Segmentation Analysis-Restaurants Data

September 7, 2019

[1]:
```
import numpy as np
import pandas as pd
import json
import csv
import os
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import nltk
from nltk.corpus import stopwords
import re
from collections import Counter
from sklearn.feature_extraction.text import CountVectorizer
from wordcloud import WordCloud
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
#can't install new tensorflow cuz of macos so use an older version
# import tensorflow as tf
# from tensorflow import keras
import geopandas
import descartes
import requests
import math
import urllib
from shapely.geometry import Point
```

Objectives:

```
1. To identify profitable customers
2. To study spending habits and predict future patterns
3. To develop better products and customize service
```

Main Questions: 1. What are they spending on? 2. Who are they? 3. What is the location like? 4. How much do they spend? 5. Why do they do the things they do?

Further Questions/Hypotheses: 1. Do smokers like to go to restaurants that have a smoking area? 2. Do people who like to drink socially and casually like to go to restaurants that have a bar? 3. Do customers with high budget like to go to restaurants that have a fancy dress code? 4. Users with kids are car owners 5. Married users are on a tighter budget. 6. Single users tend to be social drinkers. 7. Students have low budgets 8. Younger users like to go places with friends 9. Users with cars can drive to restaurants farther than their home location 10. People who drink will go to restaurants that cost more because alcohol is expensive.

User Variables:

The user dataset is essential and the most important part to the segmentation analysis.

```
userID: Nominal
latitude: Numeric
longitude: Numeric
smoker: [false,true]
drink_level: [abstemious,social drinker,casual drinker]
dress_preference:[informal,formal,no preference,elegant]
ambience: [family,friends,solitary]
transport: [on foot,public,car owner]
marital_status: [single,married,widow]
hijos:[independent,kids,dependent]
birth_year
interest: [variety,technology,none,retro,eco-friendly]
personality: [thrifty-protector,hunter-ostentatious,hard-worker,conformist]
religion:[none,Catholic,Christian,Mormon,Jewish]
activity: [student,professional,unemployed,working-class]
color: [black,red,blue,green,purple,orange,yellow,white]
weight
budget: [medium,low,high]
height
```

Restaurant data:

We are going to filter out only the variables we need for the segmentation analysis

```
placeID: Nominal
latitude: Numeric
longitude: Numeric
alcohol: [No_Alcohol_Served,Wine_Beer,Full_Bar]
smoking_area: [none,only_at_bar,permitted,section,not_permitted]
dress_code: [informal,casual,formal]
accessibility: [no_accessibility,completely,partially]
price: [medium,low,high]
ambience: [familiar,quiet]
franchise:  [t,f]
area: [open,closed]
other_services: [none,internet,variety]
cuisine (added from cuisine dataset)
```

Rating data:

```
userID: Nominal
placeID: Nominal
```

```
rating: [0,1,2]
food_rating: [0,1,2]
service_rating: [0,1,2]
```

Two approaches were tested: a collaborative filter technique and a contextual approach. (i) The collaborative filter technique used only one file i.e., rating_final.csv that comprises the user, item and rating attributes. (ii) The contextual approach generated the recommendations using the remaining eight data files.

Table of Contents:

1. Section **??**
2. Section **??**
3. Section **??**
4. Section **??**

[2]:
```
# reading data
userprofile = pd.read_csv('restaurant-data-with-consumer-ratings/userprofile.
 ↪csv')
restaurantraw = pd.read_csv('restaurant-data-with-consumer-ratings/geoplaces2.
 ↪csv')
cuisine= pd.read_csv('restaurant-data-with-consumer-ratings/chefmozcuisine.csv')
rating =pd.read_csv('restaurant-data-with-consumer-ratings/rating_final.csv')
```

1 Cleaning Data

[3]:
```
#cleaning data
userprofile = userprofile.replace('?', None)
restaurantraw.rename(columns = {'Rambience': 'ambience'}, inplace = True)
#filter for the variables we want in restaurant
restaurant = restaurantraw[['placeID', 'latitude', 'longitude', 'alcohol',␣
 ↪'smoking_area','dress_code',
          ␣
 ↪'accessibility','price','ambience','franchise','area','other_services']]
restaurant['Cuisine'] = restaurant['placeID'].map(dict(zip(cuisine['placeID'],␣
 ↪cuisine['Rcuisine'])))
restaurant['Cuisine'] =restaurant['Cuisine'].replace(np.nan,None)
```

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:7:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  import sys
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
```

```
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

[4]:
```
#new features
businessmedianrating = rating.groupby('placeID').median().reset_index()
businessmedianrating.columns = [x for x in businessmedianrating if x in␣
 ↪restaurant.columns]+ ['median_' + x for x in businessmedianrating if x not in␣
 ↪restaurant.columns]
restaurant = pd.merge(restaurant, businessmedianrating, on ='placeID', how =␣
 ↪'left')
#create age for user
userprofile['age'] = 2012-userprofile['birth_year']
#create age grouop for user
def agegroup(x):
    if x < 18:
        return 'child'
    if x>= 18 and x <30:
        return 'young adult'
    if x>=30 and x<60:
        return 'adult'
    if x>60:
        return 'senior'
userprofile['age group'] = userprofile['age'].map(agegroup)
```

[5]:
```
fulldata = pd.merge(rating, restaurant, on = 'placeID', how = 'inner').
 ↪merge(userprofile, on = 'userID', how = 'inner')
```

