Bite into Data: Unveiling Food Preferences In Uttar Pradesh

A BUSINESS ANALYTICS PROJECT REPORT

Submitted to

DR. PIYUSH CHAUHAN
Associate Professor
Department of Computer Science & Engineering
Symbiosis Institute of Technology, Nagpur Campus

Submitted by

ARUSHI SHIVHARE
PRN: 22070521062
Department of Computer Science & Engineering
Symbiosis Institute of Technology, Nagpur Campus

Under the Guidance of

AMIT MAKODE SIR

Course Name: Business Analytics
Course Code: T2228
VI SEM

SYMBIOSIS
INSTITUTE OF TECHNOLOGY, NAGPUR

Table of Contents

[1] Project Definition and Data Understanding :-

Uttar Pradesh cuisine is rich and diverse, influenced by its history and geography. Dishes range from the delicate Roomali Roti to the hearty Bedmi Puri and flavorful Awadhi specialties. Key ingredients include wheat flour, lentils, vegetables, and a variety of spices.

The cuisine offers both vegetarian and non-vegetarian options, with dishes like Paneer Kundan Kaliyan and Murgh Musallam showcasing its breadth. Street food like Chaat is also very popular. The document provides details on specific dishes, their ingredients, nutritional content, and origins, highlighting the cultural significance of food in Uttar Pradesh.

Uttar Pradesh Famous Cuisines (Vegetarian)									
Food item	Ingredients	Nutrients	Description	Youtube Stats					
				Source	Video Link	Views	Likes	Comments	Total Response
Roomali Roti	Wheat flour, all-purpose flour, salt, oil, and water.	Carbohydrates: High (due to refined flour) Protein: Moderate Calcium: Low Vitamins: B-complex (from wheat flour) Iron: Low	A soft, thin, and large flatbread, "Roomali" translates to"handkerchief," reflecting its delicate texture. It's traditionally cooked on the convex side of a tawa (griddle) and pairs well with rich gravies and kebabs.	https://www.subbuskitchen.com/roomali-roti-recipe-rumali-roti-recipe/	https://youtu.be/M9Ymbiu5NqM?si=22ydfy_X6NYHsi7	14000000	33000	3808	36808
Arhar ki Dal	Pigeon peas (arhar dal), tomatoes, turmeric, cumin seeds, garlic, ginger, green chilies, and ghee.	Protein: High (plant-based protein source) Fiber: High (good for digestion) Calcium: Moderate Vitamins: B-complex, Vitamin C (from tomatoes) Iron: High	A staple lentil dish, this dal is cooked until soft and tempered with spices in ghee, resulting in a comforting and nutritious accompaniment to rice or roti.	https://www.whiskaffair.com/toor-dal-recipe/	https://youtu.be/Dz2R6ol1GJ8?si=JT7-xGkZYAz00Vgr	32000000	168000	3408	171406
Fara	Rice flour or wheat flour, chana dal (split chickpeas), cumin seeds, ginger, green chilies, and coriander leaves.	Protein: High (from dal) Fiber: High Calcium: Moderate Vitamins: B-complex (from lentils), Vitamin C Iron: High	Steamed dumplings made from a dough filled with spiced lentil mixture. Fara is a healthy snack or breakfast option, often enjoyed with chutney.	https://www.archanaskitchen.com/dal-fara-recipe-indian-stuffed-lentil-dumplings	https://youtu.be/sDjThd_UuFo?si=oTxpWQsZ5oPRbpui	6300000	73000	4314	77314
Matar Ka Nimaona	Green peas, potatoes, tomatoes, cumin seeds, ginger, garlic, green chilies, and fresh coriander.	Protein: Moderate (from peas) Fiber: High Calcium: Low Vitamins: A, C, K (from peas and coriander) Iron: Moderate	A winter delicacy, this is a spiced curry made from mashed green peas, offering rich and hearty flavor, typically enjoyed with rice or roti.	https://cookingwithsapana.com/matar-ka-nimona/	https://youtu.be/DrKy6T5HUvs?si=lblv97QRIEGi7ya_	2200000	N/A	2085	2085
Bedmi Puri and Aloo Sabzi	For Bedmi Puri: wheat flour, urad dal (black gram), fennel seeds, and spices. For Aloo Sabzi: potatoes, tomatoes, turmeric, cumin seeds, and asafoetida.	Carbohydrates: High (from deep-fried puris) Protein: Moderate Fiber: Low (in puri), moderate (in aloo sabzi) Vitamins: B-complex , Vitamin C Iron: Moderate	Bedmi Puri are deep-fried breads stuffed with a spiced urad dal mixture, served with a tangy and spicy potato curry, making for a hearty breakfast or brunch.	https://www.vegrecipesofindia.com/bedmi-puri-recipe/	https://youtu.be/yg8ty3J3zzs?si=YbmbdprwbK8xr8V	4600000	59000	1569	60569
Baati Chokha	For Baati: whole wheat flour, ghee, and baking powder. For Chokha: roasted eggplant, tomatoes, potatoes, green chilies, and mustard oil.	Protein: Moderate Fiber: High (fwhole wheat and roasted brinjal) Calcium: Moderate Vitamins: B-complex, Vitamin C (from tomatoes) Iron: High	Baati are baked, round wheat flour balls , often dipped in ghee. Chokha is a mashed mixture of roasted vegetables with spices. Together, they form a rustic and flavorful meal.	https://tastycooking.recipes/varanasi-ki-baati-chokha/	https://youtu.be/UU8mOns0Nkg?si=UY60X4xW7BWDF8AH	25000000	177000	4899	181899
Bhindi ka Salan	Okra (bhindi), peanuts, sesame seeds, coconut, tamarind pulp, onions, and spices.	Protein: Moderate Fiber: High (from okra) Calcium: Moderate Vitamins: A, C, K (from okra and spices) Iron: High	A tangy and spicy curry where okra is simmered in a rich, nutty gravy, typically served with rice or flatbreads.	https://www.vegrecipesofindia.com/bhindi-salan-recipe/	https://youtu.be/mTRjdl4NlrU?si=P7AGeknjOrCDkAQ-	165000	2000	83	2083

