A CASE STUDY ON YELP DATASET: FEATURE ANALYSIS OF HIGH-RATED RESTAURANTS

Prepared by: Ahmedur Rahman
Supervisor: Can Kavaklioglu
CKME 136 Capstone Project FALL 2018
Ryerson University

Table of Contents

Table of Figures

Table of Tables

Abstract

The goal of this case study project is to explore the viability of classification machine learning algorithms on Yelp dataset, which consists of a huge set of restaurant's data on their business attributes (i.e. hours, parking availability, price ranges, cuisine choices, start rating etc.). Our primary goal is to find the important features of a restaurant which could contribute directly towards achieving high rating on Yelp website. An opensource dataset was obtained from Yelp website and then we preprocessed and cleaned the data using python and R. Then we applied three feature selection techniques and used the selected features from each feature selection techniques with three classifiers (Logistic Regression, Random Forest and K-Nearest Neighbor) to build the models. After that we compared the performance of each classifier models to find out the best model with the highest accuracy. Later we test each model for their significance differences.

Introduction

Yelp has always been a trusted source for finding high rated restaurants around the places. Their dataset provides a huge list of data containing each restaurant's features and their ratings along with some user details, review details and check-in details. A lot of studies have been conducted to do the sentiment analysis on review text and predicting the star rating. But none of them are focused on if there is a relationship between high rating and any of the restaurant features. In this report we will try to analyze which features of a restaurant are more likely to contribute directly towards higher ratings; this result could become a great source for business owners or entrepreneurs to improve their ratings.

The theme for this project is Classification and Regression. The Yelp Dataset contains over 1.4 million business attributes like hours, parking, availability, and ambience for 188,593 businesses in 10 metropolitan areas.

We used Python and R language to preprocess the data. Then we build three classifier models using different methods (Logistic Regression/ KNN/ Random Forest) to find out important features (i.e. hours/parking/availability/location etc.) of a business to get higher rating. After that I am planning to mine association rules on high-rated businesses to get more insight about the features and how they are correlated. At a later point, I plan to run some statistical test (i.e. ANOVA) to find the significant difference between classifier models built earlier.

Literature Review

Yelp dataset is one of the publicly available dataset which comprises a comprehensive set of restaurant features, their ratings and user comments. This dataset has been used in many researches to find valuable insight from within the restaurant industry in together with various data mining and machine learning techniques.

In [CFV14], the project involved investigating potential factors that might affect the business performance. The study first selected several features that came directly from the dataset like latitude, longitude, review count, price range, accepts credit cards, has take-out, has delivery etc. Then they generated additional features using clustering on latitude and longitude to determine the location of the restaurant (whether it is in downtown, or in shopping mall etc.). The features they have derived from k-means clustering are cluster size and cluster label. Then they have used Naïve Bayes Classifier for review sentiment analysis and derived 10 features (i.e. Horrible, Awful, Disappointed, Waited etc.). After feature selection they have used various methods of feature selections (Univariate Feature Selection, Recursive Feature Elimination and Tree Based Feature Selection). Then they used these features from each method separately to several predictive models (Support Vector Machine, Multinomial Logistic Regression, Multinomial Naïve Bayes, Gaussian Discriminant Analysis, Decision Trees and Random Forest Classifier). Then they presented top 5 features generated from each model as follows:

Method Used	**Top 5 Features**
Recursive Feature Selection with Logistic Regression	Sentiment, Reservation, latitude, longitude, Review Count
Univariate Feature Selection with SVM	Sentiment, No. of reviews, Cluster Size, Reservations, Longitude
Tree based feature selection with SVM	Sentiment, Latitude, Longitude, No. of Reviews, Cluster Size

In a study [A16] on Yelp Dataset Challenge a system or model has been proposed to predict the rating of a restaurant based on its review text. The study noted that when a user gives a rating to a restaurant, it is not possible for a user to consider all aspects and many times user assign an overall star rating solely based on a specific aspect. Therefore, it is important to have a rating prediction system which will show the rating based on user review text. In this study, they used four different feature extraction methods together with 4 supervised learning algorithms; creating a total of 16 different prediction models. They used 80% training data and 20% to validate results. Four feature extraction methods that they have used: Unigrams, Unigrams & Bigrams, Unigram Bigram & Trigram and Latent Semantic Indexing (LSI). Four supervised learning they have used: Logistic Regression, Naïve Bayes Classification, Perceptrons (a linear classifier that outputs class labels instead of probabilities) and Linear Support Vector Classification (SVC). Then they compared the result of all 16 models and found that Logistic Regression achieved the highest accuracy of 64% using the top 10000 Unigrams & Bigrams as features. The second best performing system was Linear SVC (63% accuracy) using top 10000 Unigrams & Bigrams. After that they test these two models using the test dataset, they found that Linear SVC achieved 56% accuracy and Logistic Regression achieved 54% accuracy, which are lower than the validation-fold scores indication possible overfitting. So, they proposed to add/ adjust regularization parameters as their future work.

In the study of [P17], a novel approach has been proposed to predict Yelp restaurant rating not only based on business features (i.e. images, descriptions) and user features (i.e. average previous ratings) but also considering the network features (i.e. which businesses a user has rated before). In their study, they demonstrated that a mixed approach combining node level features with network information can lead to a better Yelp star rating prediction. They split their data into training (80%), validation (10%) and test (10%) dataset and applied Linear Regression, Ridge Regression, Bayesian Regression, Neural Network and Random Forest. Although in training set Random Forest had the lowest RMSE, but in both validation and test set, Neural Network outperformed as the best supervised learning method with the lowest RMSE indicating that Random Forest seems to be overfitting the data.

Another study [FK14] was done to predict the Yelp rating solely based on its review text. In their study they have used three feature selection methods and four learning methods (Linear Regression, Support Vector Regression, Support Vector Regression with normalized features and Decision Tree Regression).

The first feature selection technique was analyzing raw data and choosing top K-frequent words used in all reviews and counting their frequency. In the second feature selection method they analyzed Part-of-Speech (POS) per sentence. Based on the analysis, they selected top K frequent words amongst all. In the third feature selection technique they have extracted top K frequent adjectives. For each feature selection techniques, they have used K = 30,50,100,200,300,500,1000 and selected the one with least RMSE. In their study they found that no matter what feature selection they choose, Linear Regression always performs better than other learning model and within the feature selection techniques, Top Frequent Words from Raw Data with Linear Regression had the lowest RMSE.

In [LLJ+11], authors have argued that just doing a sentiment analysis on the review text is not enough because same text could carry different sentiments for different reviewers. Therefore, it is necessary to scale the sentiment based on reviewer and product specifics. In their study they have designed a predictor function to scale that. Then they compared their model against with Regression, Regression + PSP [PL05], SVM and SVM+PSP. They showed that their model outperformed rest of the model with the least RMSE.

