ZOMATO RESTAURANT RATING PREDICTION

-

- SUMA REDDY DESHI REDDY

INFO 5810

FALL 2019

Introduction:

Zomato is India's biggest startup company where food industry meets the technology and connects several thousands of restaurants with one thread. It was started by Deepinder Goyal and Pankaj Chaddah in the year 2008. It gives information about the restaurant, menus, user reviews and provides food delivery options from partner restaurants in select cities. Zomato was initially started by the name Foodiebay in 2008 and was renamed to Zomato in the year 2010. It is the largest restaurant detecting company in India with more than 4200 restaurants across 12 cities in the country. Internationally, it has more than 1.4 million restaurants across 10,000 cities and at present it is present in 23 countries including India, Australia and Unites states.

Statement of the problem:

Zomato is used by million every day to decide where to eat over 10,000 cities across 23 countries. Zomato thrives on various ratings that are provided by the users for the restaurant. It is significant to analyze this data and find out whether they help in directing the performance of the restaurant or whether the restaurant performance is indeed explained by various other factors like location, cuisine, average cost. The purpose of this project is to extract features from the data to predict the rating given by the customers. In this project, I have built various models like Linear Regression, Random Forest, Decision Tree, XG Boost Regressor to find out which model best suits to predict the rating of the restaurant.

Objectives of the study:

I really get fascinated by the restaurants and food served by the restaurants so I would like to help customers find the best restaurants and restaurant owners improve their business. In this project, ratings given by the customers for the restaurants are predicted.

Review of Literature:

There are couple of Research papers published based on Restaurants Reviews and ratings. The paper Opinion mining for Thai Restaurant Ratings using neural networks is about classify the restaurant based on reviews. The model proposed in this paper is extraction of review from the social networking site using text processing, artificial neural networks to classify the data as positive and negative. Mrmr feature selection technique is used for selecting the features of data set. Location, Time, and Preference Aware Restaurant Recommendation Method paper they have proposed a system where the rating is computed by offline and online calculations. The offline calculations are done by considering the user's visiting trends, users cuisine preference, discovering restaurant's popularity and cost, modeling restaurants locality and the online calculation is done by calculating restaurants' distance and generating recommendation.

Data Collection:

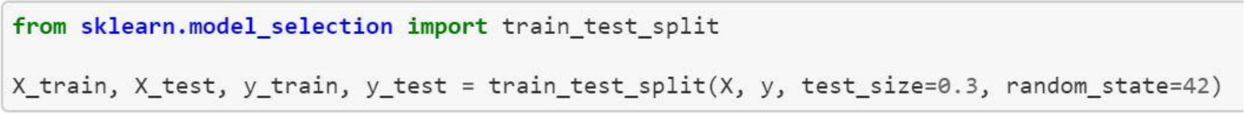
For this project, the dataset is collected from the Kaggle. The data is available in the form of two CSV files and they were put together in order to conduct the analysis. The dataset comprises of 21 columns like country, currency, average cost, ratings, latitude, longitude etc.

Restaurant ID	Identification Number
Restaurant Name	Name of the Restaurant
Country code	Country Code
City	Name of the city
Address	Address
Locality	Short Address of the Restaurant
Locality Verbose	Long Address of the Restaurant
Longitude	Longitude
Latitude	Latitude
Cuisines	Type of cuisines served
Average cost for two	Avg cost if two people visit the restaurant
Currency	currency
Has Table Booking	Can we book tables in restaurant ? yes/no
Has Online Delivery	Can we have online delivery? Yes/no

Switch to Order menu	Switch to order menu? Yes/no
Price range	Categorized price between 1-4
Aggregate rating	Categorizing rating between 1-5
Rating text	Different colors represents customer rating.
Rating color	Different rating like Excellent, very good, good, avg, poor, not rated
Votes	No, of votes received by restaurant from customers.

Splitting the data into training data and testing data:

Before the data preprocessing, the data was split in a 70%:30% ratio where 20% contains the test data and 70% comprising the training data. The splitting of training and test set is done to replicate the situation where you have past information and are building a model which helps you to test on future information which is yet to be known. Mostly, anything performed on the training data should not be informed by the test data

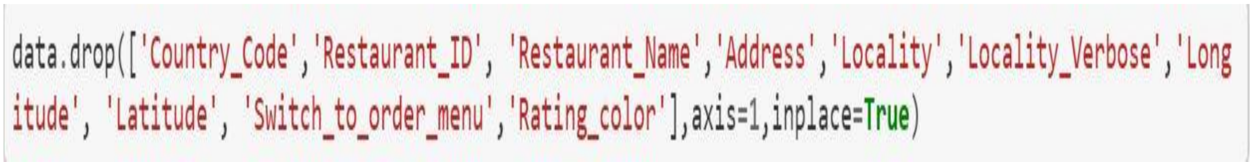


```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Here, X data frame has predictor variable and Y data frame has target variable (Rating). X train is used to train the model using the predictor variables whereas X test has same features as X train which is used to test the model. Y train is used as a target variable while training the model whereas, Y train is compared with the predicted values after testing the model.

Data Cleaning:

The data consists of many categorical columns which are not necessary for my analysis and there are null values in the column cuisine. The below mentioned columns are dropped as they contain redundant information and few of them have 100% correlation with other features and they have no significance in predicting the rating of the restaurants. There are two columns named rating color and rating text in my data which represents the same data in different format so one of the column i.e., Rating color is dropped. Moreover, Switch to order menu column does not contain switch to order option so that column is dropped as well.

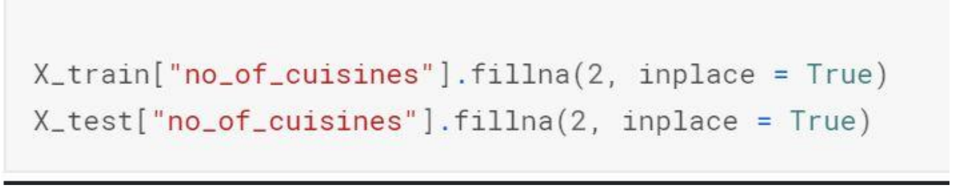


