RESTAURANT RECOMMENDATION SYSTEM

PROJECT REPORT

Machine Learning (CSE4020) Submitted

By:

DEBADITYA MITRA	**16BCE1341**
SHUBHAM JAISWAL	**16BCE1171**

Slot: E1

Name of the Faculty: **Prof. Syed Ibrahim SP (SCHOOL OF**

COMPUTER SCIENCE AND ENGINEERING)

November, 2018

CERTIFICATE

This is to certify that the project work entitled "**RESTAURANT RECOMMENDATION SYSTEM**" that is being submitted by Shubham Jaiswal and Debaditya Mitra for Machine Learning (CSE4020) is a record of bonafide work done under my supervision. The contents of this Project Work, in full or in parts, have neither been taken from any other source nor have been submitted for any other CAL course.

Signature of students:

Shubham Jaiswal

Debaditya Mitra

Signature of the Faculty:

Prof. Syed Ibrahim SP

ACKNOWLEDGEMENT

This project is an outcome of the efforts put forth by many people. Firstly, the group would like to thank the University Management and the Dean of SCSE for giving the group this opportunity to carry out this project. Secondly, our teacher, Prof. Syed Ibrahim SP for providing his valuable inputs during our Reviews and giving us the opportunity to take-up this interesting topic. This project is a group effort by all the team members and not an individual contribution. Hence, we thank all the team members also.

ABSTRACT

Recommendation Systems include simple algorithms which aim to provide the most relevant and accurate items to the user by filtering useful stuff from of a huge pool of information base. Recommendation engines discovers data patterns in the data set by learning consumers' choices and produces the outcomes that co-relates to their needs and interests.

In this project, we use reviews from registered users to generate a machine-learning model for each business and each registered user. It also defines an architecture, which uses the generated machine-learning models to support real-time personalized recommendations for restaurant searching. For rating prediction, we compare user-based collaborative filtering algorithms. The language platform that we have used for creating this restaurant recommendation system using is Python.

OBJECTIVE

Restaurant recommendation system is a very popular service whose accuracy and sophistication keeps increasing every day. The focus of the project is to develop a recommender system that would take ratings provided by users on certain places and would predict what those users would rate the other, unvisited, places. In this project, a list of top-n restaurants will be generated based on consumer preferences. The recommendation system here will predict about customers based on their previous experiences or choices. These systems are trained in cross selling and up selling.

INTRODUCTION

A recommender system refers to a system that is capable of predicting the future preference of a set of items for a user, and recommend the top items. One key reason why we need a recommender system in modern society is that people have too much options to use from due to the prevalence of Internet.
It is very common that we hang out with families, friends, and coworkers when comes to lunch or dinner time. As the users of recommendation applications, people care more about how we will like a restaurant. People will tend to have happier experiences when the prediction of the recommendation system is as good as what it says. As there is a completed and big data set of user and restaurants reviews, we want to see whether we can use the latest techniques to make good predictions.

ABOUT THE DATASET

The dataset is obtained from a recommender system prototype from UCI Machine Learning Repository of Restaurant and Consumer data Dataset.
There are nine data files used in this project, which are namely:-

For Restaurants:

- chefmozaccepts.csv
- chefmozcuisine.csv
- chefmozhours4.csv
- chefmozparking.csv
- geoplaces2.csv

For Consumers:

- usercuisine.csv
- userpayment.csv
- userprofile.csv

For User-Item-Rating:

- rating_final.csv

ALGORITHMS USED:

1. **Collaborative Filter Technique**: it uses only one file i.e., rating_final.csv that comprises the user, item and rating attributes.
   There are several types of collaborative filtering algorithms :
   - **User-User Collaborative filtering:** Here we find look alike customers (based on similarity) and offer products which first customer's look alike has chosen in past. This algorithm is very effective but takes a lot of time and resources. It requires to compute every customer pair information which takes time. Therefore, for big base platforms, this algorithm is hard to implement without a very strong parallelizable system.

