Final Project

Amal Tom and Chethan Manjunath

Introduction

Zomato is an Indian multinational restaurant aggregator and food delivery company. It provides information about restaurants, average cost for two, cuisines served, ratings etc. The dataset used for analysis is picked up from kaggle.com. This analysis would be most suitable for foodies who are looking to explore cuisines from various cities of the world based on their budget. The dataset has the following components:

- 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
- Switch to order menu
- Price range
- Aggregate Rating
- Rating colour
- Rating text
- Votes

Objective

To apply the concepts from Network Theory, Machine Learning and Optimization, and draw insights that would assist a person answer the following questions

1) How to minimise the spending while visiting a set of predefined criteria of restaurants?
2) What is the shortest path in which I can visit all the restaurants?
3) If I plan to visit a particular restaurant, are there any other restaurants that I can visit on the way to minimise the travel time covering multiple restaurants?
4) If a new restaurant was introduced in a particular city, what would be the average cost for two?

5) If a new restaurant was introduced in a particular city, will the restaurant deliver food?

Exploratory Data Analysis

1) What is the mean, median, standard deviation of various attributes of data?

	Restaurant ID	Country Code	Longitude	Latitude	Average Cost for two	Price range	Aggregate rating	Votes
count	9.551000e+03	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000	9551.000000
mean	9.051128e+06	18.365616	64.126574	25.854381	1199.210763	1.804837	2.666370	156.909748
std	8.791521e+06	56.750546	41.467058	11.007935	16121.183073	0.905609	1.516378	430.169145
min	5.300000e+01	1.000000	-157.948486	-41.330428	0.000000	1.000000	0.000000	0.000000
25%	3.019625e+05	1.000000	77.081343	28.478713	250.000000	1.000000	2.500000	5.000000
50%	6.004089e+06	1.000000	77.191964	28.570469	400.000000	2.000000	3.200000	31.000000
75%	1.835229e+07	1.000000	77.282006	28.642758	700.000000	2.000000	3.700000	131.000000
max	1.850065e+07	216.000000	174.832089	55.976980	800000.000000	4.000000	4.900000	10934.000000

2) Are there null values in the dataset? There are only 9 null values in cuisines. Apart from that there are no null values.

```
Restaurant ID           0
Restaurant Name         0
Country Code            0
City                    0
Address                 0
Locality                0
Locality Verbose        0
Longitude               0
Latitude                0
Cuisines                9
Average Cost for two    0
Currency                0
Has Table booking       0
Has Online delivery     0
Is delivering now       0
Switch to order menu    0
Price range             0
Aggregate rating        0
Rating color            0
Rating text             0
Votes                   0
dtype: int64
```

3) How many restaurants are present for each country in the dataset? The dataset mostly contains restaurants from India.

```
India             8652
United States      434
United Kingdom      80
Brazil              60
UAE                 60
South Africa        60
New Zealand         40
Turkey              34
Australia           24
Phillipines         22
Indonesia           21
Singapore           20
Qatar               20
Sri Lanka           20
Canada               4
Name: Country, dtype: int64
```

Application of concepts to Dataset

Scenario

Let us imagine Adam - a foodie who recently started dating a girl and wants to take her out to restaurants in multiple cities. But he has no idea on where to start, how much it would cost etc. We will try to help him out through our analysis and provide insights so that he can impress her in the best way possible.

Optimization - Linear Programming application

Adam is planning to take his girlfriend to Manchester but his girlfriend is pretty demanding and wants to go out on dates 20 times during their visit. But it doesn't stop there as she has more specific demands:

1.They should visit an European restaurant at least 8 times.
2.They should visit a Fast Food restaurant at least 5 times.
3.They should visit an Asian restaurant at least 4 times.
4.They should have some other cuisine at least 2 times.
5.They cannot visit the same restaurant more than 3 times.
6.They cannot visit a restaurant with less than a 3.5 zomato rating.

Adam, who recently started working, does not have a lot of money and wants to spend as little as possible while satisfying all his girlfriend's demands. How can Adam achieve this?