```
data.drop(['Country_Code','Restaurant_ID', 'Restaurant_Name','Address','Locality','Locality_Verbose','Longitude', 'Latitude', 'Switch_to_order_menu','Rating_color'],axis=1,inplace=True)
```

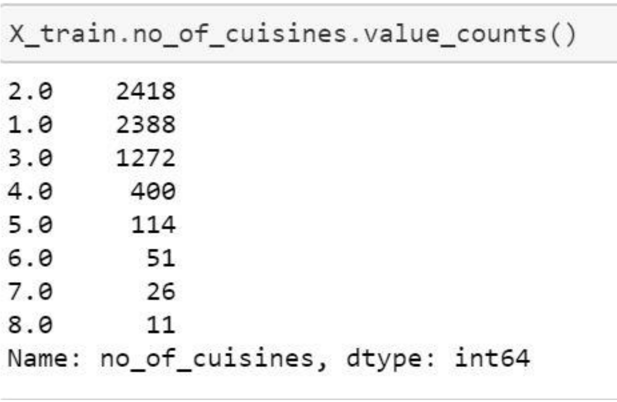
Then I have renamed the remaining feature names without gaps as gaps in the column may create trouble while indexing.

Handling missing values:

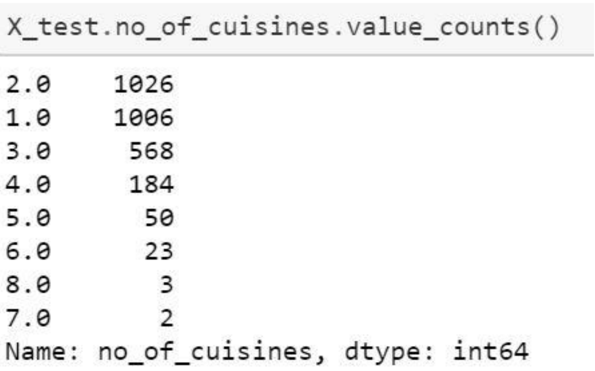
There are 9 null values in the cuisine column, those null values are treated by creating a feature named number of cuisines and replacing the null values by 2.0. As 2.0 is the mode value of number of cuisines column for both training and testing data.



```
X_train["no_of_cuisines"].fillna(2, inplace = True)
X_test["no_of_cuisines"].fillna(2, inplace = True)
```



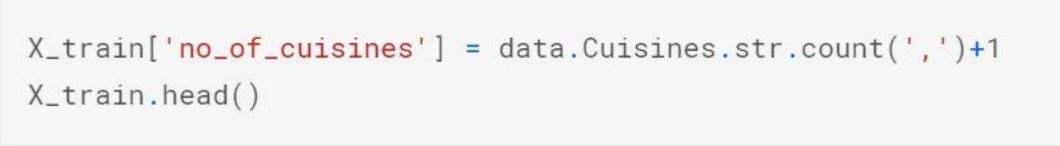
```
X_train.no_of_cuisines.value_counts()

2.0    2418
1.0    2388
3.0    1272
4.0     400
5.0     114
6.0      51
7.0      26
8.0      11
Name: no_of_cuisines, dtype: int64
```



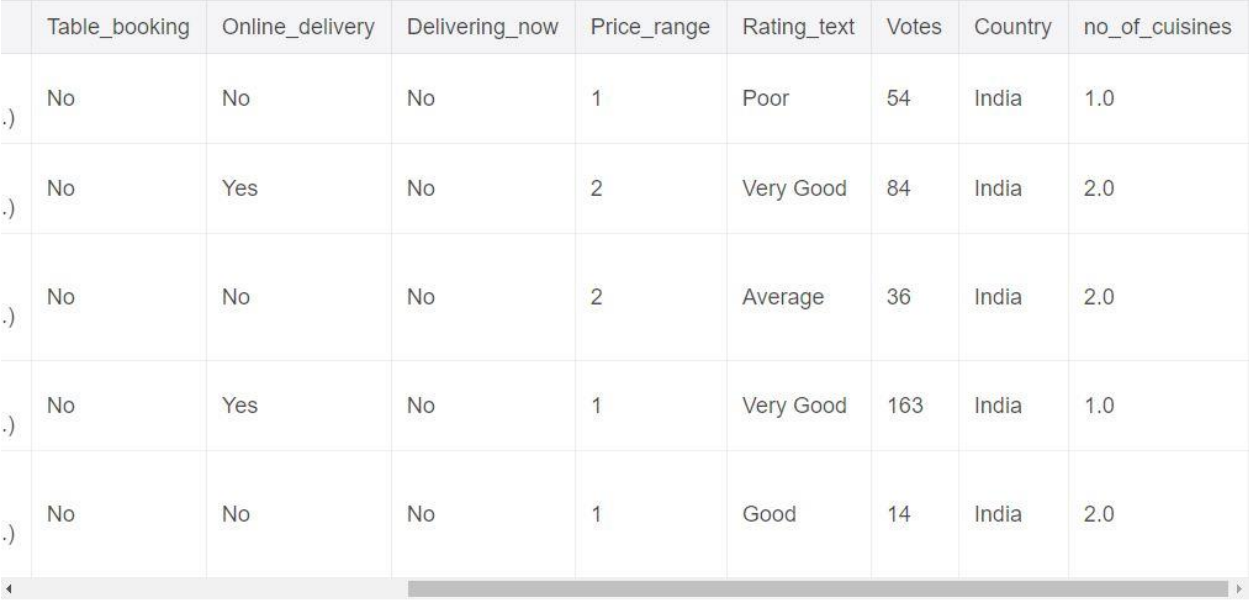
```
X_test.no_of_cuisines.value_counts()

