Configuration Manual

MSc Research Project
Programme Name

Harshit Parihar
Student ID: x21111022

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Harshit Parihar		
Student ID:	21111022		
Programme:	Msc. Data Analytics	**Year:**	2022-23
Module:	Msc. Research Project		
Lecturer:	Dr. Christian Horn		
Submission Due Date:	15th December 2022		
Project Title:	Suggesting New Restaurants To Visit Using Content Based Recommender System		
Word Count: 555	**Page Count: 11**		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……

Date: ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Harshit Parihar
Student ID: x21111022

1 Introduction

The configuration manual is a step by step procedure of the code that has been implemented in the code. The pre-processing, transformation and implementation are included in this manual.

2 System Requirement

The basic requirements of the machine for the project are-

Operating System	MacOs
Ram	8 GB
Hard Disk	256 gb ssd
Processor	M1 chip

Basic tools used are-

- Microsoft Excel
- Python 3.7
- Anaconda Jupyter Notebook

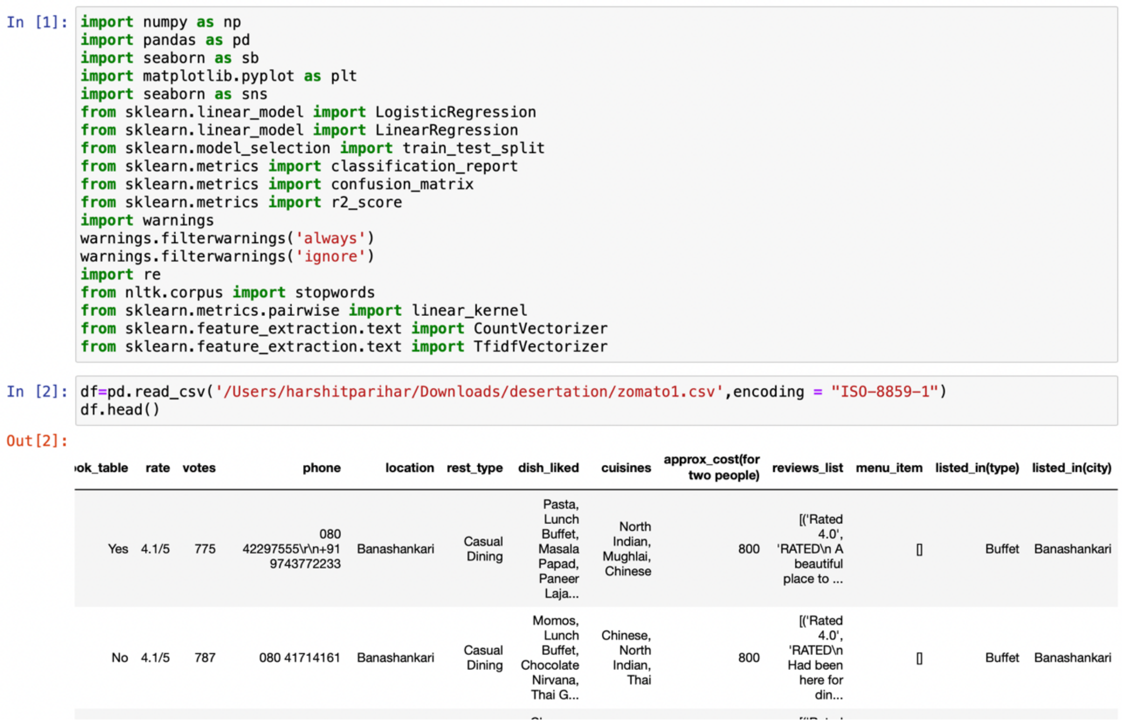
Microsoft Excel is used to view the data. Jupyter is used to write code. Python is the language used.

3 Dataset Selection

The dataset selected were from Kaggle and were data scraped from Zomato and uploaded to Kaggle. Two datasets were used for this research.