In another study [K17] on Yelp dataset, the author has proposed a model which is a hybrid of neural network, decision tree and logistic regression to predict the review rating given the location (longitude, latitude) and business category. This model predicts if a new restaurant is opened in a certain location, whether it will belong to positive class (start rating greater or equal to 4) or negative class (less than 4). They have used a feature engineering technique called Extremely Randomized Trees [GEW06]. Extremely Randomized Trees is a tree induction algorithm that selects splits both attribute and cut-point totally or partially at random and in extreme cases it is capable of building totally randomized tree, whose structure is independent of output values of learning sample.. Then they compared their model with the results from Random Forest and Neural Network and showed that this model outperforms other models with a accuracy of 67.3% and highest Area Under Curve (AUC) of 73%.

Another study [SC00] was done on 63 Toronto restaurants to find out which attributes or features have statistically significant effect on customers' ratings. The source of their data was from Zagat Survey and Toronto Life Magazine. Then they surveyed those restaurants' managers to get data about average check, restaurant volume and restaurant size. All the restaurants used in the study were similar in size and volume. The study basically focused on nine attributes (i.e. Food, Décor, Service etc.) and finding out which attribute contributes most to the average check. They gathered the average check data from Zagat Foundation (which collects data from customers) and then they validated the information directly with the restaurants. They found it to be largely consistent. The correlation between the average check reported by the foundation and managers was 0.78, which indicates the strong level of agreement between two parties. The methodology they have used to find out important features that affect positively or negatively is Two-tailed correlation significance with α = .05 level. Some interesting findings from their study are as follows:

- 81% of their sample's high rated restaurant had smoking section, 63% offered catering and 54% offered takeout.
- Three features that were not offered by all but still showed strong presence: dress code, parking and outside dining.
- Three uncommon attributes: internet presence, late-night menu and some sort of entertainment (i.e. music, dancing, comedy)

Dataset

For this project, we are using the Yelp Open Dataset [Y18]. The dataset contains information about 188,593 businesses in 10 metropolitan areas consisting of almost 6 million user reviews and 280,992 pictures.

Our research question is to identify important business features that significantly contributes towards the higher ratings (3.5 or more start ratings). So, our dataset will be limited to business dataset. Yelp's business dataset is in JSON format and contains the following:

business.json

Contains business data including location data, attributes, and categories.

```
{
    // string, 22 character unique string business id
    "business_id": "tnhfDv5Il8EaGSXZGiuQGg",

    // string, the business's name
    "name": "Garaje",

    // string, the neighborhood's name
    "neighborhood": "SoMa",

    // string, the full address of the business
    "address": "475 3rd St",

    // string, the city
    "city": "San Francisco",

    // string, 2 character state code, if applicable
    "state": "CA",

    // string, the postal code
    "postal code": "94107",

    // float, latitude
    "latitude": 37.7817529521,

    // float, longitude
    "longitude": -122.39612197,

    // float, star rating, rounded to half-stars
    "stars": 4.5,

    // interger, number of reviews
    "review_count": 1198,

    // integer, 0 or 1 for closed or open, respectively
    "is_open": 1,
    // object, business attributes to values. note: some attribute values
might be objects
    "attributes": {
        "RestaurantsTakeOut": true,
        "BusinessParking": {
            "garage": false,
            "street": true,
            "validated": false,
            "lot": false,
            "valet": false
        },
    },

    // an array of strings of business categories
    "categories": [
        "Mexican",
        "Burgers",
        "Gastropubs"
    ],

    // an object of key day to value hours, hours are using a 24hr clock
    "hours": {
        "Monday": "10:00-21:00",
        "Tuesday": "10:00-21:00",
        "Friday": "10:00-21:00",
        "Wednesday": "10:00-21:00",
        "Thursday": "10:00-21:00",
        "Sunday": "11:00-18:00",
        "Saturday": "10:00-21:00"
    }
}
```

From the above dataset, based on our research question, We are planning to use the following attributes:

```
     // float, star rating, rounded to half-stars
    "stars": 4.5,

    // interger, number of reviews
    "review_count": 1198,

    // integer, 0 or 1 for closed or open, respectively
    "is_open": 1,

    // object, business attributes to values. note: some attribute values
might be objects
    "attributes": {
        "RestaurantsTakeOut": true,
        "BusinessParking": {
            "garage": false,
            "street": true,
            "validated": false,
            "lot": false,
            "valet": false
        },
    },

    // an array of strings of business categories
    "categories": [
        "Mexican",
        "Burgers",
        "Gastropubs"
    ],

    // an object of key day to value hours, hours are using a 24hr clock
    "hours": {
        "Monday": "10:00-21:00",
        "Tuesday": "10:00-21:00",
        "Friday": "10:00-21:00",
        "Wednesday": "10:00-21:00",
        "Thursday": "10:00-21:00",
        "Sunday": "11:00-18:00",
        "Saturday": "10:00-21:00"
    }
```

Approach

Following is the high level of approach taken to solve the research question. In the following sections, they have been further discussed in detail.