2.0    1026
1.0    1006
3.0     568
4.0     184
5.0      50
6.0      23
8.0       3
7.0       2
Name: no_of_cuisines, dtype: int64
```

Feature Engineering:

The column cuisine has several hundreds of different cuisines and few of the same cuisines are written in different spellings. So, it gets very difficult to rectify those. Therefore, while building a model, when encoding will be done, it will be heavy on computing and may lead to the curse of dimensionality. So, I have created a new feature named number of feature column which counts the number of cuisines that a restaurant offers.

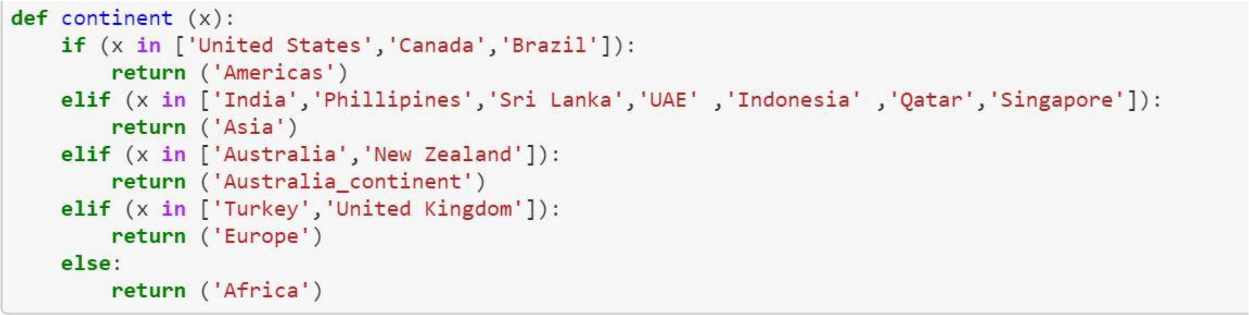