4 Importing required dataset and libraries

Review based model



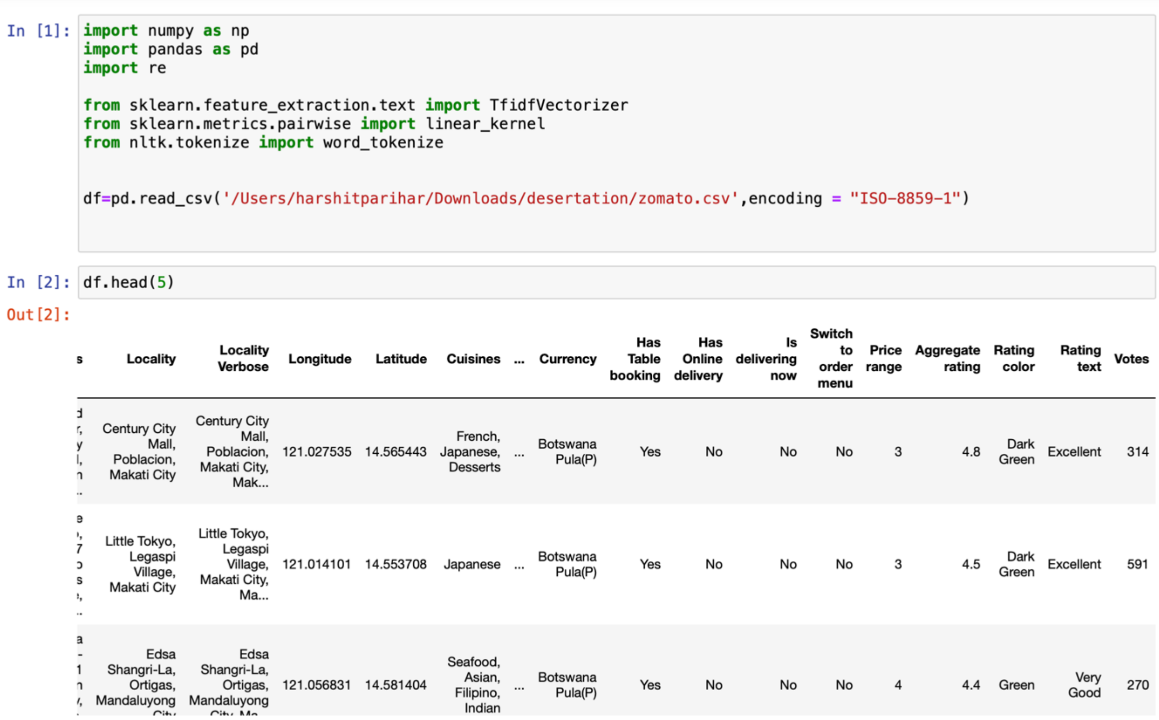
In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
import re
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:
```
df=pd.read_csv('/Users/harshitparihar/Downloads/desertation/zomato1.csv',encoding = "ISO-8859-1")
df.head()
```

Out[2]:

ok_table	rate	votes	phone	location	rest_type	dish_liked	cuisines	approx_cost(for two people)	reviews_list	menu_item	listed_in(type)	listed_in(city)
Yes	4.1/5	775	080 42297555\r\n+91 9743772233	Banashankari	Casual Dining	Pasta, Lunch Buffet, Masala Papad, Paneer Laja...	North Indian, Mughlai, Chinese	800	[('Rated 4.0', 'RATED\n A beautiful place to ...	[]	Buffet	Banashankari
No	4.1/5	787	080 41714161	Banashankari	Casual Dining	Momos, Lunch Buffet, Chocolate Nirvana, Thai G...	Chinese, North Indian, Thai	800	[('Rated 4.0', 'RATED\n Had been here for din...	[]	Buffet	Banashankari

Location based model



In [1]:
```
import numpy as np
import pandas as pd
import re

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from nltk.tokenize import word_tokenize


df=pd.read_csv('/Users/harshitparihar/Downloads/desertation/zomato.csv',encoding = "ISO-8859-1")
```

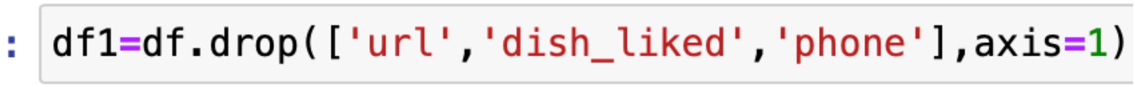
In [2]:
```
df.head(5)
```

Out[2]:

s	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
d r, y l, n ..	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
e , 7 ɔ s , ..	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
a - 1 n /,	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

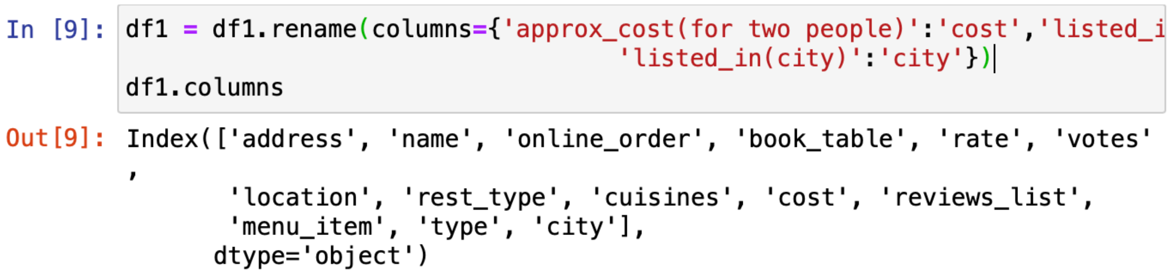
5 Pre-processing(Review Based)