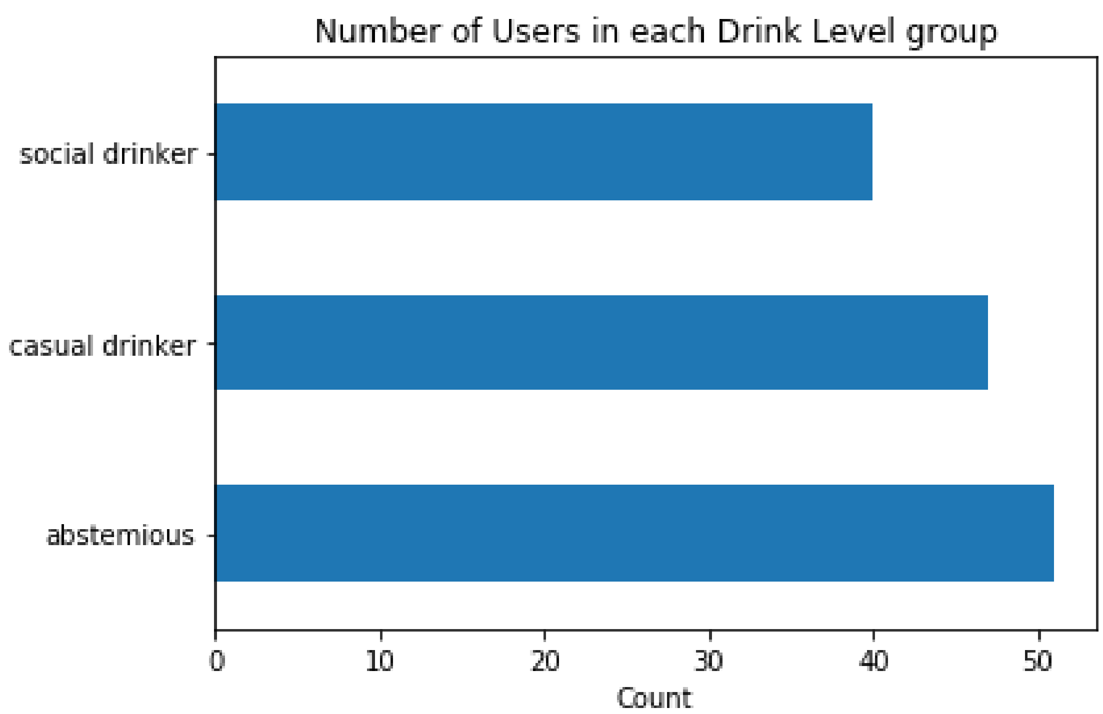
[6]:
```
fulldata.shape
```

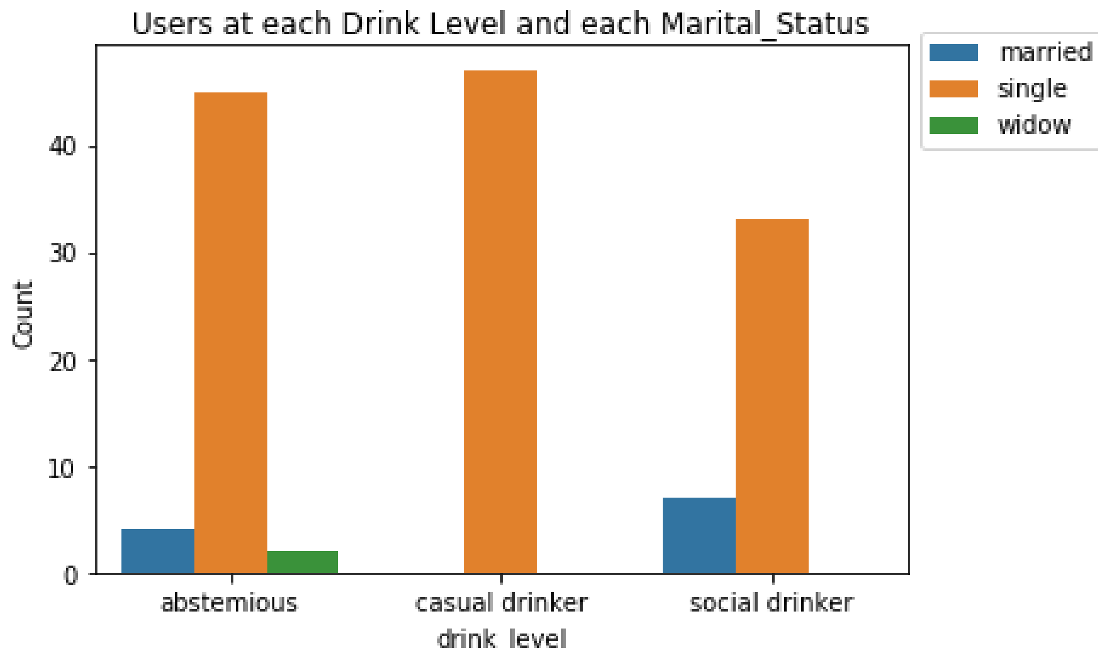
[6]:
```
(1161, 40)
```

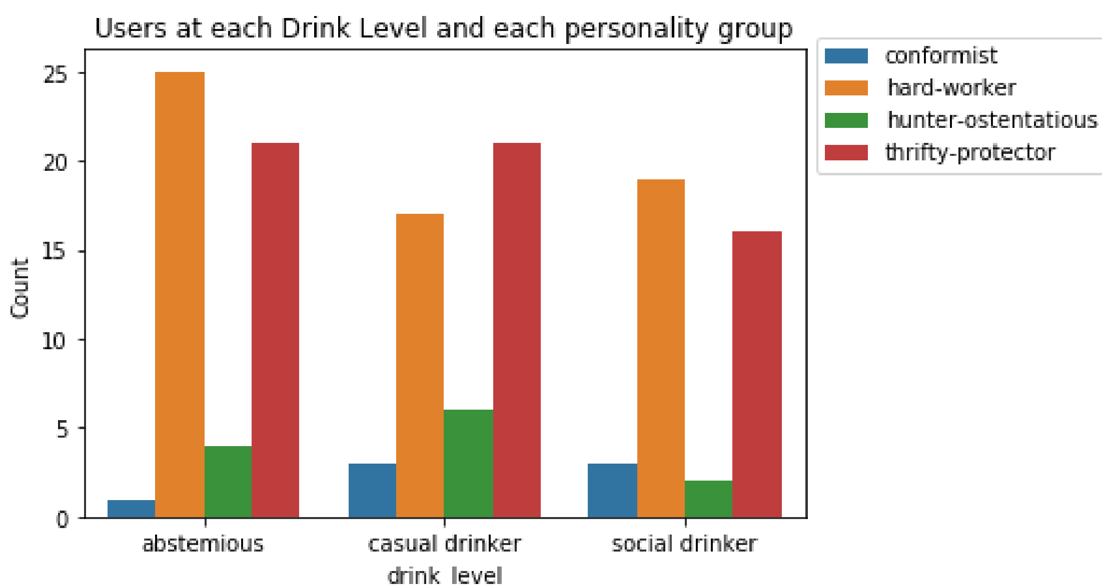
2 Who are the Users?

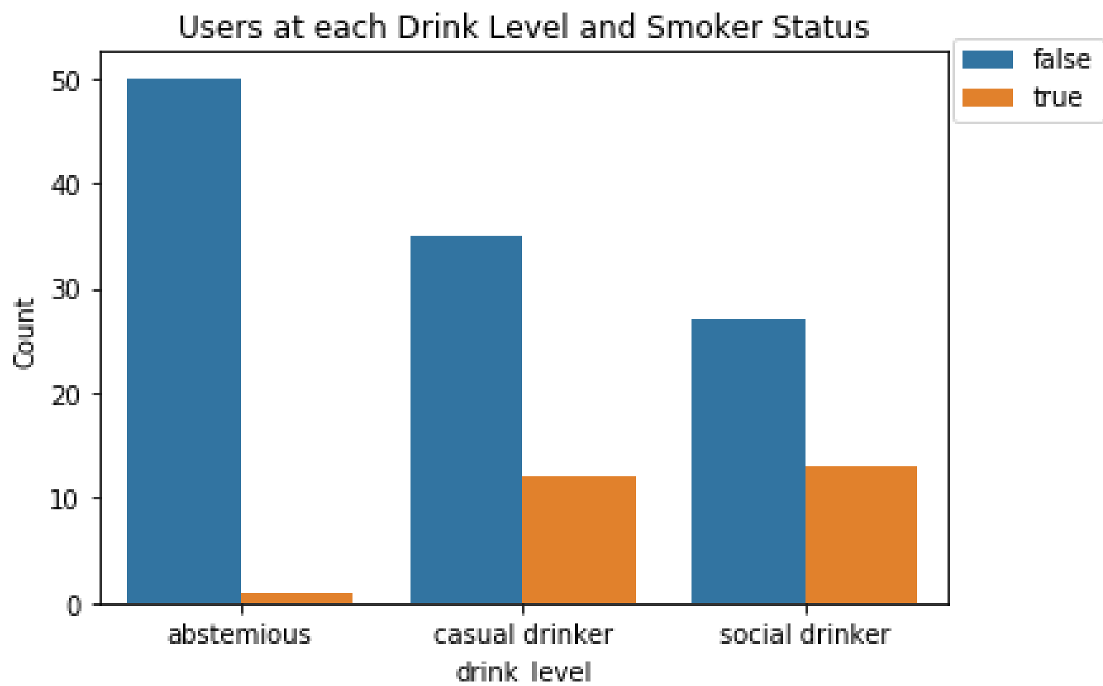
[7]:
```
userprofile['drink_level'].value_counts().plot(kind = 'barh')
plt.xlabel("Count")
plt.title('Number of Users in each Drink Level group')
plt.show();

sns.barplot(x = 'drink_level', y ='userID', hue = 'marital_status', data=␣
 ↪userprofile.groupby(['drink_level', 'marital_status']).count().reset_index())
plt.title("Users at each Drink Level and each Marital_Status")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1, 1.05))
```