```
X_train['no_of_cuisines'] = data.Cuisines.str.count(',')+1
X_train.head()
```



	Table_booking	Online_delivery	Delivering_now	Price_range	Rating_text	Votes	Country	no_of_cuisines
.)	No	No	No	1	Poor	54	India	1.0
.)	No	Yes	No	2	Very Good	84	India	2.0
.)	No	No	No	2	Average	36	India	2.0
.)	No	Yes	No	1	Very Good	163	India	1.0
.)	No	No	No	1	Good	14	India	2.0

Then, a new feature named continent is created which contains respective countries that belong to a continent. This feature is used to give the model some additional information. Below, a function is created which helps in assigning continents to their respective countries.

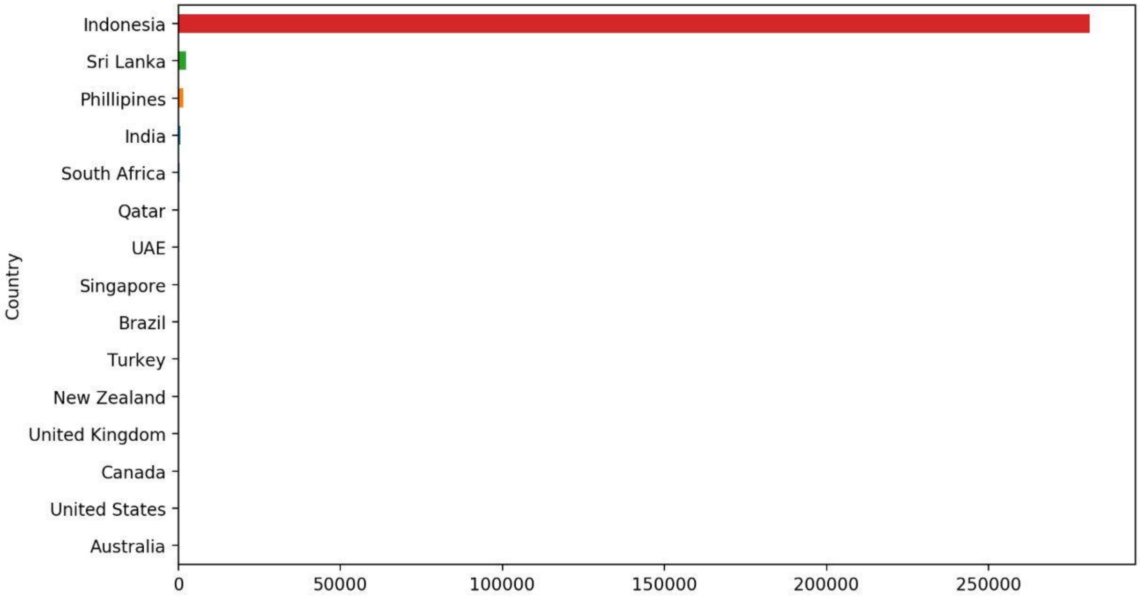


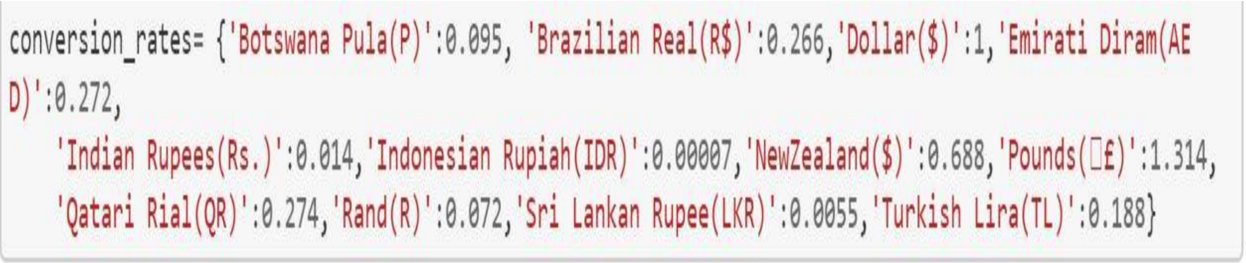
```
def continent (x):
    if (x in ['United States','Canada','Brazil']):
        return ('Americas')
    elif (x in ['India','Phillipines','Sri Lanka','UAE' ,'Indonesia' ,'Qatar','Singapore']):
        return ('Asia')
    elif (x in ['Australia','New Zealand']):
        return ('Australia_continent')
    elif (x in ['Turkey','United Kingdom']):
        return ('Europe')
    else:
        return ('Africa')
```

As model reads the numeric values, values have been assigned to the rating text with Excellent being the highest (5) and poor being the lowest (1). These rating texts are replaced by those numbers in train and test data so that this feature can be included in the model. Encoding is also done for online booking, delivering now, table booking columns by assigning binary digits 1 to yes and 0 to No.

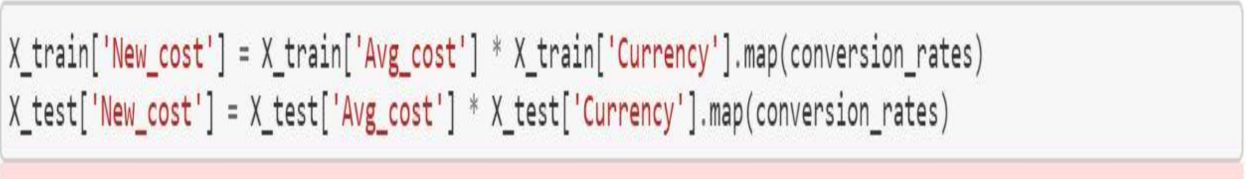
Normalization:

Initially, lets plot the average cost of restaurants at different countries. From the below plot, its obvious that the cost the customers pay in Indonesia is very high comparing to other countries. This is because, in the given data average cost is mentioned in their respective currency. So, I have standardized the currency to dollar.





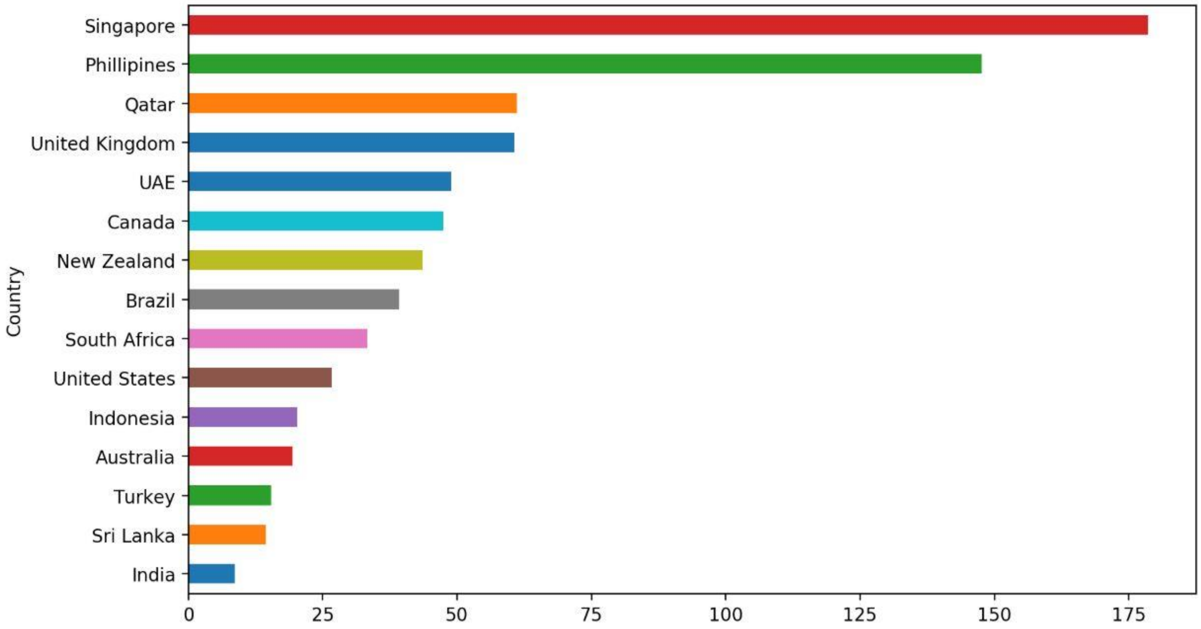
```
conversion_rates= {'Botswana Pula(P)':0.095, 'Brazilian Real(R$)':0.266,'Dollar($)':1,'Emirati Diram(AE
D)':0.272,
    'Indian Rupees(Rs.)':0.014,'Indonesian Rupiah(IDR)':0.00007,'NewZealand($)':0.688,'Pounds(□£)':1.314,
    'Qatari Rial(QR)':0.274,'Rand(R)':0.072,'Sri Lankan Rupee(LKR)':0.0055,'Turkish Lira(TL)':0.188}
```



```
X_train['New_cost'] = X_train['Avg_cost'] * X_train['Currency'].map(conversion_rates)
X_test['New_cost'] = X_test['Avg_cost'] * X_test['Currency'].map(conversion_rates)
```

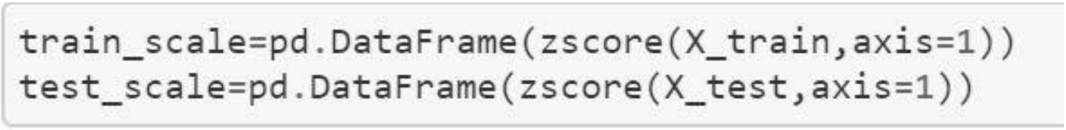
Standardization, the plot of average cost at different After countries is shown below,

After the standardization, t



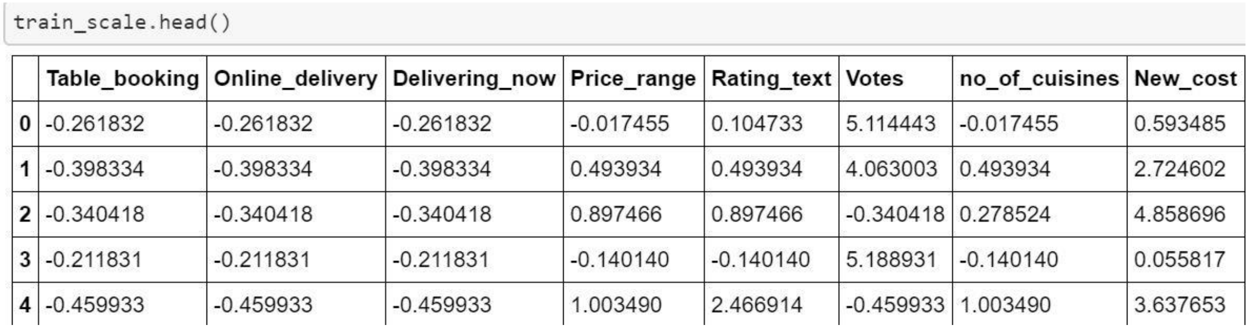
Data Scaling:

As range of values of data are varying widely, data scaling is done to standardize the range of features of data. Z score is applied here for the purpose of data scaling.



