**RESTAURANT REVIEW PREDICTION**

**A MINI PROJECT REPORT**

***Submitted by***

**B.SIVA SUNDARI(953622104098)**

**P.SWATHIGA(953622104105)**

**R.SATHIYA PUSHPA(953622104093)**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**RAMCO INSTITUTE OF TECHNOLOGY,**

**RAJAPALAYAM**

**FEBRAURY 2024**

**CS3361 DATA SCIENCE LABORATORY**

**BONAFIDE CERTIFICATE**

Certified that this miniproject report **“RESTAURANT REVIEW PREDICTION”** is the bonafide work of **“SATHIYAPUSHPA R(953622104093),SIVA SUNDARI B(953622104098),SWATHIGA P (953622104105)”** who carried out the miniproject work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| S.Vijaya amaladevi M.E. CSE | Dr.K.Vijayalakshmi M.E., Ph.D. |
| **Faculty In-charge** | **HEAD OF THE DEPARTMENT** |
| Assistant Professor | Department of Computer Science |
| Department of Computer Science and | and Engineering |
| Engineering | Ramco Institute of Technology |
| Ramco Institute of Technology | North Venganallur Village |
| North Venganallur Village | Rajapalayam – 626117 |
| Rajapalayam– 626117 |  |

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ABSTRACT**

The restaurant review system is an essential tool in today's digital age, providing users with personalized recommendations based on their preferences and past dining experiences. This system aims to simplify the restaurant selection process by aggregating user data and suggesting suitable dining establishments. Similar to movie recommendation systems, the reliability of a restaurant review system is crucial for enhancing user satisfaction and engagement. However, challenges such as poor recommendation quality and scalability issues need to be addressed for optimal performance.

Restaurants play a significant role in social life, offering diverse cuisines and dining experiences to cater to different preferences and occasions. The system categorizes restaurants based on various factors such as cuisine type, price range, ambiance, and location, making it easier for users to find restaurants that meet their specific criteria. By leveraging user data and feedback, the system can provide accurate and relevant recommendations, enhancing the overall dining experience for users.

Overall, the restaurant review system serves as a valuable tool for both users and businesses in the food industry. Users can discover new dining options and make informed decisions, while businesses can improve customer interactions and increase customer satisfaction. Despite facing challenges, the system continues to evolve, leveraging advanced technologies to provide better recommendations and enhance the dining experience for all.

**TABLE OF CONTENT:**

|  |  |
| --- | --- |
| **Content** | **Page no** |
| 1.INTRODUCTION | 5 |
| 1.1 Restaurant Review System | 5 |
| 1.2Project objectives | 6 |
| 1.3Project specification | 6 |
| 1.4Modules | 6 |
| 2.SYSTEM SPECIFICATION | 6 |
| 2.1Hardware specification | 6 |
| 2.2Software specification | 6 |
| 3.PACKAGES | 6 |
| 3.1 Numpy | 6 |
| 3.2Pandas | 7 |
| 3.3Seaborn | 7 |
| 3.4Warnings | 7 |
| 3.5Matplotlib | 8 |
| 4.APENDIX | 9 |
| 4.1Source code | 9 |
| 4.2Screenshots | 12 |
| 5.CONCLUSION | 25 |
| 6.FUTURE WORK | 25 |
| 7.REFERENCE | 25 |

**1.INTRODUCTION**

**1.1 Restaurant Review System**

The restaurant review system is a cornerstone of modern dining, revolutionizing how people discover and choose dining experiences. This system operates on the principle of information filtering, predicting user preferences based on past behavior and providing tailored recommendations. It's an integral part of the digital landscape, aiding users in navigating the vast array of dining options available. Restaurants are not just places to eat; they are venues for socializing, celebrating, and experiencing different cuisines and cultures. The system's ability to categorize restaurants based on various factors such as cuisine type, price range, and location helps users find the perfect dining spot for any occasion.

One of the key objectives of a restaurant review system is to simplify the decision-making process for users by offering personalized recommendations. These recommendations are based on a user's dining history, ratings, and preferences, making the dining experience more enjoyable and convenient. However, ensuring the reliability of these recommendations is crucial, as inaccurate or irrelevant suggestions can lead to user dissatisfaction. To address this, the system must continuously analyze and update user data to improve the accuracy of its recommendations.

Despite their importance, restaurant review systems face several challenges, including maintaining recommendation quality and scalability. The sheer volume of restaurants and user reviews poses a challenge in ensuring that recommendations remain relevant and useful. Additionally, the system must contend with biases in user ratings and preferences, which can skew recommendations. To overcome these challenges, advanced algorithms and machine learning techniques are employed to analyze user data and provide more accurate recommendations.

In conclusion, the restaurant review system is a vital tool in the modern dining landscape, helping users discover new dining experiences and making informed choices. By leveraging user data and advanced algorithms, these systems provide personalized recommendations that enhance the overall dining experience. However, ensuring the reliability and scalability of these systems remains a challenge, requiring continuous innovation and improvement to meet the evolving needs of users.

**1.2PROJECT OBJECTIVES**

The primary objective of this project is to develop a restaurant review system that offers personalized restaurant recommendations to users based on their preferences and past dining experiences. By leveraging user data and machine learning algorithms, the system aims to enhance the quality of restaurant suggestions and improve user satisfaction.

**1.3PROJECT SPECIFICATION**

The restaurant review system will be designed to:Collect and analyze user data, including ratings and reviews.Generate personalized restaurant recommendations for users.Implement machine learning algorithms for recommendation purposes.Provide a user-friendly interface for easy navigation and interaction.

**1.4MODULES**

* Numpy
* Pandas
* Matplotlib.pyplot
* Plotly.graph\_objects
* Seaborn
* Warnings

**2.SYSTEM SPECIFICATION**

**2.1Hardware specification**

* Processor: Intel Core i5 or higher.
* RAM: 8GB or higher.
* Storage: 256GB SSD or higher.
* Display: Full HD resolution monitor.
* Input Devices: Keyboard and mouse.

**2.2Software Specification**

* Operating System: Windows 10
* Programming Language: Python.
* Libraries: Pandas, NumPy, Matplotlib, Seaborn.
* Development Environment: Google colab

**3.PACKAGES**

**3.1 NUMPY**

* NumPy is a Python library used for working with arrays.
* It also has functions for working in domain of linear algebra, fourier transform, and matrices.
* NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
* NumPy stands for Numerical Python.

**INSTALLING NUMPY PACKAGE**

pip install numpy

* 1. **PANDAS**
* Pandas is a Python library used for working with data sets.
* It has functions for analyzing, cleaning, exploring, and manipulating data.
* The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

**INSTALLING PANDAS PACKAGE**

pip install pandas

**3.3 SEABORN:**

Purpose: Seaborn is a statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

**Key Features:**

* Simplifies the creation of complex visualizations with fewer lines of code.
* Integration with Pandas DataFrames.
* Support for various plot types like heatmaps, violin plots, and more.