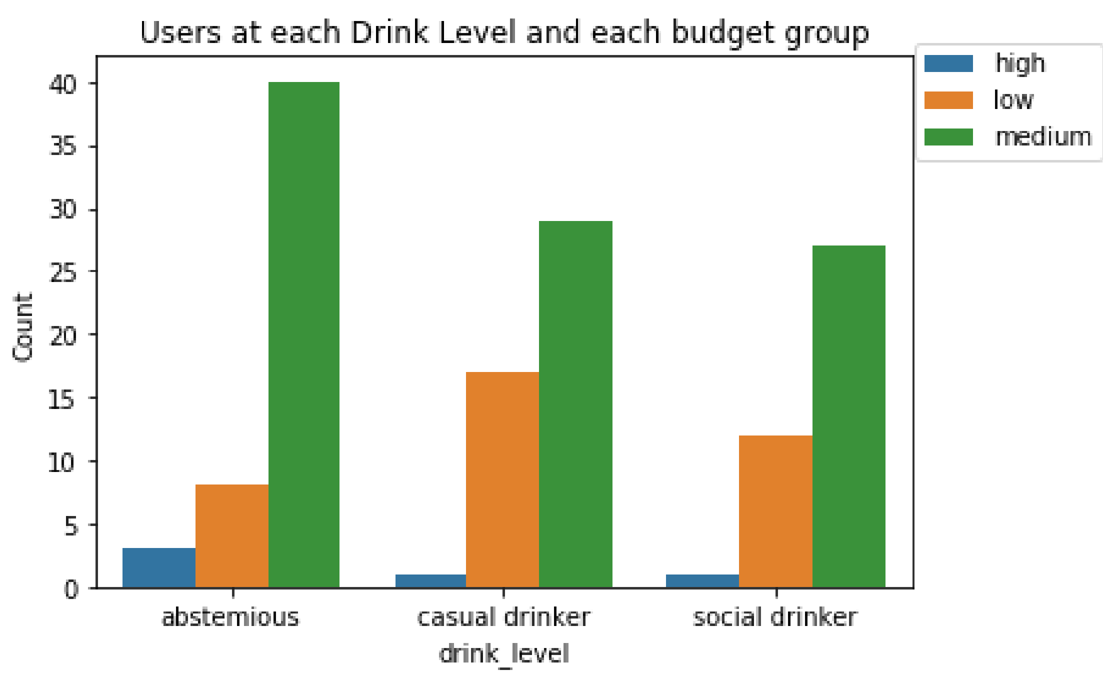
```
plt.show();

sns.barplot(x = 'drink_level', y ='userID', hue = 'personality', data=␣
↪userprofile.groupby(['drink_level', 'personality']).count().reset_index())
plt.title("Users at each Drink Level and each personality group")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1, 1.05))
plt.show();

sns.barplot(x = 'drink_level', y ='userID', hue = 'smoker', data= userprofile.
↪groupby(['drink_level', 'smoker']).count().reset_index())
plt.title("Users at each Drink Level and Smoker Status")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.20, 1.05))
plt.show();

sns.barplot(x = 'drink_level', y ='userID', hue = 'budget', data= userprofile.
↪groupby(['drink_level', 'budget']).count().reset_index())
plt.title("Users at each Drink Level and each budget group")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.25, 1.05))
plt.show();
```




Users at each Drink Level and each Marital_Status
married
single
widow
Count
0
10
20
30
40
abstemious
casual drinker
social drinker
drink_level

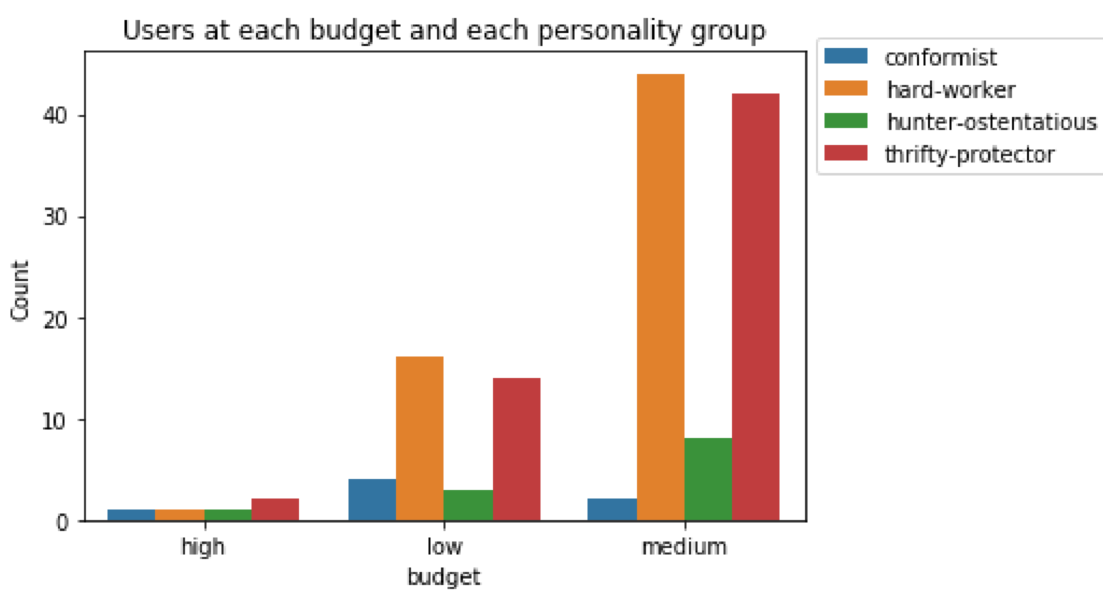

Users at each Drink Level and each personality group
conformist
hard-worker
hunter-ostentatious
thrifty-protector
Count
0
5
10
15
20
25
abstemious
casual drinker
social drinker
drink_level

