



Final Project: Yelp Dataset

**INFO7390 33162 Advances Data Sci/Architecture
SEC 01 - Spring 2016**

**Team 5:
Madhav Sahu
Rohit Singh
Shrenik Jain**

INTRODUCTION

The review website Yelp not only connects customers with businesses, but also allows customers to rate their experiences. There are millions of data consisting of user/business information, reviews, votes, and so on.

From this dataset, we will attempt to answer the following questions:

- 1) Predict the rating of a restaurant.
- 2) Identify which attributes of a restaurant can affect its ratings on yelp.
- 3) Is it possible, if we improve the ambience, parking, and so on of a restaurant than it can result in a better rating on yelp?
- 4) If anyone is planning to open a new restaurant of particular ethnicity type than what can be desired attributes for getting a decent rating on yelp?

To answer these questions, we have build model focusing on various factors that contribute to success of a restaurant like parking, waiter service, ambience of restaurant, location, price, waiting time and so on. This model will identify the key factors that lead to success of popular restaurants, based on the data collected from users, reviews and business dataset. This model can also be used to establish a new restaurant as well.

DATA ACCESS

Our data was downloaded from the Yelp Dataset Challenge web page. The URL for that page is http://www.yelp.com/dataset_challenge Click on the ‘Get the Data’ button and complete a form to download.

The data includes information on the businesses that have been reviewed, the reviews, the user/reviewer, user check-ins, and user provided tips. Yelp defines the data as follows:

The Challenge Dataset:

- 1.6M reviews and 500K tips by 366K users for 61K businesses
- 481K business attributes, e.g., hours, parking availability, ambience.
- Social network of 366K users for a total of 2.9M social edges.
- Aggregated check-ins over time for each of the 61K businesses

Cities:

U.K.: Edinburgh

Germany: Karlsruhe

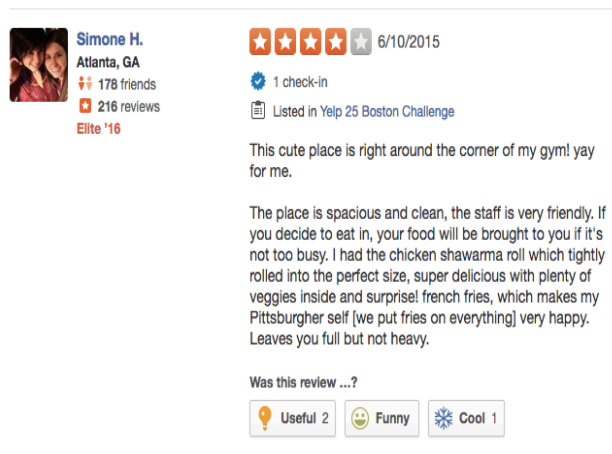
Canada: Montreal and Waterloo

U.S.: Pittsburgh, Charlotte, Urbana-Champaign, Phoenix, Las Vegas, Madison

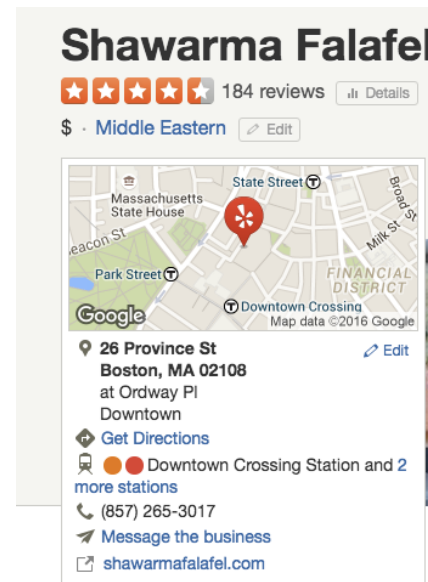
From the data, we focused only on records associated with restaurants in US. The processing of consolidating and cleaning the data is outlined in the sections that follow.

DATA UNDERSTANDING

To collect the data about restaurants, Yelp.com was used for collecting information for ratings and reviews. Sample of a restaurant review and restaurant rating on Yelp is provided below. As we can see below provide the information regarding the rating of restaurant provided by user and how many people found it useful. The rating of restaurant is the average of total ratings given by users for a particular restaurant.



User's Review



Business Overall Rating

The yelp datasets have following files in JSON:

1. Business File →

Attributes:

'type': 'business', 'business_id': (encrypted business id), 'name': (business name), 'neighborhoods': [(hood names)], 'full_address': (localized address), 'city': (city), 'state': (state), 'latitude': latitude, 'longitude': longitude, 'stars': (star rating, rounded to half-stars), 'review_count': review count, 'categories': [(localized category names)], 'open': True / False (corresponds to closed, not business hours), 'hours': { (day_of_week): { 'open': (HH:MM), 'close': (HH:MM) }, }, 'attributes': { (attribute_name): (attribute_value), }

2. Review File →

Attributes:

'type': 'review', 'business_id': (encrypted business id), 'user_id': (encrypted user id), 'stars': (star rating, rounded to half-stars), 'text': (review text), 'date': (date, formatted like '2012-03-14'), 'votes': {(vote type): (count)}

3. User File →

Attributes:

'type': 'user', 'user_id': (encrypted user id), 'name': (first name), 'review_count': (review count), 'average_stars': (floating point average, like 4.31), 'votes': {(vote type): (count)}, 'friends': [(friend

user_ids]], 'elite': [(years_elite)], 'yelping_since': (date, formatted like '2012-03'), 'compliments': { (compliment_type): (num_compliments_of_this_type), }, 'fans': (num_fans)

4. Check-in File →

Attributes:

```
'type': 'checkin', 'business_id': (encrypted business id), 'checkin_info': {  
  '0-0': (number of checkins from 00:00 to 01:00 on all Sundays),  
  '1-0': (number of checkins from 01:00 to 02:00 on all Sundays),  
  ...  
  '14-4': (number of checkins from 14:00 to 15:00 on all Thursdays),  
  ...  
  '23-6': (number of checkins from 23:00 to 00:00 on all Saturdays)  
} # if there was no checkin for a hour-day block it will not be in the dict
```

5. Tip File →

Attributes:

```
'type': 'tip', 'text': (tip text), 'business_id': (encrypted business id), 'user_id': (encrypted user id),  
'date': (date, formatted like '2012-03-14'), 'likes': (count)
```

DATA EXTRACTION AND CONSOLIDATION:

The file that we downloaded from yelp were in JSON file. The first step was to extract the data in csv format. We did our data extraction in R. The following is small R code that extracted the JSON data in csv format.

```
install.packages("jsonlite");  
library("jsonlite")  
##### Business File #####  
#assign file to yelp variable  
yelp<- "/Users/Shrenik/Desktop/Adv Data Science/Final Project/yelp_dataset_challenge_academic_dataset/yelp_academic_dataset_business.json"  
#read in file  
business <- stream_in(file(yelp))  
#Flatten JSON file  
business <- flatten(business,recursive = TRUE)  
#create dataframe that work with write  
business<- data.frame(lapply(business, as.character), stringsAsFactors = FALSE)  
#import csv file  
write.csv(business, file = "/Users/Shrenik/Desktop/Adv Data Science/Final Project/Dataset/business.csv",row.names=TRUE, na="")
```

DATA CLEANING:

Our aim was to determine the rating of restaurants of US. And the data contained all the businesses that were registered in yelp. So we first removed all rows that were not a restaurant and made sure that the data contains restaurants that are in US only. Below is sample code of how we removed the rows as follow:

```
business <- business[grepl("restaurant", business$categories), ]  
business <- business[!grepl("SCB",business$state),]  
business <- business[!grepl("BW",business$state),]  
business <- business[!grepl("RP",business$state),]  
business <- business[!grepl("QC",business$state),]
```

In the above code SCB, BW are the states of UK and Germany respectively. We similarly removed all the states that were not in US.

After having data just for restaurant, our second step was to determine the ethnicity of each restaurant and create different subsets with ethnicity type and then merge it together to create a new business file with ethnicity type.

```
#####
ethnicity_chinese <- business[grepl("Chinese", business$categories), ]
ethnicity_german <- business[grepl("German", business$categories), ]
ethnicity_american <- business[grepl("American", business$categories), ]
ethnicity_indian <- business[grepl("Indian", business$categories), ]
#####
ethnicity_chinese$type <- rep("Chinese", nrow(ethnicity_chinese))
ethnicity_german$type <- rep("German", nrow(ethnicity_german))
ethnicity_american$type <- rep("American", nrow(ethnicity_american))
ethnicity_indian$type <- rep("Indian", nrow(ethnicity_indian))
#####
business <- rbind(ethnicity_chinese, ethnicity_german, ethnicity_american, ethnicity_indian, ethnicity_japanese, ethnicity_korean,
                  ethnicity_greek, ethnicity_italian, ethnicity_mexican, ethnicity_irish, ethnicity_caribbean, ethnicity_pakistani,
                  ethnicity_brazilian)
```

Above is snippet of R code that we ran for creating a new business file with ethnicity type. After new business file with different ethnicity, our 3rd step was to clean the column names. As all attributes were considered in model building and as column name seemed to be ambiguous we ran following R code for cleaning the column names.

```
# Function to clean the names of the variables
clean.names <- function(df){
  colnames(df) <- gsub("attributes[^[:alnum:]]", "", colnames(df))
  colnames(df) <- gsub("[^[:alnum:]]", "", colnames(df))
  colnames(df) <- tolower(colnames(df))
  return(df)
}

#The column names will look like the following after flattening and extracting :
business <- clean.names(business)
colnames(business)
```

R code 1

Running the above snippet code will give the following result from dataset:

attributes.Ambience.romantic	attributes.Ambience.intimate	attributes.Ambience.classy	attributes.Ambience.hipster	attributes.Ambience.divey	attributes.Ambience.tourist
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE

Before running the R code 1

ambience.romantic	ambience.intimate	ambience.classy	ambience.hipster	ambience.divey	ambience.touristy	ambience.trendy	ambience.upscale	ambience.casual
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

After Running the R code 1

DATA TRANSFORMATION AND REDUCTION:

After cleaning the column names, our aim was to get the average weighted star rating and user's review count for a restaurant. For this we extracted the reviews dataset and ran the following R code to get a new column user review count and average weighted star rating:

```
review <- sqldf("SELECT business_id,Count(stars) AS reviewcount, SUM(stars * useful) / SUM(useful) AS Weighted_stars
FROM reviews GROUP BY business_id")
```