Let us look at the data snapshot

Restaurant ID	Restaurant Name	Primary_Cuisine	Secondary_Cuisine	Tertiary_Cuisine	Average Cost for two	Aggregate rating
6800280	Monolos Playhouse Restaurant	American	Fast Food	Desserts	30	3.3
6800443	Santos	Pizza	Fast Food	None	20	3.3
6801867	Manchester House	British	None	None	85	4.0
6800235	The Grill On The Alley	Steak	Seafood	Grill	55	4.4
6801051	Nawaab	Pakistani	Indian	Afghani	35	3.9
6800577	Jamie's Italian	Italian	None	None	50	3.9
6800569	Chaophraya	Thai	None	None	70	4.3
6801374	Solita	American	Burger	Grill	50	4.9
6801963	Almost Famous Burgers	Burger	American	None	40	4.0
6801329	Home Sweet Home	British	Burger	Cafe	30	4.1
6801395	Teacup	British	Contemporary	None	45	4.1
6800538	Archies	Fast Food	American	None	15	3.4
6800593	Zouk Tea Bar & Grill	Indian	Seafood	None	50	3.6
6800908	Mughli	Indian	Curry	None	35	4.5
6800678	Lahore	Indian	Grill	None	25	3.7
6800892	Gaucho	Argentine	American	None	80	4.5
6800263	Akbars	Indian	None	None	30	4.2
6801039	San Carlo	Italian	None	None	25	4.3

We preprocessed the data to split the cuisines into multiple columns and then created dummy columns to indicate the type of cuisine served by them.

Restaurant ID	Restaurant Name	Average Cost for two	Aggregate rating	Primary_Cuisine_Type_Asian	Primary_Cuisine_Type_European	Primary_Cuisine_Type_Fast Food	Primary_Cuisine_Type_Others
6800280	Monolos Playhouse Restaurant	30	3.3	0	0	1	0
6800443	Santos	20	3.3	0	0	1	0
6801867	Manchester House	85	4.0	0	1	0	0
6800235	The Grill On The Alley	55	4.4	0	0	0	1
6801051	Nawaab	35	3.9	1	0	0	0
6800577	Jamie's Italian	50	3.9	0	1	0	0
6800569	Chaophraya	70	4.3	1	0	0	0
6801374	Solita	50	4.9	0	0	1	0
6801963	Almost Famous Burgers	40	4.0	0	0	1	0
6801329	Home Sweet Home	30	4.1	0	1	0	0
6801395	Teacup	45	4.1	0	1	0	0
6800538	Archies	15	3.4	0	0	0	1

Using the pulp package in python the problem is formulated

```
df3_4 = df3_3[df3_3["Aggregate rating"]>3.5]
```

```
restaurant = pulp.LpVariable.dicts('Restaurant',(name for name in df3_4['Restaurant Name']),
                                   lowBound=0, upBound=3,
                                   cat='Continuous')
```

```
#add objective
prob += lpSum([restaurant[df3_4.loc[i,'Restaurant Name']]*int(df3_4.loc[i,'Average Cost for two'])
               for i in df3_4.index])

#add contraints
#European restaurant for atleast 8 times
prob += lpSum([restaurant[df3_4.loc[i,'Restaurant Name']]*int(df3_4.loc[i,'Primary_Cuisine_Type_European'])
               for i in df3_4.index]) >=8
#Fast food restaurants for atleast 5 times
prob += lpSum([restaurant[df3_4.loc[i,'Restaurant Name']]*int(df3_4.loc[i,'Primary_Cuisine_Type_Fast Food'])
               for i in df3_4.index]) >=5
#Asian restaurant for atleast 4 times
prob += lpSum([restaurant[df3_4.loc[i,'Restaurant Name']]*int(df3_4.loc[i,'Primary_Cuisine_Type_Asian'])
               for i in df3_4.index])>=4
#other cuisine for atleast twice
prob += lpSum([restaurant[df3_4.loc[i,'Restaurant Name']]*int(df3_4.loc[i,'Primary_Cuisine_Type_Others'])
               for i in df3_4.index])>=2

|
```

