```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

DATA IMPORT

```
#Importing csv files
df1= pd.read_csv("Bangalore_Restaurants.csv")
df2= pd.read_csv("Pune Restaurants.csv")
```

```
df1.head()
```

	Restaurant_Name	Category	Pricing_for_2	Locality	Dining_Rating	Dining
0	Burma Burma	Asian, Burmese, Bubble Tea, Salad, Tea, Desser...	1500	Indiranagar, Bangalore	4.9	
1	Windmills Craftworks	Continental, Fast Food, Kebab, Beverages, Ital...	2500	Windmills Craftworks, Bangalore	4.9	
2	CTR Shri Sagar	South Indian	150	Malleshwaram, Bangalore	4.9	
3	Brahmin's Coffee Bar	South Indian	100	Basavanagudi, Bangalore	4.9	
4	Milano Ice Cream	Desserts, Ice Cream, Beverages	400	Indiranagar, Bangalore	4.9	

```
df2.head()
```

	Restaurant_Name	Category	Pricing_for_2	Locality	Dining_Rating	Dining_
0	Santè Spa Cuisine	Continental, Healthy Food, Mediterranean	1200	Koregaon Park, Pune	4.9	
1	Le Plaisir	Cafe, Italian, Continental, Salad, Sandwich, P...	1000	Deccan Gymkhana, Pune	4.9	
2	Gong	Chinese, Sushi, Asian, Momos, Beverages	1700	Balewadi High Street, Baner, Pune	4.9	
3	The French Window Patisserie	Cafe, Desserts, French, Bakery, European	600	Koregaon Park, Pune	4.9	
4	Savya Rasa	South Indian, Mangalorean,	2100	Koregaon	4.9	

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5109 entries, 0 to 5108
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant_Name       5109 non-null   object 
 1   Category              5109 non-null   object 
 2   Pricing_for_2         5109 non-null   int64  
 3   Locality              5109 non-null   object 
 4   Dining_Rating         5101 non-null   float64
 5   Dining_Review_Count   5101 non-null   float64
 6   Delivery_Rating       3697 non-null   float64
 7   Delivery_Rating_Count 5101 non-null   float64
 8   Website               5109 non-null   object 
 9   Address               5109 non-null   object 
 10  Phone_No              5109 non-null   object 
 11  Latitude              5109 non-null   float64
 12  Longitude             5109 non-null   float64
dtypes: float64(6), int64(1), object(6)
memory usage: 519.0+ KB
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4797 entries, 0 to 4796
Data columns (total 15 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Restaurant_Name       4797 non-null   object 
 1   Category              4797 non-null   object 
 2   Pricing_for_2         4797 non-null   int64  
 3   Locality              4797 non-null   object 
 4   Dining_Rating          4797 non-null   float64
 5   Dining_Review_Count    4797 non-null   int64  
 6   Delivery_Rating        3226 non-null   float64
 7   Delivery_Rating_Count  4797 non-null   int64  
 8   Website                4797 non-null   object 
 9   Address                4797 non-null   object 
 10  Phone_No               4797 non-null   object 
 11  Latitude               4797 non-null   float64
 12  Longitude              4797 non-null   float64
 13  Known_for1             4155 non-null   object 
 14  Known_for2             1078 non-null   object 
dtypes: float64(4), int64(3), object(8)
memory usage: 562.3+ KB
```

```
print(df1.shape)
df2.shape
```

```
(5109, 13)
(4797, 15)
```

```
print(df1.describe())
df2.describe()
```

```
       Pricing_for_2  Dining_Rating  Dining_Review_Count  Delivery_Rating  \
count    5109.000000    5101.000000          5101.000000      3697.000000   
mean      568.790370       3.765340           276.544011         3.835110   
std       508.652835       0.304306           873.119616         0.292891   
min       100.000000       3.300000             0.000000         2.500000   
25%       300.000000       3.500000            13.000000         3.700000   
50%       400.000000       3.700000            46.000000         3.900000   
75%       600.000000       3.900000           194.000000         4.000000   
max      6000.000000       4.900000         25500.000000         4.800000   

       Delivery_Rating_Count     Latitude    Longitude  
count            5101.000000  5109.000000  5109.000000  
mean             1655.631249    12.971842    77.584676  
std              4491.941931     0.498430     1.466164  
min                 0.000000     0.033482     0.000000  
25%                12.000000    12.918861    77.578063  
50%               147.000000    12.964121    77.614647  
75%              1073.000000    13.000115    77.651366  
max             75700.000000    37.249009    77.815213  
```