We used the useful vote and star rating given by user to determine the average weighted star rating. We removed the reviews which had useful vote as 0. The following is the review table that we wanted to use for our model.

business_id	reviewcount	Weighted_stars
8PlrSa0WsLKIUWdVlptszg	68	5
b4zV86JJ9Ez4MfzaWO0umQ	54	5
iavfj2f10OCQ2Be178L_Rg	54	5
sZxz-chNjkq_AzXpO09Gew	51	5
wUd2aOHxWrKmYtel0OH5rQ	48	5

Review Table with Weighted Stars Rating

After creating a modified review table, our final step was to determine the important attributes of restaurants that can have impact on restaurant's rating. We removed the columns that contains more than 70% of values as N/A. Then we were left with 41 attributes out of 98 attributes of business file. Then we merged the business file with review file and we also reduced the similar attributes into a single attribute with weightage. The following is final R code snippet that we ran for our modified dataset:

```
business_review <- sqldf("SELECT business_id, name, type, city, fulladdress, state, stars, Weighted_stars,
review2.reviewcount AS userreviewcount,business.reviewcount AS Business_ReviewCount, year, pricerange,
waiterservice, wheelchairaccessible, havstv, alcohol, takeout, attire, goodforkids,
goodfordancing, goodforgroups,

CASE WHEN parkinggarage='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN parkingstreet='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN parkingvalidated= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN parkinglot= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN parkingvalet= 'TRUE' THEN 1 ELSE 0 END
AS parking,

CASE WHEN goodfordessert='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN goodforlatenight='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN goodforlunch= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN goodfordinner= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN goodforbrunch= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN goodforbreakfast= 'TRUE' THEN 1 ELSE 0 END
AS goodforfood,

CASE WHEN ambienceromantic='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambienceintimate= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambienceclassy= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambiencehipster= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambiencediver='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambiencefamily= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambiencehipster= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN ambiencecasual= 'TRUE' THEN 1 ELSE 0 END
AS ambience,

CASE WHEN musicdj='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN musicbackgroundmusic='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN musicjukebox= 'TRUE' THEN 1 ELSE 0 END
+ CASE WHEN musiclive='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN musicvideo='TRUE' THEN 1 ELSE 0 END
+ CASE WHEN musickaraoke= 'TRUE' THEN 1 ELSE 0 END AS music

FROM business
INNER JOIN review2 ON business.businessid = review2.business_id ")
```

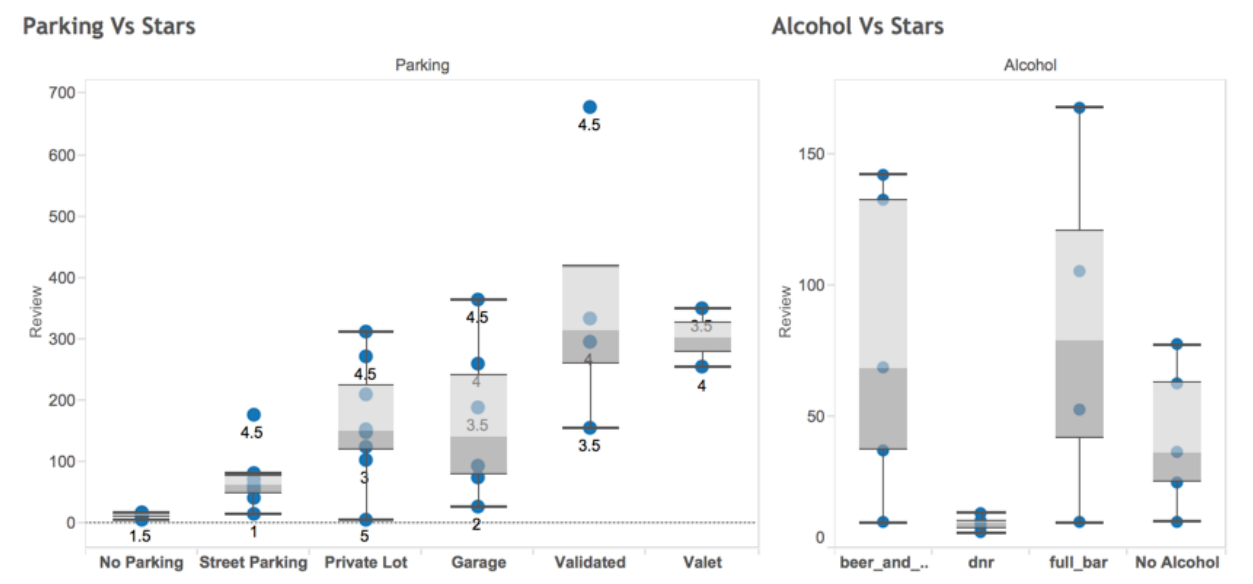
DATA VISUALIZATION:

After reducing the attributes using R code to just important factors which had most influencing role in getting the predicted rating value during modelling, we decided to visualize the box plot of all important factors.

The factors Alcohol, Parking, Ambience and Alcohol have been plotted below giving us the details of rating received and number of reviews for that category.