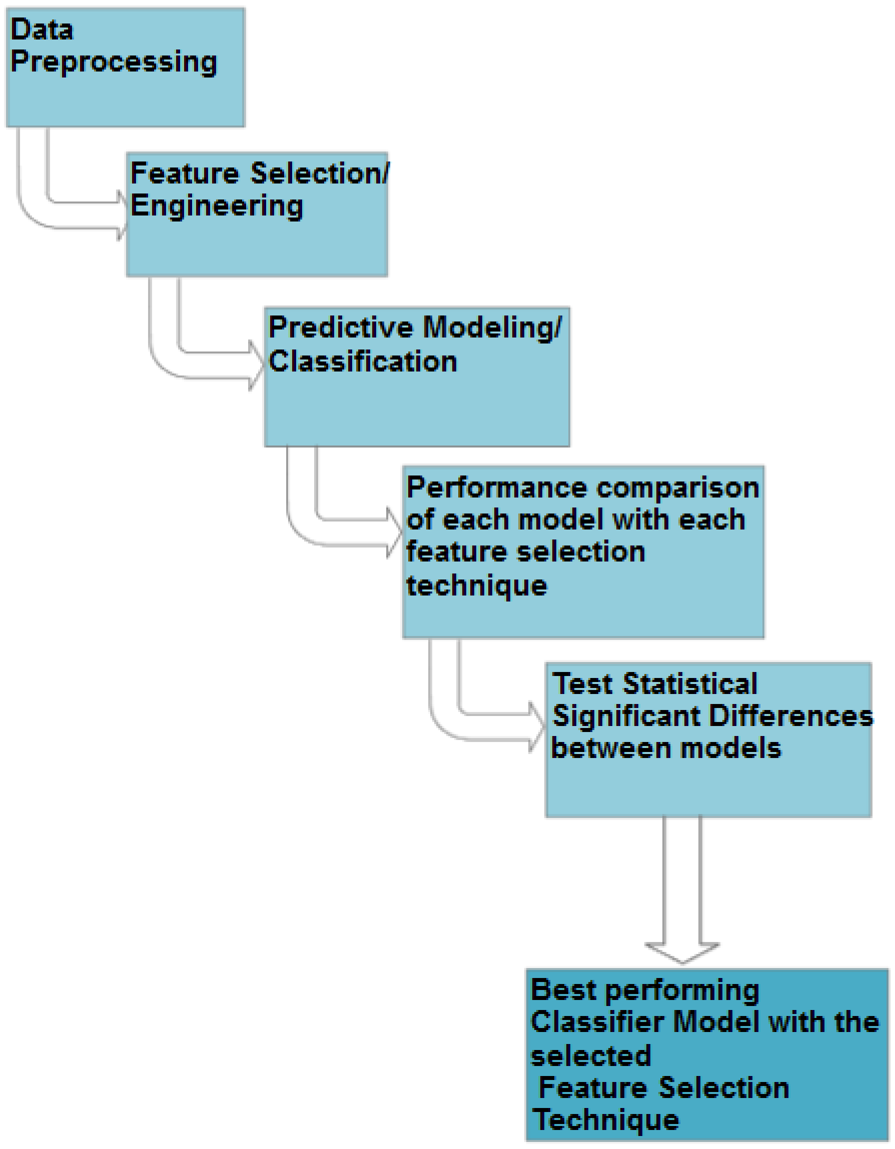

Data Preprocessing
Feature Selection/ Engineering
Predictive Modeling/ Classification
Performance comparison of each model with each feature selection technique
Test Statistical Significant Differences between models
Best performing Classifier Model with the selected Feature Selection Technique

Figure 1: High Level Overview of Approach

Step 1: Data Preprocessing

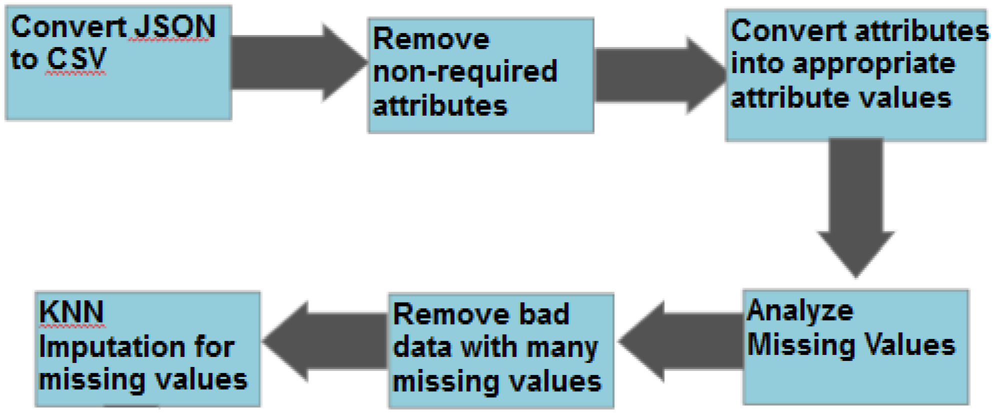

Convert JSON to CSV
Remove non-required attributes
Convert attributes into appropriate attribute values
Analyze Missing Values
Remove bad data with many missing values
KNN Imputation for missing values

Figure 2: Data Preprocessing Steps

Convert JSON to CSV:

Convert the business.json to output.csv by prep.py.

After converting it to CSV file the file has 89 attributes in total.

Remove unrelated attributes/ transform attributes:

Some of the attributes are not related to our research question like business_id, city, full_address, latitude, longitude, name, state, type, attire etc. So, we remove those attributes from the file and then we are left with 71 attributes in total to start our preprocessing.

- Import the raw data into R dataframe
- Create our new class attribute called high_rated which is yes or 1 for all restaurants where the rating is equal or above 3.5

- Process all the opening hour & closing hour of the restaurant and create a new attribute called open_7_days
- Process all the parking attributes and create a new attribute called has_somekind_of_parking
- Process all the music related facilities for a restaurant and merge them into a new attribute called has_somekind_of_music
- Wifi attribute in the original dataset is a nominal attribute with three factors (free, paid, no). Convert this attribute into a new attribute called offer_free_wifi
- Count all the cuisines a restaurant offer and sum them up for each restaurant into a new attribute called no_of_cuisine
- Now delete all the attributes that are no longer required because of merging them into their new attribute
- At this point our dataframe has 18 variables as follows:

```
> str(refined_df)
'data.frame':	72742 obs. of  18 variables:
 $ review_count         : int  4 4 3 5 5 20 3 21 7 4 ...
 $ Noise.Level          : Factor w/ 4 levels "average","loud",..: 1 NA NA NA NA 1 NA 2 NA NA ...
 $ Alcohol              : Factor w/ 3 levels "beer_and_wine",..: 3 NA NA NA NA 2 NA 2 NA NA ...
 $ Price_Range          : int  1 1 NA NA 2 1 NA 1 NA NA ...
 $ Delivery             : int  0 NA NA NA NA 0 NA 0 NA NA ...
 $ Outdoor_Seating      : int  0 0 NA NA NA 0 NA 1 NA NA ...
 $ Good_for_Groups      : int  1 1 NA NA NA 1 NA 1 NA NA ...
 $ Good_for_Kids        : int  1 NA NA 1 NA 1 0 0 NA 1 ...
 $ Accepts_Credit_Cards  : int  1 1 NA NA 0 1 NA 1 NA NA ...
 $ Takes_Reservations    : int  0 NA NA NA NA 0 NA 0 NA NA ...
 $ Take_Out             : int  1 NA NA NA NA 1 NA 1 NA NA ...
 $ Number_of_Checkins    : int  0 0 0 9 0 23 0 55 5 0 ...
 $ high_rated           : num  1 0 1 0 0 1 0 1 0 1 ...
 $ open_7_days          : num  0 0 0 0 1 0 0 0 0 0 ...
 $ has_somekind_of_parking: num  0 NA NA NA 0 0 NA 1 NA NA ...
 $ has_somekind_of_music  : num  NA NA NA NA NA NA NA NA NA NA ...
 $ offer_free_wifi       : num  0 0 0 0 0 0 0 1 0 0 ...
 $ no_of_cuisine         : int  1 1 0 0 0 3 0 3 0 0 ...
```

Missing Value Treatment:

At this point our dataframe has 18 variables and 72,742 observations. We notice many missing values in those observations.

```
> sapply(refined_df, function(x) sum(is.na(x)))
      review_count        Noise.Level           Alcohol          Price_Range
                 0              53615             51659                26610
```

```
         Delivery       Outdoor_Seating       Good_for_Groups        Good_for_Kids
           51497              48846              49446              45077
  Accepts_Credit_Cards     Takes_Reservations           Take_Out     Number_of_Checkins
             0            51999              51539               0
       high_rated          open_7_days has_somekind_of_parking   has_somekind_of_music
             0               0              31567              69884
     offer_free_wifi        no_of_cuisine
             0               0
```

From the above command we can see that has_somekind_of_parking has 96% missing values. So, we can safely delete that attribute.