1.1 Business Problem Statement

This project wants to understand what kinds of food people in and around Uttar Pradesh, India, like to eat. It looks at how things like their culture, what they prefer to eat (like vegetarian or non-vegetarian), and how old they are affect their food choices. The goal is to help restaurants and food businesses know what people like so they can make better decisions about their food and how to attract customers who are traveling or eating out.

```
In [ ]: print("1. Project Definition and Data Understanding\n")
```

```
1. Project Definition and Data Understanding
```

```python
# Import required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings("ignore")
```

1.2 Key Questions

- The study seeks to answer the following:

- **What food do people who visit Uttar Pradesh like to eat the most?** This means figuring out which kinds of food or specific dishes travelers enjoy when they come to UP.
- **Does knowing about the special local foods of UP make people want to travel there?** This looks at whether people are more likely to visit UP if they know about its unique food.
- **How do things like age, money, and where people come from affect what food they like and how often they visit UP?** This explores if different groups of people have different food tastes and if that changes how often they travel to Uttar Pradesh.

1.3 Data Sources

- To understand the food preferences related to travel in and around Uttar Pradesh, we gathered information directly from various people. We used a standard set of questions that covered things like their age, gender, where they live, what they usually eat, if they've traveled before, the kinds of food they enjoy, and if they know about the special dishes of UP. By collecting these details from a diverse group, we aim to get a clear picture of the different food choices and travel habits.

In [ ]:
```python
df = pd.read_csv("/content/_Cuisines of Uttar Pradesh - Awareness Survey_  (Respor
```

In [ ]:
```python
csv_path = "/mnt/data/_Cuisines of Uttar Pradesh - Awareness Survey_  (Responses)
```