**3.4 WARNINGS:**

Purpose: The `warnings` module is part of Python's standard library and provides a way to handle warnings in a flexible manner.

**Key Features:**

* Allows filtering and controlling how warnings are issued or ignored.
* Useful for alerting developers about potential issues without terminating the program.

**3.5 MATPLOTLIB**

* Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy.
* As such, it offers a viable open source alternative to **MATLAB.** Developers can also use matplotlib’s APIs(Application Programming Interfaces) to embed plots inGUI applications.

A Python matplotlib script is structured so that a fewlines of code are all that is required in most instancesto generate a visual data plot.

The matplotlib scripting layer overlays two APIs:

* The pyplot API is a hierarchy of Python codeobjects topped by matplotlib.pyplot
* An OO (Object-Oriented) API collection of objectsthat can be assembled with greater flexibility thanpyplot. This API provides direct access to Matplotlib’sbackend layers.

**Matplotlib and Pyplot in Python :**

The pyplot API has a convenient MATLAB-style statefulinterface. In fact, matplotlib was originally written as an open source alternative for MATLAB. The OO API and its interface is more customizable and powerful than pyplot, but considered more difficult to use. As a result, the pyplot interface is more commonly used, and is referred to by default in this article.

Understanding matplotlib’s pyplot API is key to understanding how to work with plots:

* **matplotlib.pyplot.figure**: Figure is the top-level container. It includes everything visualized in a plot including one or more Axes.
* **matplotlib.pyplot.axes**: Axes contain most of the elements in a plot: Axis, Tick, Line2D, Text, etc., and sets the coordinates. It is the area in which data is plotted. Axes include the X-Axis, Y-Axis, and possibly a Z-Axis, as well.

**Installing Matplotlib :**

pip install matplotlib

**4.APPENDIX**

**4.1 SOURCE CODE**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.graph\_objects as go

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

pd.set\_option("display.max\_rows", None)

pd.set\_option("display.max\_columns", None)

usercuisine = pd.read\_csv("usercuisine.csv")

userpayment = pd.read\_csv("userpayment.csv")

userprofile = pd.read\_csv("userprofile.csv")

geoplaces = pd.read\_csv("geoplaces.csv")

chefmozaccepts = pd.read\_csv("chefmozaccepts.csv")

chefmozcuisine = pd.read\_csv("chefmozcuisine.csv")

chefmozhours = pd.read\_csv("chefmozhours4.csv")

chefmozparking = pd.read\_csv("chefmozparking.csv")

ratings = pd.read\_csv("ratings.csv")

import pandas as pd

def dfdetails(dfs):

    for df\_name, df in dfs.items():

        print(f"DataFrame: {df\_name}")

        print(f"\n{df.head(3)}\n")

        print("\nData Summary:")

        print(df.describe())

        # Check for missing values

        print("\nMissing Values:")

        print(df.isnull().sum())

        # Check for duplicate rows

        print("\nDuplicate Rows:")

        print(df.duplicated().sum())

        categorical\_cols = df.select\_dtypes(include=["object"]).columns

        if len(categorical\_cols) > 0:

            print("\nUnique Values in Categorical Columns:")

            for col in categorical\_cols:

                print(f"\n{col}:")

                print(df[col].unique())

        numerical\_cols = df.select\_dtypes(include=["int64", "float64"]).columns

        if len(numerical\_cols) > 0:

            print("\nNumerical Column Statistics:")

            for col in numerical\_cols:

                print(f"\n{col}:")

                print(f"Minimum: {df[col].min()}")

                print(f"Maximum: {df[col].max()}")

                print(f"Mean: {df[col].mean()}")

                print(f"Median: {df[col].median()}")

                print(f"Standard Deviation: {df[col].std()}")

        print(f"Shape: {df.shape}")

        print(f"Columns: {', '.join(df.columns)}\n")

        print({df.info()})

        print("\n=======================================================================================================================\n")

data\_frames = {

    "userprofile": userprofile,

    "ratings": ratings

}

dfdetails(data\_frames)

In this code, the `warnings` module is used to suppress warnings that might be generated during the execution of the code. Here's a detailed explanation of each line:`import warnings`: This imports the `warnings` module, which provides a way to handle warnings that might occur during program execution.`warnings.filterwarnings("ignore")`: This line sets the warning filter to ignore all warnings. This means that any warnings that would normally be displayed during the execution of the code will be suppressed.`pd.set\_option("display.max\_rows", None)`: This line sets the maximum number of rows to display in a DataFrame to `None`, meaning that all rows will be displayed.`pd.set\_option("display.max\_columns", None)`: This line sets the maximum number of columns to display in a DataFrame to `None`, meaning that all columns will be displayed.Read all csv files using pd.read\_csv`def dfdetails(dfs)`: This line defines a function called `dfdetails` that takes a dictionary of DataFrames (`dfs`) as input.`for df\_name, df in dfs.items():`: This line iterates over the items (key-value pairs) in the dictionary `dfs`, where `df\_name` is the name of the DataFrame and `df` is the DataFrame itself.`print(f"DataFrame: {df\_name}")`: This line prints the name of the current DataFrame being processed.`print(f"\n{df.head(3)}\n")`: This line prints the first 3 rows of the current DataFrame.`print("\nData Summary:")`: This line prints a header for the data summary section.`print(df.describe())`: This line prints the summary statistics (count, mean, std, min, 25%, 50%, 75%, max) for the numerical columns in the current DataFrame.`print("\nMissing Values:")`: This line prints a header for the missing values section.`print(df.isnull().sum())`: This line prints the number of missing values in each column of the current DataFrame.`print("\nDuplicate Rows:")`: This line prints a header for the duplicate rows section.`print(df.duplicated().sum())`: This line prints the number of duplicate rows in the current DataFrame.

`categorical\_cols = df.select\_dtypes(include=["object"]).columns`: This line selects the columns in the current DataFrame that are of type `object` (categorical columns).`if len(categorical\_cols) > 0:`: This line checks if there are any categorical columns in the current DataFrame.`print("\nUnique Values in Categorical Columns:")`: This line prints a header for the unique values in categorical columns section.`for col in categorical\_cols:`: This line iterates over the categorical columns in the current DataFrame.`print(f"\n{col}:")`: This line prints the name of the current categorical column.`print(df[col].unique())`: This line prints the unique values in the current categorical column.`numerical\_cols = df.select\_dtypes(include=["int64", "float64"]).columns`: This line selects the columns in the current DataFrame that are of type `int64` or `float64` (numerical columns).`if len(numerical\_cols) > 0:`: This line checks if there are any numerical columns in the current DataFrame.`print("\nNumerical Column Statistics:")`: This line prints a header for the numerical column statistics section.`for col in numerical\_cols:`: This line iterates over the numerical columns in the current DataFrame.`print(f"\n{col}:")`: This line prints the name of the current numerical column.`print(f"Minimum: {df[col].min()}")`: This line prints the minimum value in the current numerical column.`print(f"Maximum: {df[col].max()}")`: This line prints the maximum value in the current numerical column.`print(f"Mean: {df[col].mean()}")`: This line prints the mean value of the current numerical column.`print(f"Median: {df[col].median()}")`: This line prints the median value of the current numerical column.`print(f"Standard Deviation: {df[col].std()}")`: This line prints the standard deviation of the current numerical column.`print(f"Shape: {df.shape}")`: This line prints the shape (number of rows and columns) of the current DataFrame.`print(f"Columns: {', '.join(df.columns)}\n")`: This line prints the column names of the current DataFrame.`print({df.info()})`: This line prints the information about the current DataFrame, including the data type of each column and the memory usage.