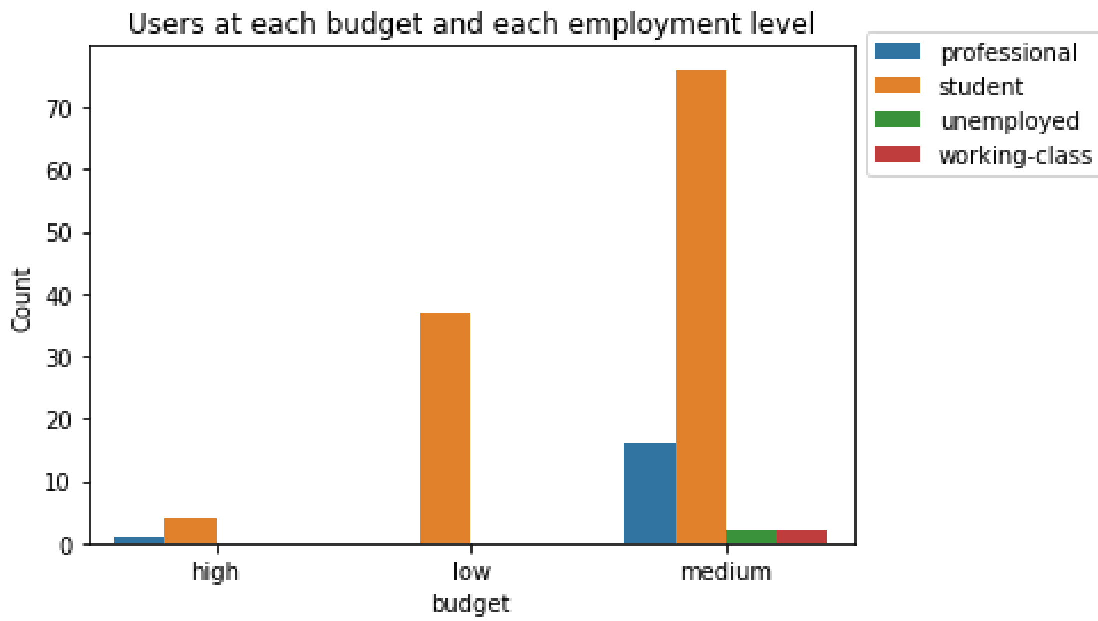


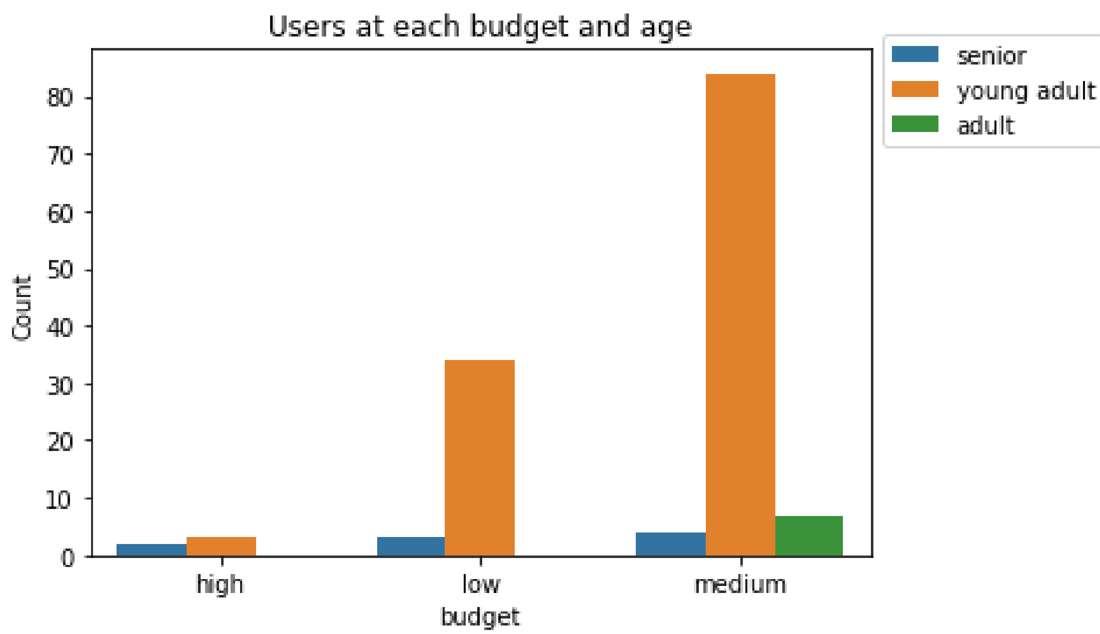


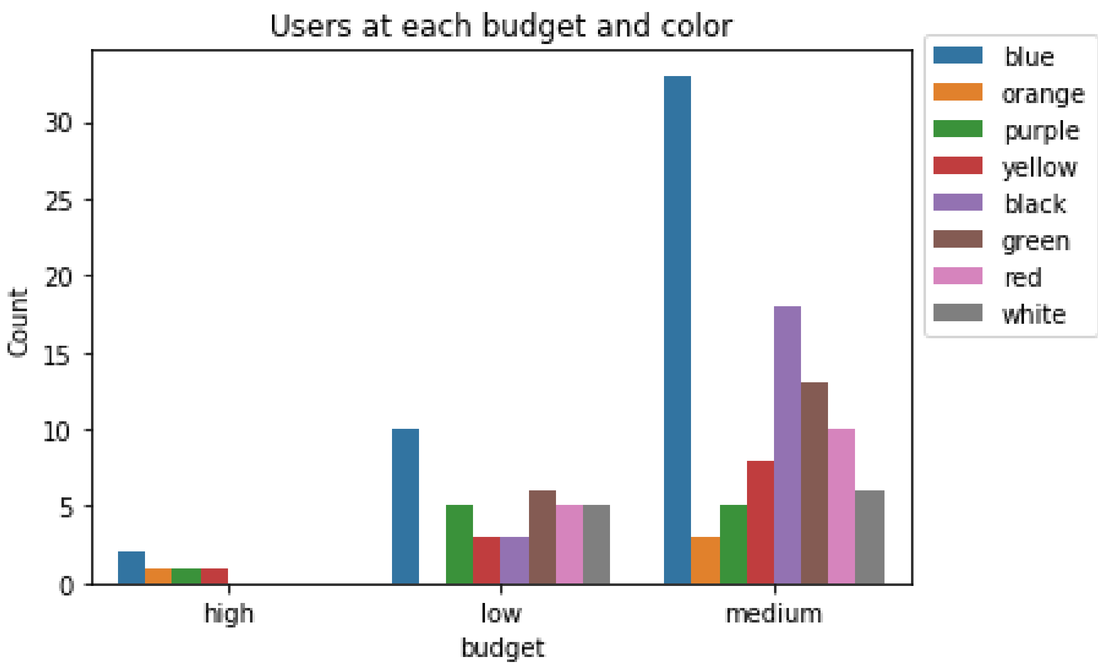
[8]:
```
sns.barplot(x = 'budget', y ='userID', hue = 'personality', data= userprofile.
    ↪groupby(['budget', 'personality']).count().reset_index())
plt.title("Users at each budget and each personality group")
```

```
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1, 1.05))
plt.show();

sns.barplot(x = 'budget', y ='userID', hue = 'activity', data= userprofile.
 ↪groupby(['budget', 'activity']).count().reset_index())
plt.title("Users at each budget and each employment level")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1, 1.05))
plt.show();

sns.barplot(x = 'budget', y ='userID', hue = 'age group', data= userprofile.
 ↪groupby(['budget', 'age group']).count().reset_index())
plt.title("Users at each budget and age")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1, 1.05))
plt.show();

sns.barplot(x = 'budget', y ='userID', hue = 'color', data= userprofile.
 ↪groupby(['budget', 'color']).count().reset_index())
plt.title("Users at each budget and color")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1, 1.05))
plt.show();
```




