ECE 9014-2: Group Project 2 & Individual Assignment 2

Restaurant Recommendation System

Group ID: *11*

Sahana Chakravarty (251183920)

Group Members

- Sahana Chakravarty (251183920) – schakr55@uwo.ca – (100%)
- Tejas Khatri (251131104)- tkhatri2@uwo.ca – (100%)
- Urva Doshi (251166816) - udoshi@uwo.ca – (50%)
- Muhammad Khan (Not Provided) - mkha83@uwo.ca – (0%)

Department of Electrical & Computer Engineering
The University of Western Ontario

Table of Content

1 Instructions

- Submission due date: ***Monday 29 November 2021 at 11:59 PM***. Don't leave the submission to the last minute. ***This deadline includes a 48-hour bonus for everyone, including students with an approved academic consideration request. The deadline will not be extended for anyone.***
- Late submissions are *not accepted*.
- Each *individual* must submit the followings, *in order* and *naming convention of the questions* in this hand out, in a *single pdf file* through OWL:
    - *Group deliverables*
    - *Individual deliverable*
- The submitted scripts must be *well organized*, *easy to read*, and *ready to test*.
- Name your file with the following format: *GroupID_LastName_FirstNameInitial_GI2.pdf*
  Example: The file name for John Nash in group 2 is G*2_Nash_J_GI2.pdf*
- Fill out the cover page with corresponding information.

- Marking:

    - Group Project is out of 100, which corresponds to 100/4=25 of the final mark.
    - Individual Assignment is out of 90, which corresponds to 90/6=15 of the final mark.
    - Format of Submission: A submission will lose 20% of the Individual Assignment mark if:
        - The format/organization/order of submission is different from the hand out.
        - The name/group number/project name is/are not mentioned on the cover page.
        - The submission file name and format does not follow the instruction above.

2 Group Deliverables

In this exercise, the team defines an analytical/predictive task as a problem statement using the chosen dataset. Then the team has to answer the following questions.

2.1 Problem Statement

2.1.1 Definition

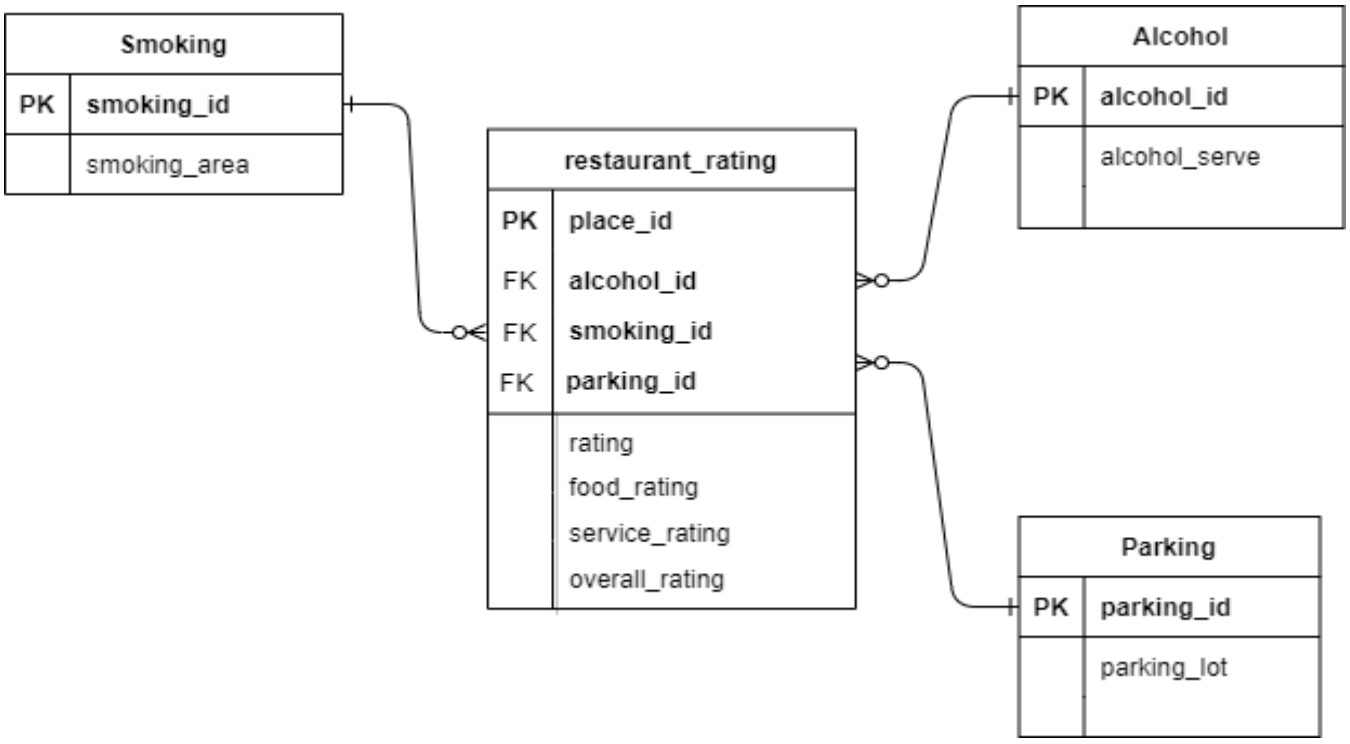
Provide a description of your proposed problem statement in one sentence, limited to maximum 30 words. Example: Admission rate prediction to the emergency room of the St. Joseph hospital. [10/100]

Overall restaurant rating prediction based on three features such as smoking area availability, alcohol served or not, parking availability.

2.2 Conceptual Model

2.2.1 Diagram

Provide a diagram (schema) representing the conceptual model for addressing the proposed problem statement (must contain at least three dimension tables). The schema must include a placeholder for analytical/inference results. [10/100]



2.2.2 Description

Provide the schema type, description, and the reasoning behind the proposed schema. [10/100]

The database organizational structure used in our data model is a Star Schema.

A star schema is a type of modelling approach adopted by relational data warehouses where there is a single fact table with one or more smaller dimensional tables in a star shape.

The reason for choosing a star schema is that our data model is highly denormalized to 3NF with one fact table called restaurant_rating and 3 dimension tables called smoking, alcohol and parking and has no subdimension tables. Also, it resembles that of a star-shape.

2.2.3 DDL Scripts

Provide the SQL scripts for the definition of the fact table and the dimension tables, including the allocated placeholder for the analytical/inference results. [10/100]

```
CREATE TABLE IF NOT EXISTS parking(
   Parking_id int PRIMARY KEY,
   Parking_lot char(25)
);
```

```
CREATE TABLE IF NOT EXISTS Smoking(
Smoking_id int PRIMARY KEY,
Smoking_area char(25)
);
```

```
CREATE TABLE IF NOT EXISTS Alcohol(
Alcohol_id int PRIMARY KEY,
Alcohol_serve char(25)
);
```

```
CREATE TABLE IF NOT EXISTS Restaurant_Rating(
place_id int Primary key,
Parking_id int,
Smoking_id int,
Alcohol_id int,
Rating int,
Food_rating int,
Service_rating int,
overall_rating int
);
```

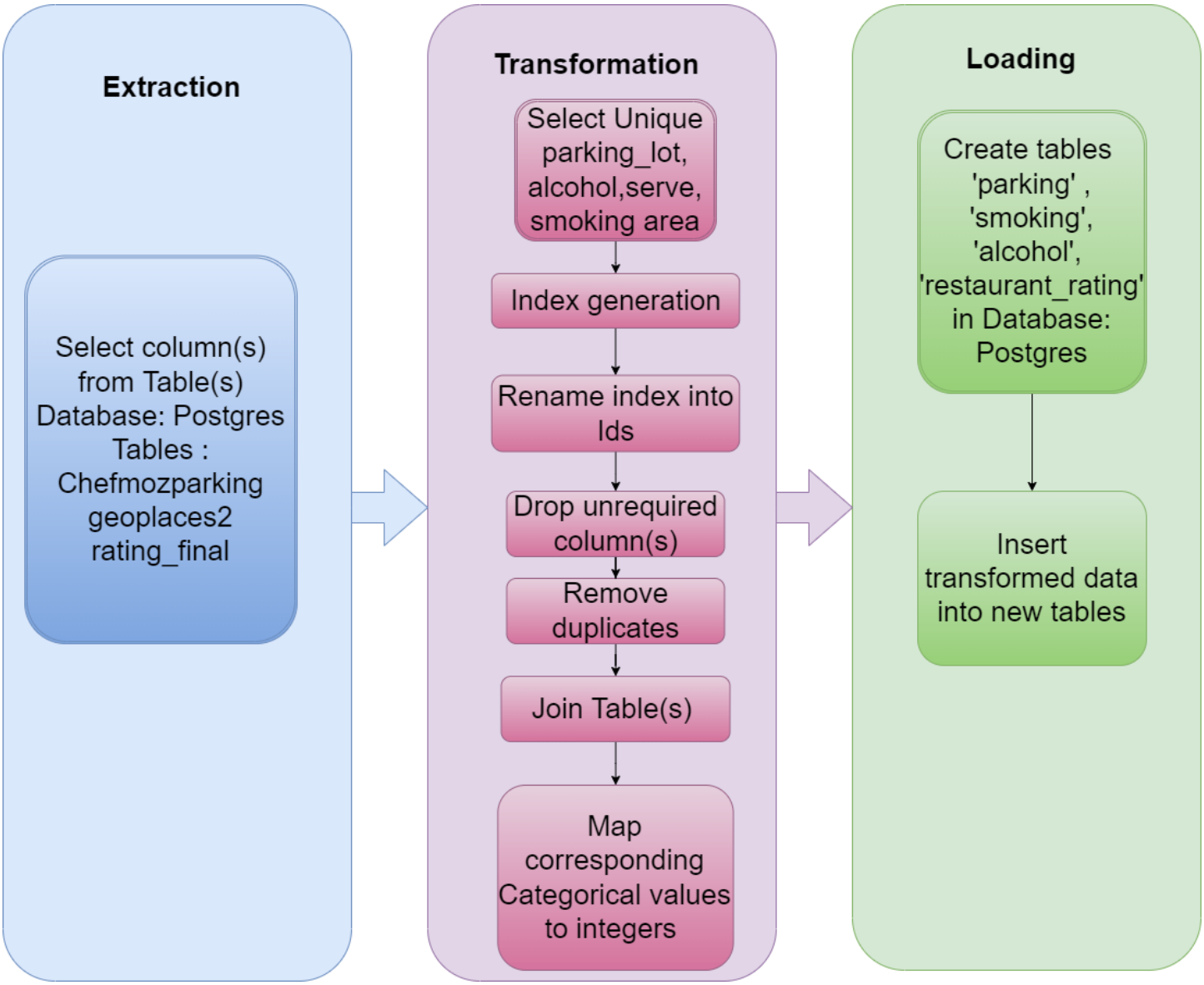
```
ALTER TABLE "restaurant_rating"
ADD FOREIGN KEY ("parking_id") REFERENCES "parking" ("parking_id")

ALTER TABLE "restaurant_rating"
ADD FOREIGN KEY ("smoking_id") REFERENCES "smoking" ("smoking_id")
```

```
ALTER TABLE "restaurant_rating"
ADD FOREIGN KEY ("alcohol_id") REFERENCES "alcohol" ("alcohol_id")
```

The placeholder column used is called overall rating in ‘Restaurant_Rating’ table.