The solution for this problem is

```
Visit to Restaurant_Manchester_House will be :  0.0
Visit to Restaurant_The_Grill_On_The_Alley will be :  2.0
Visit to Restaurant_Nawaab will be :  0.0
Visit to Restaurant_Jamie's_Italian will be :  0.0
Visit to Restaurant_Chaophraya will be :  0.0
Visit to Restaurant_Solita will be :  2.0
Visit to Restaurant_Almost_Famous_Burgers will be :  3.0
Visit to Restaurant_Home_Sweet_Home will be :  3.0
Visit to Restaurant_Teacup will be :  2.0
Visit to Restaurant_Zouk_Tea_Bar_&_Grill will be :  0.0
Visit to Restaurant_Mughli will be :  0.0
Visit to Restaurant_Lahore will be :  3.0
Visit to Restaurant_Gaucho will be :  0.0
Visit to Restaurant_Akbars will be :  1.0
Visit to Restaurant_San_Carlo will be :  3.0
Visit to Restaurant_Mr_Cooper's_House_&_Garden___The_Midland will be :  0.0
Visit to Restaurant_The_French_by_Simon_Rogan___The_Midland will be :  0.0

The minimum amount he will spend in a month is  690.0
```

Adam would need to visit ‘The Grill on Alley’ **2 times**, ‘Solita’ **2 times**, ‘Almost Famous Burgers’ **3 times**, ‘Home Sweet Home’ **3 times**, ‘Teacup’ **2 times**, ‘Lahore’ **3 times**, ‘Akbars’ **1 time** and ‘San Carlo’ **3 times** to minimise the cost.

The minimum amount that Adam would end up spending is **690 pounds.**

Output from Linear optimizer:

```
Problem MODEL has 4 rows, 17 columns and 17 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 4 (0) rows, 16 (-1) columns and 16 (-1) elements
0  Obj 170.99999 Primal inf 15.199996 (4)
4  Obj 690
Optimal - objective value 690
After Postsolve, objective 690, infeasibilities - dual 0 (0), primal 0 (0)
Optimal objective 690 - 4 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.02


'Optimal'
```

Application of Graph Theory

Scenario 1 : Adam with his girlfriend plans to visit Mumbai for a week-long trip. He decides to take her out to one restaurant each day. Adam wants to know if there are any other restaurants closer than the restaurant of interest in the same path he would take so that he can pay a visit to the closer restaurant than the one intended and reduce the expenditure on Uber.

To answer Adam's question, we use the following subset data.

Restaurant ID	Restaurant Name	Country Code	City	Longitude	Latitude
35217	Joey's Pizza	1	Mumbai	72.829976	19.126630
18447068	Cafe Hydro	1	Mumbai	72.862381	19.221315
18458563	The American Joint	1	Mumbai	72.841347	19.223840
18075122	The Fusion Kitchen	1	Mumbai	72.848923	19.254567
18233317	145 Kala Ghoda	1	Mumbai	72.832585	18.927584
18237753	Tea Villa Cafe	1	Mumbai	72.833984	19.055831
18388642	Grandmama's Cafe	1	Mumbai	72.827808	19.091458
18463285	Mumbai Vibe	1	Mumbai	72.832658	19.065838
16527711	The Rolling Pin	1	Mumbai	72.825203	18.994049
18313566	Farzi Cafe	1	Mumbai	72.827650	19.003517
49003	SpiceKlub	1	Mumbai	72.825553	18.994237
18441580	Tea Villa Cafe	1	Mumbai	72.972281	19.207222
34757	Joey's Pizza	1	Mumbai	72.834715	19.178321
18408295	Stacks And Racks	1	Mumbai	72.836191	19.181300

To detect the current location of Adam, we have to input the city, longitude and latitude. Using the Longitude and Latitude the distance from one restaurant to every other restaurant is computed. Using distance as weight, a network is created.