Users at each budget and each employment level
professional
student
unemployed
working-class
Count
0
10
20
30
40
50
60
70
high
low
medium
budget

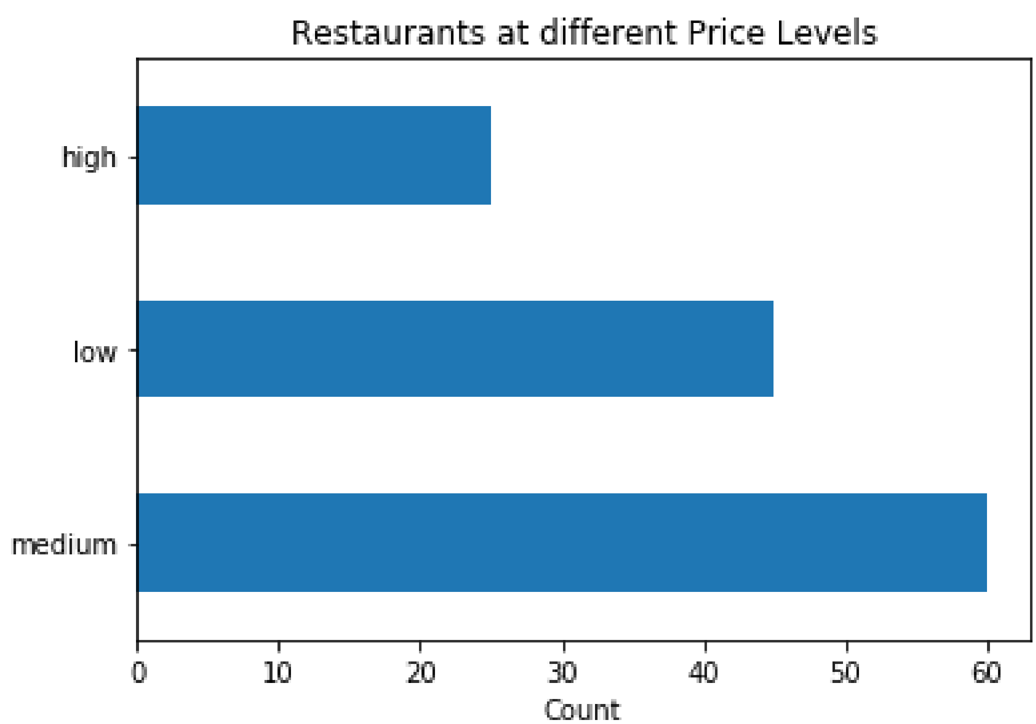

Users at each budget and age
senior
young adult
adult
Count
0
10
20
30
40
50
60
70
80
high
low
medium
budget