2.2.4 ETL Process

Propose and describe a simple ETL process to populate a data cube suitable for the proposed project. [10/100]



ETL, which stands for extract, transform and load, is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system.

For the Extraction phase of our ETL Process, since we already had data in staging area in 9 tables, our problem statement required use of only 3 tables such as Chefmozparking, geoplaces2 and final rating. Selection of relevant columns such as parking, alcohol, smoking

area, rating, food_rating, service_rating and place_id was done from the above 3 tables. Since we implemented our scripts in python, all the selected columns were stored in respective dataframes.

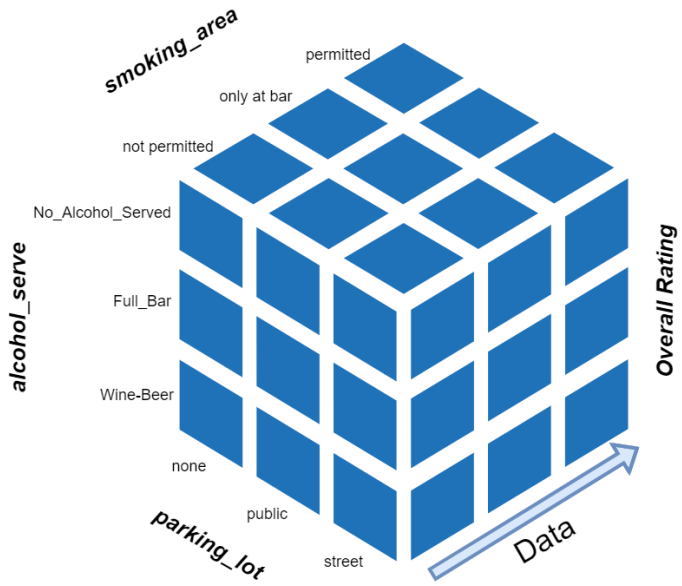
In the Transformation phase, we used the unique function to find categorical data in columns such as parking_lot, alcohol and smoking_area. We generated indexes for the unique categories and renamed them into corresponding ids. Further, we dropped irrelevant columns and removed duplicates.

We found out that there were 938 restaurants in total and only 130 restaurants had rating in them. Since we did not want to consider the restaurants which did not have any rating, we removed those restaurants.

Then, to fetch all the matching columns on the basis of place ids, we did a left outer join. After that, mapping was done for respective categorical unique values with integers. These integers were assumed to convert the categorical values to numerical form in order to encode. For instance, 'Wine-Beer':2, 'No_Alcohol_Served':1, 'Full_Bar':0.

For the Loading process, we created 4 target tables where 3 of them were dimension tables called ‘parking’, ‘smoking’ and ‘alcohol’ and one fact table called ‘restaurant_rating’ and data was inserted into these tables.

Thus, we implemented a data cube consisting of smoking_area, parking_lot, alcohol serve and overall rating.



2.2.5 ETL Scripts

Provide the SQL scripts for the proposed ETL process. [10/100]

Extract Script:

```
from sqlalchemy import create_engine

from numpy import unique

import pandas as pd

dma_engine = create_engine('postgresql://druid:FoolishPassword@localhost:5432/postgres')
```

```
# create pandas table from query and connection
def create_pandas_table(sql_query, conn_engine=dma_engine):
    table = pd.read_sql_query(sql_query, con=conn_engine)
    return table
alco_df = create_pandas_table('select alcohol from geoplaces2', dma_engine)
smok_df = create_pandas_table('select smoking_area from geoplaces2', dma_engine)
geo_df = create_pandas_table('select * from geoplaces2', dma_engine)
park_df = create_pandas_table('select * from chefmozparking', dma_engine)
rating = create_pandas_table('select * from rating_final', dma_engine)
```

Transform Script :

Parking Transformation:

```
df= pd.DataFrame({'parking_lot':unique(park_df.parking_lot)})
df.reset_index(level=0, inplace=True)
parking = df.rename({'index': 'parking_id'}, axis=1)
```

	parking_id	parking_lot
0	0	fee
1	1	none
2	2	public
3	3	street
4	4	valet parking
5	5	validated parking
6	6	yes

Alcohol Transformation:

```
alco_df = pd.DataFrame({'alcohol_serve':unique(alco_df.alcohol)})
alco_df.reset_index(level=0, inplace=True)
alcohol = alco_df.rename({'index': 'alcohol_id'}, axis=1)
```

	alcohol_id	alcohol_serve
0	0	Full_Bar
1	1	No_Alcohol_Served
2	2	Wine-Beer

Smoking Transformation:

```
smok_df = pd.DataFrame({'smoking_area':unique(smok_df.smoking_area)})
smok_df.reset_index(level=0, inplace=True)
smoking = smok_df.rename({'index': 'smoking_id'}, axis=1)
```

	smoking_id	smoking_area
0	0	none
1	1	not permitted
2	2	only at bar
3	3	permitted
4	4	section

Restaurant_rating Transformation:

```
geo_info = geo_df[['place_id','alcohol','smoking_area']]
rating.drop(['user_id'], axis=1,inplace=True)
rating_clean= rating.drop_duplicates(subset=['place_id'])
df_res = pd.merge(left=geo_info, right=park_df, how="left", on="place_id")
df_res = pd.merge(left=df_res, right=rating_clean, how="left", on="place_id")
```

```
df_res
```

	place_id	alcohol	smoking_area	parking_lot	rating	food_rating	service_rating
0	134999	No_Alcohol_Served	none	none	2	2	2
1	132825	No_Alcohol_Served	none	none	2	2	2
2	135106	Wine-Beer	only at bar	none	2	2	2
3	132667	No_Alcohol_Served	none	none	1	2	2
4	132613	No_Alcohol_Served	permitted	yes	2	2	2
...	...	...	...	...	...	...	...
125	132866	No_Alcohol_Served	not permitted	yes	2	2	1
126	135072	No_Alcohol_Served	none	none	1	2	2
127	135109	Wine-Beer	not permitted	none	0	0	0
128	135019	No_Alcohol_Served	none	none	2	2	2
129	132877	No_Alcohol_Served	none	none	0	0	0

130 rows × 7 columns

```
# 'Wine-Beer':2, 'No_Alcohol_Served':1, 'Full_Bar':0
```

```
df_res.alcohol = df_res.alcohol.map({'Wine-Beer':2, 'No_Alcohol_Served':1, 'Full_Bar':0})

# 'none':0, 'not permitted':1, 'only at bar':2, 'permitted':3, 'section':4

df_res.smoking_area = df_res.smoking_area.map({'none':0, 'not permitted':1, 'only at bar':2,
'permitted':3, 'section':4})


# 'fee':0, 'none':1, 'public':2, 'yes':6,'street':3, 'valet parking':4, 'validated parking':5

df_res.parking_lot = df_res.parking_lot.map({'fee':0, 'none':1, 'public':2, 'yes':6,
                                             'street':3, 'valet parking':4, 'validated parking':5})

df_res = df_res.rename(columns={'alcohol': 'alcohol_id', 'smoking_area': 'smoking_id',
'parking_lot': 'parking_id'})
```

Load Script :

Definition to load data into database

```
# Loading Dataframe to Database

def insert_dataframe_to_database(table_name, dataframe, engine, index_id=None,
override='replace'):
    if index_id is not None:
        dataframe.to_sql(table_name, con=engine, if_exists=override, index_label=index_id)
        print('Data inserted successfully into '+table_name)
    else:
        dataframe.to_sql(table_name, con=engine, if_exists=override, index=False)
        print('Data inserted successfully into ' + table_name)
```

Loading parking dimension into database

```
insert_dataframe_to_database('parking',parking,dma_engine,'append')
```

```
select* from parking
```

parking_id	parking_lot
0	fee
1	none
2	public
3	street
4	valet parking
5	validated parking
6	yes

Loading smoking dimension into database

insert_dataframe_to_database('smoking',smoking,dma_engine,'append')

```
select * from smoking
```

smoking_id	smoking_area
0	none
1	not permitted
2	only at bar
3	permitted
4	section

Loading alcohol dimension into database

insert_dataframe_to_database('alcohol',alcohol,dma_engine,'append')

```
select * from alcohol
```

alcohol_id	alcohol_serve
0	Full_Bar
1	No_Alcohol_Served
2	Wine-Beer

Loading restaurant_rating fact into database

insert_dataframe_to_database('restaurant_rating',df_res,dma_engine,'append')

place_id	alcohol_id	smoking_id	parking_id	rating	food_rating	service_rating	overall_rating
134999	1	0	1	2	2	2	*NULL*
132825	1	0	1	2	2	2	*NULL*
135106	2	2	1	2	2	2	*NULL*
132667	1	0	1	1	2	2	*NULL*
132613	1	3	6	2	2	2	*NULL*
135040	2	0	6	0	0	0	*NULL*

2.3 Analytical/Predictive Model

2.3.1 Description

Propose and discuss an analytical/predictive model to address the problem statement. Discuss the type of the model (e.g. supervised, unsupervised, classification, regression, etc.) and the reasoning behind this approach. [10/100]