Overall, this code defines a function to provide detailed information about the DataFrames, including data summary, missing values, duplicate rows, unique values in categorical columns, and statistics for numerical columns. It also sets some display options for Pandas to show all rows and columns, and suppresses warnings.

**4.2Screenshots**

**OUTPUT:**

DataFrame: userprofile

userID latitude longitude smoker drink\_level dress\_preference \

0 U1001 22.139997 -100.978803 false abstemious informal

1 U1002 22.150087 -100.983325 false abstemious informal

2 U1003 22.119847 -100.946527 false social drinker formal

ambience transport marital\_status hijos birth\_year interest \

0 family on foot single independent 1989 variety

1 family public single independent 1990 technology

2 family public single independent 1989 none

personality religion activity color weight budget height

0 thrifty-protector none student black 69 medium 1.77

1 hunter-ostentatious Catholic student red 40 low 1.87

2 hard-worker Catholic student blue 60 low 1.69

Data Summary:

latitude longitude birth\_year weight height

count 138.000000 138.000000 138.000000 138.000000 138.000000

mean 21.810389 -100.291857 1984.702899 64.869565 1.667536

std 1.552529 0.869916 14.655364 17.214332 0.130473

min 18.813348 -101.054680 1930.000000 40.000000 1.200000

25% 22.126030 -100.983000 1987.000000 53.000000 1.600000

50% 22.150497 -100.937788 1989.000000 65.000000 1.690000

75% 22.186642 -99.183251 1991.000000 74.750000 1.750000

max 23.771030 -99.067106 1994.000000 120.000000 2.000000

Missing Values:

userID 0

latitude 0

longitude 0

smoker 0

drink\_level 0

dress\_preference 0

ambience 0

transport 0

marital\_status 0

hijos 0

birth\_year 0

interest 0

personality 0

religion 0

activity 0

color 0

weight 0

budget 0

height 0

dtype: int64

Duplicate Rows:

0

Unique Values in Categorical Columns:

userID:

['U1001' 'U1002' 'U1003' 'U1004' 'U1005' 'U1006' 'U1007' 'U1008' 'U1009'

'U1010' 'U1011' 'U1012' 'U1013' 'U1014' 'U1015' 'U1016' 'U1017' 'U1018'

'U1019' 'U1020' 'U1021' 'U1022' 'U1023' 'U1024' 'U1025' 'U1026' 'U1027'

'U1028' 'U1029' 'U1030' 'U1031' 'U1032' 'U1033' 'U1034' 'U1035' 'U1036'

'U1037' 'U1038' 'U1039' 'U1040' 'U1041' 'U1042' 'U1043' 'U1044' 'U1045'

'U1046' 'U1047' 'U1048' 'U1049' 'U1050' 'U1051' 'U1052' 'U1053' 'U1054'

'U1055' 'U1056' 'U1057' 'U1058' 'U1059' 'U1060' 'U1061' 'U1062' 'U1063'

'U1064' 'U1065' 'U1066' 'U1067' 'U1068' 'U1069' 'U1070' 'U1071' 'U1072'

'U1073' 'U1074' 'U1075' 'U1076' 'U1077' 'U1078' 'U1079' 'U1080' 'U1081'

'U1082' 'U1083' 'U1084' 'U1085' 'U1086' 'U1087' 'U1088' 'U1089' 'U1090'

'U1091' 'U1092' 'U1093' 'U1094' 'U1095' 'U1096' 'U1097' 'U1098' 'U1099'

'U1100' 'U1101' 'U1102' 'U1103' 'U1104' 'U1105' 'U1106' 'U1107' 'U1108'

'U1109' 'U1110' 'U1111' 'U1112' 'U1113' 'U1114' 'U1115' 'U1116' 'U1117'

'U1118' 'U1119' 'U1120' 'U1121' 'U1122' 'U1123' 'U1124' 'U1125' 'U1126'

'U1127' 'U1128' 'U1129' 'U1130' 'U1131' 'U1132' 'U1133' 'U1134' 'U1135'

'U1136' 'U1137' 'U1138']

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']

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']

budget:

['medium' 'low' '?' 'high']

Numerical Column Statistics:

latitude:

Minimum: 18.813348

Maximum: 23.77103

Mean: 21.810389239130433

Median: 22.150496500000003

Standard Deviation: 1.5525285961971305

longitude:

Minimum: -101.05468

Maximum: -99.067106

Mean: -100.29185721014493

Median: -100.9377885

Standard Deviation: 0.869916451392283

birth\_year:

Minimum: 1930

Maximum: 1994

Mean: 1984.7028985507247

Median: 1989.0

Standard Deviation: 14.655364190844091

weight:

Minimum: 40

Maximum: 120

Mean: 64.8695652173913

Median: 65.0

Standard Deviation: 17.214332038938092

height:

Minimum: 1.2

Maximum: 2.0

Mean: 1.667536231884058

Median: 1.69

Standard Deviation: 0.1304725358812126

Shape: (138, 19)

Columns: userID, latitude, longitude, smoker, drink\_level, dress\_preference, ambience, transport, marital\_status, hijos, birth\_year, interest, personality, religion, activity, color, weight, budget, height

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 138 entries, 0 to 137

Data columns (total 19 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 userID 138 non-null object

1 latitude 138 non-null float64

2 longitude 138 non-null float64

3 smoker 138 non-null object

4 drink\_level 138 non-null object

5 dress\_preference 138 non-null object

6 ambience 138 non-null object

7 transport 138 non-null object

8 marital\_status 138 non-null object

9 hijos 138 non-null object

10 birth\_year 138 non-null int64

11 interest 138 non-null object

12 personality 138 non-null object

13 religion 138 non-null object

14 activity 138 non-null object

15 color 138 non-null object

16 weight 138 non-null int64

17 budget 138 non-null object

18 height 138 non-null float64

dtypes: float64(3), int64(2), object(14)

memory usage: 20.6+ KB

{None}

=======================================================================================================================

DataFrame: ratings

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

Data Summary:

placeID rating food\_rating service\_rating

count 1161.000000 1161.000000 1161.000000 1161.000000

mean 134192.041344 1.199828 1.215332 1.090439

std 1100.916275 0.773282 0.792294 0.790844

min 132560.000000 0.000000 0.000000 0.000000

25% 132856.000000 1.000000 1.000000 0.000000

50% 135030.000000 1.000000 1.000000 1.000000

75% 135059.000000 2.000000 2.000000 2.000000

max 135109.000000 2.000000 2.000000 2.000000

Missing Values:

userID 0

placeID 0

rating 0

food\_rating 0

service\_rating 0

dtype: int64

Duplicate Rows:

0

Unique Values in Categorical Columns:

userID:

['U1077' 'U1068' 'U1067' 'U1103' 'U1107' 'U1044' 'U1070' 'U1031' 'U1082'

'U1023' 'U1060' 'U1015' 'U1123' 'U1021' 'U1026' 'U1083' 'U1108' 'U1012'

'U1093' 'U1030' 'U1066' 'U1127' 'U1017' 'U1100' 'U1133' 'U1118' 'U1072'

'U1080' 'U1063' 'U1074' 'U1117' 'U1020' 'U1051' 'U1055' 'U1053' 'U1035'

'U1064' 'U1081' 'U1001' 'U1014' 'U1088' 'U1056' 'U1134' 'U1106' 'U1126'

'U1124' 'U1018' 'U1125' 'U1119' 'U1076' 'U1022' 'U1094' 'U1116' 'U1105'

'U1075' 'U1007' 'U1115' 'U1024' 'U1132' 'U1005' 'U1120' 'U1004' 'U1037'

'U1052' 'U1036' 'U1038' 'U1071' 'U1114' 'U1084' 'U1049' 'U1019' 'U1069'

'U1113' 'U1104' 'U1079' 'U1045' 'U1009' 'U1016' 'U1047' 'U1095' 'U1034'

'U1078' 'U1099' 'U1008' 'U1040' 'U1121' 'U1110' 'U1042' 'U1013' 'U1054'

'U1111' 'U1128' 'U1062' 'U1101' 'U1135' 'U1032' 'U1048' 'U1092' 'U1085'

'U1109' 'U1102' 'U1098' 'U1046' 'U1122' 'U1138' 'U1033' 'U1089' 'U1003'

'U1091' 'U1027' 'U1029' 'U1086' 'U1137' 'U1090' 'U1061' 'U1041' 'U1059'

'U1112' 'U1057' 'U1025' 'U1097' 'U1006' 'U1131' 'U1073' 'U1058' 'U1002'

'U1096' 'U1136' 'U1010' 'U1028' 'U1050' 'U1129' 'U1087' 'U1065' 'U1039'

'U1130' 'U1043' 'U1011']

Numerical Column Statistics:

placeID:

Minimum: 132560

Maximum: 135109

Mean: 134192.04134366926

Median: 135030.0

Standard Deviation: 1100.9162751276554

rating:

Minimum: 0

Maximum: 2

Mean: 1.1998277347114557

Median: 1.0

Standard Deviation: 0.7732822819258615

food\_rating:

Minimum: 0

Maximum: 2

Mean: 1.215331610680448

Median: 1.0

Standard Deviation: 0.7922942990574078

service\_rating:

Minimum: 0

Maximum: 2

Mean: 1.0904392764857882

Median: 1.0

Standard Deviation: 0.7908440907254338

Shape: (1161, 5)

Columns: userID, placeID, rating, food\_rating, service\_rating

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1161 entries, 0 to 1160

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 userID 1161 non-null object

1 placeID 1161 non-null int64

2 rating 1161 non-null int64

3 food\_rating 1161 non-null int64

4 service\_rating 1161 non-null int64

dtypes: int64(4), object(1)

memory usage: 45.5+ KB

{None}

cuisine\_counts = usercuisine["Rcuisine"].value\_counts()

top\_n = 10

top\_cuisine\_counts = cuisine\_counts.nlargest(top\_n)

other\_count = cuisine\_counts.sum() - top\_cuisine\_counts.sum()

categories\_to\_plot = list(top\_cuisine\_counts.index)

categories\_to\_plot.append("All Others")

counts\_to\_plot = list(top\_cuisine\_counts.values)

counts\_to\_plot.append(other\_count)

plt.figure(figsize=(10, 6))

plt.barh(categories\_to\_plot, counts\_to\_plot)

plt.xlabel("Count")

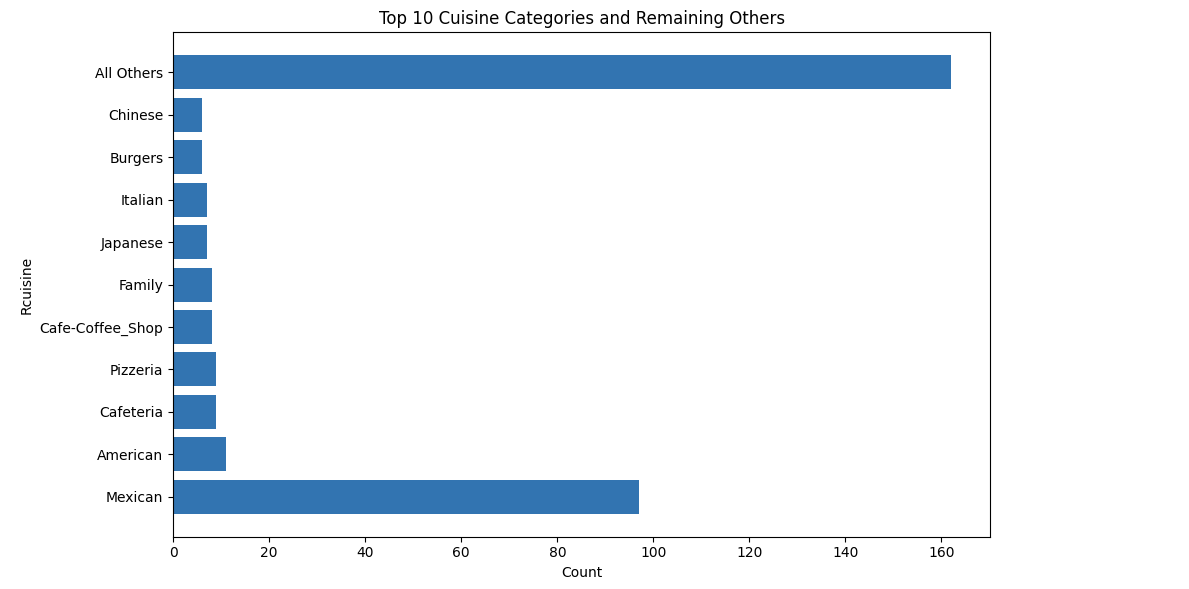
plt.ylabel("Rcuisine")

plt.title(f"Top {top\_n} Cuisine Categories and Remaining Others")

plt.tight\_layout()  # To prevent overlapping labels

plt.show()

**OUTPUT:**

****

payment\_counts = userpayment["Upayment"].value\_counts()

plt.figure(figsize=(6,6))