2. **Contextual approach**: it generates the recommendations using the remaining eight data files.

A content based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link). Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

3. **K-Means Clustering:** K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly *k* different clusters of greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance.
   K-Means is relatively an efficient method. However, we need to specify the number of clusters, in advance and the final results are sensitive to initialization and often terminates at a local optimum. Unfortunately there is no global theoretical method to find the optimal number of clusters. A practical approach is to compare the outcomes of multiple runs with different *k* and choose the best one based on a predefined criterion. In general, a large *k* probably decreases the error but increases the risk of overfitting.

SCREENSHOTS:

READING THE DATASET:

```
In [99]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         import scipy as stats
         %matplotlib inline
```

```
In [100]: chefmozaccepts_df=pd.read_csv('RCdata/chefmozaccepts.csv')
          chefmozcuisine_df=pd.read_csv('RCdata/chefmozcuisine.csv')
          chefmozhours4_df=pd.read_csv('RCdata/chefmozhours4.csv')
          chefmozparking_df=pd.read_csv('RCdata/chefmozparking.csv')
          geoplaces2_df=pd.read_csv('RCdata/geoplaces2.csv',encoding='latin-1')
          rating_final_df=pd.read_csv('RCdata/rating_final.csv')

          usercuisine_df=pd.read_csv('RCdata/usercuisine.csv')
          userpayment_df=pd.read_csv('RCdata/userpayment.csv')
          userprofile_df=pd.read_csv('RCdata/userprofile.csv')
```

EXPLORING THE RESTAURANT FILES:

chefmozaccepts.csv dataset

it contains place id with payment method

```
In [3]: chefmozaccepts_df.columns
Out[3]: Index(['placeID', 'Rpayment'], dtype='object')

In [4]: chefmozaccepts_df.shape
Out[4]: (1314, 2)

In [5]: chefmozaccepts_df.info()
        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 1314 entries, 0 to 1313
        Data columns (total 2 columns):
        placeID     1314 non-null int64
        Rpayment    1314 non-null object
        dtypes: int64(1), object(1)
        memory usage: 20.6+ KB

In [6]: chefmozaccepts_df.isnull().sum()
Out[6]: placeID     0
        Rpayment    0
        dtype: int64

In [7]: chefmozaccepts_df.dtypes
Out[7]: placeID      int64
        Rpayment    object
        dtype: object

In [8]: chefmozaccepts_df['Rpayment'].describe()
Out[8]: count      1314
        unique       12
        top        cash
        freq        500
        Name: Rpayment, dtype: object
```

Out[9]:

	placeID	Rpayment
0	135110	cash
1	135110	VISA
2	135110	MasterCard-Eurocard
3	135110	American_Express
4	135110	bank_debit_cards

chefmozcuisine.csv dataset

it contains all the type of food with place id

In [10]:
```
chefmozcuisine_df.columns
```
Out[10]:
```
Index(['placeID', 'Rcuisine'], dtype='object')
```

In [11]:
```
chefmozcuisine_df.shape
```
Out[11]:
```
(916, 2)
```

In [12]:
```
chefmozcuisine_df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 916 entries, 0 to 915
Data columns (total 2 columns):
placeID     916 non-null int64
Rcuisine    916 non-null object
dtypes: int64(1), object(1)
memory usage: 14.4+ KB
```

In [13]:
```
chefmozcuisine_df['Rcuisine'].describe()
```
Out[13]:
```
count         916
unique         59
top       Mexican
freq          239
Name: Rcuisine, dtype: object
```

geoplaces2.csv dataset

In [29]:
```
geoplaces2_df.columns
```
Out[29]:
```
Index(['placeID', 'latitude', 'longitude', 'the_geom_meter', 'name', 'address',
       'city', 'state', 'country', 'fax', 'zip', 'alcohol', 'smoking_area',
       'dress_code', 'accessibility', 'price', 'url', 'Rambience', 'franchise',
       'area', 'other_services'],
      dtype='object')
```

