# CKME 136 Capstone Project – Initial Results

This report covers the initial test report and findings. All the source code and related files can be found at <https://github.com/armisho/CKME136-Yelp>

# Data Preprocessing:

## 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) |

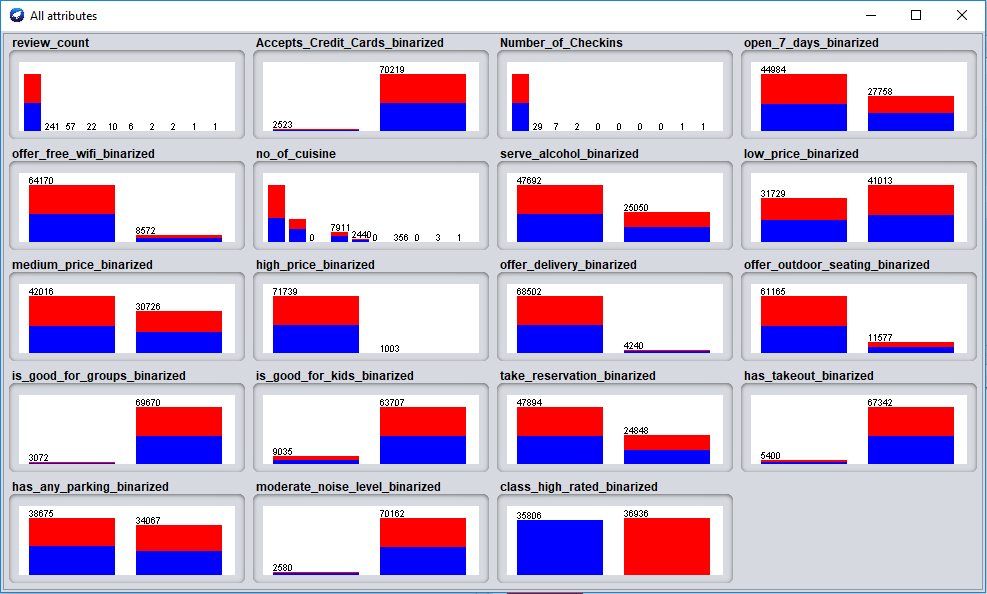
 After all missing values have been imputed we can see that now there is zero missing values.

|  |
| --- |
| > 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

|  |
| --- |
|  |



# Attribute 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 * serve\_alcohol * take\_reservation * medium\_price * open\_7\_days * low\_price * offer\_outdoor\_seating * is\_good\_for\_groups * moderate\_noise\_level | * has\_any\_parking * Number\_of\_Checkins serve\_alcohol * review\_count * take\_reservation * is\_good\_for\_groups offer\_outdoor\_seating * medium\_price * moderate\_noise\_level | * moderate\_noise\_level * offer\_free\_wifi * serve\_alcohol * no\_of\_cuisine * open\_7\_days * has\_any\_parking * Number\_of\_Checkins * Accepts\_Credit\_Cards * low\_price |

# Classification Model

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** | **Naïve Bayes** |
| **CorrelationAttributeEval** | Correctly Classified Instances 12183 **83.7435 %**  Incorrectly Classified Instances 2365 16.2565 %  === Confusion Matrix ===  a b <-- classified as  23 2340 | a = 0  25 12160 | b = 1 | Correctly Classified Instances 12184 **83.7503 %**  Incorrectly Classified Instances 2364 16.2497 %  === Confusion Matrix ===  a b <-- classified as  0 2363 | a = 0  1 12184 | b = 1 | Correctly Classified Instances 12186 **83.7641 %**  Incorrectly Classified Instances 2362 16.2359 %  === Confusion Matrix ===  a b <-- classified as  2 2361 | a = 0  1 12184 | b = 1 |
| **GainRatioAttributeEval** | Correctly Classified Instances 12185 **83.7572 %**  Incorrectly Classified Instances 2363 16.2428 %  === Confusion Matrix ===  a b <-- classified as  0 2363 | a = 0  0 12185 | b = 1 | Correctly Classified Instances 12185 **83.7572 %**  Incorrectly Classified Instances 2363 16.2428 %  === Confusion Matrix ===  a b <-- classified as  0 2363 | a = 0  0 12185 | b = 1 | Correctly Classified Instances 12185 **83.7572 %**  Incorrectly Classified Instances 2363 16.2428 %  === Confusion Matrix ===  a b <-- classified as  0 2363 | a = 0  0 12185 | b = 1 |
| **ClassifierAttributeEval** | Correctly Classified Instances 12168 **83.6404 %**  Incorrectly Classified Instances 2380 16.3596 %  === Confusion Matrix ===  a b <-- classified as  28 2335 | a = 0  45 12140 | b = 1 | Correctly Classified Instances 12186 **83.7641 %**  Incorrectly Classified Instances 2362 16.2359 %  === Confusion Matrix ===  a b <-- classified as  1 2362 | a = 0  0 12185 | b = 1 | Correctly Classified Instances 12185 **83.7572 %**  Incorrectly Classified Instances 2363 16.2428 %  === Confusion Matrix ===  a b <-- classified as  0 2363 | a = 0  0 12185 | b = 1 |

From the above test result we found that all the classifiers performed similar with a slight variation. The best classifier that outperformed other two is **Logistic Regression with ClassifierAttributeEval**.

# Next Steps

* Explore other avenues to see if we can improve the performance of the classifier model.
* Test the model for any statistically significant differences. We plan to use a statistical method **1 WAY** **ANOVA** to conduct this test.