Dropping unused columns



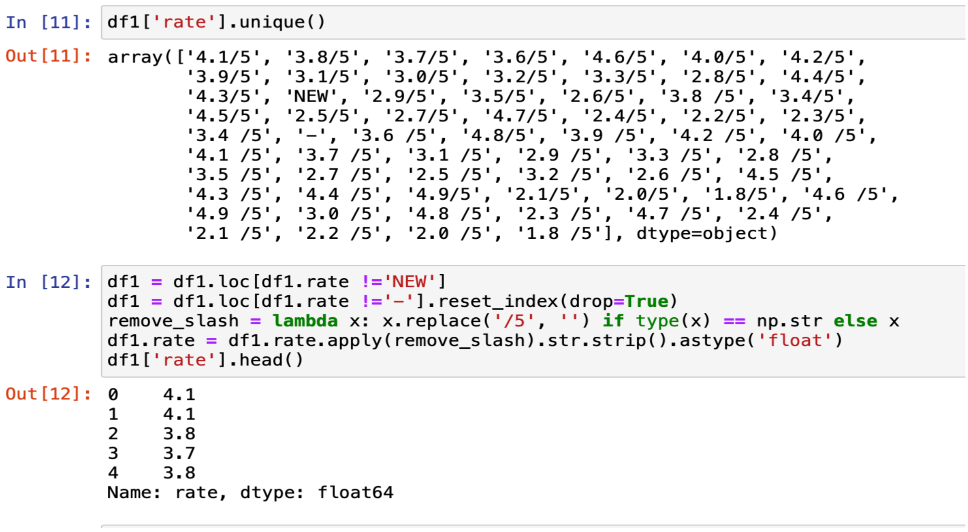
```
: df1=df.drop(['url','dish_liked','phone'],axis=1)
```

Renaming Columns



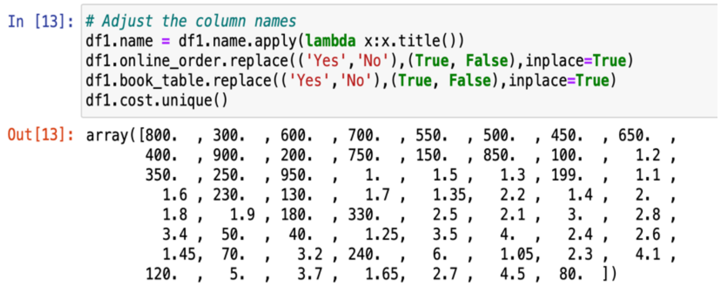
```
In [9]: df1 = df1.rename(columns={'approx_cost(for two people)':'cost','listed_i
                                    'listed_in(city)':'city'})
        df1.columns

Out[9]: Index(['address', 'name', 'online_order', 'book_table', 'rate', 'votes'
        ,
               'location', 'rest_type', 'cuisines', 'cost', 'reviews_list',
               'menu_item', 'type', 'city'],
              dtype='object')
```

Removing /5 from rate columns



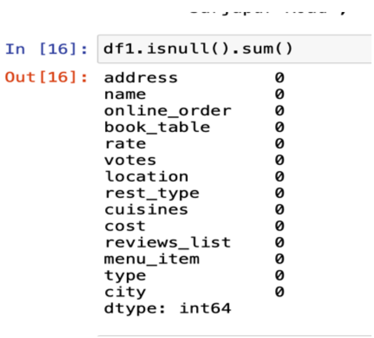
```
In [11]: df1['rate'].unique()

Out[11]: array(['4.1/5', '3.8/5', '3.7/5', '3.6/5', '4.6/5', '4.0/5', '4.2/5',
                '3.9/5', '3.1/5', '3.0/5', '3.2/5', '3.3/5', '2.8/5', '4.4/5',
                '4.3/5', 'NEW', '2.9/5', '3.5/5', '2.6/5', '3.8 /5', '3.4/5',
                '4.5/5', '2.5/5', '2.7/5', '4.7/5', '2.4/5', '2.2/5', '2.3/5',
                '3.4 /5', '-', '3.6 /5', '4.8/5', '3.9 /5', '4.2 /5', '4.0 /5',
                '4.1 /5', '3.7 /5', '3.1 /5', '2.9 /5', '3.3 /5', '2.8 /5',
                '3.5 /5', '2.7 /5', '2.5 /5', '3.2 /5', '2.6 /5', '4.5 /5',
                '4.3 /5', '4.4 /5', '4.9/5', '2.1/5', '2.0/5', '1.8/5', '4.6 /5',
                '4.9 /5', '3.0 /5', '4.8 /5', '2.3 /5', '4.7 /5', '2.4 /5',
                '2.1 /5', '2.2 /5', '2.0 /5', '1.8 /5'], dtype=object)

In [12]: df1 = df1.loc[df1.rate !='NEW']
         df1 = df1.loc[df1.rate !='-'].reset_index(drop=True)
         remove_slash = lambda x: x.replace('/5', '') if type(x) == np.str else x
         df1.rate = df1.rate.apply(remove_slash).str.strip().astype('float')
         df1['rate'].head()

Out[12]: 0    4.1
         1    4.1
         2    3.8
         3    3.7
         4    3.8
         Name: rate, dtype: float64
```

Changing yes/no to true false in book table and online order column.