To impute the missing values for rest of the attributes, we choose rpart to predict the missing value. "The limitation with DMwR::knnImputation is that it sometimes may not be appropriate to use when the missing value comes from a factor variable. Both rpart and mice has flexibility to handle that scenario. The advantage with rpart is that you just need only one of the variables to be non NA in the predictor fields.

The idea here is we are going to use rpart to predict the missing values instead of kNN. To handle factor variable, we can set the method=class while calling rpart()"[https://www.r-bloggers.com/missing-value-treatment/].

```
library(rpart)

class_mod <- rpart(Price_Range ~ . - high_rated,
data=refined_df[!is.na(refined_df$Price_Range), ], method="class", na.action=na.omit)

price_range_pred <- predict(class_mod, refined_df[is.na(refined_df$Price_Range), ])



class_mod <- rpart(Alcohol ~ . - high_rated, data=refined_df[!is.na(refined_df$Alcohol), ],
method="class", na.action=na.omit)

alcohol_pred <- predict(class_mod, refined_df[is.na(refined_df$Alcohol), ])



class_mod <- rpart(Delivery ~ . - high_rated, data=refined_df[!is.na(refined_df$Delivery), ],
method="class", na.action=na.omit)
```

```
delivery_pred <- predict(class_mod, refined_df[is.na(refined_df$Delivery), ])



class_mod <- rpart(Outdoor_Seating ~ . - high_rated,
data=refined_df[!is.na(refined_df$Outdoor_Seating), ], method="class", na.action=na.omit)

outdoor_seating_pred <- predict(class_mod, refined_df[is.na(refined_df$Outdoor_Seating), ])



class_mod <- rpart(Good_for_Groups ~ . - high_rated,
data=refined_df[!is.na(refined_df$Good_for_Groups), ], method="class", na.action=na.omit)

good_for_group_pred <- predict(class_mod, refined_df[is.na(refined_df$Good_for_Groups),
])



class_mod <- rpart(Good_for_Kids ~ . - high_rated,
data=refined_df[!is.na(refined_df$Good_for_Kids), ], method="class", na.action=na.omit)

good_for_kids_pred <- predict(class_mod, refined_df[is.na(refined_df$Good_for_Kids), ])



class_mod <- rpart(Takes_Reservations ~ . - high_rated,
data=refined_df[!is.na(refined_df$Takes_Reservations), ], method="class", na.action=na.omit)

takes_reservations_pred <- predict(class_mod,
refined_df[is.na(refined_df$Takes_Reservations), ])



class_mod <- rpart(Take_Out ~ . - high_rated, data=refined_df[!is.na(refined_df$Take_Out),
], method="class", na.action=na.omit)

take_out_pred <- predict(class_mod, refined_df[is.na(refined_df$Take_Out), ])



class_mod <- rpart(has_somekind_of_parking ~ . - high_rated,
data=refined_df[!is.na(refined_df$has_somekind_of_parking), ], method="class",
na.action=na.omit)
```

```
has_somekind_of_parking_pred <- predict(class_mod,
refined_df[is.na(refined_df$has_somekind_of_parking), ])



imputeNaFromPrediction <- function(df,col_index, mtx) {

  for(row in (1:nrow(mtx))) {

    rowNum <- rownames(mtx)[row]

    value <- colnames(mtx)[which.max(mtx[row,])]

    df[rowNum,col_index] <- value

  }

  return (df)

}



refined_df <-imputeNaFromPrediction(refined_df,4,price_range_pred)

refined_df <- imputeNaFromPrediction(refined_df,3,alcohol_pred)

refined_df <-imputeNaFromPrediction(refined_df,5,delivery_pred)

refined_df <-imputeNaFromPrediction(refined_df,6,outdoor_seating_pred)

refined_df <-imputeNaFromPrediction(refined_df,7,good_for_group_pred)

refined_df <-imputeNaFromPrediction(refined_df,8,good_for_kids_pred)

refined_df <-imputeNaFromPrediction(refined_df,10,takes_reservations_pred)

refined_df <-imputeNaFromPrediction(refined_df,11,take_out_pred)

refined_df <-imputeNaFromPrediction(refined_df,15,has_somekind_of_parking_pred)
```

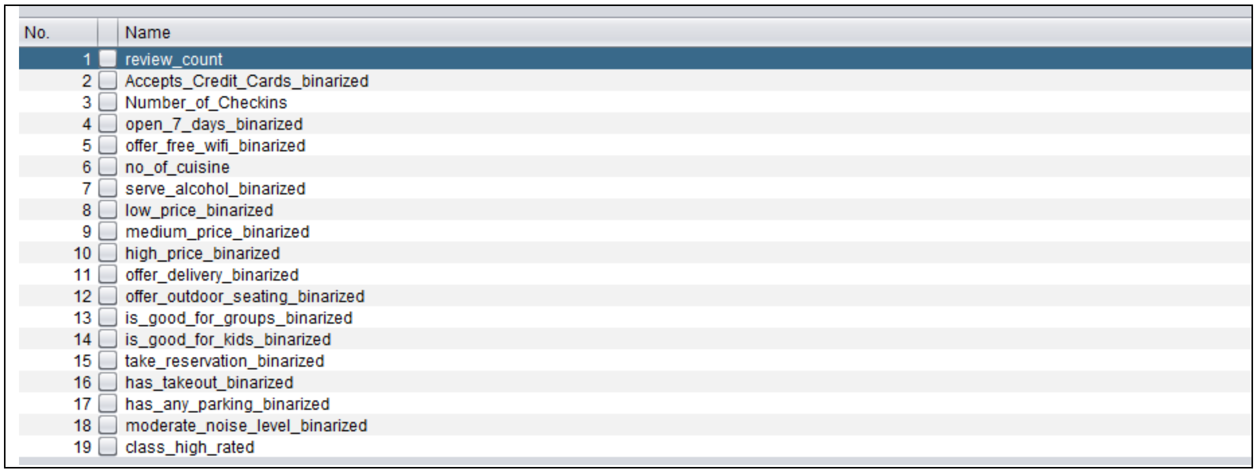
After all missing values have been imputed we can see that now there is zero missing values.

```
> sapply(refined_df, function(x) sum(is.na(x)))
```

```
        review_count          Noise.Level            Alcohol          Price_Range
                   0                    0                  0                    0
            Delivery      Outdoor_Seating      Good_for_Groups        Good_for_Kids
                   0                    0                  0                    0
 Accepts_Credit_Cards    Takes_Reservations           Take_Out    Number_of_Checkins
                   0                    0                  0                    0
          high_rated          open_7_days has_somekind_of_parking      offer_free_wifi
                   0                    0                  0                    0
        no_of_cuisine
                   0
>
```