	Pricing_for_2	Dining_Rating	Dining_Review_Count	Delivery_Rating	Deliver
count	4797.000000	4797.000000	4797.000000	3226.000000	
mean	571.867834	3.629748	182.439233	3.837260	
std	425.309842	0.350933	482.759166	0.284843	
min	100.000000	3.000000	0.000000	2.400000	
25%	300.000000	3.400000	11.000000	3.700000	
50%	450.000000	3.600000	36.000000	3.900000	
75%	700.000000	3.900000	137.000000	4.000000	
max	4300.000000	4.900000	8152.000000	4.500000	

```
df1.corr()
```

	Pricing_for_2	Dining_Rating	Dining_Review_Count	Delivery_
Pricing_for_2	1.000000	0.427258	0.336568	0
Dining_Rating	0.427258	1.000000	0.504614	0
Dining_Review_Count	0.336568	0.504614	1.000000	0
Delivery_Rating	0.048483	0.283819	0.131079	1
Delivery_Rating_Count	-0.074333	0.184954	0.115024	0
Latitude	0.066454	-0.001037	-0.006284	-0
Longitude	0.016578	0.015331	0.008022	0

```
df2.corr()
```

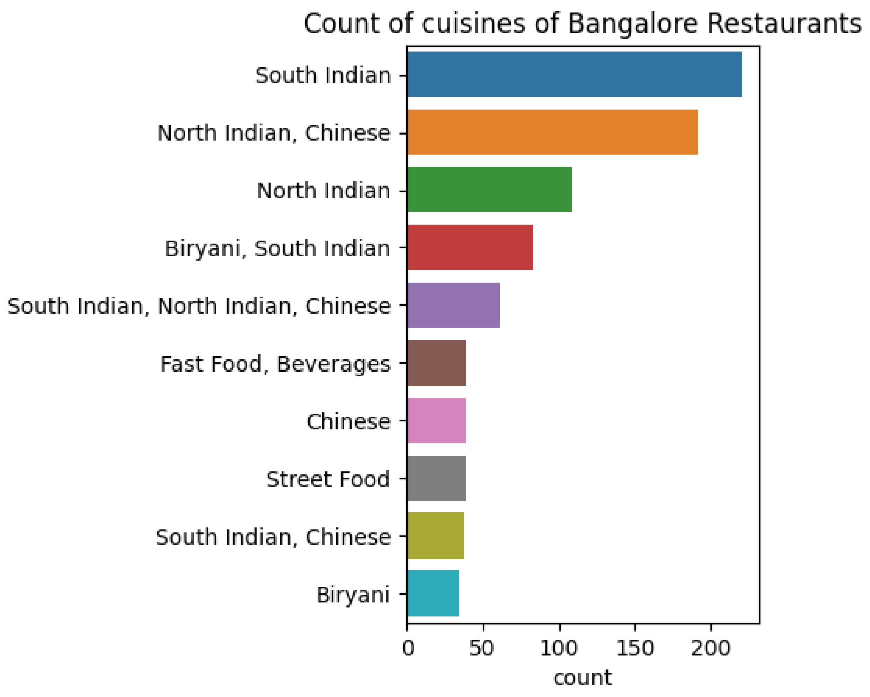
	Pricing_for_2	Dining_Rating	Dining_Review_Count	Delivery_
Pricing_for_2	1.000000	0.358431	0.367111	0
Dining_Rating	0.358431	1.000000	0.552285	0
Dining_Review_Count	0.367111	0.552285	1.000000	0
Delivery_Rating	0.051422	0.309405	0.154079	1
Delivery_Rating_Count	-0.017301	0.271891	0.283132	0
Latitude	0.045618	-0.027593	-0.013555	-0
Longitude	-0.047282	-0.007071	0.011389	0

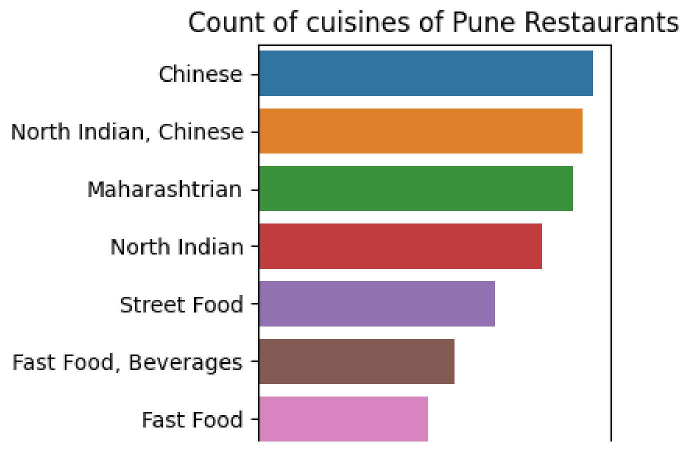
VISUALISATION

```
from collections import Counter
cuisine=df1['Category'].value_counts()[:10]
```