```
train_scale=pd.DataFrame(zscore(X_train,axis=1))
test_scale=pd.DataFrame(zscore(X_test,axis=1))
```

Data scaling has transformed the original data into,

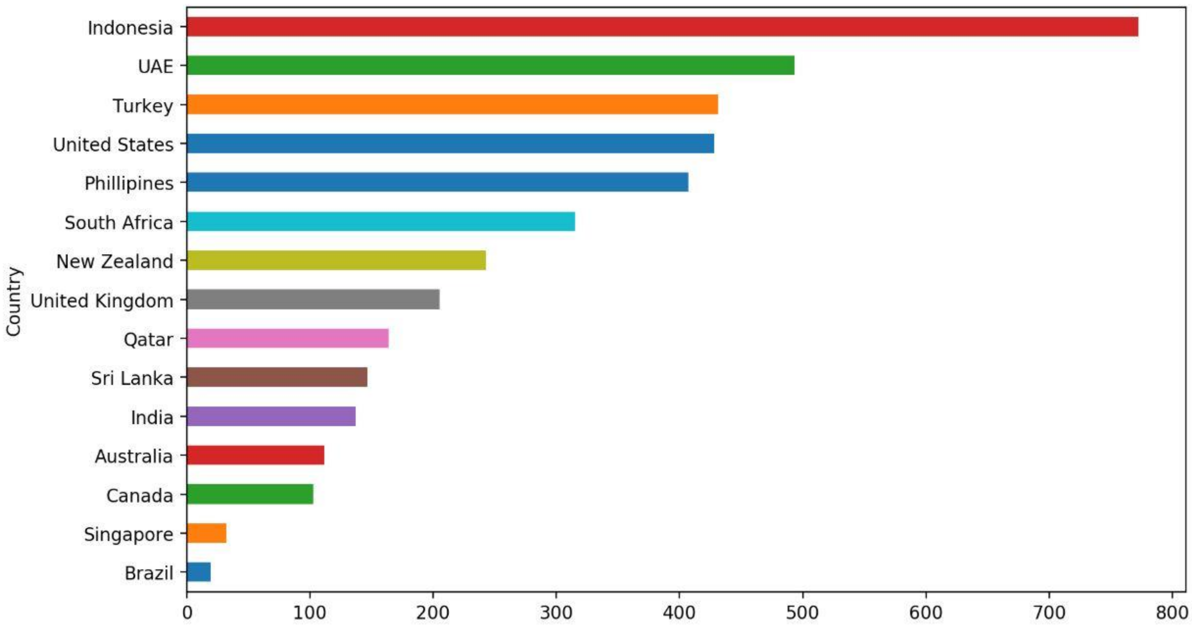