In [30]:
```
geoplaces2_df.head()
```
Out[30]:

	placeID	latitude	longitude	the_geom_meter	name	address	city	state
0	134999	18.915421	-99.184871	0101000020957F000088568DE356715AC138C0A525FC46...	Kiku Cuernavaca	Revolucion	Cuernavaca	Morelos
1	132825	22.147392	-100.983092	0101000020957F00001AD016568C4858C1243261274BA5...	puesto de tacos	esquina santos degollado y leon guzman	s.l.p.	s.l.p.
2	135106	22.149709	-100.976093	0101000020957F0000649D6F21634858C119AE9BF528A3...	El Rincón de San Francisco	Universidad 169	San Luis Potosi	San Luis Potosi
3	132667	23.752697	-99.163359	0101000020957F00005D67BCDDED8157C1222A2DC8D84D...	little pizza Emilio Portes Gil	calle emilio portes gil	victoria	tamaulipas
4	132613	23.752903	-99.165076	0101000020957F00008EBA2D06DC8157C194E03B7B504E...	carnitas_mata	lic. Emilio portes gil	victoria	Tamaulipas

5 rows × 21 columns

K-MEANS CLUSTERING:

```
hs=userprofile_df.drop(['dress_preference','ambience','hijos','religion','color','height'], axis=1)

hs['longitude']=hs['longitude'].fillna(sum(hs['longitude'])/len(hs['longitude']))
#hs['laitude']=hs['latitude'].fillna(sum(hs['latitude'])/len(hs['latitude']))
hs['birth_year']=hs['birth_year'].fillna(sum(hs['birth_year'])/len(hs['birth_year']))
hs['weight']=hs['weight'].fillna(sum(hs['weight'])/len(hs['weight']))



hs['smoker'] =hs['smoker'].fillna(0)
ls=hs['smoker']
ls=ls.replace({'false': 0})
ls=ls.replace({'true': 1})
hs['smoker']=ls


hs['marital_status'] =hs['marital_status'].fillna(0)
lst=hs['marital_status']
lst=lst.replace({'single': 0})
lst=lst.replace({'widow': 1})
lst=lst.replace({'married': 2})
hs['marital_status']=lst


ls1=hs['drink_level']
ls1=ls1.replace({'abstemious': 0})
ls1=ls1.replace({'social drinker': 1})
ls1=ls1.replace({'drink_level': 2})
ls1=ls1.replace({'casual drinker': 3})
hs['drink_level']=ls1


hs['transport']=hs['transport'].fillna(0)
ls2=hs['transport']
ls2=ls2.replace({'on foot': 0})
ls2=ls2.replace({'public': 1})
ls2=ls2.replace({'transport': 2})
ls2=ls2.replace({'car owner': 3})
hs['transport']=ls2
```

In [37]:

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
from recommendations_list import recommendation_lists
%matplotlib inline
kmeans = KMeans(n_clusters = hs.shape[0]//20)

def recommendations_lists():
    import random
    rs={1:['132733','132665','135030','135084','135076','135062','135041','135045'],2:['135058','135084','135063','135088','1350
86','135063','135042','134992'],3:['135041','132870','135063','135088','135051','135068','135042','134754'],4:['135045','132870'
,'135063','132954','135013','135000','135042','135073'],5:['132825','135051','132925','135088','135051','132723','132834','13499
9'],6:['132613','134996','132925','135088','132830','132723','132834','132861'],7:['132723','135051','132862','135088','135051',
'132925','135047','134996'],8:['132766','132733','132925','135044','135051','132872','132665','135041']};
    recommendation_listss=[];
    for i in rs:
        recommendation_listss.append(rs[i]);
    recommendation_listsss=recommendation_listss[random.randint(1,8)]
    return recommendation_listsss