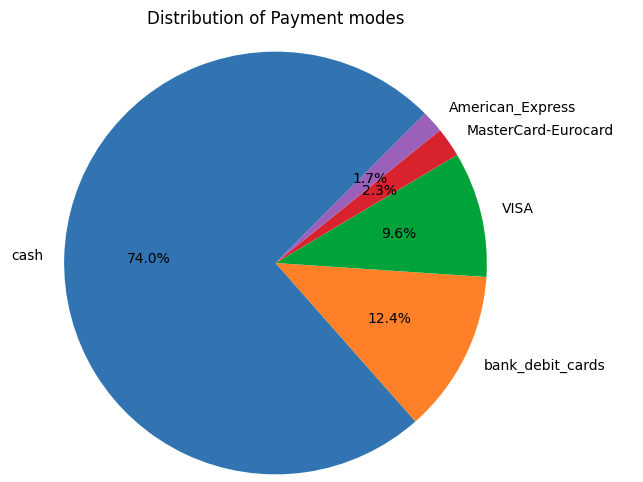
plt.pie(payment\_counts, labels=payment\_counts.index, autopct='%1.1f%%',startangle = 45)

plt.title("Distribution of Payment modes")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

userprofile.columns

**OUTPUT:**

Index(['userID', 'latitude', 'longitude', 'smoker', 'drink\_level',

'dress\_preference', 'ambience', 'transport', 'marital\_status', 'hijos',

'birth\_year', 'interest', 'personality', 'religion', 'activity',

'color', 'weight', 'budget', 'height'],

dtype='object')

import pandas as pd

import matplotlib.pyplot as plt

df = userprofile[['smoker', 'drink\_level']]

# Group and count the occurrences of each combination

grouped\_counts = df.groupby(['smoker', 'drink\_level']).size().unstack(fill\_value=0)

# Plot the grouped bar chart

plt.figure(figsize=(6, 6))

grouped\_counts.plot(kind='bar', stacked=True)

plt.xlabel('Smoker')

plt.ylabel('Count')

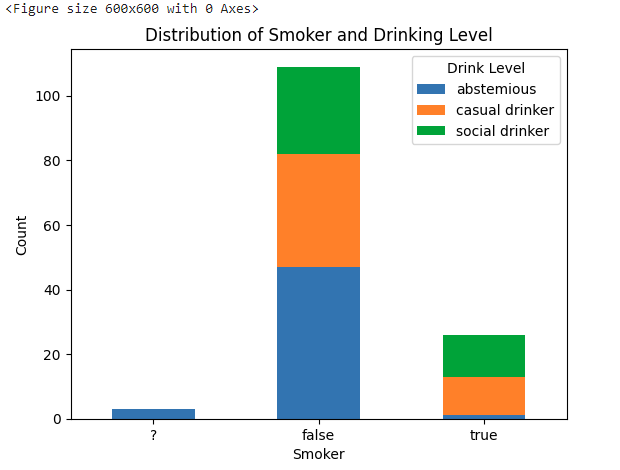
plt.title('Distribution of Smoker and Drinking Level')

plt.legend(title='Drink Level', loc='upper right')

plt.xticks(rotation=0)  # Keep the x-axis labels horizontal for better readability

plt.show()

**OUTPUT:**

****

df = userprofile

ambience\_counts = df['ambience'].value\_counts()

plt.figure(figsize=(6,6))

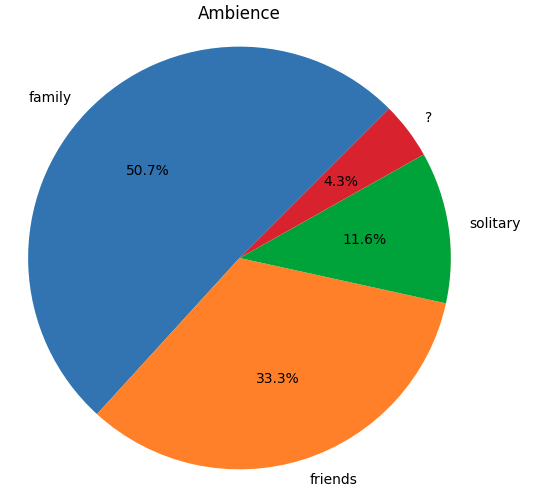
plt.pie(ambience\_counts, labels=ambience\_counts.index, autopct='%1.1f%%',startangle=45)

plt.title("Ambience")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

df = chefmozhours

ambience\_counts = df['days'].value\_counts()

plt.figure(figsize=(6,6))

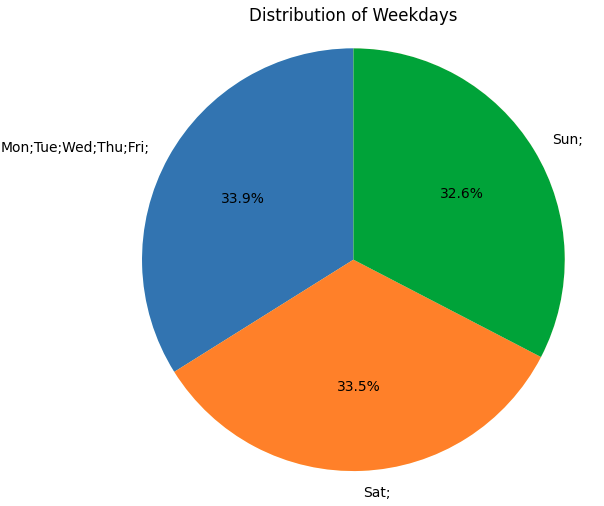
plt.pie(ambience\_counts, labels=ambience\_counts.index, autopct='%1.1f%%',startangle=90)

plt.title("Distribution of Weekdays")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

df = geoplaces

# Exclude rows with the value '?'

df\_filtered = df[df['url'] != '?']

# Group the data by 'url' and 'price' and calculate the counts

url\_price\_counts = df\_filtered.groupby(['url', 'price']).size().unstack(fill\_value=0)

# Create a bar chart with hue="price"

plt.figure(figsize=(12, 6))

url\_price\_counts.plot(kind='bar', stacked=True, ax=plt.gca())

plt.xlabel("URL")

plt.ylabel("Count")

plt.title("Number of Restaurants with Each URL (Grouped by Price)")