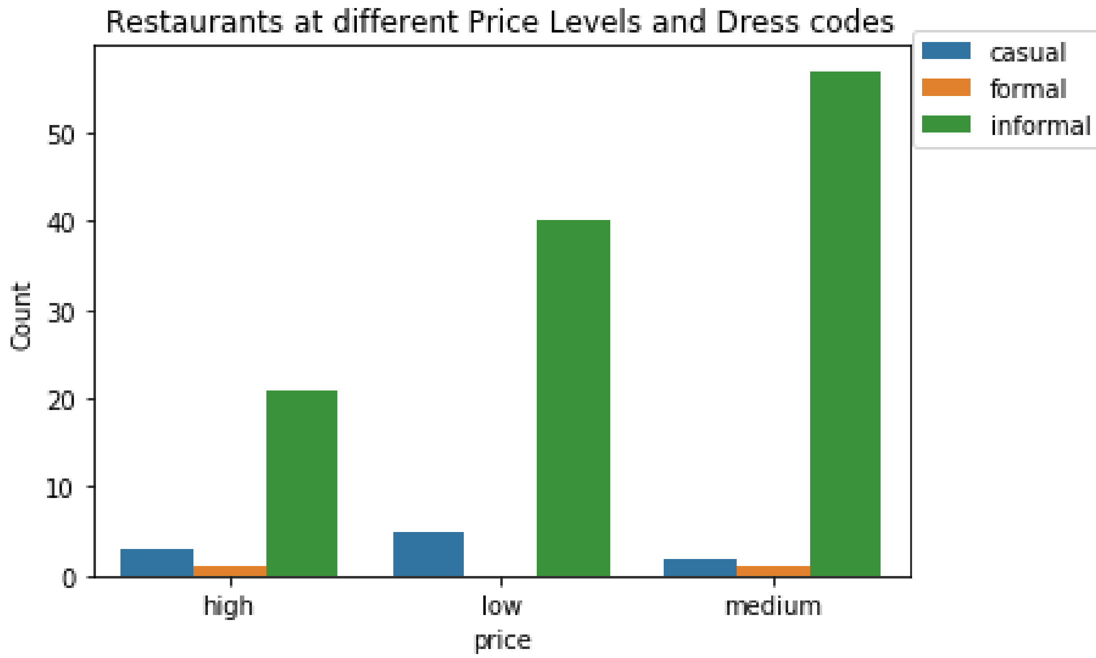


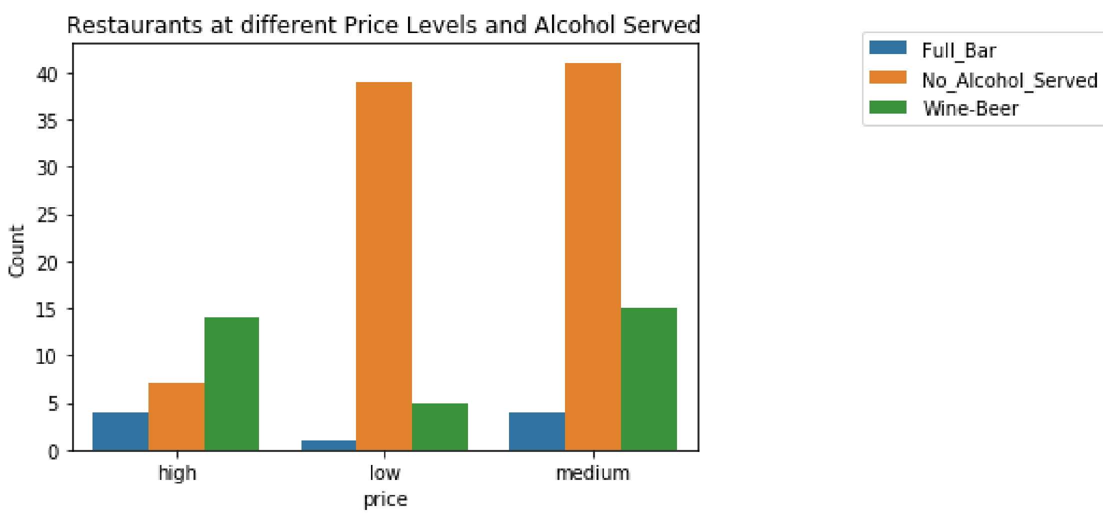
3 Restaurant Details

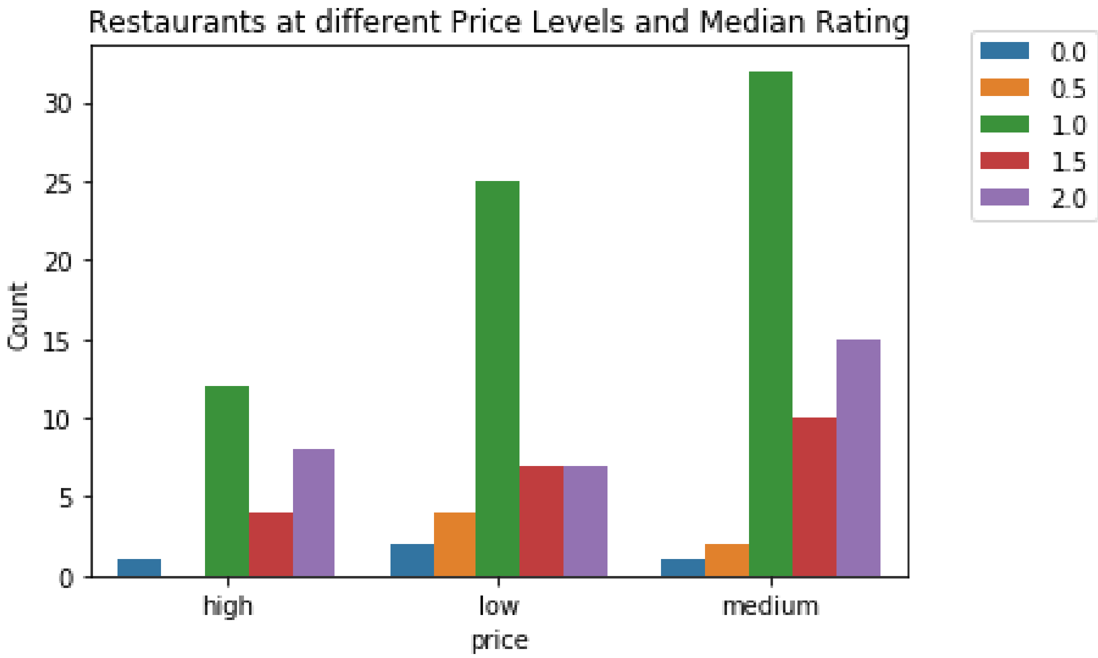
[9]:
```
restaurant['price'].value_counts().plot(kind = 'barh')
plt.xlabel('Count')
plt.title("Restaurants at different Price Levels")
plt.show();

sns.barplot(x = 'price', y ='placeID', hue = 'dress_code', data= restaurant.
 →groupby(['price', 'dress_code']).count().reset_index())
plt.title("Restaurants at different Price Levels and Dress codes")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.25, 1.05))
plt.show();


sns.barplot(x = 'price', y ='placeID', hue = 'alcohol', data= restaurant.
 →groupby(['price', 'alcohol']).count().reset_index())
plt.title("Restaurants at different Price Levels and Alcohol Served")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.25, 1.05))
plt.show();

sns.barplot(x = 'price', y ='placeID', hue = 'median_rating', data= restaurant.
 →groupby(['price', 'median_rating']).count().reset_index())
```

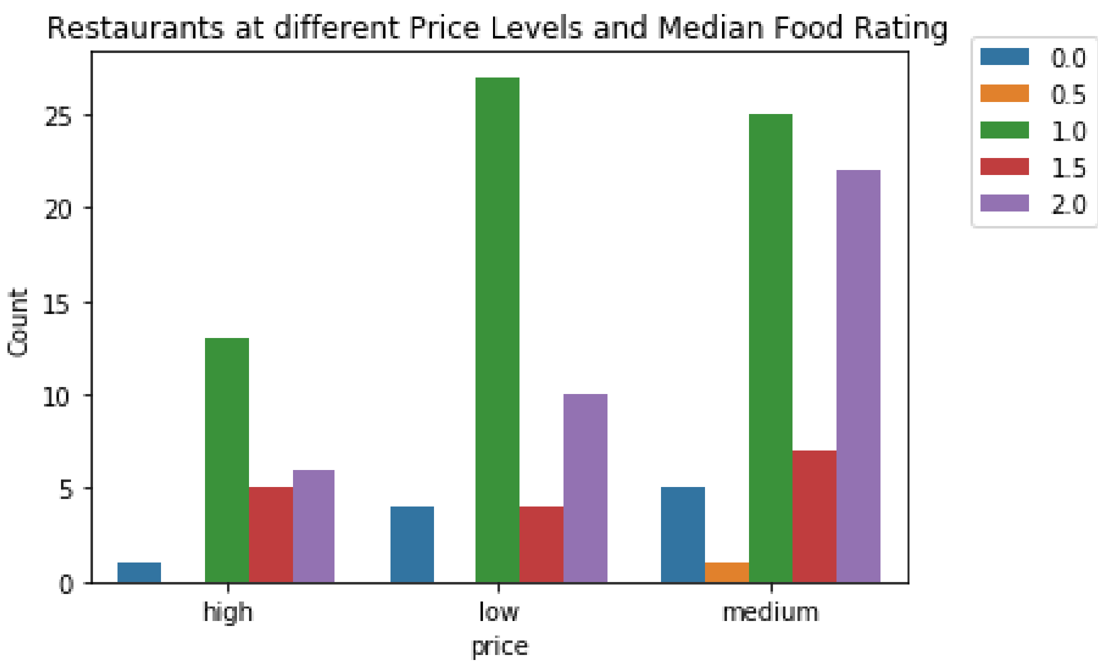
```
plt.title("Restaurants at different Price Levels and Median Rating")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.25, 1.05))
plt.show();

sns.barplot(x = 'price', y ='placeID', hue = 'median_food_rating', data=␣
↪restaurant.groupby(['price', 'median_food_rating']).count().reset_index())
plt.title("Restaurants at different Price Levels and Median Food Rating")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.25, 1.05))
plt.show();

sns.barplot(x = 'price', y ='placeID', hue = 'median_rating', data= restaurant.
↪groupby(['price', 'median_rating']).count().reset_index())
plt.title("Restaurants at different Price Levels and Median Rating")
plt.ylabel("Count")
plt.legend(bbox_to_anchor=(1.25, 1.05))
plt.show();
```