X = hs.copy()
X = hs.drop('userID', 1)

hs.head()
```

Out[37]:

	userID	latitude	longitude	smoker	drink_level	transport	marital_status	birth_year	interest	personality	...	Southeast_Asian	Burmese	(
0	U1001	22.139997	-100.978803	0	0	0	0	1989	2	0	...	0	0	(
1	U1002	22.150087	-100.983325	0	0	1	0	1990	1	1	...	0	0	(
2	U1003	22.119847	-100.946527	0	1	1	0	1989	0	3	...	0	0	(
3	U1004	18.867000	-99.183000	0	0	1	0	1940	2	3	...	0	0	(
4	U1005	22.183477	-100.959891	0	0	1	0	1992	0	0	...	0	0	(

5 rows x 117 columns

In [42]:
```python
rs1=list(rating_final_df['userID'])
rs=list(rating_final_df['placeID'])
js=list(rating_final_df['rating'])
lsn={}
lsn[rs1[0]]={str(rs[0]):js[0]}
kr=0
for i in rs1:
    lsn[i]={str(rs[kr]):js[kr]}
    kr=kr+1


lsn
```

Out[42]:
```
{'U1001': {'135051': 1},
 'U1002': {'135085': 1},
 'U1003': {'135059': 2},
 'U1004': {'132958': 2},
 'U1005': {'135032': 2},
 'U1006': {'132884': 2},
 'U1007': {'135038': 1},
 'U1008': {'135054': 1},
 'U1009': {'135079': 1},
 'U1010': {'135076': 1},
```

In [45]:
```python
rsts={}
for i in lsn1:
    if(i in rsts):
        rsts[i].append(sum(lsn1[i])/len(lsn1[i]));
    else:
        rsts[i]=[sum(lsn1[i])/len(lsn1[i])];
rsts
```

Out[45]:
```
{132560: [0.5],
 132561: [0.75],
 132564: [1.25],
 132572: [1.0],
 132583: [1.0],
 132584: [1.3333333333333333],
 132594: [0.5999999999999998],
 132608: [1.0],
 132609: [0.5999999999999998],
 132613: [1.1666666666666667],
 132626: [1.25],
 132630: [1.1666666666666667],
 132654: [0.25],
 132660: [1.3999999999999999],
 132663: [0.5],
 132665: [0.8000000000000004],
 132667: [1.25],
 132668: [1.0],
 132706: [0.75],
 132715: [1.0],
 132717: [1.3333333333333333],
 132723: [1.4166666666666667],
 132732: [0.625],
 132733: [1.3],
 132740: [0.75],
 132754: [1.4615384615384615],
 132755: [1.8],
 132766: [0.6666666666666663],
```

In [50]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from recommendations_list import recommendation_lists
from recommendations_data import reclists
from test_data import tests
from train_data import trains
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns

def similarity_score(person1,person2):
    both_viewed = {}
    for item in dataset[person1]:
        if item in dataset[person2]:
            both_viewed[item] = 1

        # Conditions to check they both have an common rating items
        if len(both_viewed) == 0:
            return 0

        # Finding Euclidean distance
        sum_of_eclidean_distance = []

        for item in dataset[person1]:
            if item in dataset[person2]:
                sum_of_eclidean_distance.append(pow(dataset[person1][item] - dataset[person2][item],2))
        sum_of_eclidean_distance = sum(sum_of_eclidean_distance)

        return 1/(1+sqrt(sum_of_eclidean_distance))



def pearson_correlation(person1,person2):
    both_rated = {}
    for item in dataset[person1]:
        if item in dataset[person2]:
            both_rated[item] = 1
```

```
def user_reommendations(person):
    totals = {}
    simSums = {}
    sim =[]
    for other in dataset:
        if other == person:
            continue
        sim = pearson_correlation(person,other)

        if sim <=0:
            continue
        for item in dataset[other]:
            if item not in dataset[person] or dataset[person][item] == 0:
                totals.setdefault(item,0)
#                 print("Sim",sim,person,other)
                totals[item] += dataset[other][item]* sim
                simSums.setdefault(item,0)
                simSums[item]+= sim

    rankings = [(total/simSums[item],item) for item,total in totals.items()]
    rankings.sort()

    rankings.reverse()
#     print("rankings", rankings)
    recommendation_list = [recommend_item for score,recommend_item in rankings]

    return recommendation_list


print("SIMILARITY SCORES\n")
print(similarity_score('U1004','U1001'),"\n")
print("MOST SIMILAR USERS\n")
print(most_similar_users('U1002',3))
print("\nPEARSON_CORRELATION\n")
print(pearson_correlation('U1002','U1001'))
print("\nUSER Recommendation\n")
print(user_reommendations('U1002'))
```

```
SIMILARITY SCORES

0.4142135623730951

MOST SIMILAR USERS

[(1.0, 'U1022'), (1.0, 'U1016'), (0.9999999999999987, 'U1020')]

PEARSON_CORRELATION

0.2857142857142859

USER Recommendation

['135089', '135085', '135073', '135039', '135034', '135030', '134996', '132668', '135118', '135083', '135033', '135031']
```