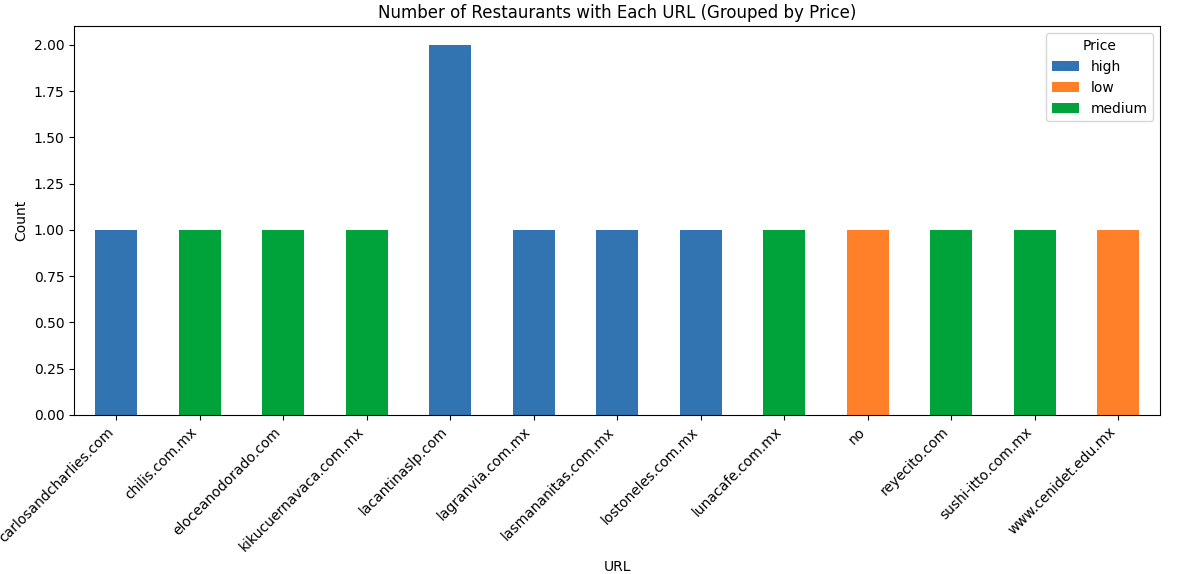
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels and align them to the right

plt.legend(title="Price", loc="upper right")  # Add legend for price categories

plt.tight\_layout()  # To prevent overlapping labels

plt.show()

**OUTPUT:**

****

df = userprofile

ambience\_counts = df['religion'].value\_counts()

plt.figure(figsize=(6,6))

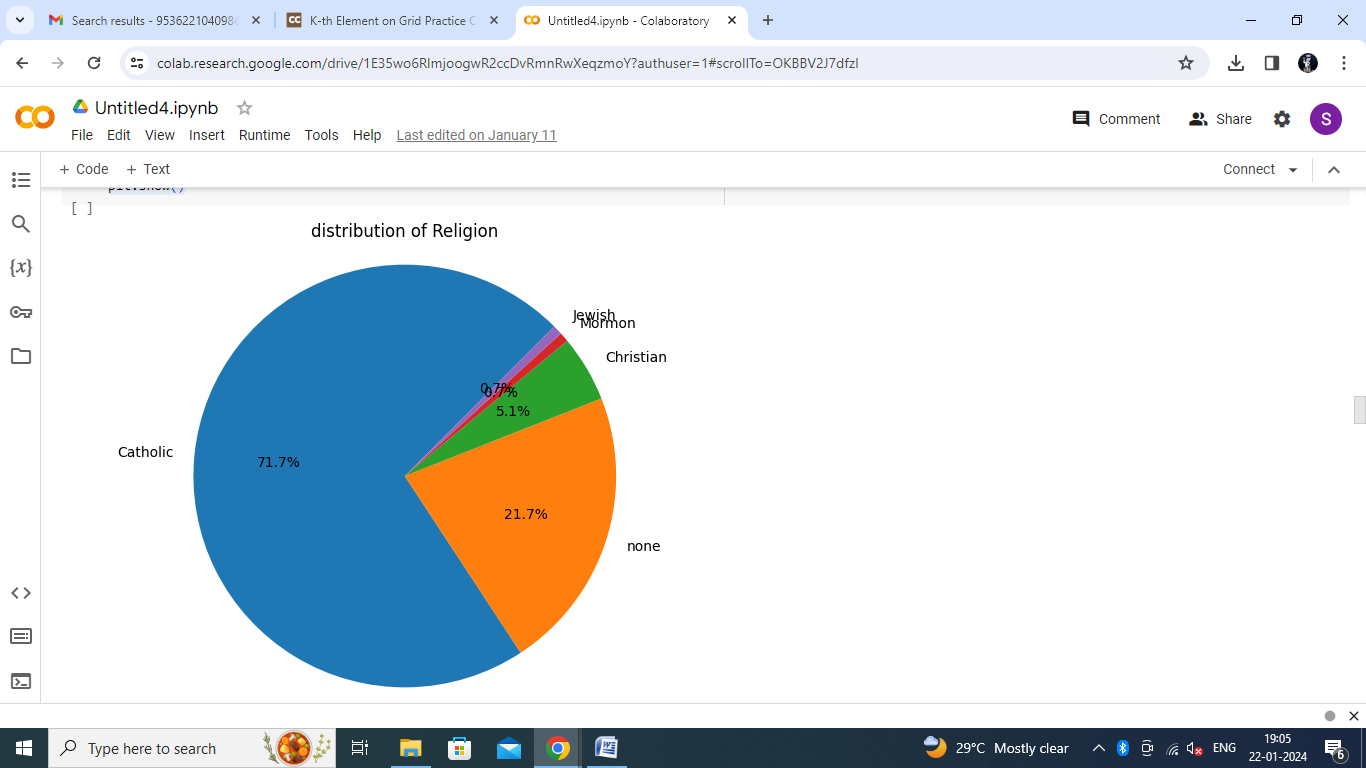
plt.pie(ambience\_counts, labels=ambience\_counts.index, autopct='%1.1f%%',startangle=45)

plt.title("distribution of Religion")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

accessibility = geoplaces['accessibility'].value\_counts()

plt.figure(figsize=(6,6))

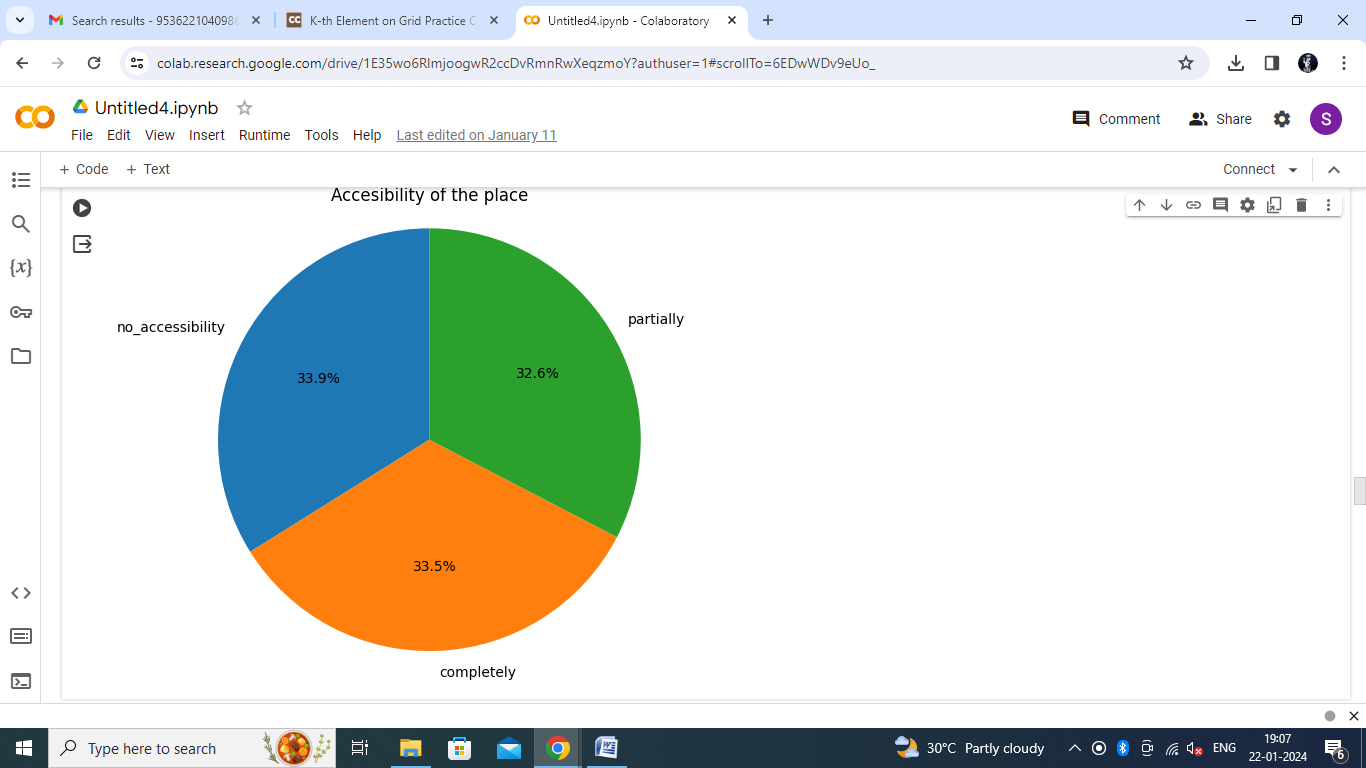
plt.pie(ambience\_counts, labels=accessibility.index, autopct='%1.1f%%',startangle=90)