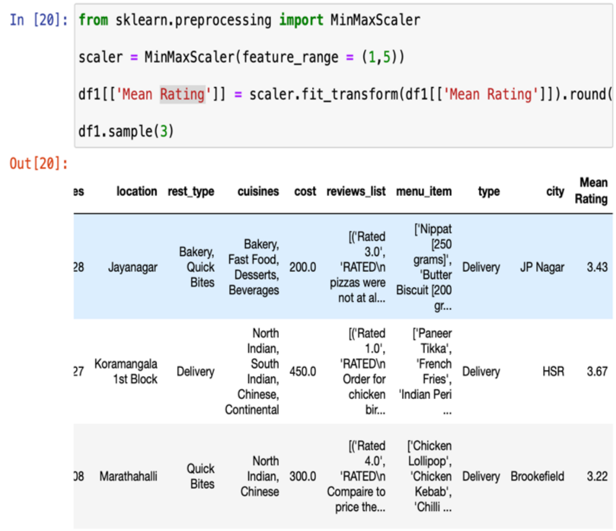
```
In [13]: # Adjust the column names
         df1.name = df1.name.apply(lambda x:x.title())
         df1.online_order.replace(('Yes','No'),(True, False),inplace=True)
         df1.book_table.replace(('Yes','No'),(True, False),inplace=True)
         df1.cost.unique()

Out[13]: array([800.  , 300.  , 600.  , 700.  , 550.  , 500.  , 450.  , 650.  ,
                400.  , 900.  , 200.  , 750.  , 150.  , 850.  , 100.  ,   1.2 ,
                350.  , 250.  , 950.  ,   1.  ,   1.5 ,   1.3 , 199.  ,   1.1 ,
                  1.6 , 230.  , 130.  ,   1.7 ,   1.35,   2.2 ,   1.4 ,   2.  ,
                  1.8 ,   1.9 , 180.  , 330.  ,   2.5 ,   2.1 ,   3.  ,   2.8 ,
                  3.4 ,  50.  ,  40.  ,   1.25,   3.5 ,   4.  ,   2.4 ,   2.6 ,
                  1.45,  70.  ,   3.2 , 240.  ,   6.  ,   1.05,   2.3 ,   4.1 ,
                120.  ,   5.  ,   3.7 ,   1.65,   2.7 ,   4.5 ,  80.  ])
```

Checking for null values.



```
In [16]: df1.isnull().sum()

Out[16]: address         0
         name            0
         online_order    0
         book_table      0
         rate            0
         votes           0
         location        0
         rest_type       0
         cuisines        0
         cost            0
         reviews_list    0
         menu_item       0
         type            0
         city            0
         dtype: int64
```

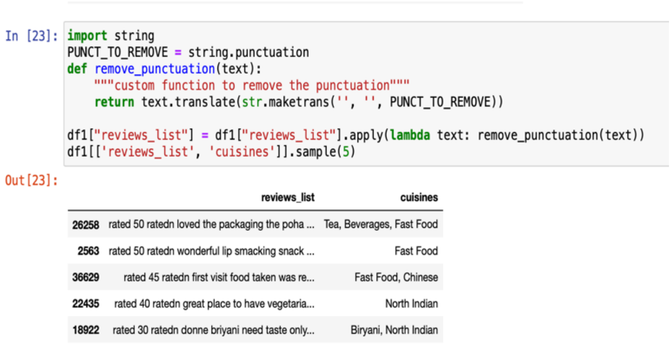
Creating new column mean rating for aggregate rating of reviews.



```
In [20]: from sklearn.preprocessing import MinMaxScaler

         scaler = MinMaxScaler(feature_range = (1,5))

         df1[['Mean Rating']] = scaler.fit_transform(df1[['Mean Rating']]).round(

         df1.sample(3)

Out[20]:
```

es	location	rest_type	cuisines	cost	reviews_list	menu_item	type	city	Mean Rating
28	Jayanagar	Bakery, Quick Bites	Bakery, Fast Food, Desserts, Beverages	200.0	[('Rated 3.0', 'RATED\n pizzas were not at al...	['Nippat [250 grams]', 'Butter Biscuit [200 gr...	Delivery	JP Nagar	3.43
27	Koramangala 1st Block	Delivery	North Indian, South Indian, Chinese, Continental	450.0	[('Rated 1.0', 'RATED\n Order for chicken bir...	['Paneer Tikka', 'French Fries', 'Indian Peri ...	Delivery	HSR	3.67
08	Marathahalli	Quick Bites	North Indian, Chinese	300.0	[('Rated 4.0', 'RATED\n Compaire to price the...	['Chicken Lollipop', 'Chicken Kebab', 'Chilli ...	Delivery	Brookefield	3.22

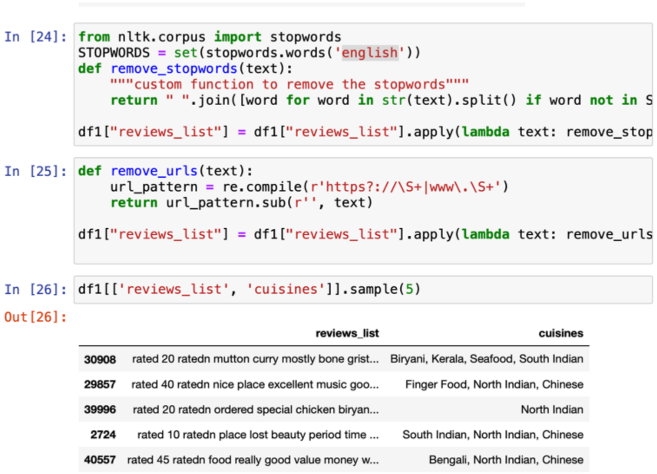
Removing punctuation from reviews



```
In [23]: import string
         PUNCT_TO_REMOVE = string.punctuation
         def remove_punctuation(text):
             """custom function to remove the punctuation"""
             return text.translate(str.maketrans('', '', PUNCT_TO_REMOVE))

         df1["reviews_list"] = df1["reviews_list"].apply(lambda text: remove_punctuation(text))
         df1[['reviews_list', 'cuisines']].sample(5)
Out[23]:
```