In [51]:
```
from collections import Counter
```

In [52]:
```
def get_liked_places(user):
    return [place for place in dataset[user] if dataset[user][place] >= 2]


def calculate_recall(user):
    liked_places = get_liked_places(user)
    predicted_places = user_reommendations(user)
    if len(liked_places) == 0:
        return 0.0
    else:
        return len(intersect(liked_places, predicted_places)) / len(liked_places)

def calculate_precision(user, liked_places_count):
    predicted_places = user_reommendations(user)
    if len(predicted_places) == 0:
        return 0.0
    else:
        return liked_places_count / len(predicted_places)
```

In [60]:
```
from sklearn.model_selection import train_test_split
columns=['userID','placeID','rating','food_rating', 'service_rating']
df = pd.DataFrame(rating_final_df,columns=columns)
df.drop(['food_rating', 'service_rating'], axis=1)
df.head()
```

Out[60]:

	userID	placeID	rating	food_rating	service_rating
0	U1077	135085	2	2	2
1	U1077	135038	2	2	1
2	U1077	132825	2	2	2
3	U1077	135060	1	2	2
4	U1068	135104	1	1	2

In [101]:
```
datasets
```

Out[101]:
```
{'U1001': {'135032': 3, '135038': 3, '135058': 1, '135076': 2, '135084': 2},
 'U1002': {'135032': 1, '135038': 2, '135058': 2, '135076': 3, '135084': 1},
 'U1003': {'135031': 1, '135032': 2, '135058': 2, '135076': 3, '135084': 3},
 'U1004': {'135032': 3, '135038': 3, '135058': 1, '135076': 3, '135084': 3},
 'U1005': {'135032': 2, '135038': 2, '135058': 1, '135076': 3, '135084': 2},
 'U1006': {'135032': 3, '135058': 1, '135076': 1, '135084': 2},
 'U1007': {'135032': 3, '135033': 1, '135058': 1, '135076': 2, '135084': 2},
 'U1008': {'135032': 3, '135038': 3, '135058': 3, '135076': 3, '135084': 1},
 'U1009': {'135032': 1, '135039': 3, '135058': 1, '135076': 1, '135084': 2},
 'U1010': {'135032': 3, '135058': 3, '135076': 3, '135084': 1},
 'U1011': {'135015': 2, '135018': 2, '135060': 3},
 'U1012': {'135032': 3, '135038': 2, '135058': 2, '135076': 2, '135083': 1},
 'U1013': {'135013': 3, '135015': 2, '135018': 1, '135028': 2, '135060': 2},
 'U1014': {'135015': 2, '135032': 1, '135034': 3, '135039': 3, '135060': 3},
 'U1015': {'135015': 2, '135018': 2, '135060': 1},
 'U1016': {'135032': 1, '135038': 2, '135058': 2, '135076': 3, '135083': 2},
 'U1017': {'135048': 2, '135058': 2, '135104': 1, '135126': 3, '135132': 1},
```

