ML HACKATHON

Project Title: Zomato Restaurants Rating

Team Name: Knowhere

Team Members:

- Shubhang Mittal (IMT2016049)
- Ashish Singh (IMT2016077)
- Rahil Vijay (IMT2016062)

Problem Statement:

To predict the rating of the restaurants based on different attributes and parameters

Dataset Source:

https://www.kaggle.com/shrutimehta/zomato-restaurants-data#zomato.csv

Why we chose this dataset:

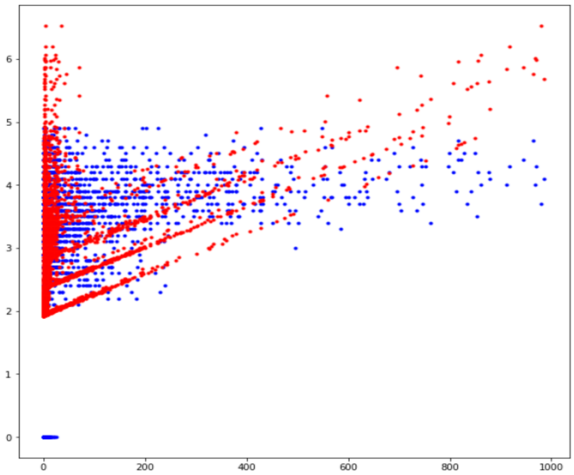
We chose this dataset because it has an adequate amount of data and all the necessary information (attributes) required for the prediction of the rating of restaurants. Since the dataset consists of the restaurants from all over the world, our model is not limited to India.

Approaches:

Our model uses regression algorithms instead of classification algorithms, so we tried different approaches to make our model better and more accurate.

- First, we used linear regression for our model.
  But, after using linear regression, we found out that our data is non-linear and hence doesn't follow linear regression. And therefore, the accuracy was bad.

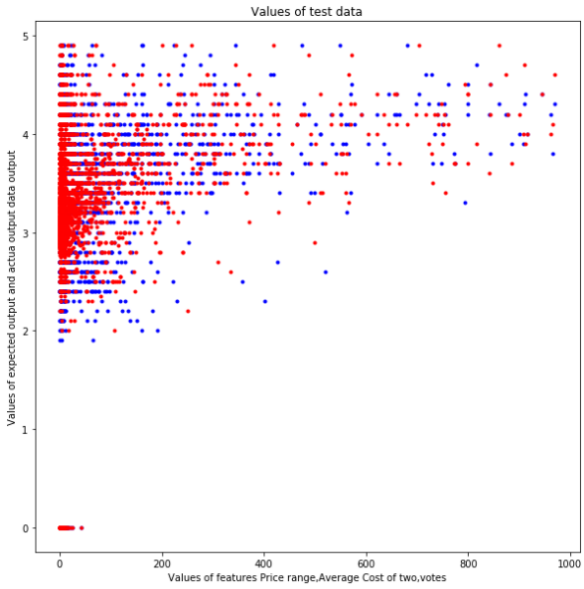
```
In [16]: plt.figure(figsize=(10,10))
         # plt.plot(train_X,train_Y,'g.')
         plt.plot(test_X,test_Y,"b.")
         plt.plot(test_X,y_pred,'r.')
         plt.show()
```




```
In [17]: lm.score(test_X,test_Y)
Out[17]: 0.2786326289099138
```

- Then, we used decision tree for our model.
  Although we got a good accuracy using this model, we thought of improving it more. That is why, we thought of going with random forest instead of decision tree.

```
In [13]: #plot between y actual and y_predicted with respect to test input
plt.figure(figsize=(10,10))
plt.plot(test_X,test_Y,"b.")
plt.plot(test_X,y_pred,"r.")
plt.title('Values of test data')
plt.xlabel("Values of features Price range,Average Cost of two,votes")
plt.ylabel("Values of expected output and actua output data output")
plt.show()
```




```
In [12]: #our accuracy
regressor.score(test_X, test_Y)
```

```
Out[12]: 0.9154302116655946
```

- Finally, we decided to go with Random Forest and the accuracy had significantly improved from what we got from decision tree.

```
In [58]: #plot between y actual and y_predicted with respect to test input
plt.figure(figsize=(10,10))
plt.plot(test_X,test_Y,"b.")
plt.plot(test_X,y_pred,"r.")
plt.title('Values of test data')
plt.xlabel("Values of features Price range,Average Cost of two,votes")
plt.ylabel("Values of expected output and actua output data output")
plt.show()
```


```
In [57]: #our accuracy
regressor.score(test_X, test_Y)
```

```
Out[57]: 0.9339318094107611
```

Analysis of Dataset:

Our dataset consists of various attributes such as ‘Restaurant ID’, ‘Restaurant Name’, ‘Country Code’, ‘City’, ‘Address’, ‘Locality’, ‘Locality Verbose’, ‘Longitude’, ‘Latitude’, ‘Cuisines’, ‘Average Cost for two’, ‘Currency’, ‘Has Table booking’, ‘Has Online delivery’, ‘Is delivering now’, ‘Switch to order menu’, ‘Price range’, ‘Aggregate rating’, ‘Rating color’, ‘Rating text’ and ‘Votes’.