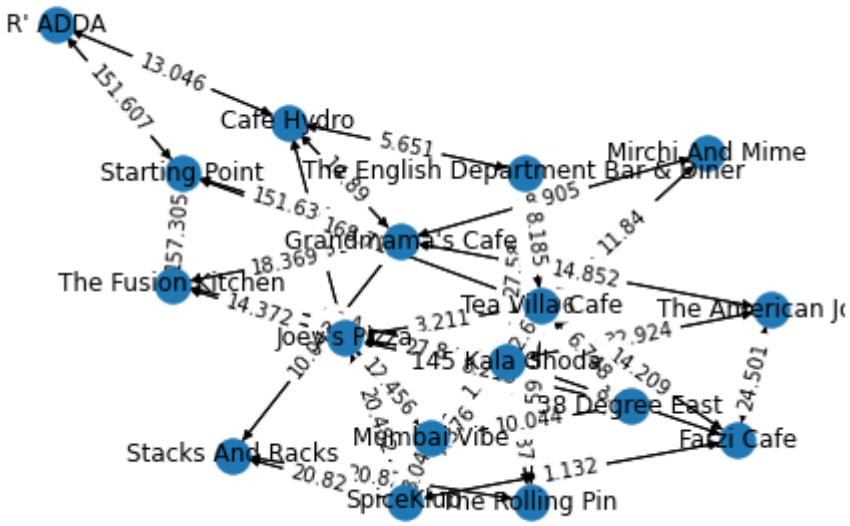
Code to input the current location

```
City=input("Enter City to visit : ")
Long=float(input("Enter Longitude : "))
Lat=float(input("Enter Latitude : "))
```

```
Enter City to visit : Mumbai
Enter Longitude : 71.4
Enter Latitude : 18.9
```

Function to compute distance

```python
from math import radians, cos, sin, asin, sqrt
def distance(lat1, lat2, lon1, lon2):

    # The math module contains a function named
    # radians which converts from degrees to radians.
    lon1 = radians(lon1)
    lon2 = radians(lon2)
    lat1 = radians(lat1)
    lat2 = radians(lat2)

    # Haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon / 2)**2

    c = 2 * asin(sqrt(a))

    # Radius of earth in kilometers. Use 3956 for miles
    r = 6371

    # calculate the result
    return(c * r)
```