In [64]:
```
for col in geoplaces2_df.columns:
    if geoplaces2_df[col].dtype == object:
        print(col, (geoplaces2_df[col].str.contains('\?') == True).sum())
```

```
the_geom_meter 0
name 0
address 27
city 18
state 18
country 28
fax 130
zip 74
alcohol 0
smoking_area 0
dress_code 0
accessibility 0
price 0
url 116
Rambience 0
franchise 0
area 0
other_services 0
```

In [65]:
```
columns_with_na_vals= ['address','city','state','country','fax','zip','url']
```

In [66]:
```
for col in columns_with_na_vals:
    geoplaces2_df[col] = (geoplaces2_df[col].replace({'?': np.NaN}))
```

In [67]:
```
geoplaces2_df.head()
```

Out[67]:

	placeID	latitude	longitude	the_geom_meter	name	address	city	state
0	134999	18.915421	-99.184871	0101000020957F000088568DE356715AC138C0A525FC46...	Kiku Cuernavaca	Revolucion	Cuernavaca	Morelos
1	132825	22.147392	-100.983092	0101000020957F00001AD016568C4858C1243261274BA5...	puesto de tacos	esquina santos degollado y leon guzman	s.l.p.	s.l.p.
2	135106	22.149709	-100.976093	0101000020957F0000649D6F21634858C119AE9BF528A3...	El Rincón de San Francisco	Universidad 169	San Luis Potosi	San Luis Potosi

Out[68]:

e_geom_meter	name	address	city	state	country	fax	...	alcohol	smoking_area	dress_code	
0A525FC46...	Kiku Cuernavaca	Revolucion	Cuernavaca	Morelos	Mexico	NaN	...	No_Alcohol_Served	none	informal	no
261274BA5...	puesto de tacos	esquina santos degollado y leon guzman	s.l.p.	s.l.p.	mexico	NaN	...	No_Alcohol_Served	none	informal	co
E9BF528A3...	El Rincón de San Francisco	Universidad 169	San Luis Potosi	San Luis Potosi	Mexico	NaN	...	Wine-Beer	only at bar	informal	pa
2A2DC8D84D...	little pizza Emilio Portes Gil	calle emilio portes gil	victoria	tamaulipas	NaN	NaN	...	No_Alcohol_Served	none	informal	co
E03B7B504E...	carnitas_mata	lic. Emilio portes gil	victoria	Tamaulipas	Mexico	NaN	...	No_Alcohol_Served	permitted	informal	co
AAEFD2CA2...	Restaurant los Compadres	Camino a Simon Diaz 155 Centro	San Luis Potosi	SLP	Mexico	NaN	...	Wine-Beer	none	informal	no
2FECBF84F...	Taqueria EL amigo	Calle Mezquite Fracc Framboyanes	Cd Victoria	Tamaulipas	Mexico	NaN	...	No_Alcohol_Served	none	casual	co
31D2A31A8...	shi ro ie	NaN	NaN	NaN	NaN	NaN	...	Wine-Beer	section	informal	no
51EB22C4E...	Pollo_Frito_Buenos_Aires	tampico	victoria	Tamaulipas	Mexico	NaN	...	No_Alcohol_Served	not permitted	informal	co
EEBB73A991...	la Estrella de Dimas	Villa de Pozos 192 Villa de Pozos	San Luis Potosi	SLP	Mexico	NaN	...	No_Alcohol_Served	none	informal	no
F85FA9791...	Restaurante 75	Villa de Pozos 4497 Villa de Pozos	San Luis Potosi	SLP	Mexico	NaN	...	No_Alcohol_Served	none	informal	no

Concatenating all restaurant file together of common column placeID:

In [69]:
```
restaurant_all = np.concatenate((chefmozaccepts_df.placeID, chefmozcuisine_df.placeID, chefmozhours4_df.placeID, chefmozparking_
df.placeID, geoplaces2_df.placeID))
restaurant_all = np.sort(np.unique(restaurant_all) ) # All the unique placeID's
print(len(restaurant_all))
```
```
938
```