	reviews_list	cuisines
26258	rated 50 ratedn loved the packaging the poha ...	Tea, Beverages, Fast Food
2563	rated 50 ratedn wonderful lip smacking snack ...	Fast Food
36629	rated 45 ratedn first visit food taken was re...	Fast Food, Chinese
22435	rated 40 ratedn great place to have vegetaria...	North Indian
18922	rated 30 ratedn donne briyani need taste only...	Biryani, North Indian

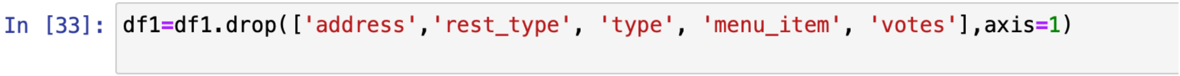
Removing stopwords and urls from reviews.



```
In [24]: from nltk.corpus import stopwords
         STOPWORDS = set(stopwords.words('english'))
         def remove_stopwords(text):
             """custom function to remove the stopwords"""
             return " ".join([word for word in str(text).split() if word not in S

         df1["reviews_list"] = df1["reviews_list"].apply(lambda text: remove_stop

In [25]: def remove_urls(text):
             url_pattern = re.compile(r'https?://\S+|www\.\S+')
             return url_pattern.sub(r'', text)

         df1["reviews_list"] = df1["reviews_list"].apply(lambda text: remove_urls

In [26]: df1[['reviews_list', 'cuisines']].sample(5)
Out[26]:
```

	reviews_list	cuisines
30908	rated 20 ratedn mutton curry mostly bone grist...	Biryani, Kerala, Seafood, South Indian
29857	rated 40 ratedn nice place excellent music goo...	Finger Food, North Indian, Chinese
39996	rated 20 ratedn ordered special chicken biryan...	North Indian
2724	rated 10 ratedn place lost beauty period time ...	South Indian, North Indian, Chinese
40557	rated 45 ratedn food really good value money w...	Bengali, North Indian, Chinese

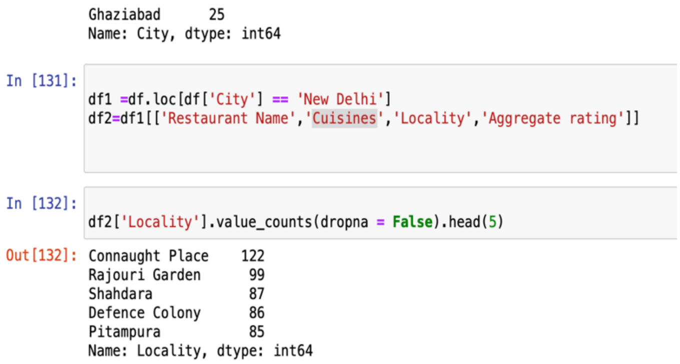
Dropping more columns.



```
In [33]: df1=df1.drop(['address','rest_type', 'type', 'menu_item', 'votes'],axis=1)
```

6 Pre-processing(Location Based)

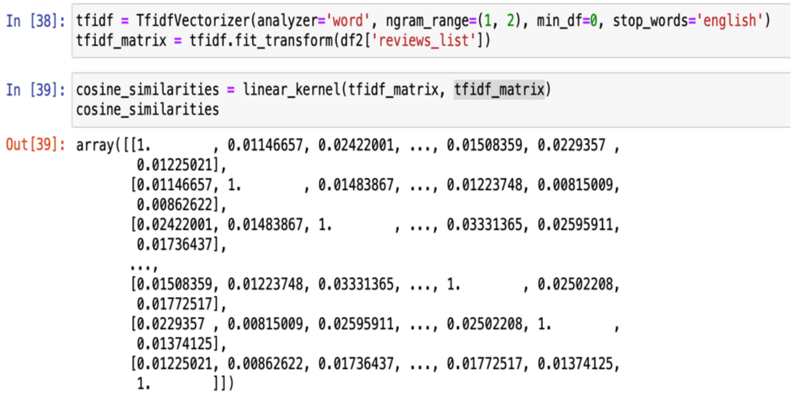
Extracting Data of New Delhi from dataframe.



```
Ghaziabad      25
Name: City, dtype: int64

In [131]:
df1 =df.loc[df['City'] == 'New Delhi']
df2=df1[['Restaurant Name','Cuisines','Locality','Aggregate rating']]

In [132]:
df2['Locality'].value_counts(dropna = False).head(5)

Out[132]: Connaught Place    122
Rajouri Garden      99
Shahdara            87
Defence Colony      86
Pitampura           85
Name: Locality, dtype: int64
```