1. In this Box plot we can see how different type of parking affect the rating of restaurants. When there is no parking provided by the restaurant the box plot clearly shows that the rating goes down also the reviews received by the restaurant also dips. Whereas when the restaurant has a garage parking allotted for the users, the ratings received are higher and the number of user ratings also increase.

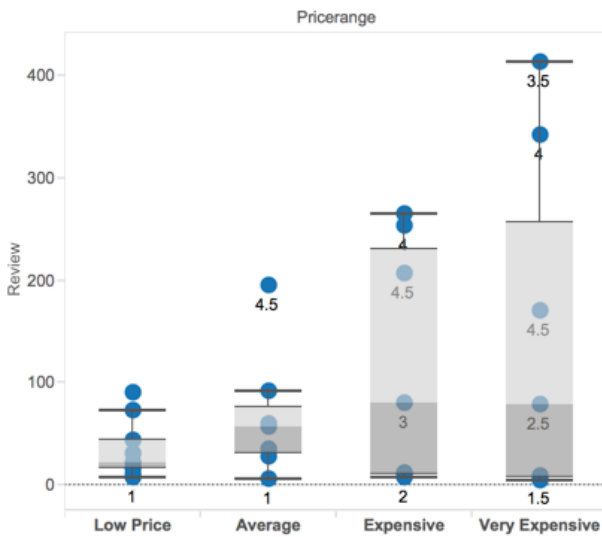
Similarly, the restaurant rating received also differ when the Alcohol option provided by the restaurant change. When the restaurant provides beer and wine option the reviews received are higher and the ratings are higher towards 3.5-4.5.



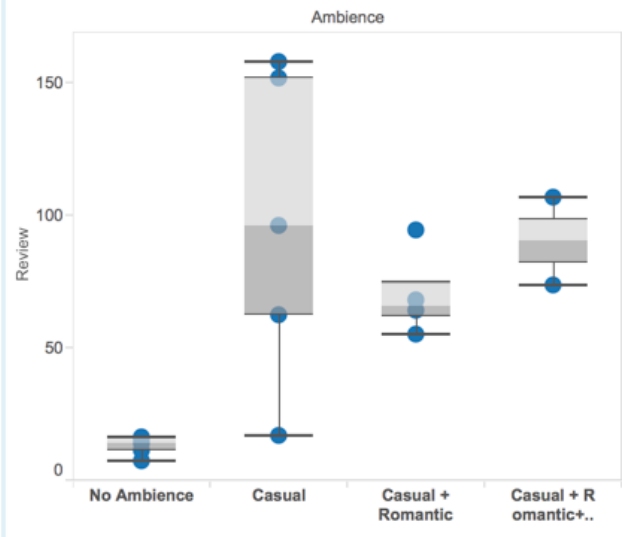
In this below Box plot we can see how different price range of restaurant affect the rating of restaurants. When the price range is low the box plot clearly shows that the rating goes down also the reviews received by the restaurant also dips. Whereas when the restaurant has very expensive price range the ratings received are higher and the number of user ratings also increase.

Similarly, the restaurant rating received also differ when the Ambience option provided by the restaurant change. When the restaurant provides casual ambience the reviews received are higher and the ratings are higher towards 3.5-4.5.

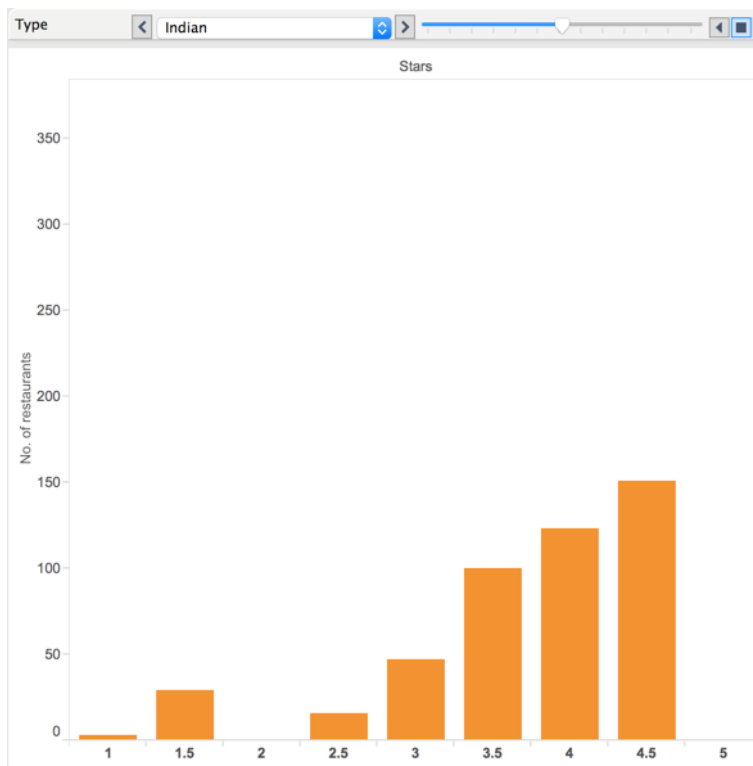
Price Range Vs Stars



Ambience Vs Stars



Ethnicity Vs Ratings



The graph here shows how different ethnicity have performed in US and received a different rating. Here we can see all the types of ethnicity having different rating received and different number of reviews.