In [ ]:
```python
# Display basic info
print(df.info())
print("\nSample Data:\n", df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 13 columns):
 #   Column                                                                                Non-Null Count  Dtype
---  ------                                                                                --------------  -----
 0   Timestamp                                                                             121 non-null    object
 1   Full Name                                                                             121 non-null    object
 2     Age Group                                                                           121 non-null    object
 3   State of Residence                                                                    121 non-null    object
 4   Have you ever visited Uttar Pradesh?                                                  121 non-null    object
 5   What is your dietary preference?                                                      121 non-null    object
 6   Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Vegetarian)       118 non-null    object
 7   Which of the following Uttar Pradesh's dishes do you know about or have you tried?  (Non-Vegetarian)   77 non-null     object
 8   Would you like to visit Uttar Pradesh in future? (rate:1-5)                           121 non-null    int64
 9   For what purpose would you like to visit Uttar Pradesh                                103 non-null    object
 10  What kind of stay would to prefer in Uttar Pradesh?                                   103 non-null    object
 11  What kind of food would you like to try in Uttar Pradesh?                             103 non-null    object
 12  Below are the top 10 famous cuisines of Uttar Pradesh, choose that which of the dishes would you try.  103 non-null    object
dtypes: int64(1), object(12)
memory usage: 12.4+ KB
None

Sample Data:
            Timestamp          Full Name   Age Group   State of Residence  \
0  2/5/2025 20:56:40    Arushi Shivhare         18-25       Maharashtra
...
1                                          NaN
2  Lucknawi Biryani (Lucknow), Baati Chokha (Vara...
3  Lucknawi Biryani (Lucknow), Bedai & Jalebi (Ag...
4  Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanpu...
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

1.4 Data Dictionary

- Each variable collected was defined with:

- **Name**: Variable name used in the dataset.
- **Type**: Categorical or numerical.
- **Description**: Explanation of what the variable represents.
- **Encoding Scheme**: For categorical variables, the numerical representation used.

```
Data Dictionary:

                                                                                                  Column Name
Timestamp                                                                                          Timestamp
Full Name                                                                                          Full Name
  Age Group                                                                                       Age Group
State of Residence                                                                             State of Residence
Have you ever visited Uttar Pradesh?                                                 Have you ever visited Uttar Pradesh?
What is your dietary preference?                                                        What is your dietary preference?
Which of the following Uttar Pradesh's dishes d...  Which of the following Uttar Pradesh's dishes ...
Which of the following Uttar Pradesh's dishes d...  Which of the following Uttar Pradesh's dishes ...
Would you like to visit Uttar Pradesh in future...  Would you like to visit Uttar Pradesh in futur...
For what purpose would you like to visit Uttar ...  For what purpose would you like to visit Uttar...
What kind of stay would to prefer in Uttar Prad...  What kind of stay would to prefer in Uttar Pra...
What kind of food would you like to try in Utta...  What kind of food would you like to try in Utt...
Below are the top 10 famous cuisines of Uttar P...  Below are the top 10 famous cuisines of Uttar ...

                                                    Data Type  Missing Values  \
Timestamp                                              object               0
Full Name                                              object               0
  Age Group                                            object               0
State of Residence                                     object               0
Have you ever visited Uttar Pradesh?                   object               0
What is your dietary preference?                       object               0
Which of the following Uttar Pradesh's dishes d...     object               3
...
For what purpose would you like to visit Uttar ...             20
What kind of stay would to prefer in Uttar Prad...             36
What kind of food would you like to try in Utta...             22
Below are the top 10 famous cuisines of Uttar P...             68
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
In [ ]: # Data dictionary creation
data_dict = pd.DataFrame({
 "Column Name": df.columns,
 "Data Type": df.dtypes,
 "Missing Values": df.isnull().sum(),
 "Unique Values": df.nunique()
})
print("\nData Dictionary:\n", data_dict)
```

[2] Data Collection and Integration :-

2.1 Data Source and Collection

So, we created a survey on Google Forms, which is like an online questionnaire. Many different people filled it out, giving us their answers to our questions about food and travel. Once we had all the responses, we took that information and put it into a CSV file. Think of a CSV file like a simple spreadsheet where all the answers are organized in rows and columns. This way, we had a big collection of data that included responses from people of all different age groups – young, old, and in between – and also from people living in various different regions. This variety in the people who answered helps us get a broader understanding of food preferences.