In [70]:
```
restaurant_all
```
Out[70]:
```
array([132001, 132002, 132003, 132004, 132005, 132006, 132007, 132008,
       132009, 132010, 132012, 132013, 132014, 132015, 132016, 132017,
       132018, 132019, 132020, 132021, 132022, 132023, 132024, 132025,
       132026, 132028, 132030, 132031, 132083, 132087, 132092, 132094,
       132096, 132097, 132098, 132100, 132101, 132102, 132103, 132105,
       132106, 132107, 132108, 132109, 132111, 132114, 132115, 132116,
       132118, 132119, 132120, 132121, 132125, 132126, 132127, 132128,
       132130, 132131, 132132, 132133, 132136, 132137, 132138, 132145,
       132146, 132147, 132148, 132155, 132156, 132157, 132159, 132160,
       132161, 132162, 132163, 132164, 132165, 132166, 132167, 132171,
       132174, 132175, 132177, 132180, 132182, 132184, 132186, 132187,
       132191, 132192, 132201, 132203, 132204, 132206, 132207, 132208,
       132209, 132210, 132211, 132212, 132213, 132214, 132215, 132216,
       132217, 132218, 132220, 132221, 132223, 132225, 132226, 132228,
       132229, 132230, 132231, 132232, 132233, 132234, 132235, 132237,
       132238, 132239, 132240, 132242, 132243, 132244, 132245, 132246,
       132247, 132249, 132250, 132253, 132255, 132257, 132258, 132262,
       132265, 132266, 132268, 132269, 132270, 132271, 132272, 132273,
       132274, 132275, 132280, 132281, 132283, 132284, 132289, 132292,
       132293, 132294, 132295, 132296, 132297, 132298, 132299, 132300,
       132301, 132302, 132306, 132310, 132312, 132315, 132316, 132317,
       132318, 132319, 132321, 132324, 132326, 132328, 132329, 132330,
       132331, 132332, 132333, 132335, 132336, 132337, 132339, 132341,
       132342, 132343, 132344, 132345, 132346, 132347, 132348, 132349,
```

Exploring the user item rating files:

In [71]:
```
rating_final_df.columns
```
Out[71]:
```
Index(['userID', 'placeID', 'rating', 'food_rating', 'service_rating'], dtype='object')
```

In [72]:
```
rating_final_df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1161 entries, 0 to 1160
Data columns (total 5 columns):
userID            1161 non-null object
placeID           1161 non-null int64
rating            1161 non-null int64
food_rating       1161 non-null int64
service_rating    1161 non-null int64
dtypes: int64(4), object(1)
memory usage: 45.4+ KB
```

In [73]:
```
rating_final_df.shape
```
Out[73]:
```
(1161, 5)
```

In [74]:
```
rating_final_df.isnull().sum()
```
Out[74]:
```
userID            0
placeID           0
rating            0
food_rating       0
service_rating    0
dtype: int64
```

In [75]:
```
rating_final_df[['rating','food_rating','service_rating']].describe()
```
Out[75]:

	rating	food_rating	service_rating
count	1161.000000	1161.000000	1161.000000
mean	1.199828	1.215332	1.090439
std	0.773282	0.792294	0.790844
min	0.000000	0.000000	0.000000
25%	1.000000	1.000000	0.000000
50%	1.000000	1.000000	1.000000
75%	2.000000	2.000000	2.000000
max	2.000000	2.000000	2.000000

In [76]:
```
rating_final_df.head()
```

Out[76]:

	userID	placeID	rating	food_rating	service_rating
0	U1077	135085	2	2	2
1	U1077	135038	2	2	1
2	U1077	132825	2	2	2
3	U1077	135060	1	2	2
4	U1068	135104	1	1	2

In [77]:
```
len(rating_final_df.placeID.unique())
```

Out[77]: 130

CONCLUSION

The project was completed and our team successfully created our Restaurant Recommendation System. We provided our users with various features, which would, enhanced the user experience. Our project helps many users thus, our objective of this project was completed and the user textures a full and satisfactory experience.