DATA MODELING:

We used following tools to run the regression model:

- R Studio
- Azure

1. R Studio

After cleaning the Dataset in we run the cleaned data using R Studio.

We used random forest as they handle most any kind of data so little pre-processing is required (missing value corrections, binning, correlation analysis, etc. generally aren't needed)

Outliers generally aren't a problem.

Below is the code used to run the model

```
##### DATA MODELING (RANDOM FOREST) #####
library("caret")
set.seed(123)
inTrain <- createDataPartition(business_review1$stars, p=0.70, list=F)
trainData <- business_review1[inTrain, ]
crossValidationData <- business_review1[-inTrain, ]
dim(trainData)
na.omit(trainData)
install.packages("randomForest")
library("randomForest")
# fit random forest
rf.1 <- randomForest(as.factor(stars) ~. , data=trainData, importance=TRUE, ntree=500)

#### calculate variable importance ####
imp<-importance(rf.1)
vars<-dimnames(imp)[[1]]
imp<-data.frame(vars=vars,imp=as.numeric(imp[,1]))
imp<-imp[order(imp$imp,decreasing=T),]
imp

# Select important variables for retraining
#set.seed(42)
selected<-c(as.character(imp[1:17,1]),'stars')
#selected

# train again
rf.2<-randomForest(stars~.,data=trainData[,selected],replace=T,ntree=500)
# Perform prediction

#### plot variable importance ####
varImpPlot(rf.1,main='Variable Importance : Final Model',pch=16,col='blue')
predict_rf <- predict(rf.1, crossValidationData, type = "class")
install.packages("e1071")
library("e1071")
confusion_matrix <- confusionMatrix(predict_rf,crossValidationData$stars)
confusion_matrix
```

	Reference								
Prediction	1	2	3	4	5	6	7	8	9
1	0	0	0	0	0	0	0	0	0
2	0	0	1	2	0	0	0	0	0
3	2	4	11	5	1	4	1	1	0
4	3	11	27	54	48	25	9	2	0
5	0	1	20	112	210	139	18	2	0
6	0	3	3	33	140	330	202	29	1
7	0	0	0	8	41	180	335	129	9
8	0	0	1	0	4	8	50	64	13
9	0	0	0	0	1	2	0	6	2

Overall Statistics

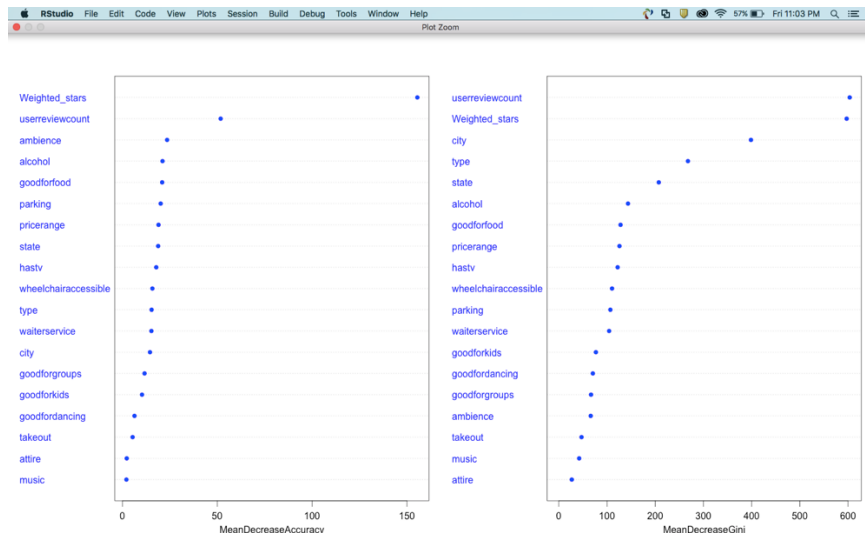
Accuracy : 0.4361
 95% CI : (0.4157, 0.4566)
 No Information Rate : 0.2982
 P-Value [Acc > NIR] : < 2.2e-16

Confusion Matrix derived from the model is as shown in figure. From confusion matrix we can see that for 3 star value of a restaurant 11 of them were predicted correctly and 13 of them were predicted wrong. And from R we got the accuracy of 43.61% in random forest.

The importance table derived from the above code was as follows:

The below importance table shows that factors such as goodforgroups, goodforfood, ambience, parking, alcohol are some important factors which can be considered as input.

vars	imp
Weighted_stars	12.1501455
goodforgroups	8.6406785
goodforfood	7.4214971
ambience	5.9910124
parking	5.9887656
userreviewcount	5.9722590
business_id	5.9667759
alcohol	5.4544845
goodforkids	5.3633933
wheelchairaccessible	4.9965672
takeout	4.9872166
city	3.8754777
hastv	2.7184643
state	1.6758664
attire	1.3794906
goodfordancing	0.6804768
music	0.0000000
type	-0.3022467
waiterservice	-1.4874499
pricerange	-1.7847673

