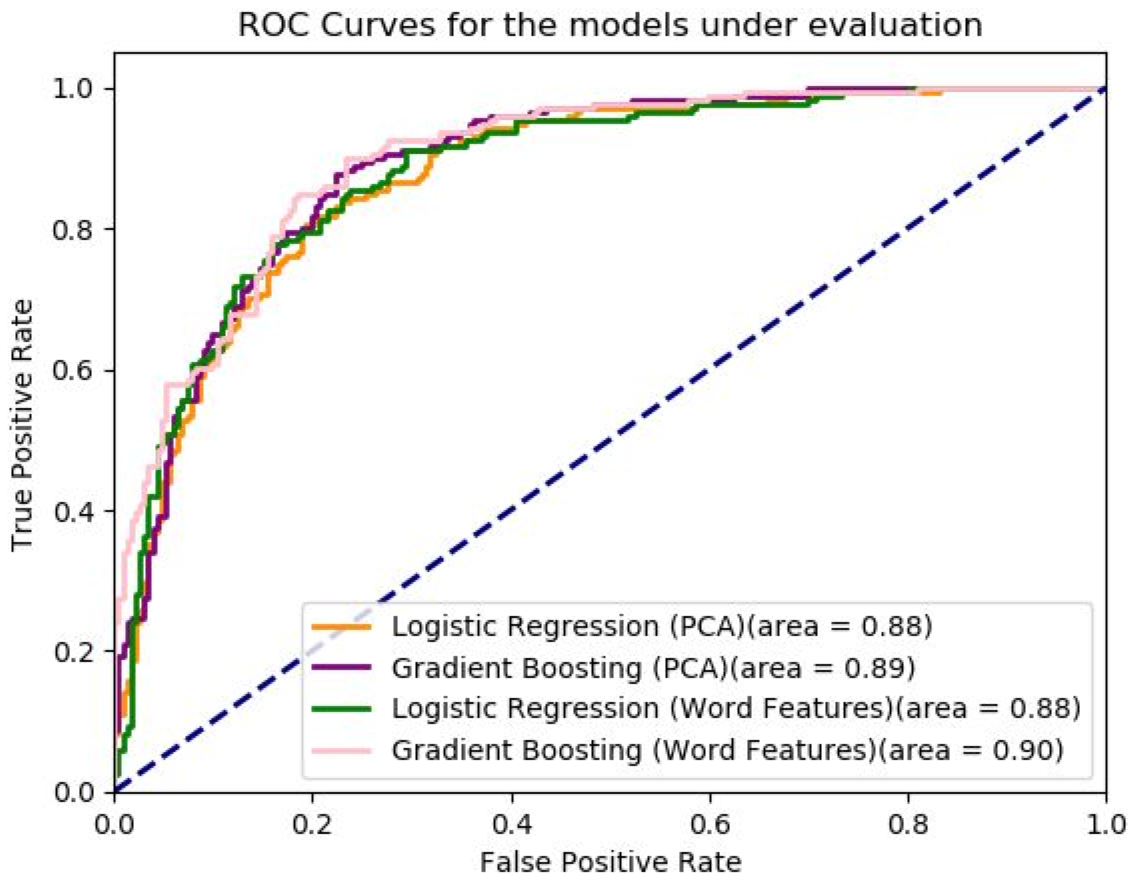


Fig. 19 ROC curves for the models under consideration after optimizing their hyperparameters

As we can see from the ROC curve graph, after tuning parameters for all the models, the ROC curves render similar results, which aligns with the results in the model accuracy table. There are still some differences between models. Gradient boosting on word features achieved the largest AUC value and its ROC curve is also closest to the top left corner, which means that under different thresholds this model tends to have a better prediction performance.

The formula of the AUC curve is the following:

$$A = \int_{x=0}^{1} \mathrm{TPR}(\mathrm{FPR}^{-1}(x))\,dx = \int_{\infty}^{-\infty} \mathrm{TPR}(T)\mathrm{FPR}'(T)\,dT = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(T' > T) f_1(T') f_0(T)\,dT'\,dT = P(X_1 > X_0)$$

The results are not exactly the same as our expectations, with four models having similar performance and word features actually getting a better one. Even though we just used 100 words as the word features, they turned out to well classify the ratings. To further explore what words or variables are helping us with classification, we extracted feature importances from Gradient Boosting and the absolute values from Logistic Regression.