```
train_scale.head()
```

	Table_booking	Online_delivery	Delivering_now	Price_range	Rating_text	Votes	no_of_cuisines	New_cost
0	-0.261832	-0.261832	-0.261832	-0.017455	0.104733	5.114443	-0.017455	0.593485
1	-0.398334	-0.398334	-0.398334	0.493934	0.493934	4.063003	0.493934	2.724602
2	-0.340418	-0.340418	-0.340418	0.897466	0.897466	-0.340418	0.278524	4.858696
3	-0.211831	-0.211831	-0.211831	-0.140140	-0.140140	5.188931	-0.140140	0.055817
4	-0.459933	-0.459933	-0.459933	1.003490	2.466914	-0.459933	1.003490	3.637653

Data Visualization:

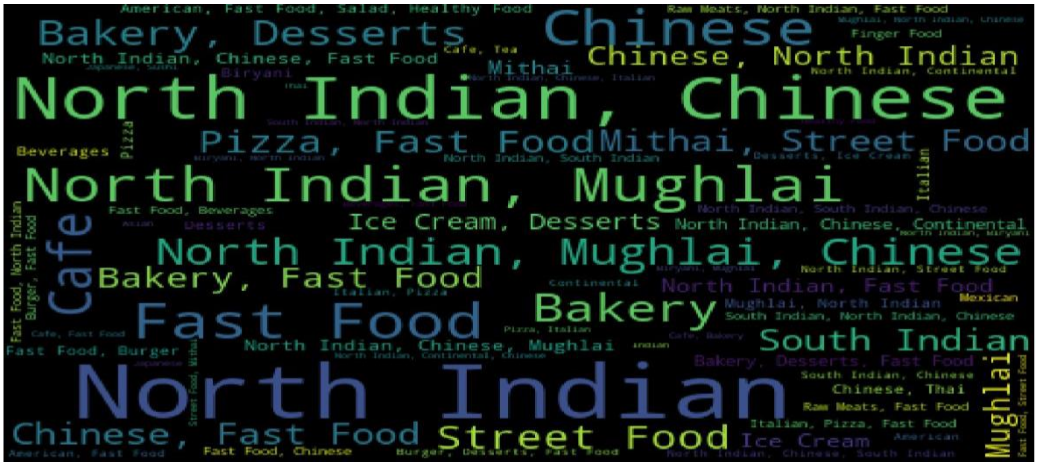
Country vs Votes



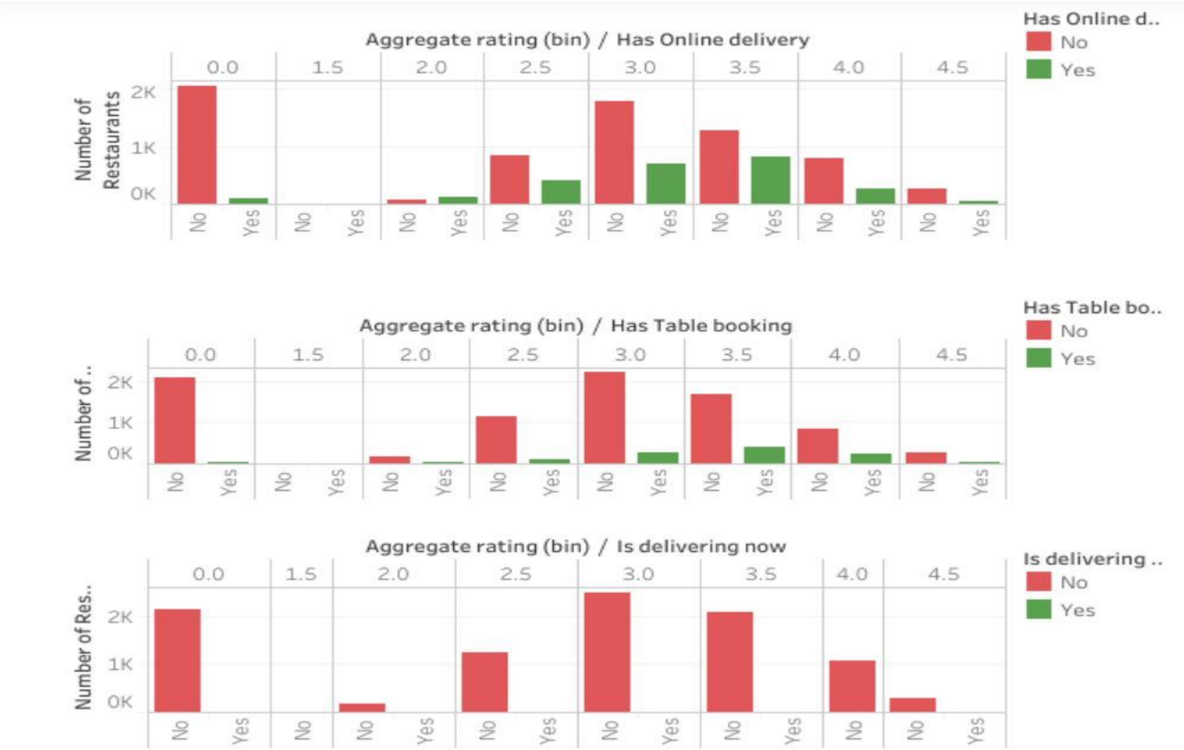
From the above figure, It is obvious that restaurants in Indonesia have highest number of votes while Brazil has lowest number of votes.

Wordcloud:

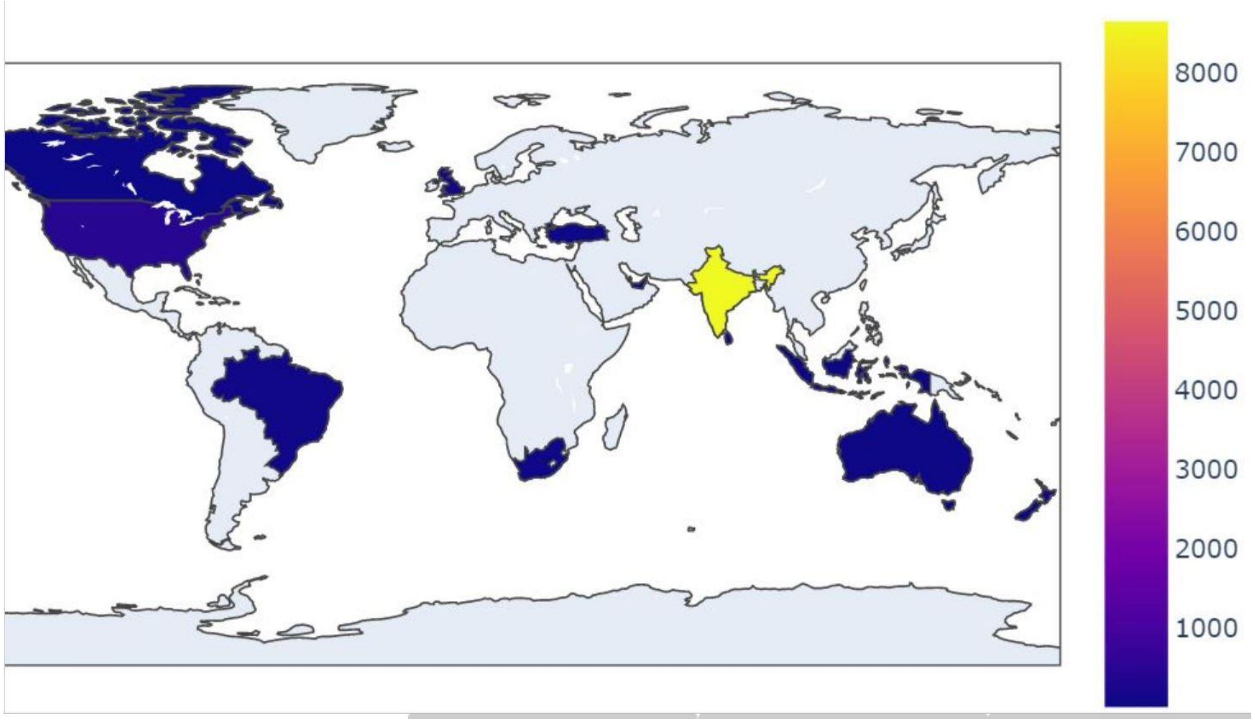
A word cloud is the graphical representation of frequently used words in the column cuisine. The height of each word in the above picture is an indication of frequency of occurrences of the words.



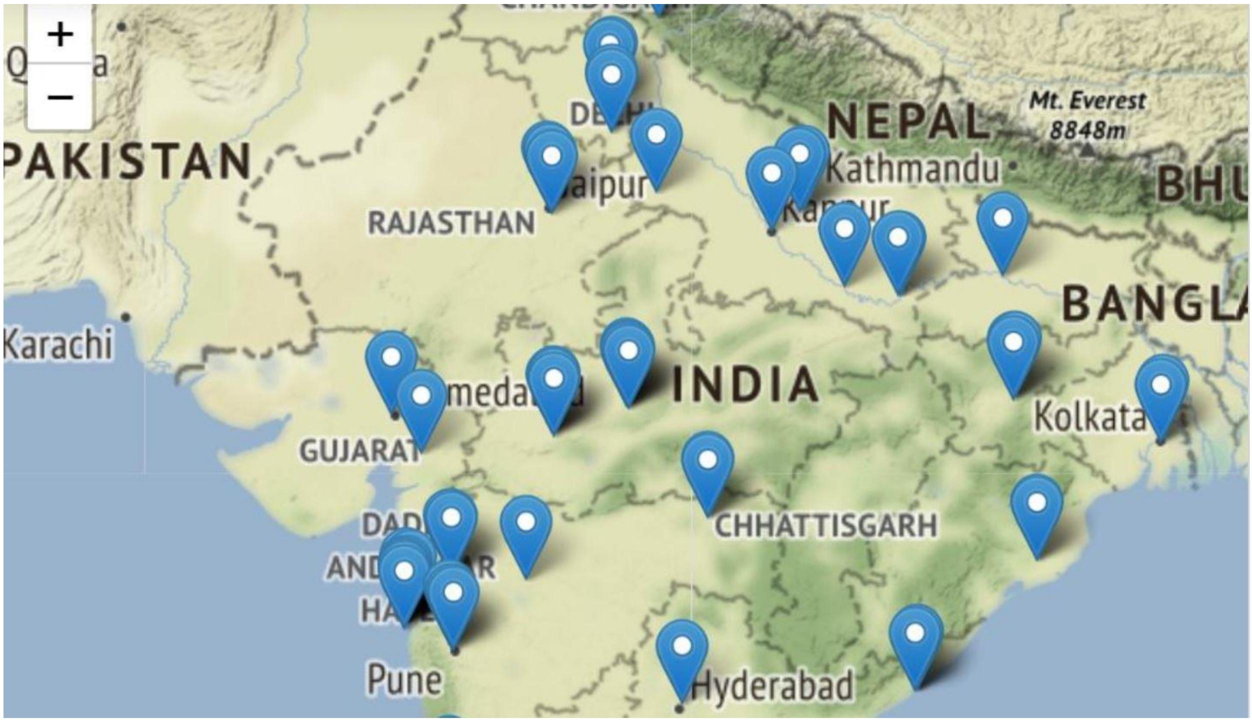
Services provided/Ratings:



Number of restaurants registered on Zomato by country:



Geographical location of restaurants:



Classification Models:

Scikit-learn is a free software machine learning algorithm for the python programming language. It contains several regression, classification and clustering algorithms like random forest, k means, gradient boosting. After cleaning the data, I have utilized various classification methods like Linear Regression, Decision Tree, Random Forest, XG Boost Regressor to predict the ratings of the restaurant.