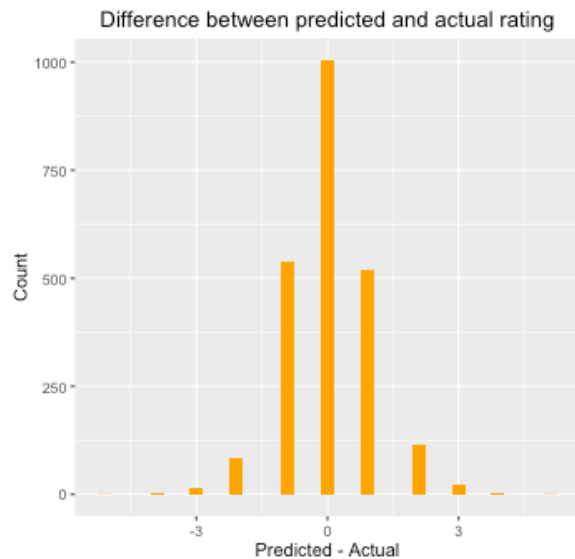


```
# Create prediction table
all.predictions <- data.frame(actual=crossValidationData$stars,random.forest = predict_rf)
all.predictions <- gather(all.predictions,key = model,value = predictions,2)
all.predictions$actual <- as.numeric(as.character( all.predictions$actual ))
all.predictions$predictions <- as.numeric(as.character( all.predictions$predictions ))
all.predictions$difference <- all.predictions$predictions-all.predictions$actual
str(all.predictions)

##plot the prediction difference
g <- ggplot(all.predictions)
g <- g + geom_histogram(aes(x=difference),fill="orange",binwidth=.3)
g <- g + ggtitle("Difference between predicted and actual rating")
g <- g + ylab("Count") + xlab("Predicted - Actual")
g

#Calculate the percentage as if how many predictions where within +1 and -1 range
difference <- as.data.frame(table(all.predictions$difference))
difference
(difference[2,2]+difference[3,2]+difference[4,2]+difference[5,2]+difference[6,2])/sum(difference[,2])
```

The prediction table derived from the model gave us a result with deviation of + 1 and -1 is as follows:



This shows that the values predicted are in range of +1 or -1 which is close to the actual rating.

2. MS Azure

After visualization and understanding the data better we build the model in Azure. We have used Boosted Decision Tree Regression, Decision Forest Regression, Linear Regression and Bayesian Linear Regression model and tried to evaluate on the basis of RMSE and AME to determine best model for our dataset.

Before building model we pre-processed the data using Meta-Data Editor and Project Column. In Meta-Data Editor we specified the categorical values like Type, State and City. In Project Column we selected the variable that were important in building a model keeping in mind about user's perspective as if inputs that we will take in the front end is relevant to user.

The following are the columns that we selected in building our model.

Begin With

ALL COLUMNS

NO COLUMNS

Include

column names

state ✕

stars ✕

type ✕

pricerange ✕

waiterservice ✕

alcohol ✕

goodforgroups ✕

goodforkids ✕

parking ✕

goodforfood ✕

ambience ✕



hastv ✕

userreviewcount ✕

Weighted_stars ✕













After pre-processing we visualized the data in Azure whether the dataset is clean or not using summarize data which gives the result of each column and its detail. Below is screen shot of how our data looks when we visualize summary data:

rows 14 columns 23

view as  






Feature	Count	Unique Value Count	Missing Value Count	Min	Max	Mean	Mean Deviation
type	7699	13	0				
state	7699	9	0				
stars	7699	9	0	1	9	5.995324	1.008533
Weighted_stars	7699	5	0	1	5	2.89648	0.680027
userreviewcount	7699	333	0	1	1114	38.247175	36.611605
pricerange	7699	5	0	1	5	1.682167	0.584436
waiterservice	7699	3	0	1	3	2.554488	0.569751
hastv	7699	3	0	1	3	2.523185	0.547604
alcohol	7699	4	0	1	4	2.978958	0.719253
goodforkids	7699	3	0	1	3	2.784258	0.348427
goodforgroups	7699	3	0	1	3	2.884271	0.209452
parking	7699	6	0	0	5	0.800104	0.454975
goodforfood	7699	7	0	0	6	0.901676	0.416933
ambience	7699	4	0	0	3	0.706196	0.425974

After the data is pre-processed we split the data in to 70% Train Data and 30% Test Data and ran all model and evaluated the models in the end. The following are results of score models:

stars	Weighted_stars	userreviewcount	pricerange	waiterservice	hastv	alcohol	goodforgroups	parking	goodforfood	ambience	Scored Labels
											
5	2	23	2	3	3	3	3	1	1	1	5.051242
5	3	6	1	2	2	4	3	1	1	1	5.783621
7	3	102	2	3	2	3	3	2	1	1	6.406258
4	1	11	1	2	2	4	3	1	1	0	4.106639
7	3	65	2	3	2	1	3	1	1	1	6.80371
6	3	27	2	3	3	3	3	1	1	1	6.107874

Score Model Results

EVALUATE MODEL:

Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
				
0.746468	0.966806	0.716603	0.508564	0.491436
0.751021	0.980631	0.720974	0.523211	0.476789

1st Row -> Bayesian Linear Regression 2nd row -> Decision Forest Regression

Results of 1st Evaluate Model

Metrics

Mean Absolute Error	0.733851
Root Mean Squared Error	0.950566
Relative Absolute Error	0.704491
Relative Squared Error	0.491621
Coefficient of Determination	0.508379