plt.title("Accesibility of the place")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

df = userprofile

ambience\_counts = df['personality'].value\_counts()

plt.figure(figsize=(6,6))

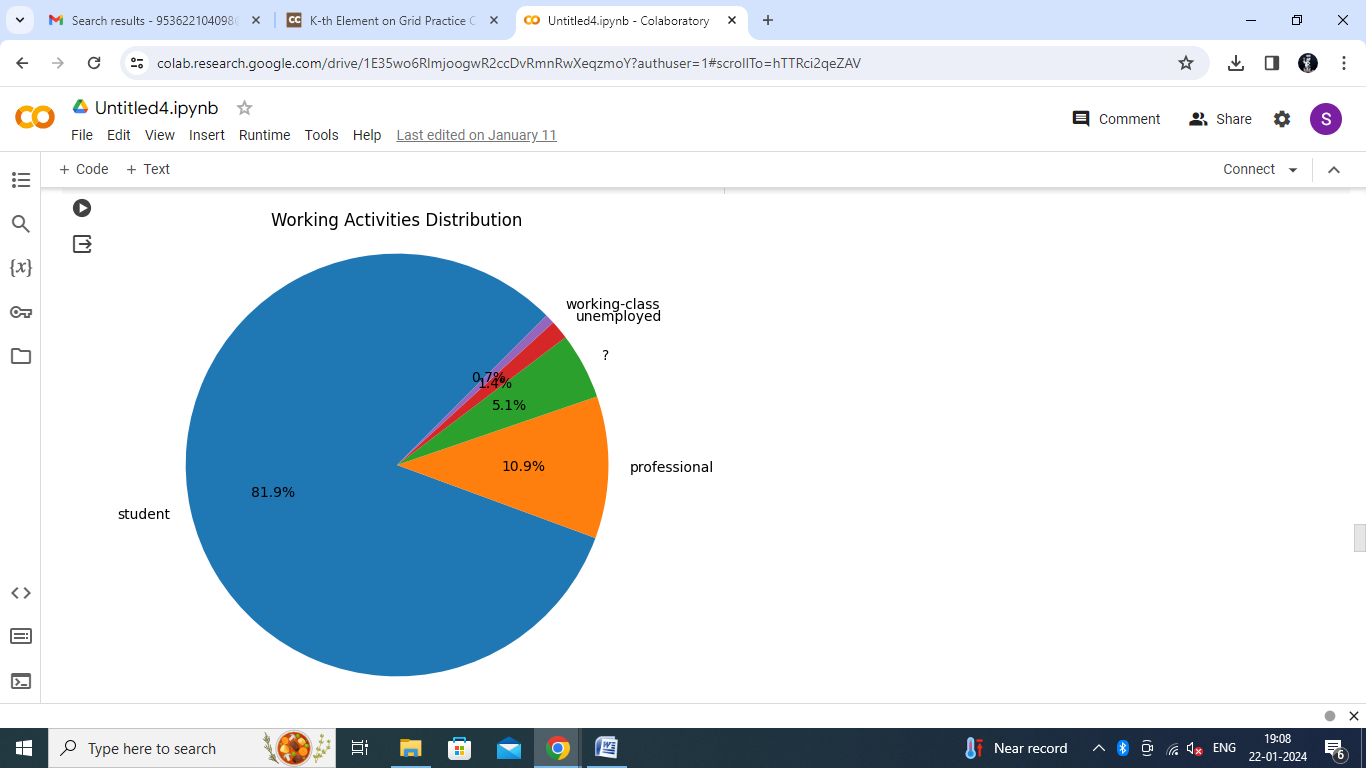
plt.pie(ambience\_counts, labels=ambience\_counts.index, autopct='%1.1f%%',startangle=45)

plt.title("Kind of Personalities")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

df = userprofile

ambience\_counts = df['interest'].value\_counts()

plt.figure(figsize=(6,6))