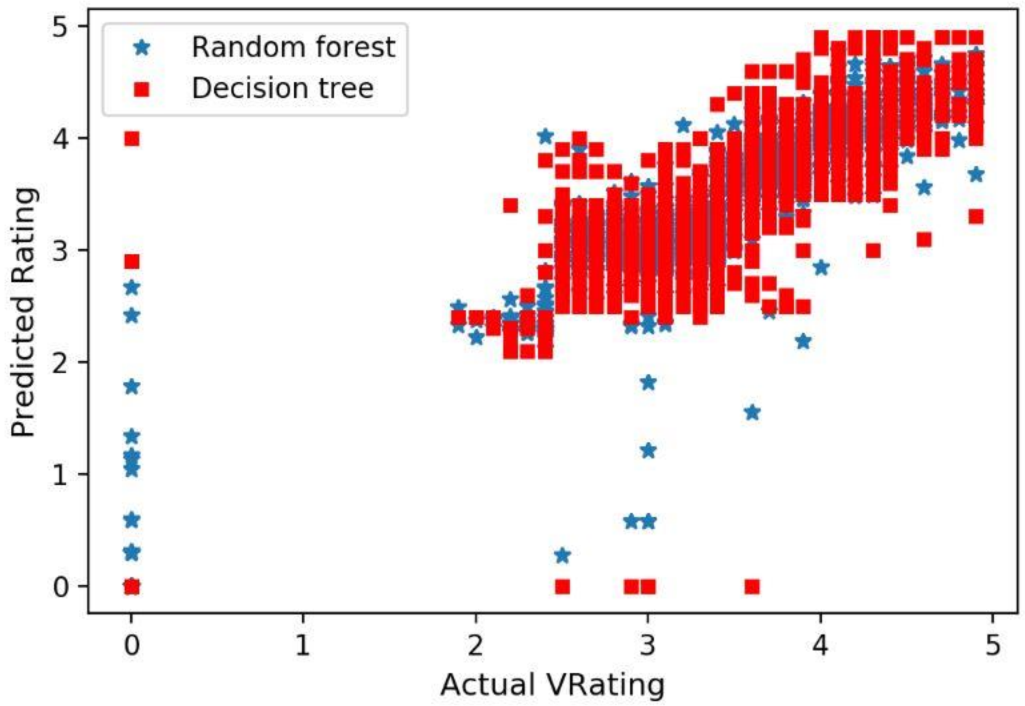
Linear Regression: Linear regression model is implemented using Scikit-learn. Linear Regression is a linear approach to model the relationship between dependent variable and one or more independent variables. When the data is fitted into the linear regression algorithm, the RMSE score obtained through Linear regression is 0.5887.

Decision Tree Regression: Decision tree regression knows feature of an object and trains the model in the structure of a tree to predict future data to get meaningful continuous output. When decision tree regressor is fit to the data, RMSE score obtained through Decision tree regressor is 0.346.

Random Forest: Random Forest is a learning method for regression, classification which operates by building a multitude of decision trees at training time and outputting the class which is the mode of the classes or mean prediction of the individual trees. RMSE score obtained through Random Forest is 0.269.

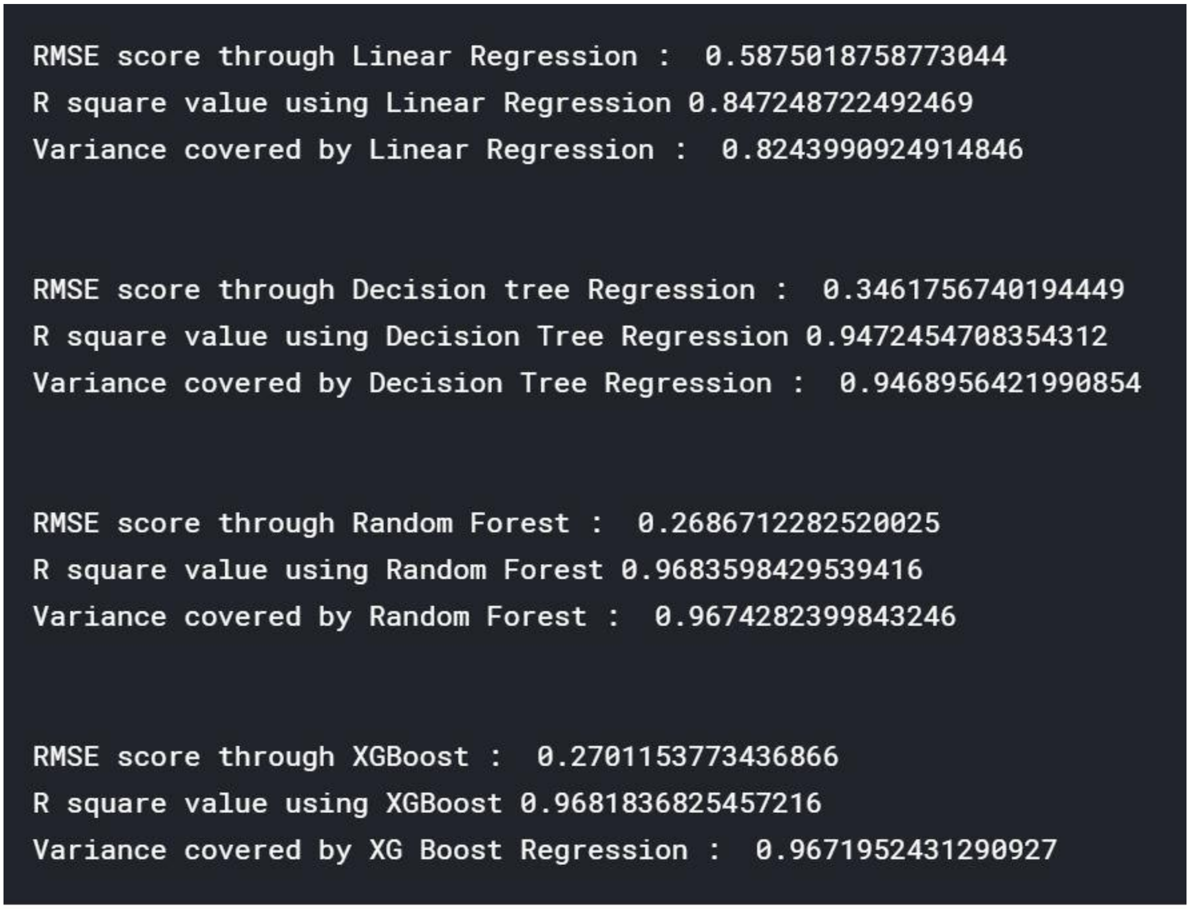
XG Boost Regressor: XG boost is an implementation of gradient boosted decision trees designed for performance and speed. RMSE score through XG Boost is 0.270.

Ideally when the models predict exactly as the actual values, there will be a straight line passing through the origin. As, 100% accuracy is impossible, a slim distribution of points is accepted. Accuracy of the model is better if the distribution is slimmer. From the below scatter plot, it is obvious that decision tree has more spread out predictions when compared to slimmer scatter points of Random Forest. It is visually proven that the Random Forest has better performance than the Decision Tree.