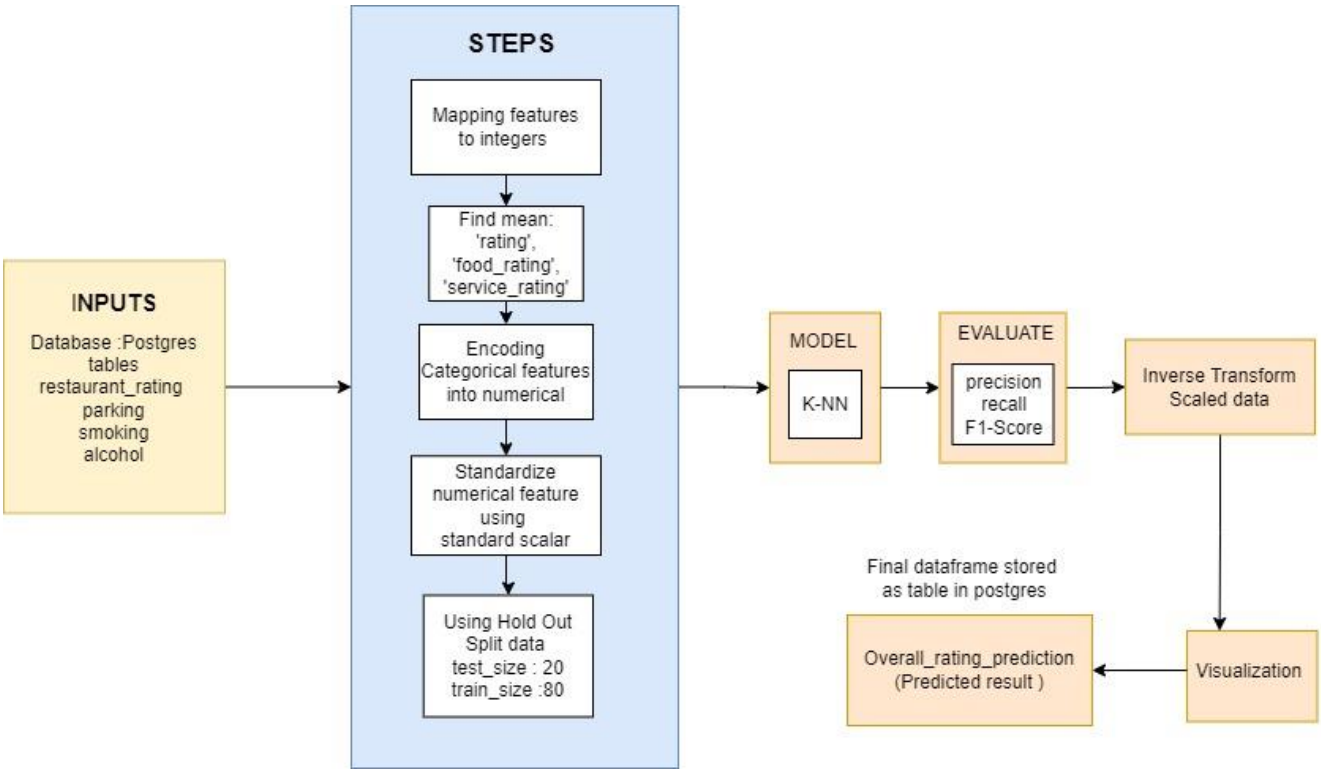
As discussed above, our problem statement is to predict the overall restaurant rating based on three features such as smoking area availability, alcohol served or not, parking availability.
We have approached the problem statement using Classification in Supervised learning where we have considered supervised learning as we have dependent and independent features in our data. The dependent feature is overall_rating and the independent features are smoking area availability, alcohol served or not, parking availability. Since the dependent feature,overall rating has three classes such as 0,1, 2 , this lead to a multi-label classification problem.

The model which we have used is K-NN (K- Nearest Neighbor classifier) that predicts the overall rating for restaurants based on selected features. (smoking area availability, alcohol served or not, parking availability)

We compared other models such as Logistic Regression, Decision Tree and Random Forest but K-NN gave us the best result.

2.3.2 Model Diagram

Provided a diagram visualizing inputs, steps, and outputs of the proposed model. [10/100]



2.3.3 Model Scripts

Provided the scripts for implementation of the proposed model (Must include scripts for querying the data, model, inserting results into the appropriate table(s)). [10/100]

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
from sqlalchemy import create_engine
#Connecting to postgres to fetch columns required for prediction
dma_engine = create_engine('postgresql://druid:FoolishPassword@localhost:5432/postgres')

# Create pandas table from query and connection
import pandas as pd
def create_pandas_table(sql_query, conn_engine=dma_engine):
 table = pd.read_sql_query(sql_query, con=conn_engine)
 return table

#Fetching columns by querying the database
data = create_pandas_table ('select r.place_id, a.alcohol_serve, p.parking_lot,
s.smoking_area , r.rating, r.food_rating, r.service_rating
from restaurant_rating as r
join parking as p on p.parking_id = r.parking_id
join smoking as s on s.smoking_id = r.smoking_id
join alcohol as a on a.alcohol_id =r.alcohol_id ,dma_engine)


# Mapping features to encode:1 if alcohol is available, 0 otherwise
data.alcohol_serve = data.alcohol_serve.map(lambda x: 0 if x == 'No_Alcohol_Served' else 1)
data.parking_lot = data.parking_lot.map({'fee':1, 'none':0, 'public':1, 'yes':2,
                                         'street':1, 'valet parking':1, 'validated parking':1})
# 1 if there is smoking area, 0 otherwise
```

```
data.smoking_area = data.smoking_area.map(lambda x: 0 if (x == 'none') | (x == 'not permitted') else 1)

#Finding mean
data['total_rating'] = data[['rating', 'food_rating','service_rating']].mean(axis=1).round()
data.drop(['rating', 'food_rating','service_rating'], axis=1, inplace=True)
X=data.iloc[:,1:4].values
Y=data.iloc[:,-1].values
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=1)
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
def models(X_train,Y_train):

 # K – Nearest Neighbor Classifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
def models(X_train,Y_train):

  # K – Nearest Neighbor Classifier
  print()
  print('K – Nearest Neighbor Classifier(KNN)')
  print()
  from sklearn.neighbors import KNeighborsClassifier
  knn= KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
  knn.fit(X_train, Y_train)
  y_pred_knn=knn.predict(X_test)
  cm=confusion_matrix(Y_test,y_pred_knn)



  ax = sns.heatmap(cm, annot = True)
  plt.title('Heatmap of Confusion Matrix', fontsize = 15)
```

```
  plt.ylabel('Actual')
  plt.xlabel('Predicted')
  plt.show()
  print(cm)
  print(classification_report(Y_test,y_pred_knn))
  print("Accuracy_Score: ",accuracy_score(Y_test,y_pred_knn))
  print()
  print('All K- fold cross val',cross_val_score(knn, X_train, Y_train, cv=5))
  print('Mean of all K-fold cross Val',np.mean(cross_val_score(knn, X_train, Y_train, cv=5)))
  print()
  return knn
model=models(X_train,Y_train)
```

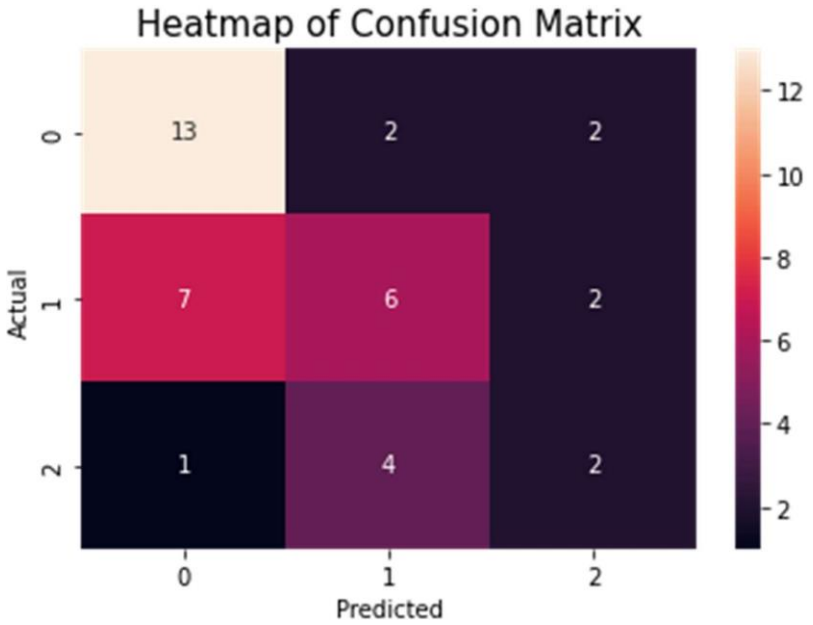
#Loading to final placeholder column