Preprocessing:

- Then we Discretize following three numeric attributes into 10 bins for better classification:
    - Review_count
    - Number_of_checkins
    - Number_of_cuisine
- Then we Binarize rest of the attributes



No.	Name
1	review_count
2	Accepts_Credit_Cards_binarized
3	Number_of_Checkins
4	open_7_days_binarized
5	offer_free_wifi_binarized
6	no_of_cuisine
7	serve_alcohol_binarized
8	low_price_binarized
9	medium_price_binarized
10	high_price_binarized
11	offer_delivery_binarized
12	offer_outdoor_seating_binarized
13	is_good_for_groups_binarized
14	is_good_for_kids_binarized
15	take_reservation_binarized
16	has_takeout_binarized
17	has_any_parking_binarized
18	moderate_noise_level_binarized
19	class_high_rated

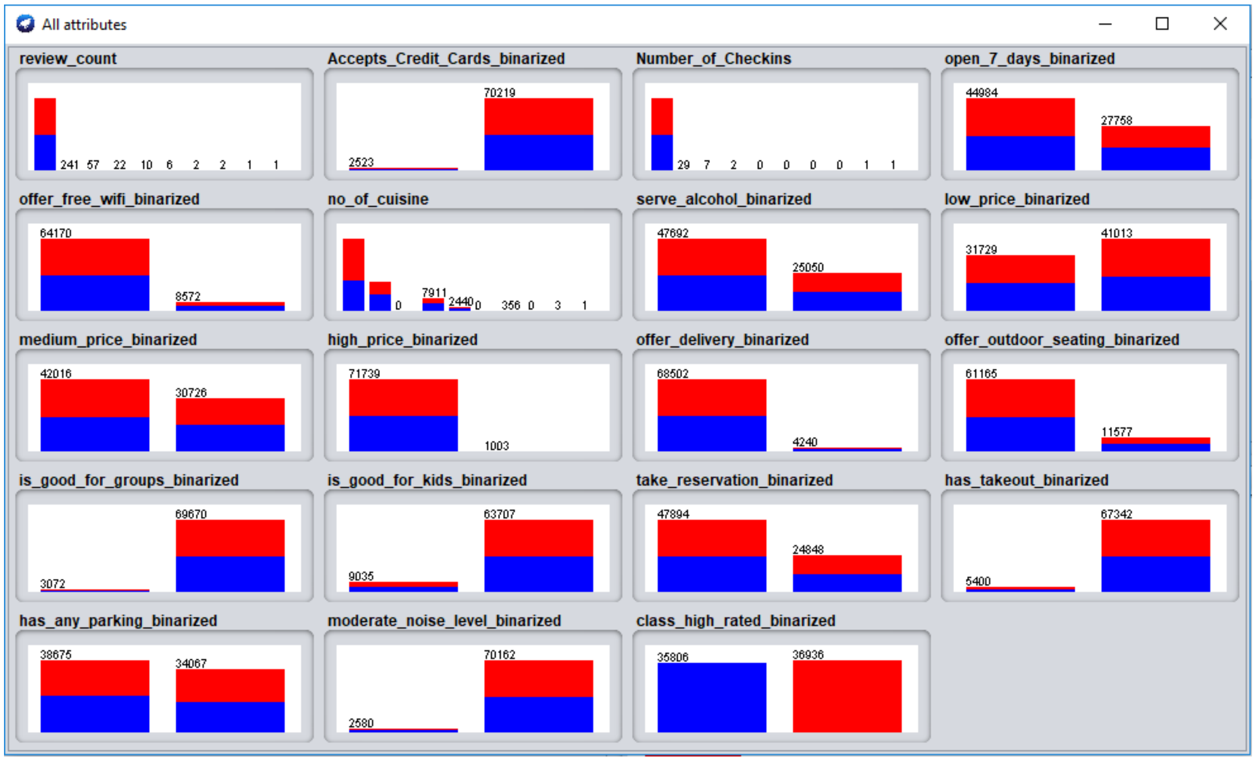


Figure 3: Data Distribution after Preprocessing

Step 2: Feature Selection

For Attribute Selection we used following three techniques:

- **CorrelationAttributeEval** : Evaluates the worth of an attribute by measuring the correlation (Pearson's) between it and the class.
- **GainRatioAttributeEval** : Evaluates the worth of an attribute by measuring the gain ratio with respect to the class.
- **ClassifierAttributeEval** : Evaluates the worth of an attribute by using a user-specified classifier.

We ran each of the algorithm on Weka with 10 fold cross validation and selected top 9 attributes.

CorrelationAttributeEval	GainRatioAttributeEval	ClassifierAttributeEval
• **has_any_parking**	• **has_any_parking**	• **moderate_noise_level**
• **serve_alcohol**	• **Number_of_Checkins**	• offer_free_wifi
• **take_reservation**	**serve_alcohol**	• **serve_alcohol**
• medium_price	• review_count	• no_of_cuisine
• **open_7_days**	• **take_reservation**	• **open_7_days**
• **low_price**	• is_good_for_groups	• **has_any_parking**
• **offer_outdoor_seating**	**offer_outdoor_seating**	• **Number_of_Checkins**
• is_good_for_groups	• medium_price	• Accepts_Credit_Cards
• **moderate_noise_level**	• **moderate_noise_level**	• **low_price**

Table 1: Output – Features Selection Techniques

Step 3: Predictive Modeling/ Classification

Next we build three classifier models with each 3 of these attribute selection techniques and then tested the model with 80%-20% split of training set and test set. Three classifiers we selected to test the model are:

- Random Forest
- Logistic Regression
- Naïve Bayes

The test result we achieved is as follows:

Feature Selection Technique	Classification Model		
	Random Forest	**Logistic Regression**	**KNN**
CorrelationAttributeEval	Correctly Classified Instances 12183 **83.7435 %** Incorrectly Classified Instances 2365 16.2565 %	Correctly Classified Instances 12184 **83.7503 %** Incorrectly Classified Instances 2364 16.2497 %	Correctly Classified Instances 12545 **86.2366 %** Incorrectly Classified Instances 2003 13.7634 %
GainRatioAttributeEval	Correctly Classified Instances 12185 **83.7572 %** Incorrectly Classified Instances 2363 16.2428 %	Correctly Classified Instances 12185 **83.7572 %** Incorrectly Classified Instances 2363 16.2428 %	Correctly Classified Instances 12546 **86.2435 %** Incorrectly Classified Instances 2002 13.7565 %
ClassifierAttributeEval	Correctly Classified Instances 12168 **83.6404 %** Incorrectly Classified Instances 2380 16.3596 %	Correctly Classified Instances 12186 **83.7641 %** Incorrectly Classified Instances 2362 16.2359 %	Correctly Classified Instances 12540 **86.2022 %** Incorrectly Classified Instances 2008 13.7978 %

Table 2: Output – Classifier performance with different Feature Selection Techniques

From the above test result we found that all the classifiers performed similar with a slight variation (~3%). The best classifier that outperformed other two is **K-Nearesr Neighbor (KNN) with GainRatioAttributeEval**.