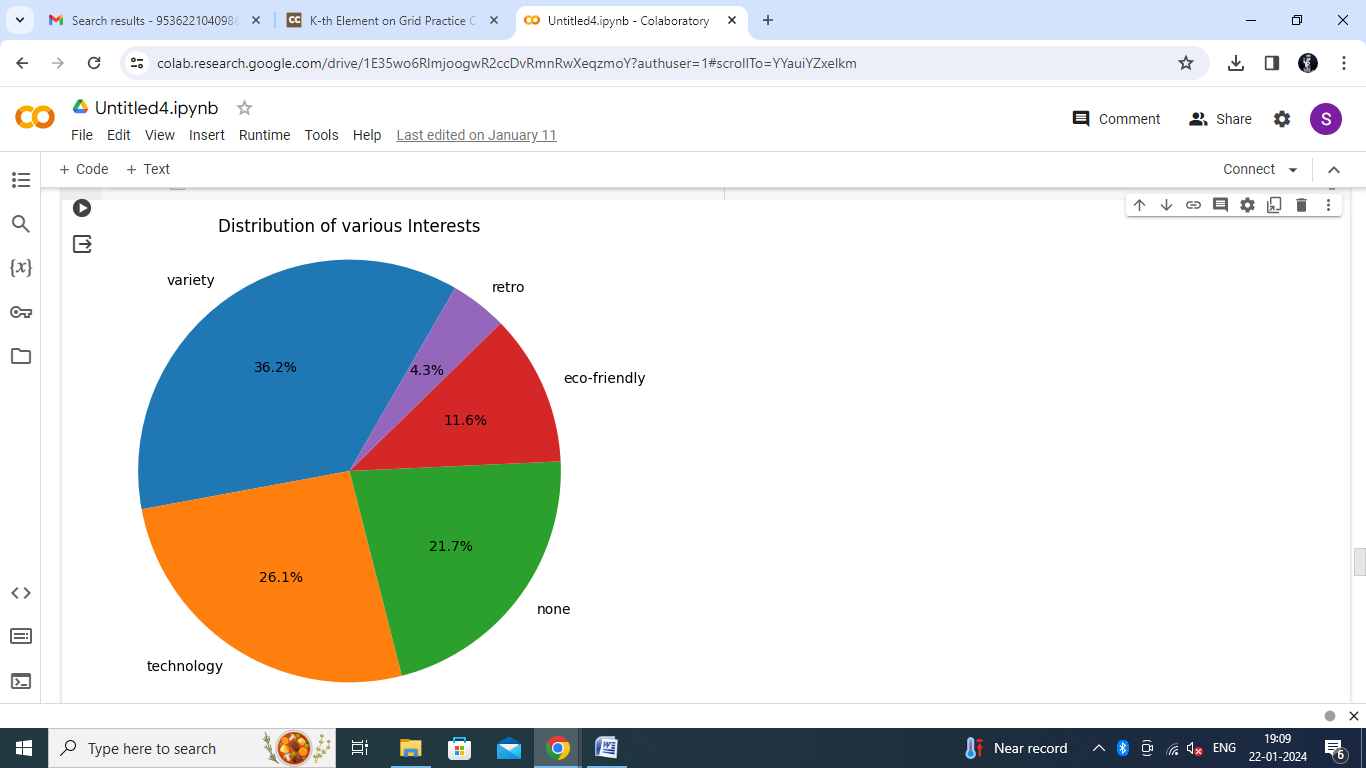
plt.pie(ambience\_counts, labels=ambience\_counts.index, autopct='%1.1f%%',startangle=60)

plt.title("Distribution of various Interests")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

df = userprofile

ambience\_counts = df['transport'].value\_counts()

plt.figure(figsize=(6,6))

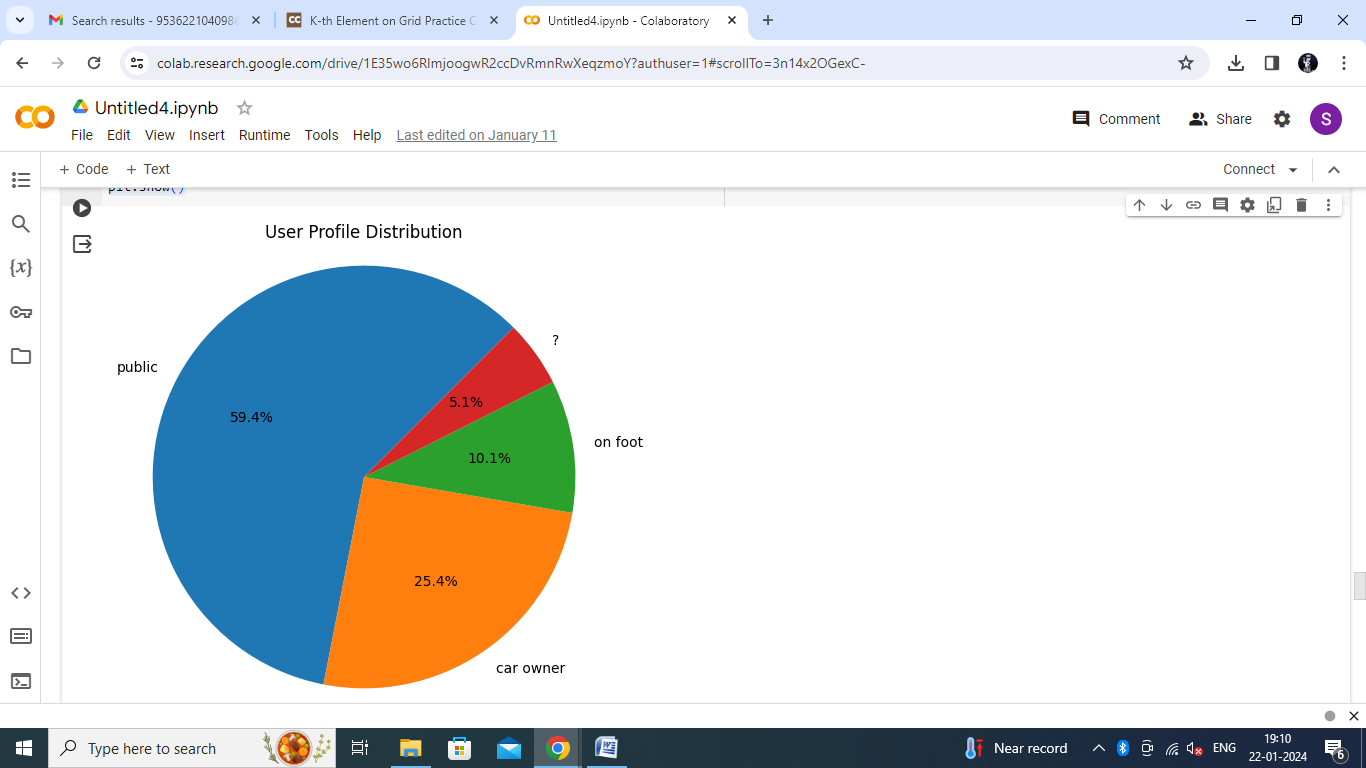
plt.pie(ambience\_counts, labels=ambience\_counts.index, autopct='%1.1f%%',startangle=45)

plt.title("User Profile Distribution")

plt.axis('equal')

plt.show()

**OUTPUT:**

****

**5.CONCLUSION**

The restaurant review system presented in this project aims to revolutionize the way users discover and choose restaurants for dining out. By providing personalized recommendations based on user preferences and past experiences, the system can enhance user satisfaction and enrich the overall dining experience. With further development and refinement, this system has the potential to become a valuable tool for both users and restaurants in the food industry.

**6.FUTURE WORK**

Future work on the restaurant review system could include:Integration with real-time data sources to provide up-to-date restaurant recommendations.Incorporation of user feedback mechanisms to continuously improve recommendation accuracy.Expansion of the system to include additional features such as restaurant reservations, menu browsing, and user reviews aggregation.

**7.REFERENCE**

WEBSITES:

<https://www.researchgate.net/publication/335873646_Restaurants_Rating_Prediction_using_Machine_Learning_Algorithms>

<https://www.ijcrt.org/papers/IJCRT2304585.pdf>

<https://www.irjmets.com/uploadedfiles/paper/volume3/issue_4_april_2021/9088/1628083371.pdf>

|  |  |
| --- | --- |
| **PERFORMANCE** |  |
| **VIVAVOCE** |  |
| **MINI PROJECT** |  |
| **TOTAL** |  |