```
from sqlalchemy import create_engine
dma_engine = create_engine('postgresql://druid:FoolishPassword@localhost:5432/postgres')
final_df.to_sql('overall_rating_prediction', dma_engine, if_exists='replace')
```

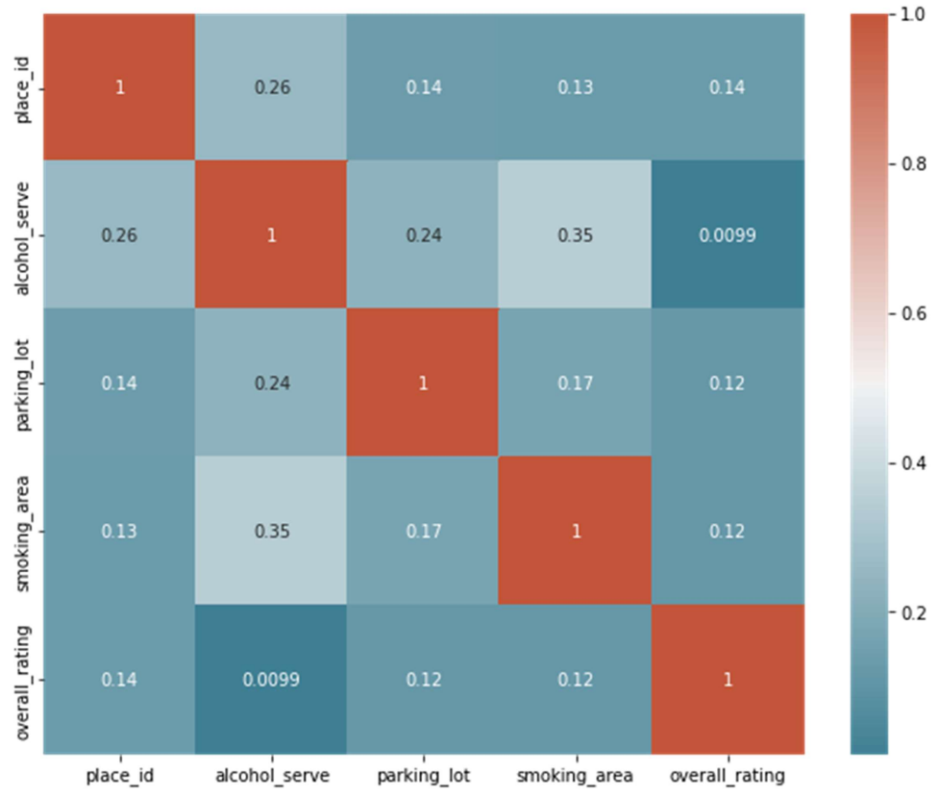
2.3.4 Outputs

Visualize analytical and inference results from the proposed model (at least three plots). [10/100]

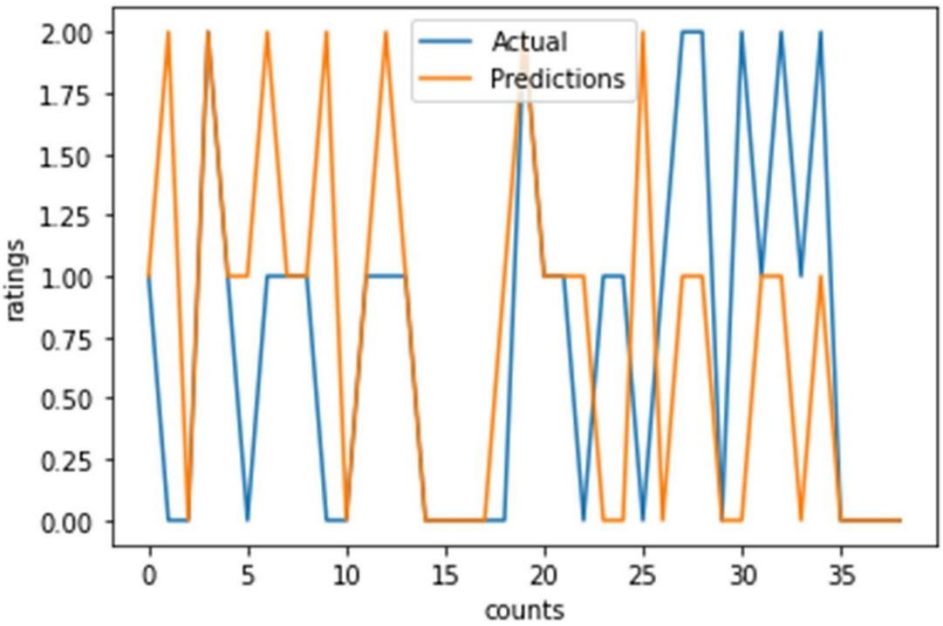
K-NN Classifier Confusion matrix



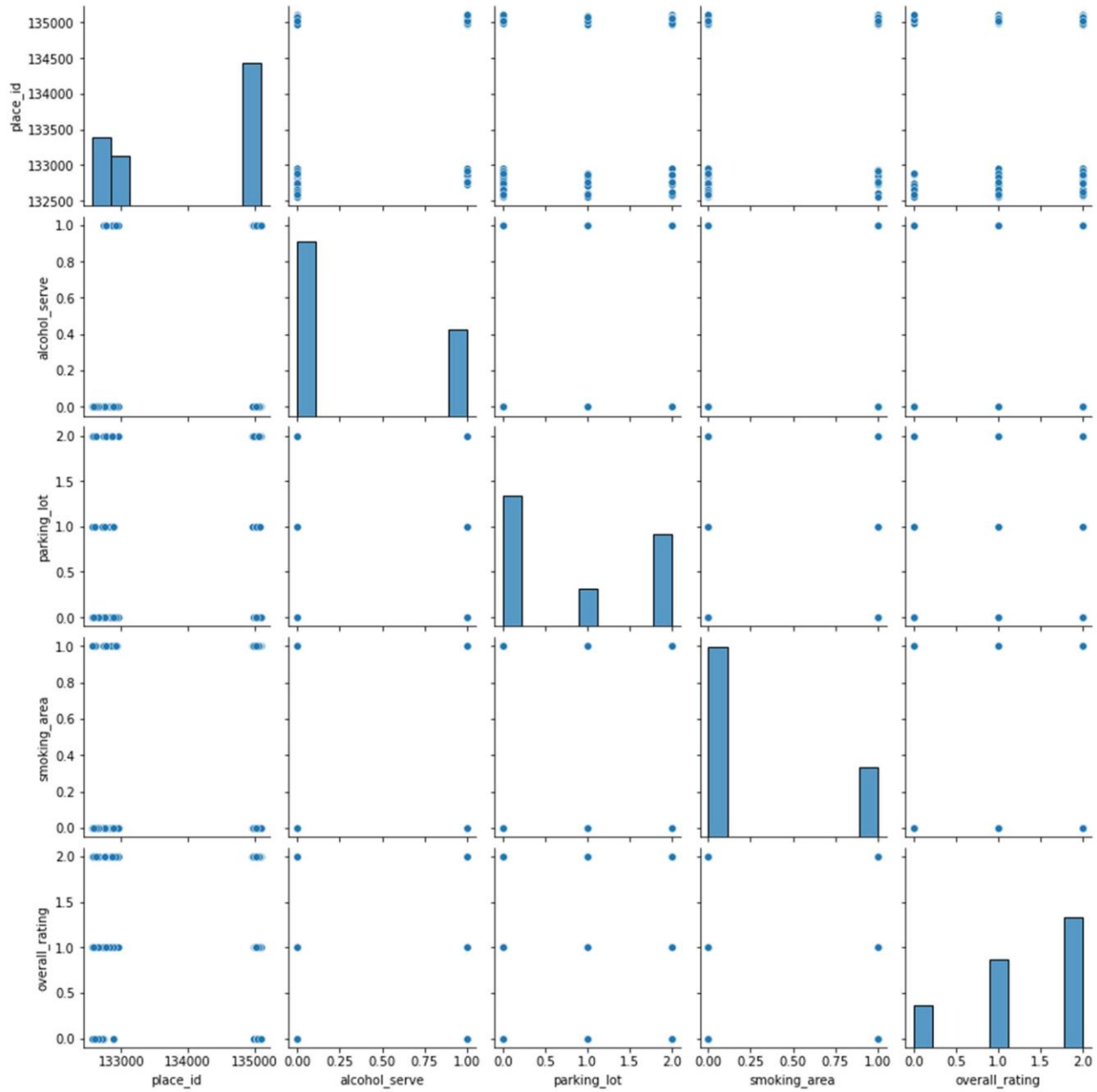
Correlation between all features with respect to target variable Heatmap



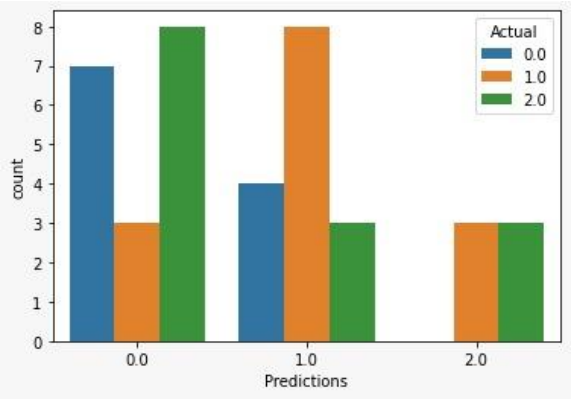
K-NN Actual and Predicted Values



Below, we can see the pair plot of Data:



Count Plot for Prediction Count



3 Individual Deliverables

In this exercise, each team member individually proposes an extension of the group project. Then the individual has to answer the following questions.

3.1 Problem Statement

3.1.1 Definition

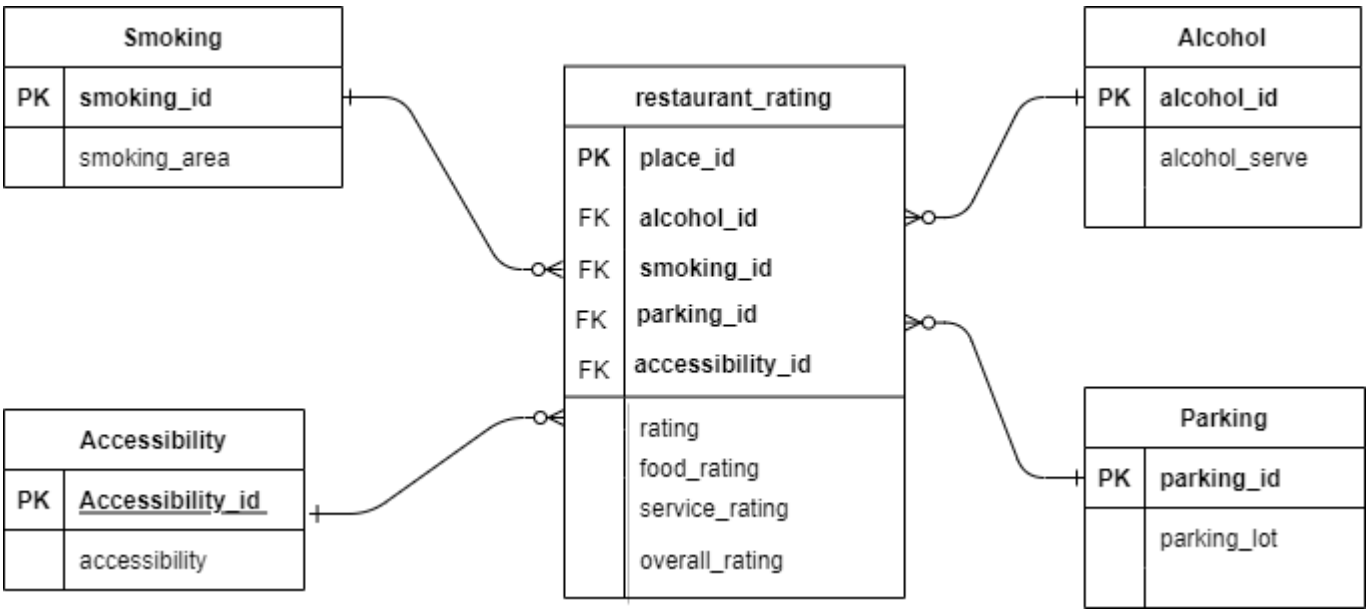
Provide a description of your proposed extension to the problem statement in the group project, limited to maximum 30 words. Example 1: Admission rate prediction to the emergency room of the St. Joseph hospital based on the accident data in the city. Example 2: Mortality rate prediction in the emergency room of the St. Joseph hospital. [10/90]

Prediction of overall ratings of restaurants based on an additional feature called accessibility apart from using existing features such as smoking area availability, alcohol served or not, parking availability.

3.2 Conceptual Model

3.2.1 Diagram

Provide a diagram (schema) representing the conceptual model for addressing the proposed problem statement (must contain at least one more dimension table than the original schema proposed in the group project). The schema must include a placeholder for analytical/inference results. [10/90]



3.2.2 Description

Provide the schema type, description, and the reasoning behind the proposed schema. [10/90]

The database organizational structure used in our data model is a Star Schema.

A star schema is a type of modelling approach adopted by relational data warehouses where there is a single fact table with one or more smaller dimensional tables in a star shape.

The reason for choosing a star schema is that the data model is highly denormalized to 3NF with one fact table called restaurant_rating and 4 dimension tables called smoking, alcohol, parking and accessibility, with no subdimension tables. Also, it resembles that of a star-shape.

3.2.3 DDL Scripts

Provide the SQL scripts for the definition of the fact table and the dimension tables. [10/90]