Metrics

Mean Absolute Error	0.748235
Root Mean Squared Error	0.969091
Relative Absolute Error	0.7183
Relative Squared Error	0.510969
Coefficient of Determination	0.489031

left metrics-> Boosted Decision Tree Regression

right metrics-> Linear Regression

Results of 2nd Evaluate Model

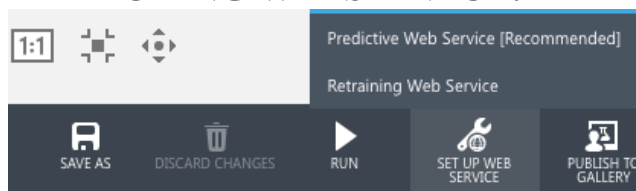
predictive model as it's RMSE and AME is **minimum** and Co-efficient of Determination is **maximum** as compared to other models.

The evaluation metrics available for regression models are: *Mean Absolute Error*, *Root Mean Squared Error*, *Relative Absolute Error*, *Relative Squared Error*, and the *Coefficient of Determination*.

The term "error" here represents the difference between the predicted value and the true value. The absolute value or the square of this difference are usually computed to capture the total magnitude of error across all instances, as the difference between the predicted and true value could be negative in some cases. The error metrics measure the predictive performance of a regression model in terms of the mean deviation of its predictions from the true values. Lower error values mean the model is more accurate in making predictions. An overall error metric of 0 means that the model fits the data perfectly.

The coefficient of determination, which is also known as R squared, is also a standard way of measuring how well the model fits the data. It can be interpreted as the proportion of variation explained by the model. A higher proportion is better in this case, where 1 indicates a perfect fit.

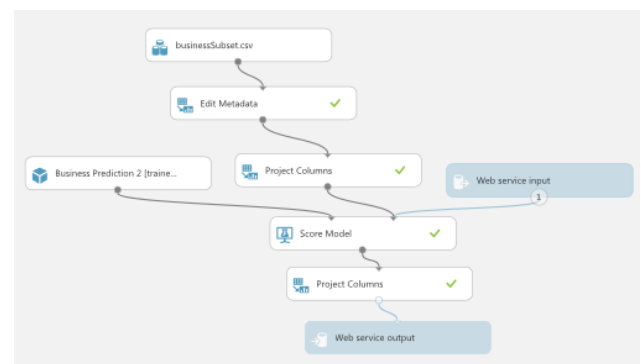
DEPLOYING WEBSERVICE:



trained model ready to be deployed as a scoring web service. Users of the web service will send input data to our model and our model will send back the prediction results.

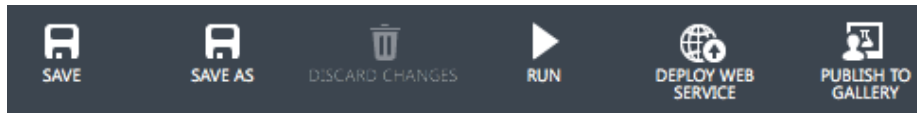
After running the predictive web service, we need to make certain changes in the predictive experiment created by azure. We need to modify web input and add a project column after the scored model in order to return the rating prediction for different attributes.

In our dataset we have Boosted Decision Tree Regression as the best predictive model. So for web service we need to select the train model of boosted decision tree regression in order to build the predictive experiment. By converting to a predictive experiment, we are getting your

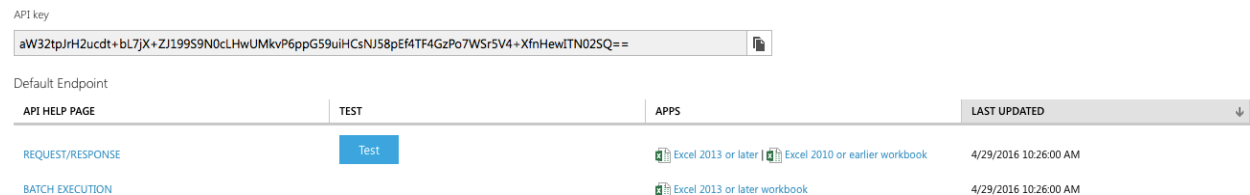


Predictive Experiment

Now we have our predictive experiment, we can deploy it as an Azure web service. Using the web service, users can send data to your model and the model will return its predictions.



After we deploy web service we are redirected to azure web service dashboard which has the API key and the Test link, on clicking the Test link a dialog pops up for the input data for the service. These are the columns expected by the scoring experiment on entering a set of data and then clicking OK the results are generated by the web service are displayed at the bottom of the dashboard.



Now we have API and the request/response service of azure web service which we can consume in our end to display the predicted value. If we Click **REQUEST/RESPONSE** under **API HELP PAGE** on the service Dashboard to view the API help page, aside from the URI, we have input and output definitions and code samples. The API input and response, for our web service, is shown below and is the payload of the API call.

