**APPENDIX**

**SOURCE CODE:**

**EXPLORATORY DATA ANALYSIS**

**Import Library**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import pingouin as pg

**Load Dataset**

zomatodata = pd.read\_csv("C:/Users/muvee/Desktop/project/Dataset/BangaloreZomatoData-2022.csv")

**Test of Reliability**

# Calculate Cronbach's alpha

zomatodata = pd.read\_csv("C:/Users/muvee/Desktop/project/Dataset/BangaloreZomatoData-2022.csv")

# Select the columns that contain the items or measures of interest

item\_cols = ['Dinner Reviews', 'Delivery Reviews', 'AverageCost']

# Calculate Cronbach's alpha

alpha = pg.cronbach\_alpha(zomatodata1[item\_cols])

print(f'the alpha is between the range {alpha} is 95% reliable')

### Test of Normality

from scipy.stats import kstest, norm

# select columns with numerical data

item\_cols = ['Dinner Reviews', 'Delivery Reviews', 'AverageCost']

data = zomatodata[item\_cols].values.flatten()

# perform Kolmogorov-Smirnov test

stat, p = kstest(data, 'norm')

print( f'the p value is {p}')

# interpret the test results

alpha = 0.05

if p > alpha:

print('The data is normally distributed (fail to reject H0)')

else:

print('The data is not normally distributed (reject H0)')

### Discriptive statistics

zomatodata.describe()

### 1).Understanding the data with head

zomatodata.head()

### 2).understanding the tail of data[¶](http://localhost:8888/notebooks/Desktop/project/Jupyter/EDA%20%20-%20Zomato%20Analysis.ipynb#2).understanding-the-tail-of-data)

zomatodata.tail()

### 3).Datatype of every columns

zomatodata.dtypes

### 4).Dataset shape[columns,rows]

zomatodata.shape

### 5).list of columns

### zomatodata.columns

### 6).columns showing unique values

### zomatodata.nunique()

### 7).cleaning the data(is null and drop coulmn)

### zomatodata.isnull().sum()

### if 'Timing' in zomatodata.columns:

### zomatodata.drop('Timing', axis=1, inplace=True)

zomatodata.isnull().sum()

### 8).Bar chart to show the number of restaurants in each location

Area\_counts = zomatodata['Area'].value\_counts().head(10)

print(Area\_counts)

#Chart

plt.figure(figsize=(10, 4))

plt.bar(Area\_counts.index, Area\_counts.values)

plt.xticks(rotation=90)

plt.xlabel('Area')

plt.ylabel('Number of Restaurants')

plt.title('Number of Restaurants in Each Area')

plt.ylim(0,1000)

plt.text(0,Area\_counts.values[0],'674',ha='center',va='bottom')

plt.text(1,Area\_counts.values[1],'484',ha='center',va='bottom')

plt.text(2,Area\_counts.values[2],'457',ha='center',va='bottom')

plt.text(3,Area\_counts.values[3],'447',ha='center',va='bottom')

plt.text(4,Area\_counts.values[4],'397',ha='center',va='bottom')

plt.text(5,Area\_counts.values[5],'292',ha='center',va='bottom')

plt.text(6,Area\_counts.values[6],'292',ha='center',va='bottom')

plt.text(7,Area\_counts.values[7],'284',ha='center',va='bottom')

plt.text(8,Area\_counts.values[8],'237',ha='center',va='bottom')

plt.text(9,Area\_counts.values[9],'232',ha='center',va='bottom')

plt.show()

### 9).We Will Count The Number Of Times A Restaurant's Name Has Been Repeated

df=zomatodata

df['count']=1

a=df.groupby(['Name']).count()['count']

#THEN WE WILL SORT IN DESCENDING ORDER AND SLICE THE TOP 5

b=a.sort\_values(ascending=False)

most\_branches=b[0:6]

most\_branches

#Chart

f = plt.figure()

f.set\_figwidth(19)

f.set\_figheight(5)

plt.plot(most\_branches,color='orange',linewidth=3,marker="D",markerfacecolor='black',markersize=8)

plt.xlabel('Restaurant')

plt.ylabel('No of branches')

plt.title('Restaurants with most branches across city')

plt.grid(axis='y')