```
CREATE TABLE IF NOT EXISTS parking(
   Parking_id int PRIMARY KEY,
   Parking_lot char(25)
);
```

```
CREATE TABLE IF NOT EXISTS Smoking(
Smoking_id int PRIMARY KEY,
Smoking_area char(25)
);
```

```
CREATE TABLE IF NOT EXISTS Alcohol(
Alcohol_id int PRIMARY KEY,
Alcohol_serve char(25)
);
```

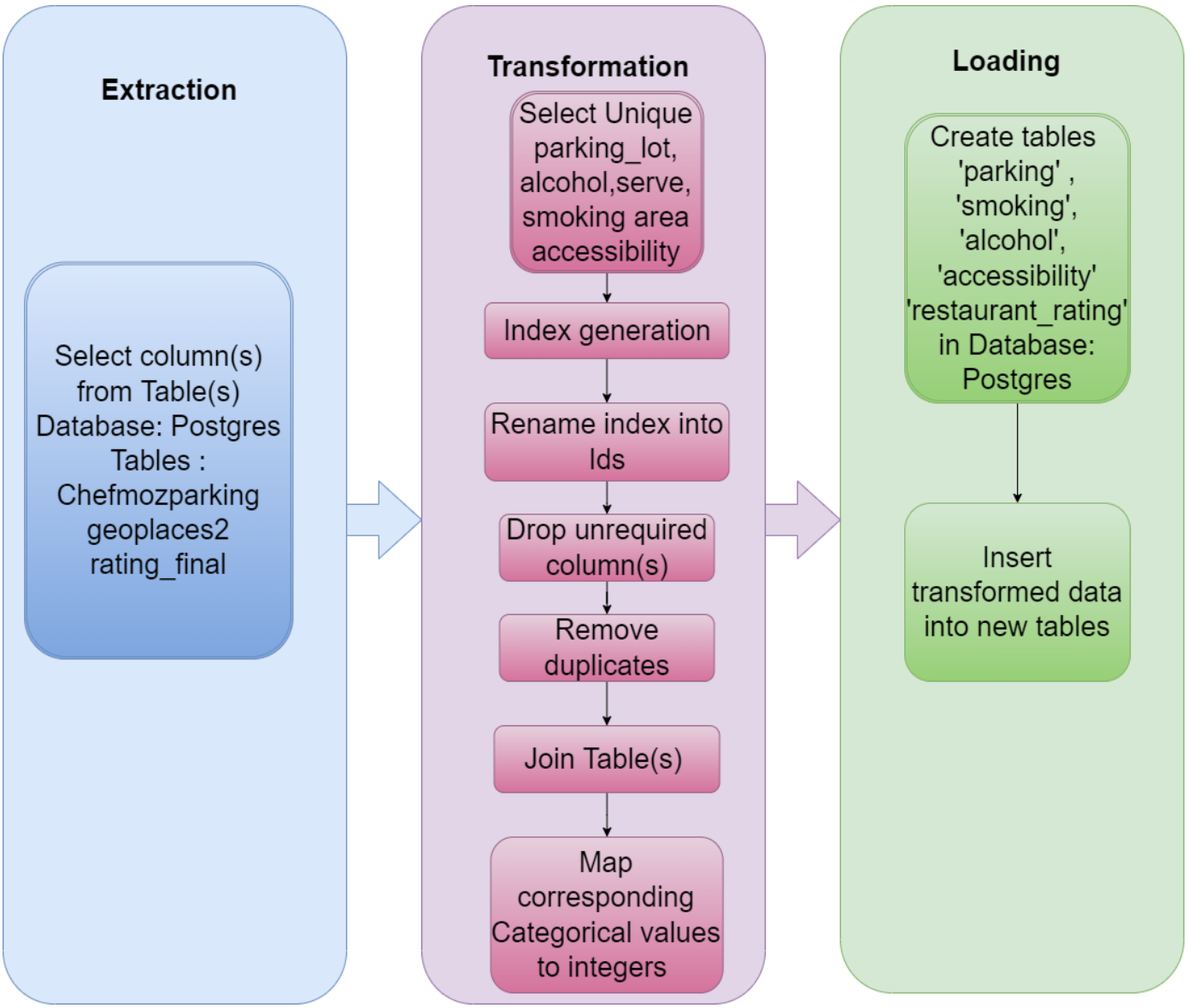
```
CREATE TABLE IF NOT EXISTS Restaurant_Rating(
place_id int,
Parking_id int,
Smoking_id int,
Alcohol_id int,
accessibility_id int,
Rating int,
Food_rating int,
Service_rating int,
overall_rating int,
PRIMARY KEY (place_id),
FOREIGN KEY(accessibility_id)
REFERENCES accessibility(accessibility_id)
);

CREATE TABLE IF NOT EXISTS Accessibility(
Accessibility_id int PRIMARY KEY,
accessibility char(25)
);
```

```
ALTER TABLE "restaurant_rating"
ADD FOREIGN KEY ("parking_id") REFERENCES "parking" ("parking_id")
```

```
ALTER TABLE "restaurant_rating"
ADD FOREIGN KEY ("smoking_id") REFERENCES "smoking" ("smoking_id")

ALTER TABLE "restaurant_rating"
ADD FOREIGN KEY ("alcohol_id") REFERENCES "alcohol" ("alcohol_id")
```

3.2.4 ETL Process

Propose and describe a simple ETL process to populate a data cube suitable for the proposed project, including the allocated placeholder for the analytical/inference results. [10/90]



ETL, which stands for extract, transform and load, is a data integration process that combines data from multiple data sources into a single, consistent data store that is loaded into a data warehouse or other target system.

For the Extraction phase of our ETL Process, since we already had data in staging area in 9 tables, our problem statement required use of only 3 tables such as Chefmozparking, geoplaces2 and final rating. Selection of relevant columns such as parking, alcohol, smoking area, rating, food_rating, service_rating, accessibility and place_id was done from the above 3 tables. Since we implemented our scripts in python, all the selected columns were stored in respective dataframes.

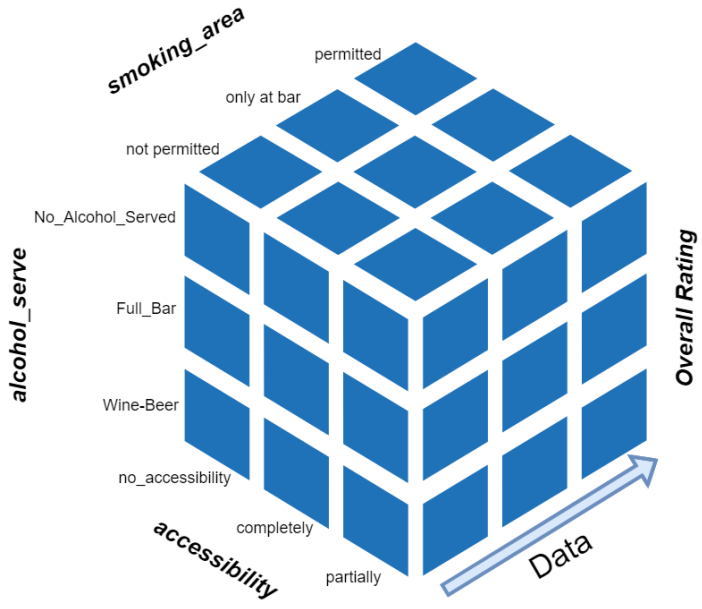
In the Transformation phase, we used the unique function to find categorical data in columns such as parking_lot, alcohol,smoking_area and accessibility. We generated indexes for the unique categories and renamed them into corresponding ids. Further, we dropped irrelevant columns and removed duplicates.

We found out that there were 938 restaurants in total and only 130 restaurants had rating in them. Since we did not want to consider the restaurants which did not have any rating, we removed those restaurants.

Then, to fetch all the matching columns on the basis of place ids, we did a left outer join. After that, mapping was done for respective categorical unique values with integers. These integers were assumed to convert the categorical values to numerical form in order to encode. For instance, 'Wine-Beer':2, 'No_Alcohol_Served':1, 'Full_Bar':0.

For the Loading process, we created 5 target tables where 3 of them were dimension tables called ‘parking’, ‘smoking’ ,‘alcohol’ , ‘accessibility’ and one fact table called ‘restaurant_rating’ and data was inserted into these tables.

Thus, we implemented a data cube consisting of smoking_area, accessibility, alcohol serve and overall rating.



3.2.5 ETL Scripts

Provide the SQL scripts for the proposed ETL process. [10/90]