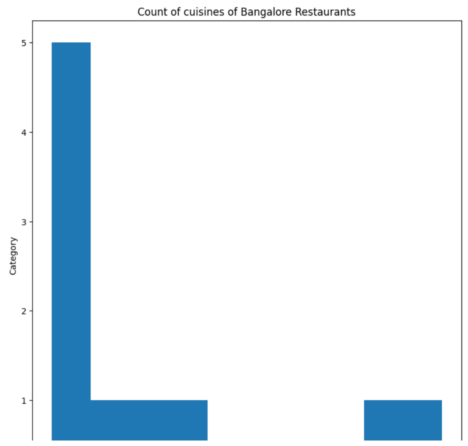
```
from collections import Counter
cuisines=df2['Category'].value_counts()[:10]
```

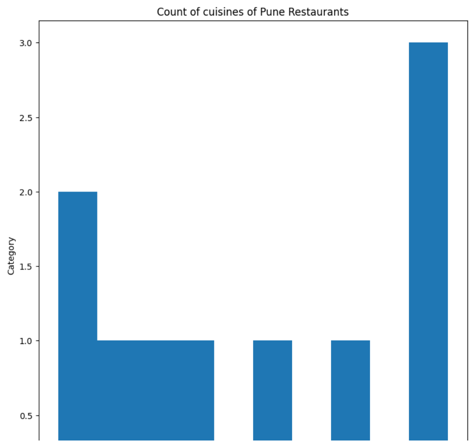
```
plt.subplot(1,2,1)
sns.barplot(x=cuisine,y=cuisine.index)
plt.title("Count of cuisines of Bangalore Restaurants")
plt.xlabel("count")
plt.show()

plt.subplot(1,2,2)
sns.barplot(x=cuisines,y=cuisines.index)
plt.title("Count of cuisines of Pune Restaurants")
plt.xlabel("count")
plt.show()
```





```
plt.figure(figsize=(20,10))
plt.subplot(1,2,1)
plt.hist(cuisine)
plt.ylabel("Category")
plt.xlabel("Count")
plt.title("Count of cuisines of Bangalore Restaurants")

plt.subplot(1,2,2)
plt.hist(cuisines)
plt.ylabel("Category")
plt.xlabel("Count")
plt.title("Count of cuisines of Pune Restaurants")
plt.show()
```





LOCATING ON MAPS

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="test")
address = '8, Omkareshwar Path, Narayan Peth, Pune, Maharashtra 411030'
location = geolocator.geocode(address)
address_final = location.address

latitude = location.latitude
longitude = location.longitude

print(latitude, longitude)
```

```
     18.5197182 73.8497737
```

```
latitude = '12.9703944526'
longitude = '77.6447132975'

coordinates = '{},{}'.format(latitude,longitude)

locator = Nominatim(user_agent="test")
Location = locator.reverse(coordinates)