### 10).Here I Filtered The Restaurants By Top Rating And Cheap Price

df = df.replace('-', np.nan)

df['total\_rating']=df['Dinner Ratings'].astype(float)+df['Delivery Ratings'].astype(float)

s=df['total\_rating'].sort\_values(ascending=False )

e=s[0:6]

k=df.groupby(['Name']).mean().sort\_values('total\_rating',ascending=False)

k

#Chart

price=k['AverageCost'][0:5]

rating=k['total\_rating'][0:5]

res\_name=['Brahmins Coffee Bar','CTR Shri Sagar','Burma Burma','The Blue Wagon - Coffee & Kitchen','Bologna']

price=price.sort\_values()

plt.bar(res\_name,rating)

plt.plot(range(5))

plt.ylim(8.5,9.45)

plt.text(0,rating[0],'₹ 100',ha='center',va='bottom')

plt.text(1,rating[1],'₹ 150',ha='center',va='bottom')

plt.text(2,rating[2],'₹ 1500',ha='center',va='bottom')

plt.text(3,rating[3],'₹ 400',ha='center',va='bottom')

plt.text(4,rating[4],'₹ 1600',ha='center',va='bottom')

plt.ylabel('RATING')

plt.xticks(rotation=270)

plt.show()

### 11-Most popular cuisines of Bangalore

Cuisines = zomatodata['Cuisines'].value\_counts()[:10]

Cuisines

#Chart

Cuisines = zomatodata['Cuisines'].value\_counts()[:10]

sns.barplot(x = Cuisines,y =Cuisines.index)

plt.xlabel('Count')

plt.title("Most popular cuisines of Bangalore")

**RESTAURANT RECOMMENDATION MODEL USING MACHING LEARNING**

**SOURCE CODE:**

**Import Library:**

import numpy as np

import pandas as pd

import difflib

import sklearn

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

**Load Dataset:**

zo=pd.read\_csv('A:/BangaloreZomatoData-2022.csv')

**Reset Index**

zo = zo.reset\_index()

**Show head**

zo.head()

**Code to Bulid the Recommentation Model:**

#Select the list of columns

selected\_features = ['Name','Cuisines','Area','Full\_Address','PopularDishes','Dinner Ratings','AverageCost']

#If there is any Na fill wit " "

for feature in selected\_features:

zo[feature] = zo[feature].fillna('')

#Coimbining the columns Together

combine\_features=zo['Name']+zo['Cuisines']+zo['Area']+zo['Full\_Address']+zo['PopularDishes']+str(zo['Dinner Ratings'])+str(zo['AverageCost'])

# vectorizer is used to convert text to numbers as specific tag

vectorizer = TfidfVectorizer()

# The transformed numerics store in this variable

feature\_vectors = vectorizer.fit\_transform(combine\_features)

#Getting similarity score

similarity = cosine\_similarity(feature\_vectors)

#Getting the name as input from user

Restraurant\_area = input('Enter the area name: ').lower()

#Which allows spacing

Restraurant\_area = Restraurant\_area.strip()

# Filter restaurants based on area

#condition

zo\_area = zo[zo['Area'].str.lower() == Restraurant\_area]

if len(zo\_area) == 0:

print(f"No restaurants found in {Restraurant\_area}")

else:

List\_of\_all\_Restaurant\_name = zo\_area['Name'].tolist()

Restraurant\_name = input(f'Enter a restaurant name in {Restraurant\_area}: ').lower()

Restraurant\_name = Restraurant\_name.strip()

# Finding the closest match for the given restaurant name

find\_close\_match = difflib.get\_close\_matches(Restraurant\_name, List\_of\_all\_Restaurant\_name)

if len(find\_close\_match) == 0:

print(f"No restaurants found with name '{Restraurant\_name}' in {Restraurant\_area}")

else:

close\_match = find\_close\_match[0]

index\_of\_the\_Restraurant = zo\_area[zo\_area.Name == close\_match]['index'].values[0]

similarity\_score = list(enumerate(similarity[index\_of\_the\_Restraurant]))

sort\_similarity\_Restraurant = sorted(similarity\_score,key = lambda x:x[1], reverse=True)

print(f'Restaurants in {Restraurant\_area} similar to {close\_match} based on rating and features: \n')

i = 1

for Restraurant in sort\_similarity\_Restraurant:

index = Restraurant[0]

Name\_from\_index = zo\_area[zo\_area.index==index]['Name'].values

if i<10:

print(i,'.',Name\_from\_index)

i+=1