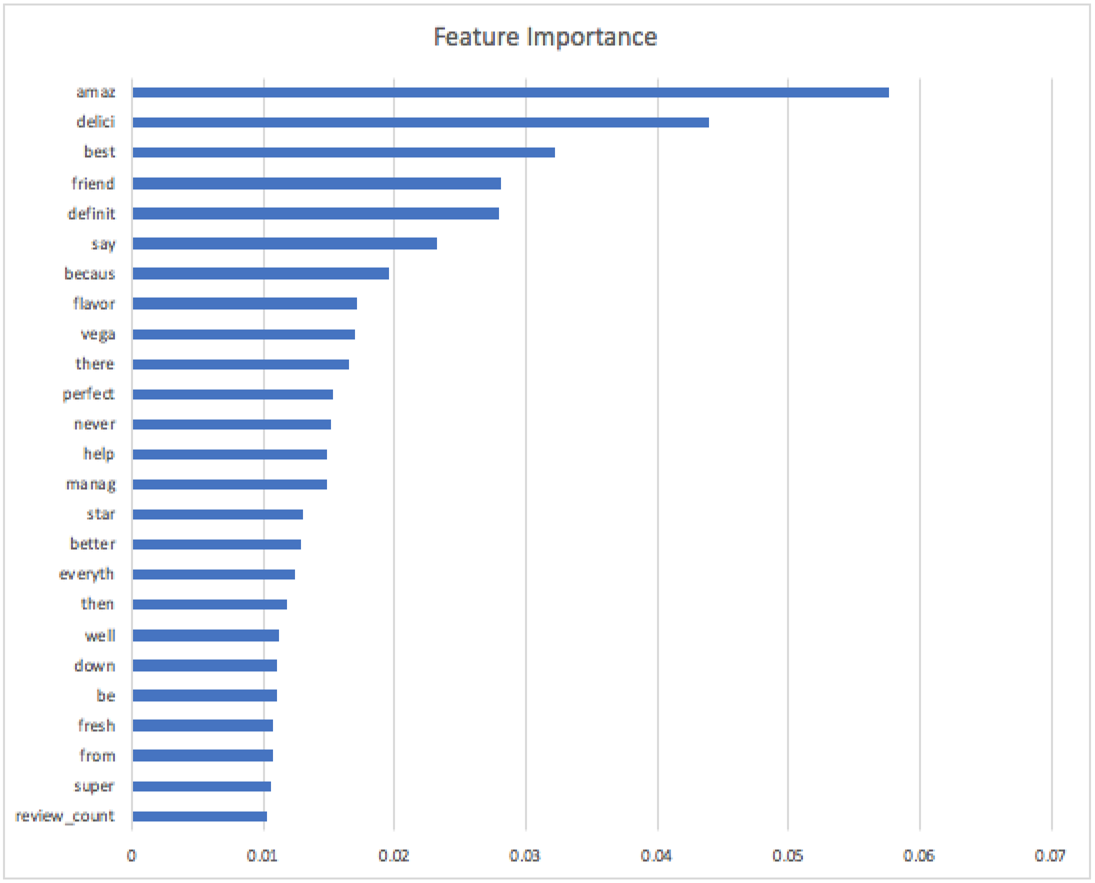


Fig. 20 Feature importances under a Gradient Boosting model for the 25 most relevant variables

The feature importances graph above shows how much different variables matter in Gradient Boosting model. Feature importance, in this case, is defined as a total decrease in node impurity weighted by the probability of reaching that node averaged over all trees of the ensemble.

Out of the 25 most important variables, only 'review_count' is a feature specific to a business in Yelp. The rest of the important variables all come from the reviews and are presented as lemmatized keywords here, which means the information from reviews dominates in the model. Some lemmatized words really make sense and can imply a positive evaluation from the customers. For example, "amaz", "delici", "best", "perfect" and "fresh". Negative words are not so obvious in this case.