```
In [ ]: print("\n2. Data Collection and Integration\n")
```

```
2. Data Collection and Integration
```

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
Timestam	Full Name	Age Grou	State of Re	Have you	What is yc	Which of t	Which of t	Would yo	For what p	What kind	What kind	Below are the top 10 famous cuisines of Uttar Pradesh, choose that which						
########	Arushi Shi	18-25	Maharashi	Yes	Non-Veget	Roomali R	Shami Kab	5	Religious T	Budget & I	Awadhi Cu	Tunday Kababi (Lucknow), Bedai & Jalebi (Agra, Mathura), Aloo Tikki Chaat						
########	Akash Sha	18-25	Maharashi	No	Vegetariar	Roomali R	None	2										
########	Alankrita F	18-25	Maharashi	No	Non-Veget	Roomali R	None	4	Heritage &	Budget & I	Awadhi Cu	Lucknawi Biryani (Lucknow), Baati Chokha (Varanasi, Prayagraj), Aloo Tikki						
########	Tanushree	18-25	Maharashi	No	Non-Veget	Roomali R	Murgh Do	4	Cultural &	Budget & I	Awadhi Cu	Lucknawi Biryani (Lucknow), Bedai & Jalebi (Agra, Mathura), Aloo Tikki Cha						
########	Gautam St	18-25	Maharashi	No	Vegetariar	Roomali R	None	4	Religious T	Budget & I	Vegetariar	Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanpur, Varanasi), Banarasi Paan						
########	Anmol Ch	18-25	Surendra r	Yes	Non-Veget	Roomali R	Shami Kab	5	Religious T	Budget & I	Awadhi Cu	Lucknawi Biryani (Lucknow), Petha (Agra), Baati Chokha (Varanasi, Prayagra						
########	Deep Gupt	18-25	MAHARAS	No	Non-Veget	Roomali R	Prawn Cur	1										
########	ABHISHEK	18-25	Maharashi	Yes	Non-Veget	Arhar ki D	Awadhi Gc	4	Wildlife &	Budget & I	Awadhi Cu	Tunday Kababi (Lucknow), Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanpur						
########	Ved Bisne	18-25	Maharashi	Yes	Non-Veget	Roomali R	Shami Kab	4	Religious T	Luxury Ho	Street Foo	Lucknawi Biryani (Lucknow), Bedai & Jalebi (Agra, Mathura), Petha (Agra), E						
########	Nakshatra	18-25	Maharashi	No	Vegetariar	None	None	3	Religious T	Budget & I	Street Foo	Aloo Tikki Chaat (Lucknow, Kanpur, Varanasi)						
########	Abhijay Ta	18-25	Maharashi	No	Non-Veget	Roomali R	Prawn Cur	3	Wildlife &	Luxury Ho	Mughlai C	Tunday Kababi (Lucknow), Lucknawi Biryani (Lucknow), Petha (Agra), Aloo						
########	Akshat Du	18-25	Maharashi	Yes	Non-Veget	Roomali R	Kakori Kat	5	Religious T	Budget & I	Awadhi Cu	Tunday Kababi (Lucknow), Bedai & Jalebi (Agra, Mathura), Malaiyyo (Varan						
########	Atharva Al	18-25	Maharashi	No	Non-Veget	Roomali R	Shami Kab	3	Religious T	Budget & I	Awadhi Cu	Lucknawi Biryani (Lucknow), Baati Chokha (Varanasi, Prayagraj), Aloo Tikki						
########	Shreyas Ka	18-25	Maharashi	Yes	Eggetarian	Roomali R	Galouti Ke	5	Religious T	Luxury Ho	Awadhi Cu	Bedai & Jalebi (Agra, Mathura), Petha (Agra), Baati Chokha (Varanasi, Praya						
########	Pranav Chi	18-25	Maharashi	No	Vegetariar	Roomali R	None	3	Heritage &	Luxury Ho	Awadhi Cu	Lucknawi Biryani (Lucknow), Petha (Agra), Baati Chokha (Varanasi, Prayagra						
########	Sangeet	18-25	Maharashi	Yes	Vegetariar	Baati Chok	None	3	Religious T	Governme	Vegetariar	Baati Chokha (Varanasi, Prayagraj)						
########	Jash Chaul	18-25	Maharashi	Yes	Vegetariar	Roomali R	Shami Kab	5	Religious T	Budget & I	Street Foo	Bedai & Jalebi (Agra, Mathura), Petha (Agra), Baati Chokha (Varanasi, Praya						
########	Anushri Ac	18-25	Maharashi	Yes	Non-Veget	Roomali R	Shami Kab	2										
########	Raghav Up	18-25	Maharashi	Yes	Eggetarian	Roomali R	Shami Kab	5	Religious T	Budget & I	Awadhi Cu	Tunday Kababi (Lucknow), Lucknawi Biryani (Lucknow), Petha (Agra), Aloo						
########	Ruchika Ka	18-25	Maharashi	Yes	Non-Veget	Roomali R	Shami Kab	5	Religious T	Dharamsh	Street Foo	Petha (Agra), Baati Chokha (Varanasi, Prayagraj), Aloo Tikki Chaat (Lucknow						
########	Saloni Mal	18-25	Maharashi	Yes	Vegetariar	Roomali R	None	3	Religious T	Budget & I	Street Foo	Baati Chokha (Varanasi, Prayagraj)						
########	Seshank ra	18-25	UP	Yes	Non-Veget	Roomali R	Shami Kab	5	Religious T	Budget & I	Awadhi Cu	Tunday Kababi (Lucknow), Lucknawi Biryani (Lucknow), Bedai & Jalebi (Agr						
########	Aryan Par	18-25	Maharashi	No	Vegetariar	Roomali R	None	3	Heritage &	Budget & I	Street Foo	Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanpur, Varanasi), Banarasi Paan						
########	Dhanashre	18-25	Maharashi	No	Eggetarian	Roomali R	None	5	Religious T	Budget & I	Street Foo	Lucknawi Biryani (Lucknow), Bedai & Jalebi (Agra, Mathura), Petha (Agra), A						
########	Aryan Mis	18-25	uttar prad	Yes	Vegetariar	Roomali R	Shami Kab	5	Religious T	Luxury Ho	Awadhi Cu	Baati Chokha (Varanasi, Prayagraj), Aloo Tikki Chaat (Lucknow, Kanpur, Var						
########	Shrishti Pr	18-25	Maharashi	No	Vegetariar	Roomali R	Awadhi Gc	5	Religious T	Budget & I	Awadhi Cu	Bedai & Jalebi (Agra, Mathura), Petha (Agra), Baati Chokha (Varanasi, Praya						
########	Arunima D	18-25	Maharashi	No	Non-Veget	Roomali R	Prawn Cur	3	Heritage &	Budget & I	Mughlai C	Lucknawi Biryani (Lucknow), Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanp						
########	Jay Vijay Y	18-25	Maharashi	No	Vegetariar	Roomali R	None	4	Religious T	Budget & I	Awadhi Cu	Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanpur, Varanasi), Malaiyyo (Vara						
########	Kashish gu	18-25	Maharashi	Yes	Non-Veget	Roomali R	Awadhi Gc	5	Religious T	Homestay	Awadhi Cu	Tunday Kababi (Lucknow), Lucknawi Biryani (Lucknow), Baati Chokha (Vara						

2.2 Data Provenance

- To make sure the information we collected was real and trustworthy:

- Every time someone answered the survey, we recorded the exact time they did it. This helps us see when responses came in.
- We also checked the internet address (IP address) of each person who filled out the form. This helped us make sure that people weren't filling out the survey multiple times from the same device.
- After we collected all the answers, we went through the data and cleaned it up. This meant getting rid of any duplicate responses, so we only counted each person's answers once. These steps help us be more confident that our data is accurate and not tampered with.

2.3 Integration Methodology

After we got all the survey answers, which were in those spreadsheet-like CSV files, we used a special tool called "pandas" with the programming language Python. Think of pandas as a helpful way to organize and work with data on a computer. If we had the answers in more than one CSV file, pandas helped us put them all together into one big, organized table. We used commands like pd.concat() to simply stack the files on top of each other, or pd.merge() if we needed to combine information based on matching details in the files. This gave us one single, easy-to-work-with dataset containing all the responses.