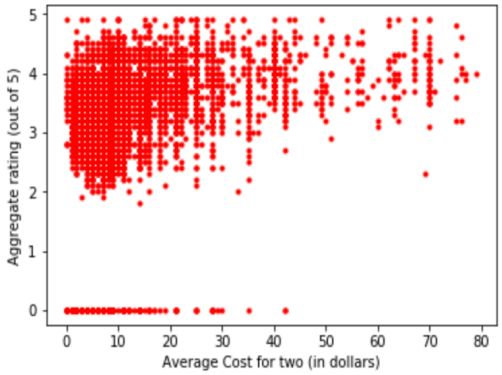
We thought of all the attributes that would be important for deciding the rating of a restaurant and finally used ‘Average Cost for two’, ‘Price range’ and ‘Votes’ to find the rating of the restaurants which resulted in good accuracy of the model.

Visualization:

Following are the dependencies of ‘Aggregate rating’ on different attributes.

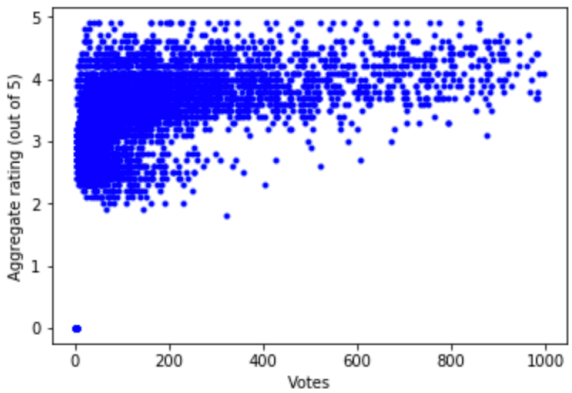
- Average Cost for two

```
In [164]: hy = data['Aggregate rating']
          hx = data['Average Cost for two']
          plt.xlabel('Average Cost for two (in dollars)', fontsize=10)
          plt.ylabel('Aggregate rating (out of 5)', fontsize=10)
          plt.plot(hx,hy,"r.")
          plt.show()
```



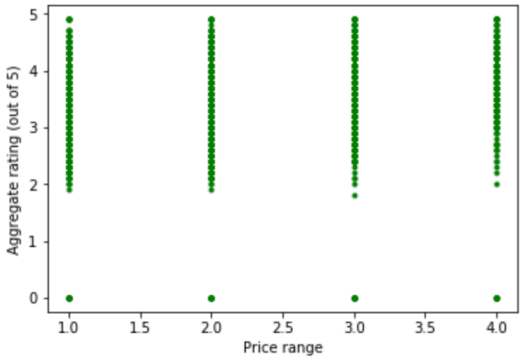
- Votes

```
In [162]: hy = data['Aggregate rating']
          hx = data['Votes']
          plt.xlabel('Votes', fontsize=10)
          plt.ylabel('Aggregate rating (out of 5)', fontsize=10)
          plt.plot(hx,hy,"b.")
          plt.show()
```



- Price range

```
In [163]: hy = data['Aggregate rating']
          hx = data['Price range']
          plt.xlabel('Price range', fontsize=10)
          plt.ylabel('Aggregate rating (out of 5)', fontsize=10)
          plt.plot(hx,hy,"g.")
          plt.show()
```



Feature Engineering:

Let's start with loading our entire dataset.

```
In [70]: data = pd.read_csv("zomato.csv")
         data.head()
```

Out[70]:

ss	Locality	Locality Verbose	Longitude	Latitude	Cuisines	...	Currency	Has Table booking	Has Online delivery	Is delivering now	Switch to order menu	Price range	Aggregate rating	Rating color	Rating text	Votes
ird or, ıry ıll, an ı...	Century City Mall, Poblacion, Makati City	Century City Mall, Poblacion, Makati City, Mak...	121.027535	14.565443	French, Japanese, Desserts	...	Botswana Pula(P)	Yes	No	No	No	3	4.8	Dark Green	Excellent	314
tle ٠o, 77 no es ıe, ı...	Little Tokyo, Legaspi Village, Makati City	Little Tokyo, Legaspi Village, Makati City, Ma...	121.014101	14.553708	Japanese	...	Botswana Pula(P)	Yes	No	No	No	3	4.5	Dark Green	Excellent	591
sa ri- , 1 en ıy, ıs, l...	Edsa Shangri-La, Ortigas, Mandaluyong City	Edsa Shangri-La, Ortigas, Mandaluyong City, Ma...	121.056831	14.581404	Seafood, Asian, Filipino, Indian	...	Botswana Pula(P)	Yes	No	No	No	4	4.4	Green	Very Good	270
ird or, ga on iM ıll, ı...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.056475	14.585318	Japanese, Sushi	...	Botswana Pula(P)	No	No	No	No	4	4.9	Dark Green	Excellent	365
ird or, ga ın, iM ıll, ı...	SM Megamall, Ortigas, Mandaluyong City	SM Megamall, Ortigas, Mandaluyong City, Mandal...	121.057508	14.584450	Japanese, Korean	...	Botswana Pula(P)	Yes	No	No	No	4	4.8	Dark Green	Excellent	229