Step 4: Performance Improvement & Comparison of each model

Next, we select top 6 features that were selected at least by two (out of the three) feature selection techniques and re-train three classification models again. Then we test each model in 10 iteration (each time with a different seed value and taking random sample of the 20% dataset for testing). We noticed an almost 5% increase of accuracy after selecting top 6 features. These top 6 features are as follows:

1. **has_any_parking**
2. **serve_alcohol**
3. **moderate_noise_level**
4. **low_price**
5. **take_reservation**
6. **open_7_days**

Results are as below:

Iteration	**Logistic Regression**		**Random Forest**		**KNN**	
	Accuracy%	**Precision%**	**Accuracy%**	**Precision%**	**Accuracy %**	**Precision%**
1	86.59	86.67	86.74	86.84	89.22	89.34
2	86.47	86.59	87.00	87.23	89.50	89.71
3	86.55	86.68	86.68	86.70	89.19	89.16
4	86.75	86.91	86.74	85.76	89.24	89.35
5	86.16	86.37	87.03	87.19	**89.54**	89.66
6	86.47	86.60	86.46	86.48	88.96	88.95
7	86.65	86.76	86.45	86.50	88.95	88.98
8	86.42	86.59	85.93	85.94	88.44	88.52
9	86.64	86.83	86.48	86.48	89.00	89.10
10	86.32	86.49	86.53	86.66	89.03	89.14

Table 3: Output – Classifier performance with selected features for 10 iteration

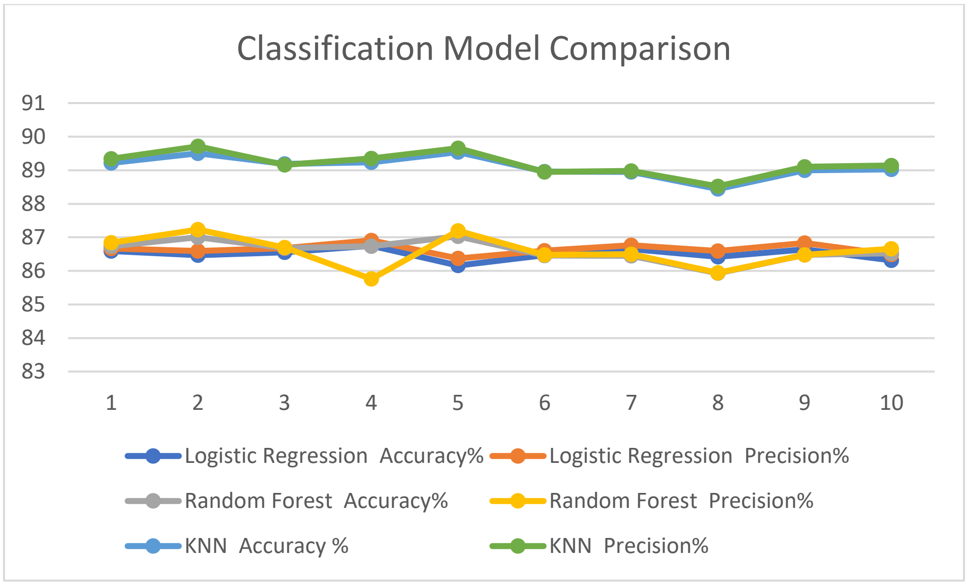


Figure 4: Performance Comparison of LR vs RF vs KNN

From the above result, it is clear that KNN is always outperforming other two models with an accuracy of almost 90%.

Step 5: Test Statistical Significance Difference

In next step we conduct a variance test to find out whether these three models are statistically significantly different. To find this out we performed a One-way ANOVA test on the accuracy of three models for 10 different iteration with the significance level set to 5% (α = 0.05).

Below is the test table:

LR	**RF**	**KNN**
86.59	86.74	89.22
86.47	87	89.5
86.55	86.68	89.19
86.75	86.74	89.24
86.16	87.03	89.54
86.47	86.46	88.96
86.65	86.45	88.95
86.42	85.93	88.44
86.64	86.48	89
86.32	86.53	89.03

Table 4: Input – ANOVA test table for three classifier model accuracy

We convert the table to a CSV and import into R and run the following test:

```
> model <- read.csv('/Users/Arahman/Documents/CKME-136/Model_accuracy.csv', header = TRUE,
sep = ",")
> attach(model)
> str(model)
'data.frame':	30 obs. of  2 variables:
 $ Model   : Factor w/ 3 levels "KNN","Linear Regression",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ Accuracy: num  86.6 86.5 86.5 86.8 86.2 ...
> is.factor(Model)
[1] TRUE
> boxplot(Accuracy~Model, main="Fig: Boxplot of Accuracy of Three Classifier Model")
> boxplot(Accuracy~Model, main="Fig: Boxplot of Accuracy of Three Classifier Model", col=
rainbow(3))
>
```

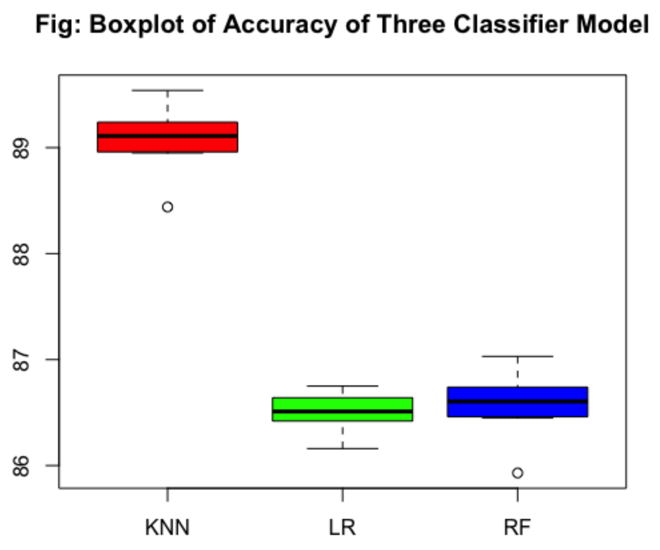
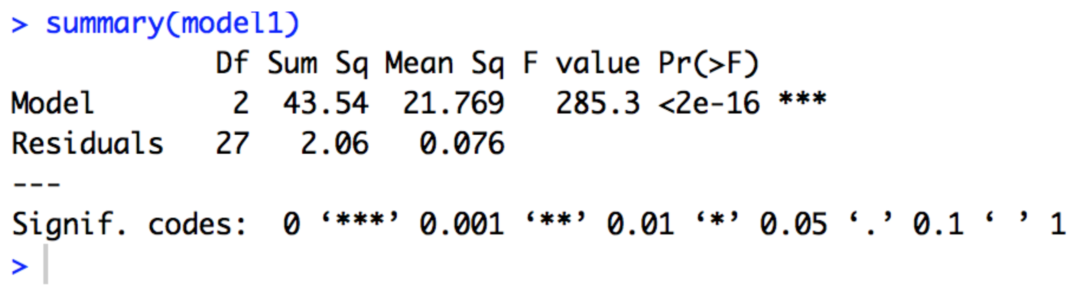