print(Location.address)
```

```
     pala, 12th Main Road, HAL 2nd Stage, Hoysala Nagara, East Zone, Bengaluru, Bangalore East, Bengaluru Urban Distric
```

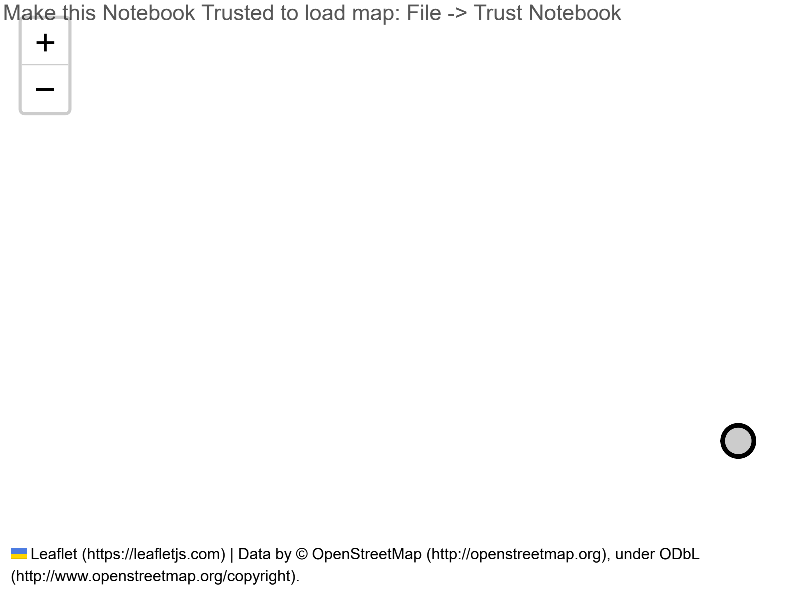
```
import math
def distance(lat1, lon1, lat2, lon2):
  p = 0.017453292519943295
  c = math.cos
  a = 0.5 - c((lat2 - lat1) * p)/2 + c(lat1 * p) * c(lat2 * p) * (1 - c((lon2 - lon1) * p))/2

  return 12742 * math.asin(math.sqrt(a))

d = distance(12.9703944526,77.6447132975,12.9732913,77.6404672)
print("The Distance is = ",d,'kms')
```

```
     The Distance is =  0.5616456785351014 kms
```

```
import folium
import matplotlib.pyplot as plt
%matplotlib inline

indiranagar_map = folium.Map(location=[latitude,longitude],zoom_start=16)
```

```
a = folium.map.FeatureGroup()
a.add_child(folium.CircleMarker([latitude,longitude],radius=10,color='black',fill_color='black',popup="IndiraNagar"))
indiranagar_map.add_child(a)
indiranagar_map
```

Make this Notebook Trusted to load map: File -> Trust Notebook


+
−
Leaflet (https://leafletjs.com) | Data by © OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

```
df1.columns
```

```
Index(['Restaurant_Name', 'Category', 'Pricing_for_2', 'Locality',
       'Dining_Rating', 'Dining_Review_Count', 'Delivery_Rating',
       'Delivery_Rating_Count', 'Website', 'Address', 'Phone_No', 'Latitude',
       'Longitude'],
      dtype='object')
```

SPLITTING AND TRAINING DATA

```
enc = LabelEncoder()
for i in (2,3,4,5,6,7,8):
  df1.iloc[:,i] = enc.fit_transform(df1.iloc[:,i])
df1.head()
```

	Restaurant_Name	Category	Pricing_for_2	Locality	Dining_Rating	Dining_Rev
0	Burma Burma	Asian, Burmese, Bubble Tea, Salad, Tea, Desser...	26	94	16	
1	Windmills Craftworks	Continental, Fast Food, Kebab, Beverages, Ital...	37	222	16	
2	CTR Shri Sagar	South Indian	1	133	16	

```
y = df1.Delivery_Rating_Count
X = df1.iloc[:,[2,3,4,5,6,7,8]]
X.head()
```

	Pricing_for_2	Locality	Dining_Rating	Dining_Review_Count	Delivery_Rating	D
0	26	94	16	856	19	
1	37	222	16	931	16	
2	1	133	16	915	17	
3	0	16	16	870	18	
4	6	94	16	848	18	

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=10)
```

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train.shape
```

```
(3576, 7)
```

```
X_test.shape
```

```
(1533, 7)
```

APPLYING MODEL

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
model = LogisticRegression()
model.fit(X_train,y_train)
```

```
y_predict = model.predict(X_test)
```

```
metrics.accuracy_score(y_test,y_predict)
```

```
0.1689497716894977
```

```
metrics.classification_report(y_test,y_predict)
```

```
'              precision    recall  f1-score   support\n\n           0       0.57
0.99      0.73       248\n           1       0.00      0.00      0.00        22\n
2       0.00      0.00      0.00        17\n           3       0.00      0.00      0.0
0        18\n           4       0.00      0.00      0.00         4\n           5
0.00      0.00      0.00        10\n           6       0.00      0.00      0.00
10\n           7       0.00      0.00      0.00         9\n           8       0.00
0.00      0.00         5\n           9       0.00      0.00      0.00         5\n
10       0.00      0.00      0.00        10\n          11       0.00      0.00      0.
00         7\n          12       0.00      0.00      0.00        11\n          13
0.00      0.00      0.00         5\n          14       0.00      0.00      0.00
```

✓ 0s completed at 12:50 AM