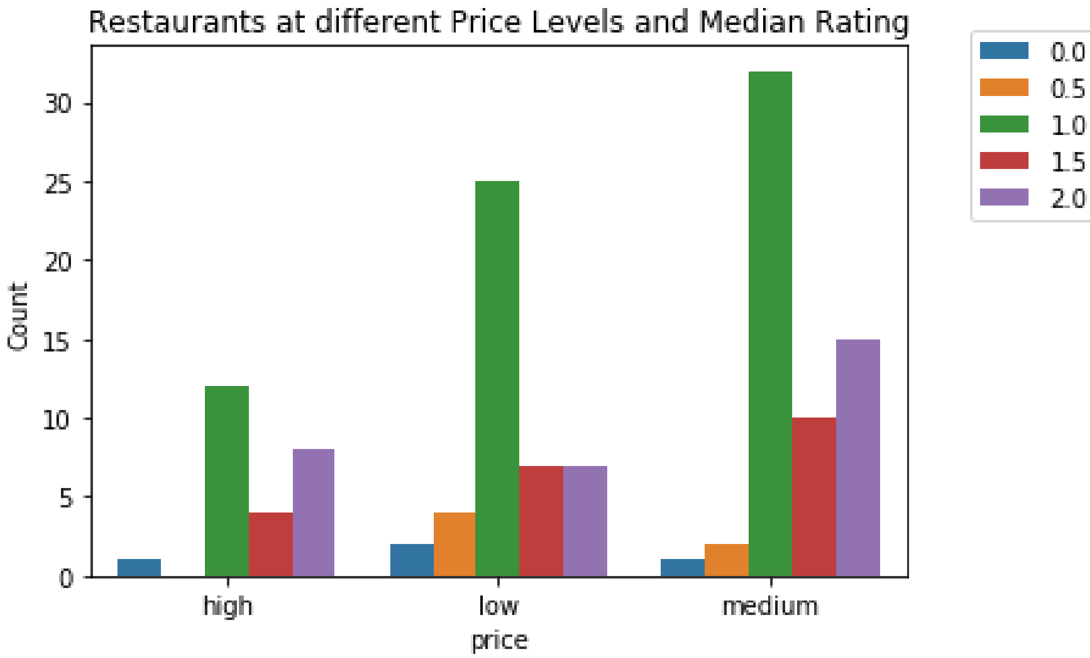



Restaurants at different Price Levels and Dress codes
casual
formal
informal
Count
0
10
20
30
40
50
high
low
medium
price


Restaurants at different Price Levels and Alcohol Served
Full_Bar
No_Alcohol_Served
Wine-Beer
Count
0
5
10
15
20
25
30
35
40
high
low
medium
price


Restaurants at different Price Levels and Median Rating
0.0
0.5
1.0
1.5
2.0
Count
30
25
20
15
10
5
0
high
low
medium
price


Restaurants at different Price Levels and Median Food Rating
0.0
0.5
1.0
1.5
2.0
Count
25
20
15
10
5
0
high
low
medium
price



4 The Type of Users at that type of Restaurant

Users goes to multiple restaurants

[10]:
```
fulldata.head(10)
```

[10]:
```
  userID  placeID  rating  food_rating  service_rating  latitude_x  \
0  U1077   135085       2            2               2   22.150802
1  U1077   135038       2            2               1   22.155651
2  U1077   132825       2            2               2   22.147392
3  U1077   135060       1            2               2   22.156883
4  U1077   135027       0            1               1   22.147145
5  U1108   135085       1            2               1   22.150802
6  U1108   132723       2            2               2   22.148934
7  U1108   135046       1            2               1   22.141282
8  U1108   135075       2            2               2   22.139573
9  U1108   132572       1            2               1   22.141647

   longitude_x             alcohol   smoking_area dress_code  ...    interest  \
0  -100.982680  No_Alcohol_Served  not permitted   informal  ...  technology
1  -100.977767  No_Alcohol_Served        section   informal  ...  technology
2  -100.983092  No_Alcohol_Served           none   informal  ...  technology
3  -100.978485  No_Alcohol_Served           none   informal  ...  technology
4  -100.974494          Wine-Beer           none   informal  ...  technology
5  -100.982680  No_Alcohol_Served  not permitted   informal  ...  technology
6  -101.019845           Full_Bar        section   informal  ...  technology
```

```
7  -101.002958  No_Alcohol_Served           none   informal  ...  technology
8  -100.991564  No_Alcohol_Served           none   informal  ...  technology
9  -100.992712  No_Alcohol_Served  not permitted   informal  ...  technology

         personality  religion activity color weight  budget  height  age  \
0  thrifty-protector  Catholic  student  blue     65  medium    1.71   25
1  thrifty-protector  Catholic  student  blue     65  medium    1.71   25
2  thrifty-protector  Catholic  student  blue     65  medium    1.71   25
3  thrifty-protector  Catholic  student  blue     65  medium    1.71   25
4  thrifty-protector  Catholic  student  blue     65  medium    1.71   25
5  thrifty-protector  Catholic  student  blue     76  medium    1.81   29
6  thrifty-protector  Catholic  student  blue     76  medium    1.81   29
7  thrifty-protector  Catholic  student  blue     76  medium    1.81   29
8  thrifty-protector  Catholic  student  blue     76  medium    1.81   29
9  thrifty-protector  Catholic  student  blue     76  medium    1.81   29

     age group
0  young adult
1  young adult
2  young adult
3  young adult
4  young adult
5  young adult
6  young adult
7  young adult
8  young adult
9  young adult

[10 rows x 40 columns]
```

[11]:
```python
# how many users goes to places that serve alcohol and out of those places, how␣
↪many are students?
fulldata.groupby(['alcohol', 'placeID', 'userID','activity']).count().
↪reset_index().groupby(['alcohol','userID','activity']).count().
↪groupby(['alcohol', 'activity']).count()
```

[11]:
```
                                  placeID  rating  food_rating  service_rating  \
alcohol           activity
Full_Bar          professional          7       7            7               7
                  student              62      62           62              62
                  unemployed            1       1            1               1
No_Alcohol_Served professional         17      17           17              17
                  student             116     116          116             116
                  unemployed            2       2            2               2
                  working-class         2       2            2               2
Wine-Beer         professional         14      14           14              14
                  student              91      91           91              91
                  unemployed            1       1            1               1
```

```
                  working-class        2       2            2              2

                                 latitude_x  longitude_x  smoking_area  \
alcohol           activity
Full_Bar          professional            7            7             7
                  student                62           62            62
                  unemployed              1            1             1
No_Alcohol_Served professional           17           17            17
                  student               116          116           116
                  unemployed              2            2             2
                  working-class           2            2             2
Wine-Beer         professional           14           14            14
                  student                91           91            91
                  unemployed              1            1             1
                  working-class           2            2             2

                                 dress_code  accessibility  price  ...  \
alcohol           activity                                         ...
Full_Bar          professional            7              7      7  ...
                  student                62             62     62  ...
                  unemployed              1              1      1  ...
No_Alcohol_Served professional           17             17     17  ...
                  student               116            116    116  ...
                  unemployed              2              2      2  ...
                  working-class           2              2      2  ...
Wine-Beer         professional           14             14     14  ...
                  student                91             91     91  ...
                  unemployed              1              1      1  ...
                  working-class           2              2      2  ...

                                 birth_year  interest  personality  religion  \
alcohol           activity
Full_Bar          professional            7         7            7         7
                  student                62        62           62        62
                  unemployed              1         1            1         1
No_Alcohol_Served professional           17        17           17        17
                  student               116       116          116       116
                  unemployed              2         2            2         2
                  working-class           2         2            2         2
Wine-Beer         professional           14        14           14        14
                  student                91        91           91        91
                  unemployed              1         1            1         1
                  working-class           2         2            2         2

                                 color  weight  budget  height  age  age group
alcohol           activity
Full_Bar          professional       7       7       7       7    7          7
```

```
                  student            62      62      62      62   62        62
                  unemployed          1       1       1       1    1         1
No_Alcohol_Served professional       17      17      17      17   17        17
                  student           116     116     116     116  116       116
                  unemployed          2       2       2       2    2         2
                  working-class       2       2       2       2    2         2
Wine-Beer         professional       14      14      14      14   14        14
                  student            91      91      91      91   91        91
                  unemployed          1       1       1       1    1         1
                  working-class       2       2       2       2    2         2

[11 rows x 37 columns]
```