7 Calculating Similarity Scores (Review Based)

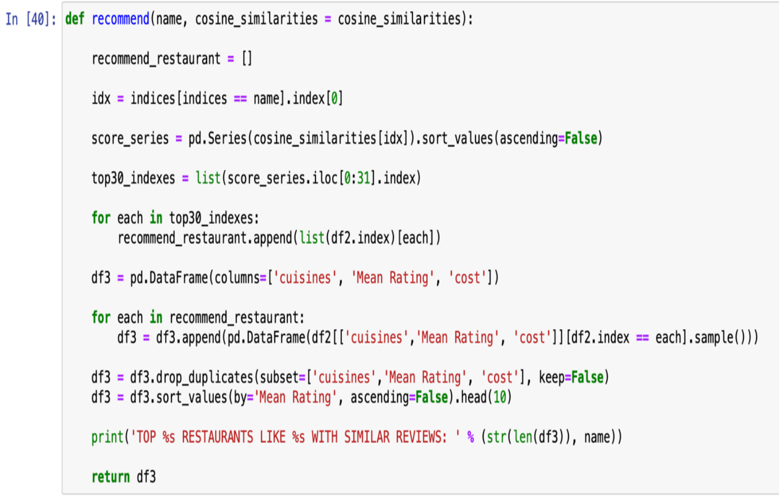
Tf-idf to vectorize words and cosine similarity to calculate scores.



```
In [38]: tfidf = TfidfVectorizer(analyzer='word', ngram_range=(1, 2), min_df=0, stop_words='english')
         tfidf_matrix = tfidf.fit_transform(df2['reviews_list'])

In [39]: cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
         cosine_similarities

Out[39]: array([[1.        , 0.01146657, 0.02422001, ..., 0.01508359, 0.0229357 ,
                 0.01225021],
                [0.01146657, 1.        , 0.01483867, ..., 0.01223748, 0.00815009,
                 0.00862622],
                [0.02422001, 0.01483867, 1.        , ..., 0.03331365, 0.02595911,
                 0.01736437],
                ...,
                [0.01508359, 0.01223748, 0.03331365, ..., 1.        , 0.02502208,
                 0.01772517],
                [0.0229357 , 0.00815009, 0.02595911, ..., 0.02502208, 1.        ,
                 0.01374125],
                [0.01225021, 0.00862622, 0.01736437, ..., 0.01772517, 0.01374125,
                 1.        ]])
```

8 Recommendation Model (Review Based)

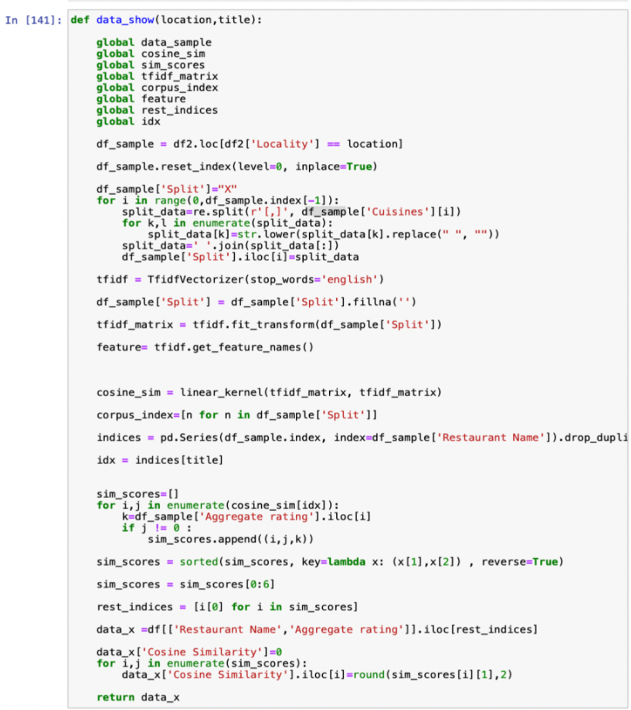
This is the first model.



```
In [40]: def recommend(name, cosine_similarities = cosine_similarities):

             recommend_restaurant = []

             idx = indices[indices == name].index[0]

             score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending=False)

             top30_indexes = list(score_series.iloc[0:31].index)

             for each in top30_indexes:
                 recommend_restaurant.append(list(df2.index)[each])

             df3 = pd.DataFrame(columns=['cuisines', 'Mean Rating', 'cost'])

             for each in recommend_restaurant:
                 df3 = df3.append(pd.DataFrame(df2[['cuisines','Mean Rating', 'cost']][df2.index == each].sample()))

             df3 = df3.drop_duplicates(subset=['cuisines','Mean Rating', 'cost'], keep=False)
             df3 = df3.sort_values(by='Mean Rating', ascending=False).head(10)

             print('TOP %s RESTAURANTS LIKE %s WITH SIMILAR REVIEWS: ' % (str(len(df3)), name))

             return df3
```