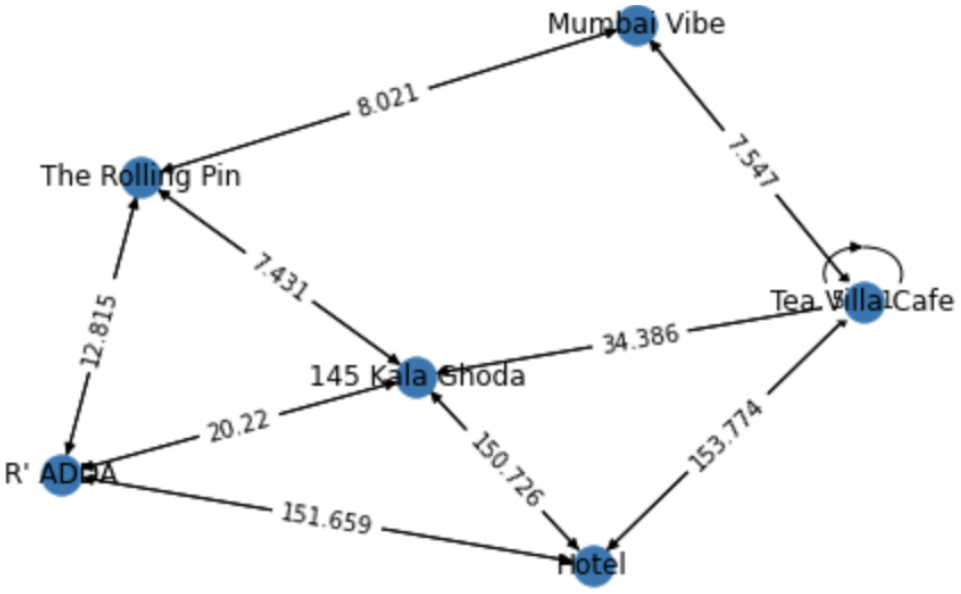


Using the single source dijkstra algorithm we can find the shortest distance from Adam's current location to the restaurant of interest. If there are any other restaurants in the same path, and are closer, the algorithm will return the values.

```
: ({'Starting Point': 0,
    'The Fusion Kitchen': 1,
    "Grandmama's Cafe": 1,
    'Tea Villa Cafe': 1,
    "R' ADDA": 1,
    '145 Kala Ghoda': 2,
    "Joey's Pizza": 2,
    'The American Joint': 2,
    'Cafe Hydro': 2,
    'Stacks And Racks': 2,
    'Mirchi And Mime': 2,
    'The Rolling Pin': 2,
    'Farzi Cafe': 2,
    'The English Department Bar & Diner': 2,
    'Mumbai Vibe': 2,
    '38 Degree East': 2,
    'SpiceKlub': 3},
```

```
{'Starting Point': ['Starting Point'],
 'The Fusion Kitchen': ['Starting Point', 'The Fusion Kitchen'],
 "Grandmama's Cafe": ['Starting Point', "Grandmama's Cafe"],
 'Tea Villa Cafe': ['Starting Point', 'Tea Villa Cafe'],
 "R' ADDA": ['Starting Point', "R' ADDA"],
 '145 Kala Ghoda': ['Starting Point', 'The Fusion Kitchen', '145 Kala Ghoda'],
 "Joey's Pizza": ['Starting Point', 'The Fusion Kitchen', "Joey's Pizza"],
 'The American Joint': ['Starting Point',
  "Grandmama's Cafe",
  'The American Joint'],
 'Cafe Hydro': ['Starting Point', "Grandmama's Cafe", 'Cafe Hydro'],
 'Stacks And Racks': ['Starting Point',
  "Grandmama's Cafe",
  'Stacks And Racks'],
 'Mirchi And Mime': ['Starting Point', "Grandmama's Cafe", 'Mirchi And Mime'],
 'The Rolling Pin': ['Starting Point', 'Tea Villa Cafe', 'The Rolling Pin'],
 'Farzi Cafe': ['Starting Point', 'Tea Villa Cafe', 'Farzi Cafe'],
 'The English Department Bar & Diner': ['Starting Point',
  'Tea Villa Cafe',
  'The English Department Bar & Diner'],
 'Mumbai Vibe': ['Starting Point', 'Tea Villa Cafe', 'Mumbai Vibe'],
 '38 Degree East': ['Starting Point', 'Tea Villa Cafe', '38 Degree East'],
 'SpiceKlub': ['Starting Point',
  'The Fusion Kitchen',
  '145 Kala Ghoda',
  'SpiceKlub']})
```

<u>Scenario 2 :</u> Adam and his girlfriend both have a sweet tooth. They decided to visit all restaurants in Mumbai which serves ‘Desserts’, using the Zomato database as a reference on a given day.They want to stop at their Hotel once to take a nap in between. How can they minimise their uber expenses



Using graph analytics, we can solve this by approximating the problem to a travelling salesman problem.

tsp = nx.approximation.traveling_salesman_problem
path3=tsp(G,cycle=False)

Where ‘G’ is the above graph. The optimal path from the analysis is:

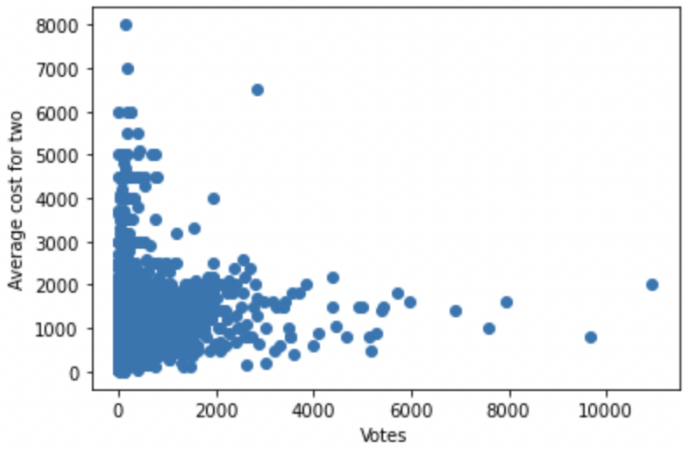
'The Rolling Pin' ➡️'Mumbai Vibe' ➡️'Tea Villa Cafe' ➡️’Hotel’ ➡️’R' ADDA’ ➡️'145 Kala Ghoda'