[ ]:

5 Collaborative Filtering Recommendation

"The process is to calculate the similarities between target user i and all other users, select the top X similar users, and take the weighted average of ratings from these X users with similarities as weights." - Towards Data Science 1. KNN Based Filtering 2. matrix factorization. A better soluton for sparse data (data with alot of missing values) a. Rating matrix = user matrix * item matrix

[12]:

```
matrix = pd.pivot_table(rating, index = 'userID', columns = 'placeID', values =␣
 ↪'rating')

matrix = matrix.fillna(0)

usertoplacepivot = pd.pivot_table(rating, index = 'placeID', columns = 'userID',␣
 ↪values = 'rating')
usertoplacepivot = usertoplacepivot.fillna(0)
```

[13]:

```
#user ids are columns and place id are rows
usertoplacematrix = matrix.values.T

#place ids are columns and user ids are rows
placetousermatrix = matrix
```

[124]:

```
import sklearn
from sklearn.decomposition import TruncatedSVD
SVD = TruncatedSVD(n_components = 129, random_state = 17)
placetousermatrix = SVD.fit_transform(matrix)
usertoplacematrix = SVD.fit_transform(matrix.values.T)

placetousercorr = np.corrcoef(placetousermatrix)
usertoplacecorr = np.corrcoef(usertoplacematrix)
```

```
//anaconda3/lib/python3.7/site-packages/numpy/lib/function_base.py:2530:
RuntimeWarning: invalid value encountered in true_divide
```

```
  c /= stddev[:, None]
//anaconda3/lib/python3.7/site-packages/numpy/lib/function_base.py:2531:
RuntimeWarning: divide by zero encountered in true_divide
  c /= stddev[None, :]
//anaconda3/lib/python3.7/site-packages/numpy/lib/function_base.py:2531:
RuntimeWarning: invalid value encountered in true_divide
  c /= stddev[None, :]
```

[142]:
```
def recommendsimilarbusinesses(place_id):
    '''
    Using matrix factorization
    Take in a place ID and recommend similar places to that place ID
    '''
    index = list(matrix.columns).index(place_id)
    print(index)
    placename =restaurantraw[restaurantraw['placeID'] == place_id]['name'].
 ↪iloc[0]

    corrvalues = usertoplacecorr[index]
    return [restaurantraw[restaurantraw['placeID'] == k]['name'].iloc[0] for k␣
 ↪in matrix.columns[(corrvalues < 1.0) & (corrvalues > 0.5)]]
```

[143]:
```
recommendsimilarbusinesses(132560)
```

```
0
```

[143]:
```
['little pizza Emilio Portes Gil', 'Taqueria EL amigo ']
```

[28]:
```
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
knn_matrix = csr_matrix(matrix.values.T)
knn_model = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
knn_model.fit(knn_matrix)

query_index = np.random.choice(matrix.values.T.shape[0])
query_index = 88
distances, indices = knn_model.kneighbors(matrix.values.T[query_index].
 ↪reshape(1,-1), n_neighbors = 3)
for i in range(0, len(distances.flatten())):
    if i == 0:
        print("Recommendations for {0}, {1}: ".format(usertoplacepivot.
 ↪index[query_index],restaurantraw[restaurantraw['placeID'] == usertoplacepivot.
 ↪index[query_index]]['name'].iloc[0]))
    else:
```

```
        print('{0} : {1}, {3}, with distance of {2}'.format(i, usertoplacepivot.
↪index[indices.flatten()[i]], distances.flatten()[i],␣
↪restaurantraw[restaurantraw['placeID'] == usertoplacepivot.index[indices.
↪flatten()[i]]]['name'].iloc[0]))
```

```
Recommendations for 135042, Restaurant Oriental Express:
1 : 135063, Restaurante Alhondiga, with distance of 0.5327246494517668
2 : 135032, Cafeteria y Restaurant El Pacifico, with distance of
0.5758224892954421
```

First, find similar users based on these characteristics Create a user matrix of features

[19]:
```
usermatrix = userprofile[['transport', 'drink_level','marital_status',␣
↪'activity', 'budget', 'color', 'age group']]

Xuser = pd.get_dummies(usermatrix, columns = [ 'transport',␣
↪'drink_level','marital_status', 'activity', 'budget', 'color', 'age group'])
```

[20]:
```
# find the three closest nearest neighbors using cosine similarities

def findnearestneighbors(userID):
    listofusers = userprofile['userID']
    index = listofusers[listofusers == userID].index[0]
    knn_model = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
    knn_model.fit(csr_matrix(Xuser.fillna(0).values))
    distances, indices = knn_model.kneighbors(Xuser.values[index].reshape(1,-1),␣
↪n_neighbors = 4)
    listofnn = []
    nn = []
    for i in range(0, len(distances.flatten())):
        if i == 0:
            print("Recommendations for {0}: ".format(listofusers[index]))
        else:
            nn.append(listofusers[indices.flatten()[i]])
            listofnn.append((listofusers[indices.flatten()[i]], distances.
↪flatten()[i]))
    userIDdict = {}
    userIDdict[listofusers[index]] = listofnn
    nn =[listofusers[index]] + nn
    return userIDdict, nn
```

[21]:
```
findnearestneighbors('U1010')
```

```
Recommendations for U1010:
```

[21]:
```
({'U1010': [('U1131', 0.14285714285714313),
   ('U1050', 0.14285714285714313),
   ('U1046', 0.2857142857142859)]},
 ['U1010', 'U1131', 'U1050', 'U1046'])
```

Get the restaurant ratings for these nearest neighbors user

[121]:
```
recommendsimilarbusinesses(132560)
```

[121]: ['little pizza Emilio Portes Gil', 'Taqueria EL amigo ']

6 classification

building a training data, a validation data, and testing data

[155]:
```
data = pd.get_dummies(fulldata[fulldata['rating'] != 1],
               columns = [k for k in fulldata.select_dtypes(exclude = ['int',␣
 ↪'float']).columns if k != 'userID'])
y = data['rating']
data.drop(columns = ['userID', 'placeID', 'latitude_x', 'longitude_x',␣
 ↪'latitude_y',
                    'longitude_y', 'birth_year', 'weight', 'height', 'age',␣
 ↪'rating'], axis = 1, inplace = True)
x = data
```

[157]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size = 0.33,␣
 ↪random_state = 10)
model = LogisticRegression(solver = 'lbfgs')
model.fit(X_train, Y_train)
predictions = model.predict(X_test)
print(accuracy_score(Y_test, predictions))
print(classification_report(Y_test, predictions))
```

```
0.9469387755102041
              precision    recall  f1-score   support

           0       0.89      0.95      0.92        78
           2       0.98      0.95      0.96       167

    accuracy                           0.95       245
   macro avg       0.93      0.95      0.94       245
weighted avg       0.95      0.95      0.95       245
```

[ ]:
```

```