```
from sqlalchemy import create_engine

from numpy import unique
```

```
import pandas as pd
dma_engine = create_engine('postgresql://druid:FoolishPassword@localhost:5432/postgres')
# create pandas table from query and connection
def create_pandas_table(sql_query, conn_engine=dma_engine):
    table = pd.read_sql_query(sql_query, con=conn_engine)
    return table
alco_df = create_pandas_table('select alcohol from geoplaces2', dma_engine)
smok_df = create_pandas_table('select smoking_area from geoplaces2', dma_engine)
geo_df = create_pandas_table('select * from geoplaces2', dma_engine)
park_df = create_pandas_table('select * from chefmozparking', dma_engine)
rating = create_pandas_table('select * from rating_final', dma_engine)
accessibility_df = create_pandas_table('select accessibility from geoplaces2', dma_engine)
```

Transform Script :

Parking Transformation:

```
df= pd.DataFrame({'parking_lot':unique(park_df.parking_lot)})
df.reset_index(level=0, inplace=True)
parking = df.rename({'index': 'parking_id'}, axis=1)
```

	parking_id	parking_lot
0	0	fee
1	1	none
2	2	public
3	3	street
4	4	valet parking
5	5	validated parking
6	6	yes

Alcohol Transformation:

```
alco_df = pd.DataFrame({'alcohol_serve':unique(alco_df.alcohol)})
alco_df.reset_index(level=0, inplace=True)
alcohol = alco_df.rename({'index': 'alcohol_id'}, axis=1)
```

	alcohol_id	alcohol_serve
0	0	Full_Bar
1	1	No_Alcohol_Served
2	2	Wine-Beer

Smoking Transformation:

```
smok_df = pd.DataFrame({'smoking_area':unique(smok_df.smoking_area)})
smok_df.reset_index(level=0, inplace=True)
smoking = smok_df.rename({'index': 'smoking_id'}, axis=1)
```

	smoking_id	smoking_area
0	0	none
1	1	not permitted
2	2	only at bar
3	3	permitted
4	4	section

Accessibility Transformation:

```
accessibility_df = pd.DataFrame({'accessibility':unique(accessibility_df.accessibility)})
accessibility_df.reset_index(level=0, inplace=True)
accessibility = accessibility_df.rename({'index': 'accessibility_id'}, axis=1)
```

```
accessibility
```

	accessibility_id	accessibility
0	0	completely
1	1	no_accessibility
2	2	partially

Restaurant_rating Transformation:

```
geo_info = geo_df[['place_id','alcohol','smoking_area','accessibility']]
rating.drop(['user_id'], axis=1,inplace=True)
rating_clean= rating.drop_duplicates(subset=['place_id'])
df_res = pd.merge(left=geo_info, right=park_df, how="left", on="place_id")
df_res = pd.merge(left=df_res, right=rating_clean, how="left", on="place_id")
```

```
df_res
```

	place_id	alcohol	smoking_area	accessibility	parking_lot	rating	food_rating	service_rating
0	134999	No_Alcohol_Served	none	no_accessibility	none	2	2	2
1	132825	No_Alcohol_Served	none	completely	none	2	2	2
2	135106	Wine-Beer	only at bar	partially	none	2	2	2
3	132667	No_Alcohol_Served	none	completely	none	1	2	2
4	132613	No_Alcohol_Served	permitted	completely	yes	2	2	2
...	...	...	...	...	...	...	...	...
125	132866	No_Alcohol_Served	not permitted	completely	yes	2	2	1
126	135072	No_Alcohol_Served	none	no_accessibility	none	1	2	2
127	135109	Wine-Beer	not permitted	no_accessibility	none	0	0	0
128	135019	No_Alcohol_Served	none	completely	none	2	2	2
129	132877	No_Alcohol_Served	none	completely	none	0	0	0

```
# 'Wine-Beer':2, 'No_Alcohol_Served':1, 'Full_Bar':0
df_res.alcohol = df_res.alcohol.map({'Wine-Beer':2, 'No_Alcohol_Served':1, 'Full_Bar':0})
# 'none':0, 'not permitted':1, 'only at bar':2, 'permitted':3, 'section':4
df_res.smoking_area = df_res.smoking_area.map({'none':0, 'not permitted':1, 'only at bar':2,
'permitted':3, 'section':4})

# 'fee':0, 'none':1, 'public':2, 'yes':6,'street':3, 'valet parking':4, 'validated parking':5
df_res.parking_lot = df_res.parking_lot.map({'fee':0, 'none':1, 'public':2, 'yes':6,
                                             'street':3, 'valet parking':4, 'validated parking':5})
#accessibility : 'no_accessibility' :0, 'completely':1, 'partially' :2
df_res.accessibility = df_res.accessibility.map({'no_accessibility' :0, 'completely':1, 'partially' :2})
df_res = df_res.rename(columns={'alcohol': 'alcohol_id', 'smoking_area': 'smoking_id',
'parking_lot': 'parking_id', 'accessibility': 'accessibility_id'})
```

```
df_res
```

	place_id	alcohol_id	smoking_id	accessibility_id	parking_id	rating	food_rating	service_rating
0	134999	1	0	0	1	2	2	2
1	132825	1	0	1	1	2	2	2
2	135106	2	2	2	1	2	2	2
3	132667	1	0	1	1	1	2	2
4	132613	1	3	1	6	2	2	2
...	...	...	...	...	...	...	...	...
125	132866	1	1	1	6	2	2	1
126	135072	1	0	0	1	1	2	2
127	135109	2	1	0	1	0	0	0
128	135019	1	0	1	1	2	2	2
129	132877	1	0	1	1	0	0	0

Load Script :

Definition to load data into database

Loading Dataframe to Database

```
def insert_dataframe_to_database(table_name, dataframe, engine, index_id=None, override='replace'):
    if index_id is not None:
        dataframe.to_sql(table_name, con=engine, if_exists=override, index_label=index_id)
        print('Data inserted successfully into '+table_name)
    else:
        dataframe.to_sql(table_name, con=engine, if_exists=override, index=False)
        print('Data inserted successfully into ' + table_name)
```

Loading parking dimension into database

```
insert_dataframe_to_database('parking',parking,dma_engine,'append')
```

```
select* from parking
```

parking_id	parking_lot
0	fee
1	none
2	public
3	street
4	valet parking
5	validated parking
6	yes

Loading smoking dimension into database

insert_dataframe_to_database('smoking',smoking,dma_engine,'append')

```
select * from smoking
```

smoking_id	smoking_area
0	none
1	not permitted
2	only at bar
3	permitted
4	section

Loading alcohol dimension into database

insert_dataframe_to_database('alcohol',alcohol,dma_engine,'append')

```
select * from alcohol
```

alcohol_id	alcohol_serve
0	Full_Bar
1	No_Alcohol_Served
2	Wine-Beer

Loading accessibility dimension into database

insert_dataframe_to_database('accessibility', accessibility,dma_engine,'append')

```
select * from accessibility
```

accessibility_id	accessibility
0	completely
1	no_accessibility
2	partially

Loading restaurant_rating fact into database

insert_dataframe_to_database('restaurant_rating',df_res,dma_engine,'append')

```
select * from restaurant_rating
```

append	place_id	alcohol_id	smoking_id	accessibility_id	parking_id	rating	food_rating	service_rating
0	134999	1	0	0	1	2	2	2
1	132825	1	0	1	1	2	2	2
2	135106	2	2	2	1	2	2	2
3	132667	1	0	1	1	1	2	2
4	132613	1	3	1	6	2	2	2
5	135040	2	0	0	6	0	0	0

3.3 Analytical/Predictive Model

3.3.1 Description

Propose and discuss an analytical/predictive model to address the proposed problem statement. Discuss the type of the model (e.g. supervised, unsupervised, classification, regression, etc.) and the reasoning behind this approach. [5/90]

As discussed above, the problem statement is to predict the overall restaurant rating based on four features such as smoking area availability, alcohol served or not, parking availability and accessibility.

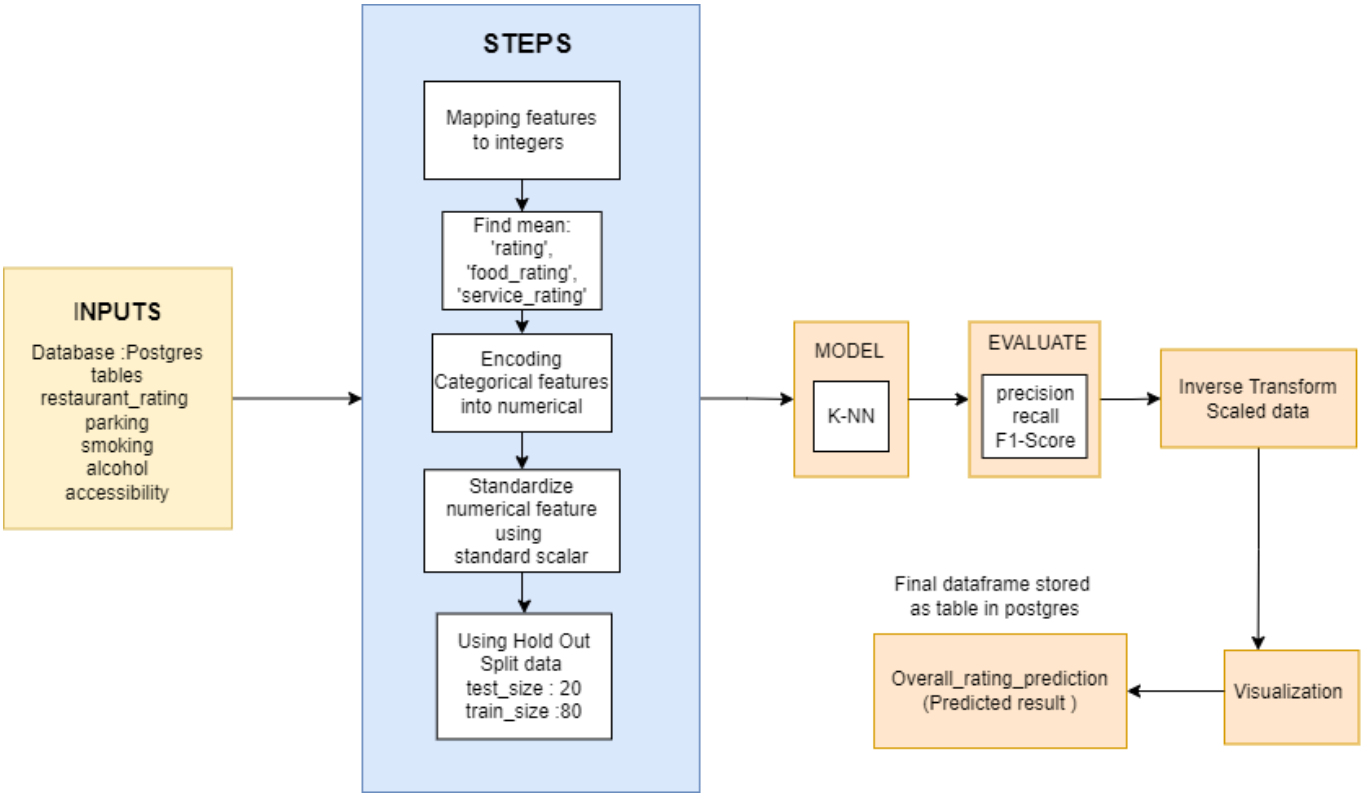
We have approached the problem statement using Classification in Supervised learning where we have considered supervised learning as we have dependent and independent features in our data. The dependent feature is overall_rating and the independent features are smoking area availability, alcohol served or not, parking availability and accessibility. Since the dependent feature, ie overall rating has three classes such as 0,1, 2, this leads to a multi-label classification problem.

The model which we have used is K-NN (K- Nearest Neighbor classifier) that predicts the overall rating for restaurants based on selected features. (smoking area availability, alcohol served or not, parking availability, accessibility)

We compared other models such as Logistic Regression, Decision Tree and Random Forest but K-NN gave us the best result.

3.3.2 Model Diagram

Provided a diagram visualizing inputs, steps, and outputs of the proposed model. [5/90]



3.3.3 Model Scripts

Provided the scripts for implementation of the proposed model (Must include scripts for querying the data, model, inserting results into the appropriate table(s)). [10/90]

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import r2_score
from sqlalchemy import create_engine
#Connecting to postgres to fetch columns required for prediction
dma_engine = create_engine('postgresql://druid:FoolishPassword@localhost:5432/postgres')

# Create pandas table from query and connection
import pandas as pd
def create_pandas_table(sql_query, conn_engine=dma_engine):
 table = pd.read_sql_query(sql_query, con=conn_engine)
```