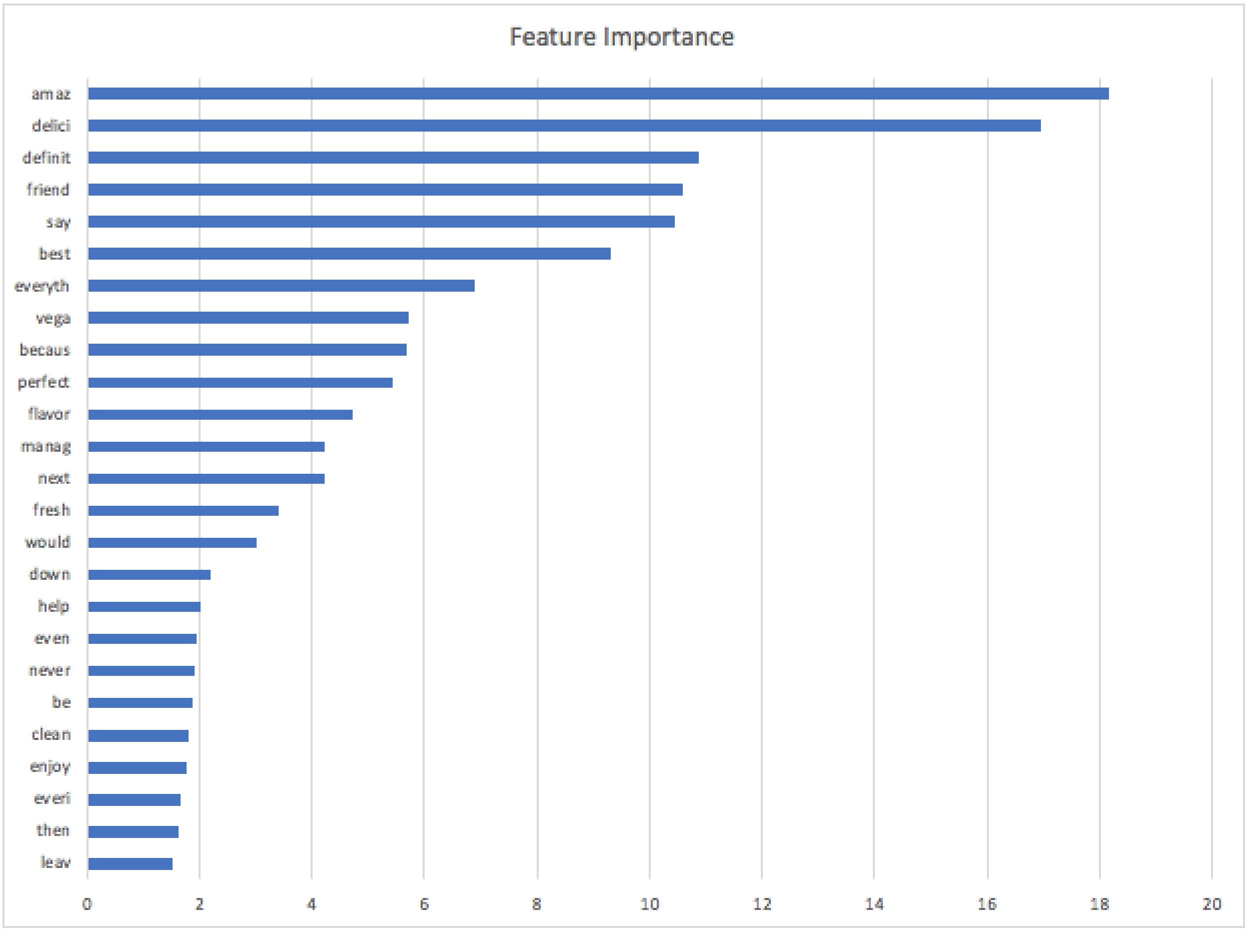


Fig. 21 Absolute value of Feature Importances under a Logistic Regression model for the 25 most relevant variables

In the case of the logistic regression model, we calculated the absolute value of the coefficients in order to identify the most relevant words for the model. All of the important variables here are just words and there is no variable from business information. Some keywords here are quite similar to what we got in the Gradient Boosting Model. This can actually be used to create a simple lexicon for sentiment analysis that we trained based on Yelp reviews and ratings. If we increase the number of keywords, a larger lexicon can be generated.