Figure 5: Boxplot of three classifier performane

```
> model1 <- aov(Accuracy~Model)
> summary(model1)
```

ANOVA:



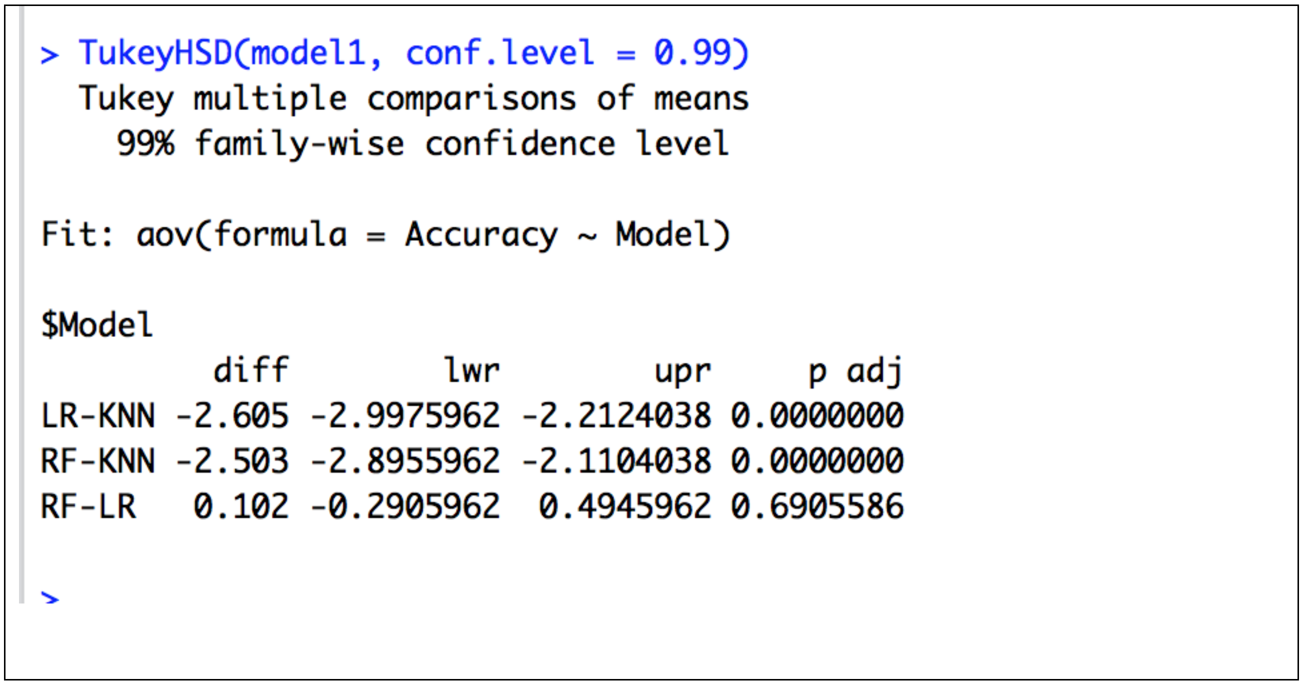
```
> summary(model1)
            Df Sum Sq Mean Sq F value Pr(>F)
Model        2  43.54  21.769   285.3 <2e-16 ***
Residuals   27   2.06   0.076
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Figure 6: Output – ANOVA test result

From the above results, it is observed that the F value is 285.3 and it is highly significant as the corresponding p-value is much less than the level of significance (1% or 0.01). **So, we can reject the null hypothesis and can say that the average accuracy of three models are not equal.**

TukeyHSD

Next, let's compare the models with each other with **TukeyHSD** test:



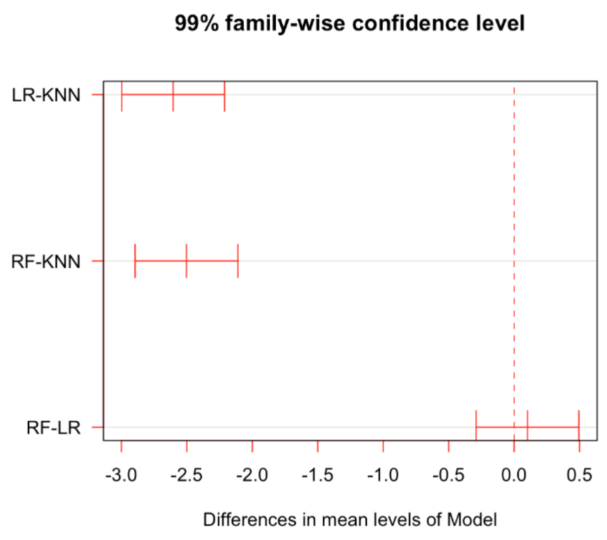
```
> TukeyHSD(model1, conf.level = 0.99)
  Tukey multiple comparisons of means
    99% family-wise confidence level

Fit: aov(formula = Accuracy ~ Model)

$Model
         diff        lwr        upr     p adj
LR-KNN -2.605 -2.9975962 -2.2124038 0.0000000
RF-KNN -2.503 -2.8955962 -2.1104038 0.0000000
RF-LR   0.102 -0.2905962  0.4945962 0.6905586

>
```

Figure 7: Output – TukeyHSD test result

From the above result, we can see that all pairs are statistically significantly different except Random Forest and Logistic Regression where the p-adjusted value is greater than the significance level (0.01). From the above result, we can also see that both pairwise difference between LR-KNN and RF-KNN is -2.605 and -2.503 respectively, which tells us that KNN has higher accuracy than both LR & RF.

Let's plot the results:

```
> plot(TukeyHSD(model1, conf.level = 0.99), las=1, col = "red")
> plotmeans(Accuracy~Model, main="Mean Plot with 99% Confidence Interval", ylab = "Accuracy %",
xlab = "Classification Model")
>
```



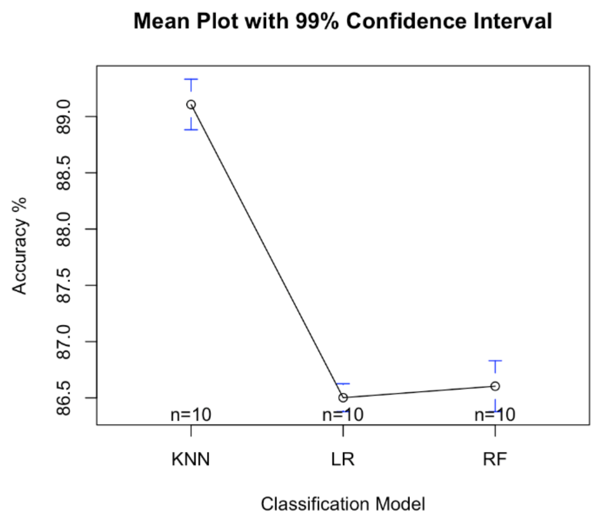


Figure 8: Differences in mean level of three classifiers with 99% confidence interval

Since the number of observation is less, it's better to test the normality for the overall residuals of the model. Where resid() function extracts the residuals of the model and stores in the object uhat.

Sharpio-Wilk Normality test

Finally, we conduct a **Sharpio-Wilk Normality test** to confirm whether the samples are taken from normal population.

```
> uhat<-resid(model1)
> shapiro.test(uhat)

	Shapiro-Wilk normality test

data:  uhat
W = 0.92439, p-value = 0.03491

>
```

Figure 9: Output - Sharpio-Wilk Normality test

From the above result, we can see that p-value is higher than the significance level (0.03>0.01). **So, we accept the null hypothesis and can say that samples are taken from the normal population.**

Step 6: Best Performing Model

From the above tests, we can conclude that clearly KNN Classifier works better with an accuracy of 90% and this classifier is significantly different than other two classifiers (Random Forest and Logistic Regression), which also confirms the assumption that the samples for comparison of there models were taken from normal population. This classifier was built on six selected attributes (validated by at least two attribute selection techniques). So, we can conclude that after analyzing Yelp dataset, following six important attributes can predict with a 90% accuracy whether a business can achieve a higher rating (>3.5):

1. has_any_parking
2. serve_alcohol
3. moderate_noise_level
4. low_price
5. take_reservation
6. open_7_days

Conclusion

An open source dataset on many restaurants was obtained from Yelp website in JSON format, converted to tabular structure, and preprocessed in R followed by exploratory analysis. Features were explored and converted to binary features and a binary class attribute (is_high_rating) was assigned accordingly. Three predictive models were trained with each feature selection technique with a different set of features. Later each model was trained on 80% of data in 10 iterations with a unique seed to choose randomly a training set. Their accuracy were then compared with each other. From the results of these training and testing phase we found out that KNN classifier worked best with an almost 90% accuracy where as Logistic Regression and Random Forest both performed similar with an accuracy of 86%. After that we ran some significance difference test (ANOVA and TukeyHSD), to compare the accuracy mean with each other for any significance difference. We found that all three models are significantly different however the pair Random Forest and Logistic Regression does not seem to be significantly different. Finally, we ran Sharpio-Wilk Normality test to confirm that the samples were taken from a normal population. In closing we can say that the KNN classification model performed well and likely to be helpful for existing restaurant owners or prospective owners to decide on which features to focus on most to achieve a high rating on Yelp website.

References:

[A16] Asghar, N. (2016). Yelp Dataset Challenge: Review Rating Prediction. *arXiv preprint arXiv:1605.05362*.

[CFV14] Carbon, K., Fujii, K., & Veerina, P. (2014). Applications of Machine Learning to Predict Yelp Ratings.

[FK14] Fan, M., & Khademi, M. (2014). Predicting a business star in yelp from its reviews text alone. *arXiv preprint arXiv:1401.0864*.

[GEW06] Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*(1), 3-42.

[K17] Kaing, D. (2017, October). Yelp business rating classification using hybrid ensemble. In *Knowledge Engineering and Applications (ICKEA), 2017 2nd International Conference on* (pp. 30-33). IEEE.

[LLJ+11]]Li, F., Liu, N., Jin, H., Zhao, K., Yang, Q., & Zhu, X. (2011, July). Incorporating reviewer and product information for review rating prediction. In *IJCAI* (Vol. 11, pp. 1820-1825).

[PL05] Pang, B., & Lee, L. (2005, June). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 115-124). Association for Computational Linguistics.

[P17] Perez, L. (2017). Predicting Yelp Star Reviews Based on Network Structure with Deep Learning. *arXiv preprint arXiv:1712.04350*.

[SC00] Susskind, A. M., & Chan, E. K. (2000). How Restaurant Features Affect Check Aberages: A Study of the Toronto Retaurant Market. *Cornell Hotel and Restaurant Administration Quarterly*, *41*(6), 56-63.

[Y18] Yelp Open Dataset. (2018). https://www.yelp.com/dataset/

Appendix A

Confusion Matrix of KNN Classifier for 10 Iterations:

KNN:
Iteration 1:

```
=== Confusion Matrix ===

    a     b   <-- classified as
   74  1539 |     a = 0
   29 12906 |     b = 1
```

Iteration 2:

```
=== Confusion Matrix ===

    a     b   <-- classified as
   76  1484 |     a = 0
   44 12944 |     b = 1
```

Iteration 3:

```
=== Confusion Matrix ===

    a     b   <-- classified as
   51  1571 |     a = 0
    2 12924 |     b = 1
```

Iteration 4:

```
=== Confusion Matrix ===

    a     b   <-- classified as
   76  1537 |     a = 0
   27 12908 |     b = 1
```

Iteration 5:

```
=== Confusion Matrix ===

    a     b   <-- classified as
   74  1492 |     a = 0
   30 12950 |     b = 1
```

Iteration 6:

```
=== Confusion Matrix ===

    a     b   <-- classified as
   74  1492 |     a = 0
   30 12950 |     b = 1
```

Iteration 7:
```
=== Confusion Matrix ===

     a     b   <-- classified as
    74  1492 |     a = 0
    30 12950 |     b = 1
```

Iteration 8:
```
=== Confusion Matrix ===

     a     b   <-- classified as
    71  1660 |     a = 0
    22 12795 |     b = 1
```

Iteration 9:
```
=== Confusion Matrix ===

     a     b   <-- classified as
    79  1574 |     a = 0
    27 12868 |     b = 1
```

Iteration 10:
```
=== Confusion Matrix ===

     a     b   <-- classified as
    70  1569 |     a = 0
    27 12882 |     b = 1
```