return table

#Fetching columns by querying the database

```
data = create_pandas_table ('select r.place_id, a.alcohol_serve, p.parking_lot,
s.smoking_area,ac.accessibility, r.rating, r.food_rating, r.service_rating
from restaurant_rating as r
join parking as p on p.parking_id = r.parking_id
join smoking as s on s.smoking_id = r.smoking_id
join alcohol as a on a.alcohol_id =r.alcohol_id
join accessibility as ac
on ac.accessibility_id = r.accessibility_id', dma_engine)
```

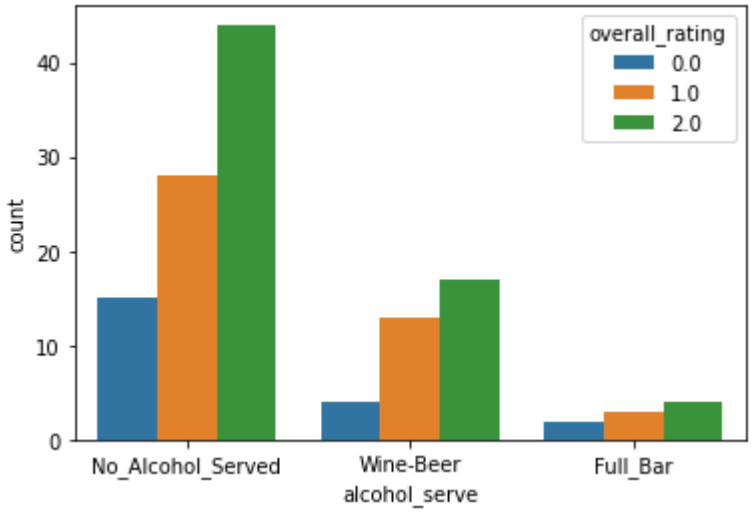
```
data
```

	place_id	alcohol_serve	parking_lot	smoking_area	accessibility	rating	food_rating	service_rating
0	134999	No_Alcohol_Served	none	none	completely	2	2	2
1	132825	No_Alcohol_Served	none	none	no_accessibility	2	2	2
2	135106	Wine-Beer	none	only at bar	partially	2	2	2
3	132667	No_Alcohol_Served	none	none	no_accessibility	1	2	2
4	132613	No_Alcohol_Served	yes	permitted	no_accessibility	2	2	2
...	...	...	...	...	...	...	...	...
125	132866	No_Alcohol_Served	yes	not permitted	no_accessibility	2	2	1
126	135072	No_Alcohol_Served	none	none	completely	1	2	2
127	135109	Wine-Beer	none	not permitted	completely	0	0	0
128	135019	No_Alcohol_Served	none	none	no_accessibility	2	2	2
129	132877	No_Alcohol_Served	none	none	no_accessibility	0	0	0

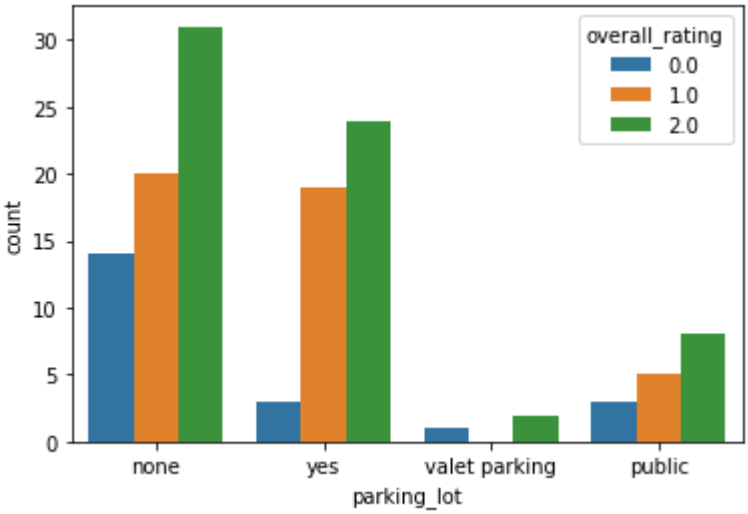
#Finding mean

```
data['overall_rating'] = data[['rating', 'food_rating','service_rating']].mean(axis=1).round()
```

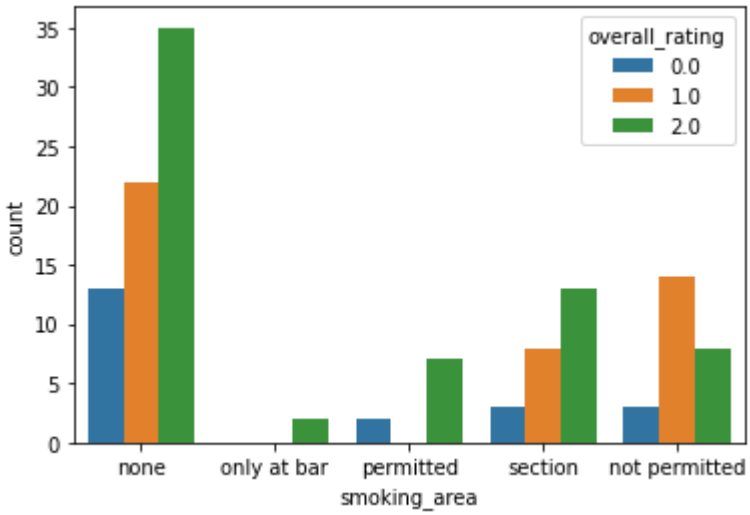
```
sns.countplot(x='alcohol_serve', hue='overall_rating', data=data)
```



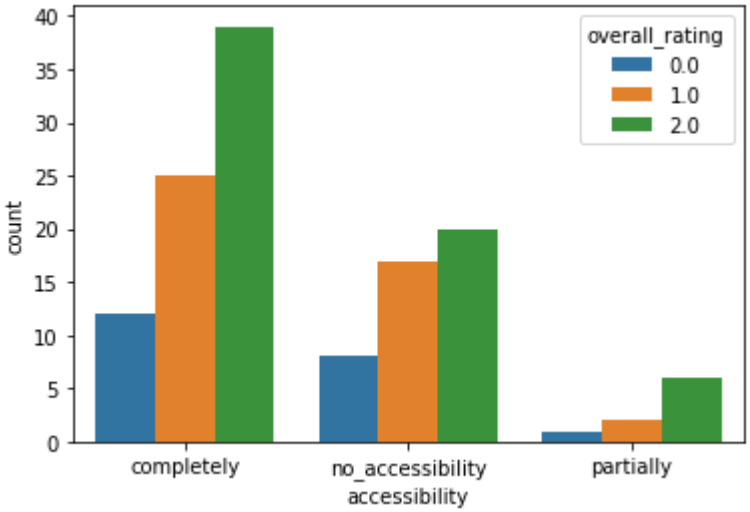
sns.countplot(x='parking_lot', hue='overall_rating', data=data)



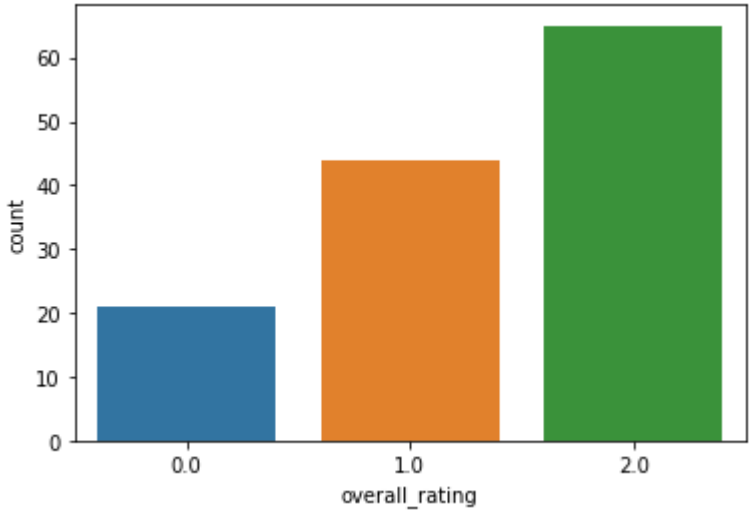
sns.countplot(x='smoking_area', hue='overall_rating', data=data)



sns.countplot(x='accessibility', hue='overall_rating', data=data)



sns.countplot(data['overall_rating'])