Machine Learning

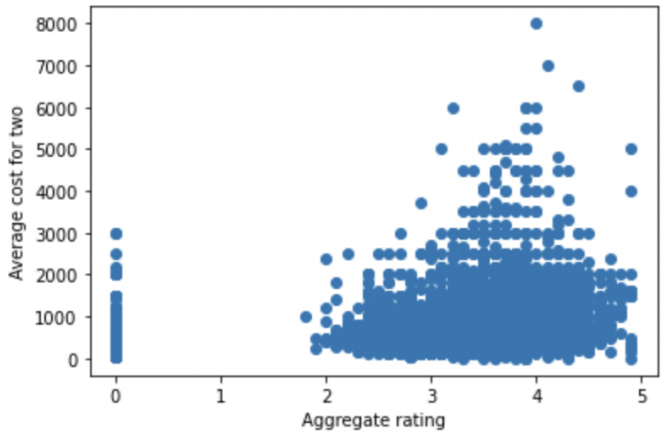
Objective : To predict the “average cost for two” given the location, ratings, votes, table bookings, delivery options etc. Prediction is made on the subset data filtering for India as the country.

Relationship check with the target variable

1) Scatter plot for votes vs average cost for two



2) Scatter plot for ratings vs average cost for two



Since categorical variables are present, dummy columns were created to encode the values

```
df5_1 = pd.get_dummies(df5, columns=['City','Has Table booking',
                                     'Has Online delivery','Is delivering now',
                                     'Switch to order menu','Rating color','Rating text'])
```

The dataset is split into 70% train and 30% test.

```
train_X,test_X,train_Y,test_Y = train_test_split(X,y,test_size = 0.3,random_state=1234)
```

The Xgboost package is used to make predictions. Since this algorithm can run in parallel, we use thread = 4. Grid search methodology is used to find the best combination of hyperparameters for the model.

```
estimator = xgbm.XGBRegressor(objective ='reg:linear',nthread = 4, seed=123)
```

```
parameters = {
    'max_depth': range (2, 8, 1),
    'n_estimators': range(5, 60, 50),
    'learning_rate': [0.1, 0.01, 0.05]
}
```

```
grid_search = GridSearchCV(
    estimator=estimator,
    param_grid=parameters,
    scoring = 'r2',
    n_jobs = -1,
    cv = 4,
    verbose=True)
```

The best estimated parameters for the model are selected as shown below. Some of the important selected parameters are Max_depth = 256, n_estimators = 55, learning_rate=0.1

```
grid_search.best_estimator_
```

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.1, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=55, n_jobs=4,
             nthread=4, num_parallel_tree=1, objective='reg:linear',
             predictor='auto', random_state=123, ...)
```

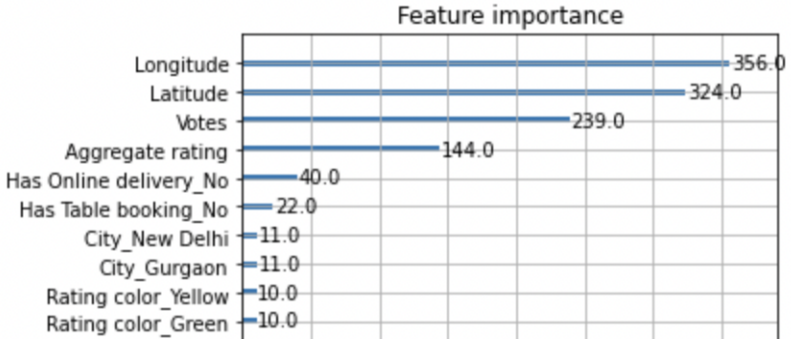
Xgboost model was fit using the best params from the grid search. The model achieved an r-sq of 55.37% on the test dataset.