Sample Request

```
{
  "Inputs": {
    "input1": {
      "ColumnNames": [
        "type",
        "state",
        "Weighted_stars",
        "userreviewcount",
        "pricerange",
        "waiterservice",
        "hastv",
        "alcohol",
        "goodforgroups",
        "parking",
        "goodforfood",
        "ambience"
      ],
      "type": "DataTable",
      "value": {
        "ColumnNames": [
          "Scored Labels"
        ],
        "ColumnTypes": [
          "Numeric"
        ],
        "Values": [
          [
            "0"
          ]
        ]
      }
    }
  }
}
```

Sample Response

```
{
  "Results": {
    "output1": {
      "type": "DataTable",
      "value": {
        "ColumnNames": [
          "Scored Labels"
        ],
        "ColumnTypes": [
          "Numeric"
        ],
        "Values": [
          [
            "0"
          ]
        ]
      }
    }
  }
}
```

.NET Framework:

We used .Net for using the API for prediction.

```
public static string PredictedSchool(InputPrediction inp)
{
    using (var client = new HttpClient())
    {
        var scoreRequest = new
        {
            Inputs = new Dictionary<string, StringTable>() {
                {
                    "input1",
                    new StringTable()
                    {
                        ColumnNames = new string[] { "Account", "Time", "day", "month", },
                        Values = new string[,] { { inp.account, inp.time, inp.day, inp.month }, { inp.account, inp.time, inp.day, inp.month }, }
                    }
                },
            },
            GlobalParameters = new Dictionary<string, string>()
            {
                {
                    " ",
                    ""
                }
            }
        };
        const string apiKey = "a+Mlyo1DFh/1lUgq2/3zFds1wpG9AO2ULIKIK4PTCLHv8cTytHay70uPqNzTxo/51oOd0u/hQYSGQwGPEsGFag==";
        client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", apiKey);
        client.BaseAddress = new Uri("https://ussouthcentral.services.azureml.net/workspaces/febda3c9e5e04052bf342c88e9df804a/services/1ede4738fc0a475490e93c");

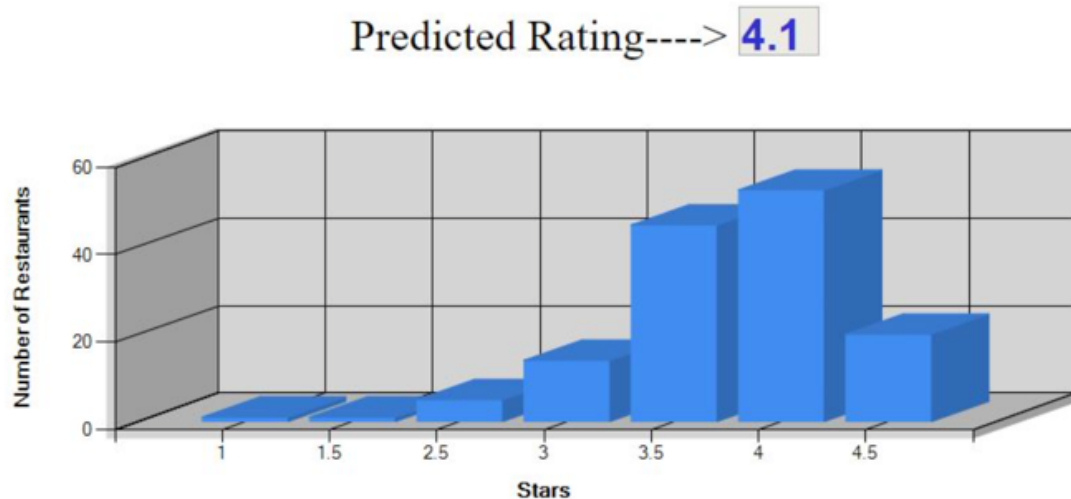
        HttpResponseMessage response = client.PostAsJsonAsync("", scoreRequest).Result;
        if (response.IsSuccessStatusCode)
        {
            string jsonDocument = response.Content.ReadAsStringAsync().Result;
            var responseBody = JsonConvert.DeserializeObject<PredictionResult>(jsonDocument);
        }
    }
}
```

We also published our website using Visual Studio and have also got a new domain for our website from GoDaddy.com. The following is the link of our website:

<http://team5finalproject.com>

The screenshot shows a web browser window with the URL team5finalproject.com. The page has a dark sidebar on the left with links: Home, User Type, Existing User, and New User. The main content area is white and contains a form for user registration. The form includes a "Good for Groups:" section with radio buttons for "Yes" (selected) and "No". Below this is a "Parking:" dropdown menu set to "No Parking". The "Food Speciality:" dropdown menu is set to "Lunch". The "Ambience:" dropdown menu is set to "Hipster". A "Submit" button is located below the form. At the bottom of the page, it displays "Predicted Rating----> 72%". The browser's address bar shows the URL, and the top of the browser window displays various tabs and a search bar.

We predicted the rating of existing users and also displayed a bar graph showing the number of restaurants received that rating in past to gauge their position using .NET and xml file.



CONCLUSION:

- From our website we can predict the rating to be received for a new restaurant to be opened depending upon the factors owner is planning to provide using .NET Framework for integrating the restful API generated from Azure Machine Learning
- We have developed a user friendly website giving existing users to get the current business rating of their business.

REFERENCE Links: (For .NET Integration)

Publishing Website:

<https://www.youtube.com/watch?v=7V8HikBP1vQ&nohtml5=False>

Machine Learning Design and Integration:

<https://www.youtube.com/watch?v=x-rAJdJUpGs&nohtml5=False>