```
In [ ]: # Already collected from Google Form (CSV)
        print("Data source: Google Form CSV export")
        print("Number of rows and columns:", df.shape)
```

```
Data source: Google Form CSV export
Number of rows and columns: (121, 13)
```

2.4 Initial Validation

- To make sure the information in our combined dataset was correct and made sense, we did a few more things:

- We checked what kind of information was in each column (like numbers or text) and made sure the values were within expected limits. For example, age shouldn't be a negative number or a ridiculously high value.
- We looked for any missing answers or strange, unusual values that didn't seem right. This helps us spot potential errors in the data.
- We also did some basic calculations, like finding averages and counts, to get a first look at the general trends in the responses. This gives us an initial understanding of the data before we do more detailed analysis.

[3] Data Cleaning and Preparation :-

3.1 Check Shape, Data Types, and Null Values

- We used specific computer commands to take a closer look at our dataset and make sure everything was in order.

- df.shape told us how big our dataset was by showing us the number of rows (representing each person's answers) and the number of columns (representing each question we asked).
- df.info() gave us a summary of the dataset, including the type of data in each column (like numbers or text) and if there were any missing values.
- df.isnull().sum() specifically showed us how many missing answers there were in each column.

These commands helped us understand the basic structure of our data and quickly identify any potential problems like missing information.

In [ ]:
```
# Dataset shape
print("Dataset shape:",df.shape)
# Data types and  nulls
print("\nData types and missing values :")
print(df.info())
# Summary o f  n u l l  value s
print("\n Missing values in each column:")
print(df.isnull().sum())
```