By closely looking at it, we figured that 'Votes' and 'Average Cost for two' played a major role in determining the 'Aggregate rating' of a restaurant.

```
In [81]: data[['Votes','Average Cost for two','Aggregate rating']].head()
```

Out[81]:

	Votes	Average Cost for two	Aggregate rating
12	294	75	4.8
14	223	65	4.3
15	29	75	3.6
16	72	79	4.0
18	118	56	4.5

We can compute some basic statistical measures on these features.

```
In [82]: data[['Votes','Average Cost for two','Aggregate rating']].describe()
```

Out[82]:

	Votes	Average Cost for two	Aggregate rating
count	9189.000000	9189.000000	9189.000000
mean	97.571444	9.720971	2.605702
std	163.963980	10.706836	1.511741
min	0.000000	0.000000	0.000000
25%	4.000000	4.000000	2.400000
50%	27.000000	7.000000	3.200000
75%	112.000000	11.000000	3.600000
max	997.000000	79.000000	4.900000

After using random forest using these two attributes, we get an accuracy of 0.93.

```
In [100]: #our accuracy
          regressor.score(test_X, test_Y)
```

```
Out[100]: 0.9325481218486282
```

So, we involved another attribute called ‘Price range’ as ‘Aggregate rating’ weekly depended on ‘Price range’ as well.

```
In [108]: data[['Votes','Average Cost for two','Price range','Aggregate rating']].describe()
```

Out[108]:

	Votes	Average Cost for two	Price range	Aggregate rating
count	9189.000000	9189.000000	9189.000000	9189.000000
mean	97.571444	9.720971	1.756230	2.605702
std	163.963980	10.706836	0.874465	1.511741
min	0.000000	0.000000	1.000000	0.000000
25%	4.000000	4.000000	1.000000	2.400000
50%	27.000000	7.000000	2.000000	3.200000
75%	112.000000	11.000000	2.000000	3.600000
max	997.000000	79.000000	4.000000	4.900000

And finally, after running random forest, we got an accuracy of 0.9339

```
In [57]: #our accuracy
         regressor.score(test_X, test_Y)
```

```
Out[57]: 0.9339318094107611
```

Model Building and training:

After trying different models, we finally selected Random Forest for our model.
We imported RandomForestRegressor from sklearn.ensemble and started building our model.

In [119]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import accuracy_score
```

Since, we had the data from all around the world, we needed to standardize the currencies of different regions to a fixed currency and we decided to convert every currency to dollars.

In [121]:
```
#storing all different currencies of different countries
cur=data['Currency']
cur = set(cur)
cur = list(cur)
```

In [122]:
```
#dictionary of different currencies and their conversion rate to dollars
dic={}
dic[cur[0]] = 0.094 #currency in dollars
dic[cur[1]] = 0.19
dic[cur[2]] = 0.000068
dic[cur[3]] = 0.27
dic[cur[4]] = 1.28
dic[cur[5]] = 0.69
dic[cur[6]] = 1
dic[cur[7]] = 0.27
dic[cur[8]] = 0.27
dic[cur[9]] = 0.071
dic[cur[10]] = 0.0057
dic[cur[11]] = 0.014
```

In [123]:
```
#converting all currencies to dollars
for i in range(9551):
    data['Average Cost for two'][i] = data['Average Cost for two'][i] * dic[data['Currency'][i]]
```

Now, we needed to segregate the data and drop the outliers.

In [124]:
```
#dropping all the outliers with avg cost for two>$80 it won't effect the data too much and it would make it more logica
data = data[data['Average Cost for two']<80]
#dropping all votes >1000 as they are working as outliers
data = data[data['Votes']<1000]
#splitting the data between train data and test data
train_Y,test_Y,train_X,test_X = train_test_split(data['Aggregate rating'],data[['Average Cost for two','Votes','Price r
train_X.describe()
```

Out[124]:

	Average Cost for two	Votes	Price range
count	7351.000000	7351.000000	7351.000000
mean	9.750646	97.194259	1.754319
std	10.795400	163.869051	0.872739
min	0.000000	0.000000	1.000000
25%	4.000000	4.000000	1.000000
50%	7.000000	27.000000	2.000000
75%	11.000000	111.000000	2.000000
max	77.000000	997.000000	4.000000

Finally, we train our data using random forest model.

In [127]:
```
#training of the model
regressor = RandomForestRegressor(n_estimators = 300)
regressor.fit(train_X, train_Y)
```

Out[127]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=300, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

Validation

We used K-fold cross validation to validate our model and calculated mean accuracy and standard deviation accuracy for our model.

In [128]:
```
# K-fold cross validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = regressor, X = train_X, y = train_Y, cv = 10)
mean_accuracy = accuracies.mean()
std_accuracy = accuracies.std()
std_accuracy
```

Test Metrices

- Accuracy score

In [57]:
```
#our accuracy
regressor.score(test_X, test_Y)
```

Out[57]: 0.9339318094107611