```
[909]: pred3 = grid_search.predict(test_X)
       rmse3 = np.sqrt(MSE(test_Y,pred3))
       print(rmse3)
       rsq2 = r2_score(test_Y,pred3)
       print(rsq2)
```

```
392.68868896669017
0.5537272667048454
```

Visualising feature importance:

```
from xgboost import plot_importance
ax=plot_importance(xgbr)
fig = ax.figure
fig.set_size_inches(5, 12)
pyplot.show()
```



From the graph, we can see that the longitude and latitude, Votes and aggregate rating are the most important features in the model.

Application of decision tree

Objective: To predict if a restaurant present in a certain location has a home delivery service or not.

If Adam finds a new restaurant other than the ones in the list of restaurants present in the dataset, and is unsure if the restaurant has delivery service or not, the prediction made here will come in handy for Adam to plan a visit or order food accordingly.

Data Exploration

We can see that the Average Cost for two is an important differentiator for understanding whether a restaurant will deliver or not

From the above graph we can see that Votes is an important differentiator for understanding whether a restaurant will deliver or not

Model Building

Steps Undertaken:

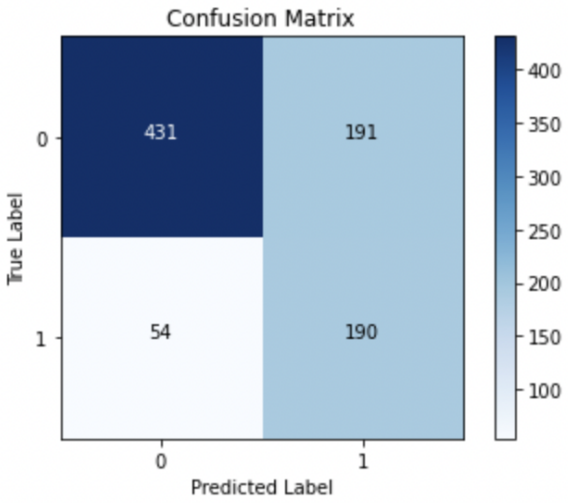
1. Data Treatment: We use a subset of data i.e., by filter for India as a country to make predictions.
2. Data Split: Split the dataset to 90% train and 10% test

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.10,random_state=1)
```

3. Used one hot encoding to treat the categorical variables
4. Used grid search the find the optimal parameters for the model using AUC-ROC as the primary metric
5. Constructed the decision tree using the optimal parameters. The optimal parameters are (max_depth - 6, n_estimators=30, learning_rate=0.3)

Model Results:

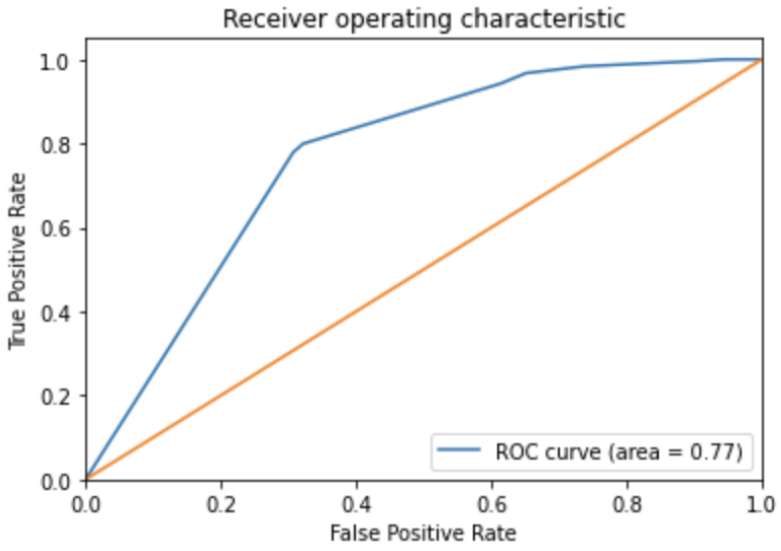
Confusion Matrix



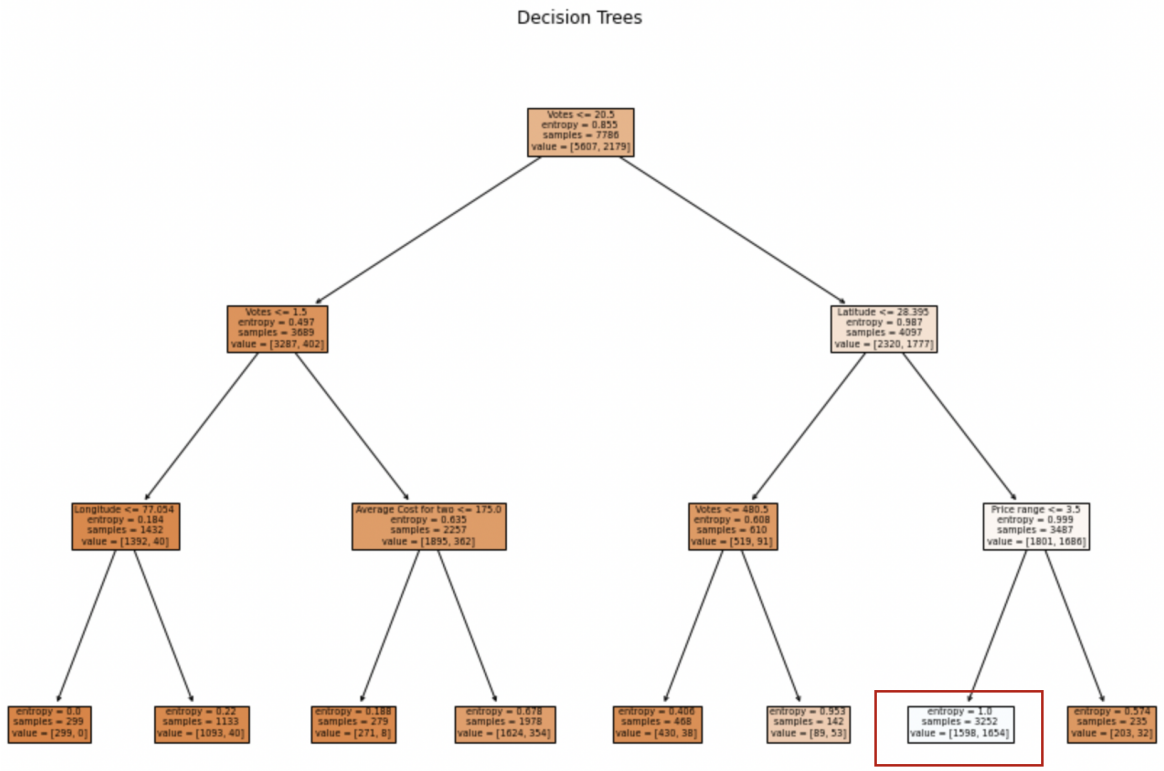

```
In [908]: print('Accuracy:',round(metrics.accuracy_score(y_test,y_pred),4))
          print('Recall:',round(metrics.recall_score(y_test,y_pred),4))
          print('Precision:',round(metrics.precision_score(y_test,y_pred),4))

          Accuracy: 0.7171
          Recall: 0.7787
          Precision: 0.4987
```

The Recall for the model is 0.7787, so our model is able to identify ~78% of the restaurants offering online delivery. But the precision is low at 0.4987, which means almost half of the restaurants which the model predicts as delivering online are not true.

Plotting ROC and AUC curve



Decision Tree Plots



The leaf (highlighted in red box) has the highest share of online deliveries. This leaf can be reached by partitioning the data on the basis of:

Votes>20.5 ➡ Latitude>28.395 ➡ Price range >3.5

Conclusion

Using the optimization and prediction method suggested above, Adam successfully could meet his girlfriend's demands as well as save money by optimally spending on restaurants and Uber. Also, the suggestions made by graphs, helped Adam to impress his girlfriend with smart recommendations and further strengthened their bonding.