9 Recommendation Model (Location Based)

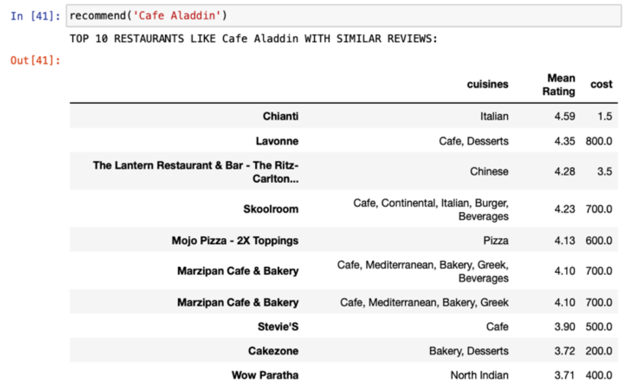
This is the location based model.



```
In [141]: def data_show(location,title):

              global data_sample
              global cosine_sim
              global sim_scores
              global tfidf_matrix
              global corpus_index
              global feature
              global rest_indices
              global idx

              df_sample = df2.loc[df2['Locality'] == location]

              df_sample.reset_index(level=0, inplace=True)

              df_sample['Split']="X"
              for i in range(0,df_sample.index[-1]):
                  split_data=re.split(r'[,]', df_sample['Cuisines'][i])
                  for k,l in enumerate(split_data):
                      split_data[k]=str.lower(split_data[k].replace(" ", ""))
                  split_data=' '.join(split_data[:])
                  df_sample['Split'].iloc[i]=split_data

              tfidf = TfidfVectorizer(stop_words='english')

              df_sample['Split'] = df_sample['Split'].fillna('')

              tfidf_matrix = tfidf.fit_transform(df_sample['Split'])

              feature= tfidf.get_feature_names()


              cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

              corpus_index=[n for n in df_sample['Split']]

              indices = pd.Series(df_sample.index, index=df_sample['Restaurant Name']).drop_dupli

              idx = indices[title]


              sim_scores=[]
              for i,j in enumerate(cosine_sim[idx]):
                  k=df_sample['Aggregate rating'].iloc[i]
                  if j != 0 :
                      sim_scores.append((i,j,k))

              sim_scores = sorted(sim_scores, key=lambda x: (x[1],x[2]) , reverse=True)

              sim_scores = sim_scores[0:6]

              rest_indices = [i[0] for i in sim_scores]

              data_x =df[['Restaurant Name','Aggregate rating']].iloc[rest_indices]

              data_x['Cosine Similarity']=0
              for i,j in enumerate(sim_scores):
                  data_x['Cosine Similarity'].iloc[i]=round(sim_scores[i][1],2)

              return data_x
```

10 Recommendation Result(Review Based)

Café Aladdin is historical data of example user.

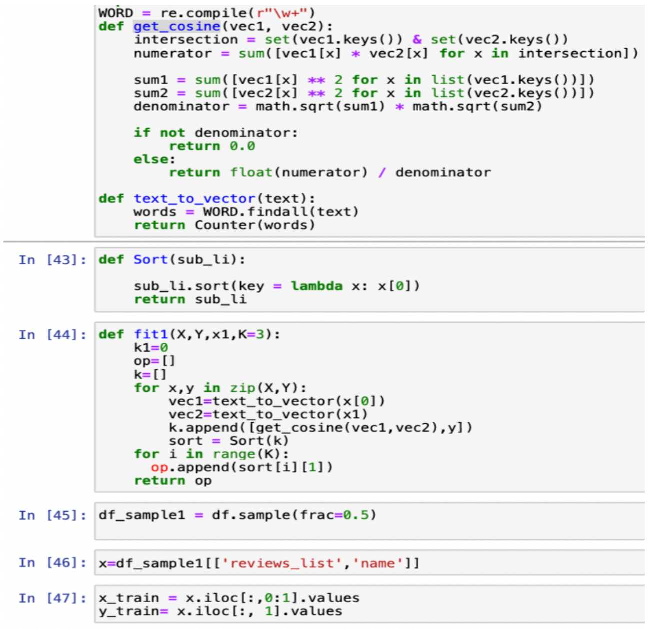


```
In [41]: recommend('Cafe Aladdin')
         TOP 10 RESTAURANTS LIKE Cafe Aladdin WITH SIMILAR REVIEWS:
Out[41]:
```

	cuisines	Mean Rating	cost
Chianti	Italian	4.59	1.5
Lavonne	Cafe, Desserts	4.35	800.0
The Lantern Restaurant & Bar - The Ritz-Carlton...	Chinese	4.28	3.5
Skoolroom	Cafe, Continental, Italian, Burger, Beverages	4.23	700.0
Mojo Pizza - 2X Toppings	Pizza	4.13	600.0
Marzipan Cafe & Bakery	Cafe, Mediterranean, Bakery, Greek, Beverages	4.10	700.0
Marzipan Cafe & Bakery	Cafe, Mediterranean, Bakery, Greek	4.10	700.0
Stevie'S	Cafe	3.90	500.0
Cakezone	Bakery, Desserts	3.72	200.0
Wow Paratha	North Indian	3.71	400.0