Result:

Variance explained by XG Boost regressor and Random Forest are almost equal. Generally, Random Forest have lower RMSE square compared to decision tree, which is obvious is our case study. Linear Regression is under performed due to the assumptions of linear Regression algorithm which is bit offset from the real case. Among the four models, Random Forest has the least RMSE score which indirectly tells that it is the most accurate model. R squared value and variance explained are almost equal.



```
RMSE score through Linear Regression :  0.5875018758773044
R square value using Linear Regression 0.847248722492469
Variance covered by Linear Regression :  0.8243990924914846


RMSE score through Decision tree Regression :  0.3461756740194449
R square value using Decision Tree Regression 0.9472454708354312
Variance covered by Decision Tree Regression :  0.9468956421990854


RMSE score through Random Forest :  0.2686712282520025
R square value using Random Forest 0.9683598429539416
Variance covered by Random Forest :  0.9674282399843246


RMSE score through XGBoost :  0.2701153773436866
R square value using XGBoost 0.9681836825457216
Variance covered by XG Boost Regression :  0.9671952431290927
```

Reference:

https://rstudio-pubs-static.s3.amazonaws.com/296592_29cd90970f7f4e18958a215416803927.ht

https://www.academia.edu/34604851/Sales_Management_Report_On_Zomato

https://www.zomato.com/blog/annual-report-19

http://ijrar.com/upload_issue/ijrar_issue_20542895.pdf