 # Mapping features:1 if alcohol is available, 0 otherwise

data.alcohol_serve = data.alcohol_serve.map(lambda x: 0 if x == 'No_Alcohol_Served' else 1)

data.parking_lot = data.parking_lot.map({'fee':1, 'none':0, 'public':1, 'yes':2,
'street':1, 'valet parking':1, 'validated parking':1})

1 if there is smoking area, 0 otherwise

data.smoking_area = data.smoking_area.map(lambda x: 0 if (x == 'none') | (x == 'not permitted') else 1)

#accessibility : 'no_accessibility' :0, 'completely':1, 'partially' :2

data.accessibility = data.accessibility.map({'no_accessibility' :0, 'completely':1, 'partially' :2})

```
data
```

	place_id	alcohol_serve	parking_lot	smoking_area	accessibility	rating	food_rating	service_rating
0	134999	0	0	0	1	2	2	2
1	132825	0	0	0	0	2	2	2
2	135106	1	0	1	2	2	2	2
3	132667	0	0	0	0	1	2	2
4	132613	0	2	1	0	2	2	2
...	...	...	...	...	...	...	...	...
125	132866	0	2	0	0	2	2	1
126	135072	0	0	0	1	1	2	2
127	135109	1	0	0	1	0	0	0
128	135019	0	0	0	0	2	2	2
129	132877	0	0	0	0	0	0	0

```
data.drop(['rating', 'food_rating','service_rating'], axis=1, inplace=True)
X=data.iloc[:,1:4].values
Y=data.iloc[:,-1].values
Y = Y.reshape(-1,1)
from imblearn.over_sampling import SMOTE
sm = SMOTE()
X_train_sm, Y_train_sm = sm.fit_resample(X, Y)
X_train,X_test,Y_train,Y_test=train_test_split(X_train_sm,Y_train_sm,test_size=0.20)
#Scaling and Transforming
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)from sklearn.model_selection import cross_val_score

from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
def models(X_train,Y_train):

 # K – Nearest Neighbor Classifier
```

```
print('K – Nearest Neighbor Classifier(KNN)')
print()
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier(n_neighbors = 1, metric = 'minkowski')
knn.fit(X_train, Y_train)
y_pred_knn=knn.predict(X_test)
cm=confusion_matrix(Y_test,y_pred_knn)


ax = sns.heatmap(cm, annot = True)
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
print(cm)
print(classification_report(Y_test,y_pred_knn))
print("Accuracy_Score: ",accuracy_score(Y_test,y_pred_knn))
print()
print('All K- fold cross val',cross_val_score(knn, X_train, Y_train, cv=5))
print('Mean of all K-fold cross Val',np.mean(cross_val_score(knn, X_train, Y_train, cv=5)))
final_df = pd.DataFrame(Y_test.reshape(-1, 1), columns=['Actual'])
final_df['Predictions'] = y_pred_knn.reshape(-1, 1)
print(final_df.head(100))
plt.plot(final_df)
plt.legend(['Actual', 'Predictions'])
plt.show()
plt.close()

sns.pairplot(data)
sns.countplot(data = final_df, x = 'Predictions', hue = 'Actual')
plt.show()
#Loading to final placeholder table ‘overall rating prediction’

from sqlalchemy import create_engine
dma_engine = create_engine('postgresql://druid:FoolishPassword@localhost:5432/postgres')
```

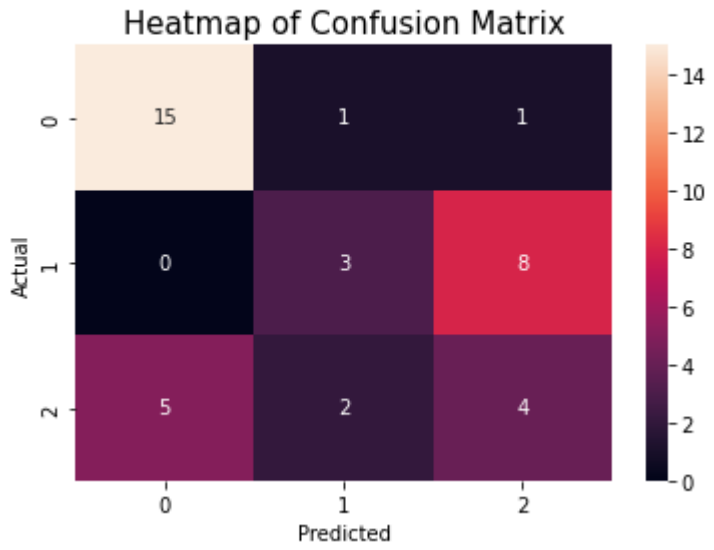
final_df.to_sql('overall_rating_prediction', dma_engine, if_exists='replace')

3.3.4 Outputs

Visualize analytical and inference results from the proposed model (at least three). [10/90]

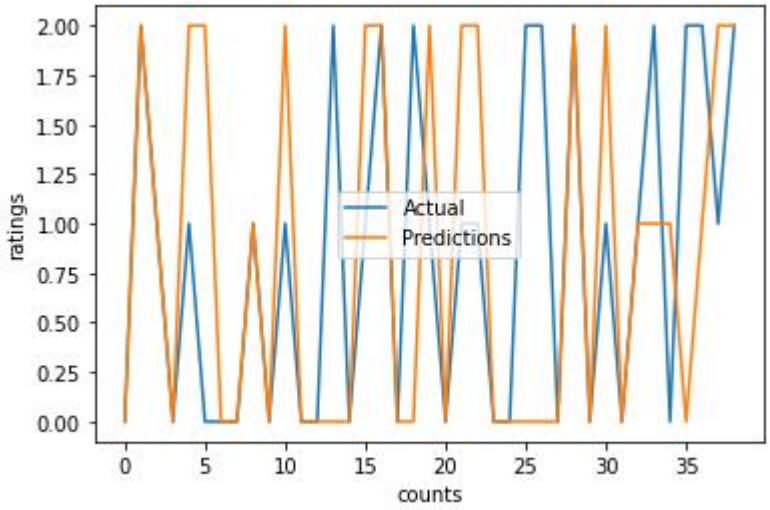
1. KNN

```
K - Nearest Neighbor Classifier(KNN)
```

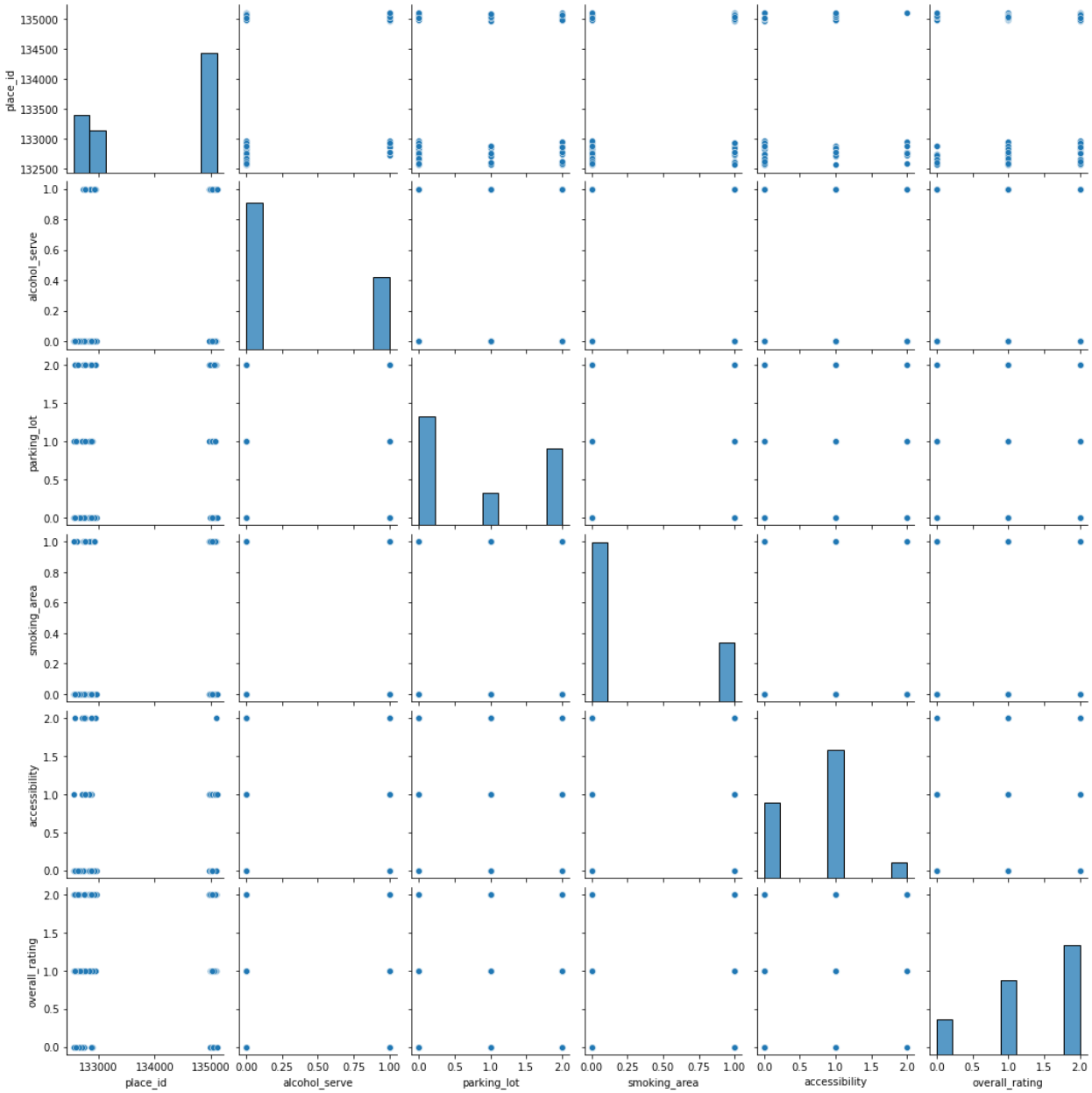


Accuracy_Score: 0.5641025641025641

2. K-NN Actual and Predicted Values



3. Pair plot



4. Count Plot for Prediction Count