11 Recommendation Model(Review Based 2)

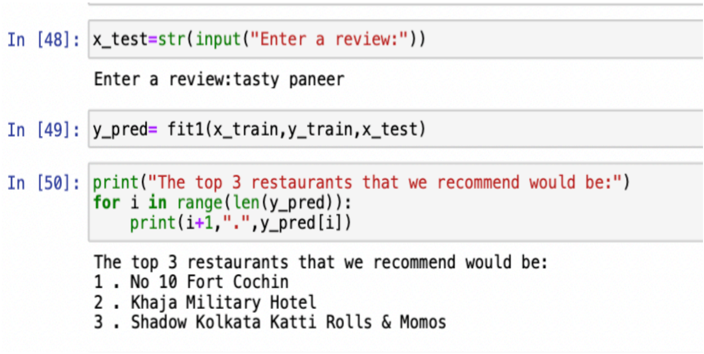
This is the second model in the same codebase of review based models.



```
          WORD = re.compile(r"\w+")
          def get_cosine(vec1, vec2):
              intersection = set(vec1.keys()) & set(vec2.keys())
              numerator = sum([vec1[x] * vec2[x] for x in intersection])

              sum1 = sum([vec1[x] ** 2 for x in list(vec1.keys())])
              sum2 = sum([vec2[x] ** 2 for x in list(vec2.keys())])
              denominator = math.sqrt(sum1) * math.sqrt(sum2)

              if not denominator:
                  return 0.0
              else:
                  return float(numerator) / denominator

          def text_to_vector(text):
              words = WORD.findall(text)
              return Counter(words)

In [43]: def Sort(sub_li):

              sub_li.sort(key = lambda x: x[0])
              return sub_li

In [44]: def fit1(X,Y,x1,K=3):
              k1=0
              op=[]
              k=[]
              for x,y in zip(X,Y):
                  vec1=text_to_vector(x[0])
                  vec2=text_to_vector(x1)
                  k.append([get_cosine(vec1,vec2),y])
                  sort = Sort(k)
              for i in range(K):
                 op.append(sort[i][1])
              return op

In [45]: df_sample1 = df.sample(frac=0.5)

In [46]: x=df_sample1[['reviews_list','name']]

In [47]: x_train = x.iloc[:,0:1].values
         y_train= x.iloc[:, 1].values
```

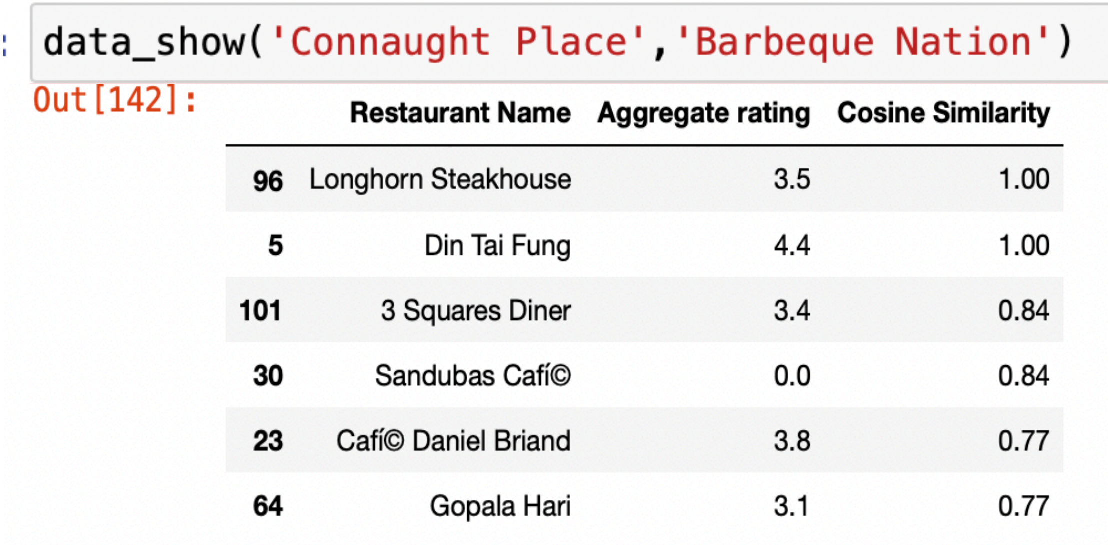
12 Recommendation Result(Review Based 2)

The text in ‘Enter A Review’is entered by the person running the code.



```
In [48]: x_test=str(input("Enter a review:"))

         Enter a review:tasty paneer

In [49]: y_pred= fit1(x_train,y_train,x_test)

In [50]: print("The top 3 restaurants that we recommend would be:")
         for i in range(len(y_pred)):
             print(i+1,".",y_pred[i])

         The top 3 restaurants that we recommend would be:
         1 . No 10 Fort Cochin
         2 . Khaja Military Hotel
         3 . Shadow Kolkata Katti Rolls & Momos
```

13 Recommendation Result(Location Based)



```
: data_show('Connaught Place','Barbeque Nation')
```

Out[142]:

	Restaurant Name	Aggregate rating	Cosine Similarity
96	Longhorn Steakhouse	3.5	1.00
5	Din Tai Fung	4.4	1.00
101	3 Squares Diner	3.4	0.84
30	Sandubas Cafí©	0.0	0.84
23	Cafí© Daniel Briand	3.8	0.77
64	Gopala Hari	3.1	0.77