```
Dataset shape: (121, 13)

Data types and missing values :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 13 columns):
 #   Column                                                                  Non-Null Count  Dtype
---  ------                                                                  --------------  -----
 0   Timestamp                                                               121 non-null    object
 1   Full Name                                                               121 non-null    object
 2     Age Group                                                             121 non-null    object
 3   State of Residence                                                      121 non-null    object
 4   Have you ever visited Uttar Pradesh?                                    121 non-null    object
 5   What is your dietary preference?                                        121 non-null    object
 6   Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Vegetarian)       118 non-null    object
 7   Which of the following Uttar Pradesh's dishes do you know about or have you tried?  (Non-Vegetarian)  77 non-null     object
 8   Would you like to visit Uttar Pradesh in future? (rate:1-5)             121 non-null    int64
 9   For what purpose would you like to visit Uttar Pradesh                  103 non-null    object
 10  What kind of stay would to prefer in Uttar Pradesh?                     103 non-null    object
 11  What kind of food would you like to try in Uttar Pradesh?               103 non-null    object
 12  Below are the top 10 famous cuisines of Uttar Pradesh, choose that which of the dishes would you try.  103 non-null    object
dtypes: int64(1), object(12)
memory usage: 12.4+ KB
None
...
What kind of stay would to prefer in Uttar Pradesh?                                   18
What kind of food would you like to try in Uttar Pradesh?                             18
Below are the top 10 famous cuisines of Uttar Pradesh, choose that which of the dishes would you try.    18
dtype: int64
```

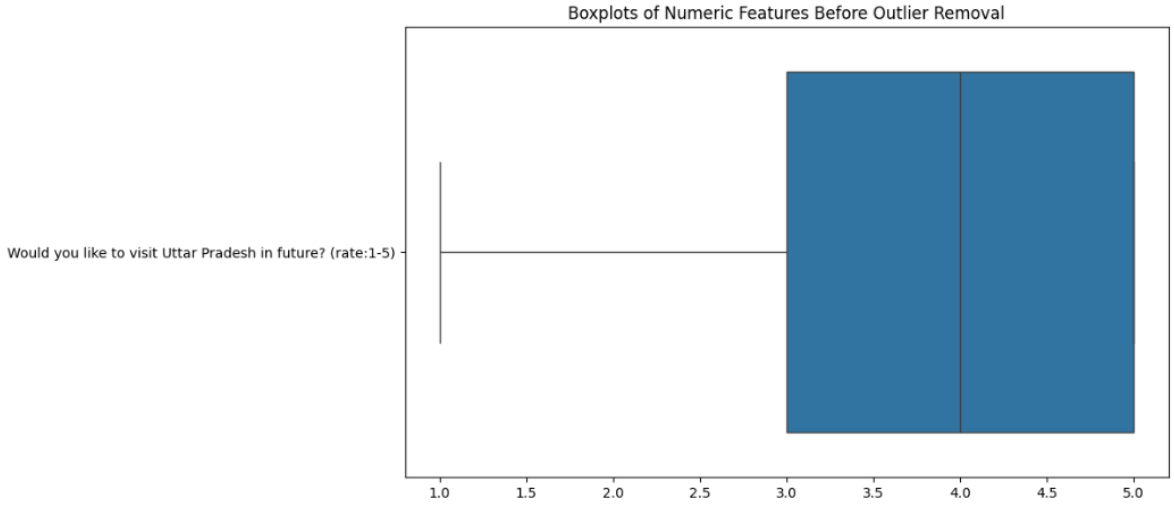
3.2 Outlier Treatment

To find any really unusual answers in our data, we used a type of visual chart called a "boxplot." Boxplots help us see the typical range of answers and easily spot any values that are far outside that range – these are what we call "extreme values" or "outliers."

In [ ]:
```python
# Handle outliers: Cap using z-score
numeric_cols = df.select_dtypes(include=np.number).columns
for col in numeric_cols:
 z_scores = np.abs(stats.zscore(df[col]))
 df = df[(z_scores < 3)]
```

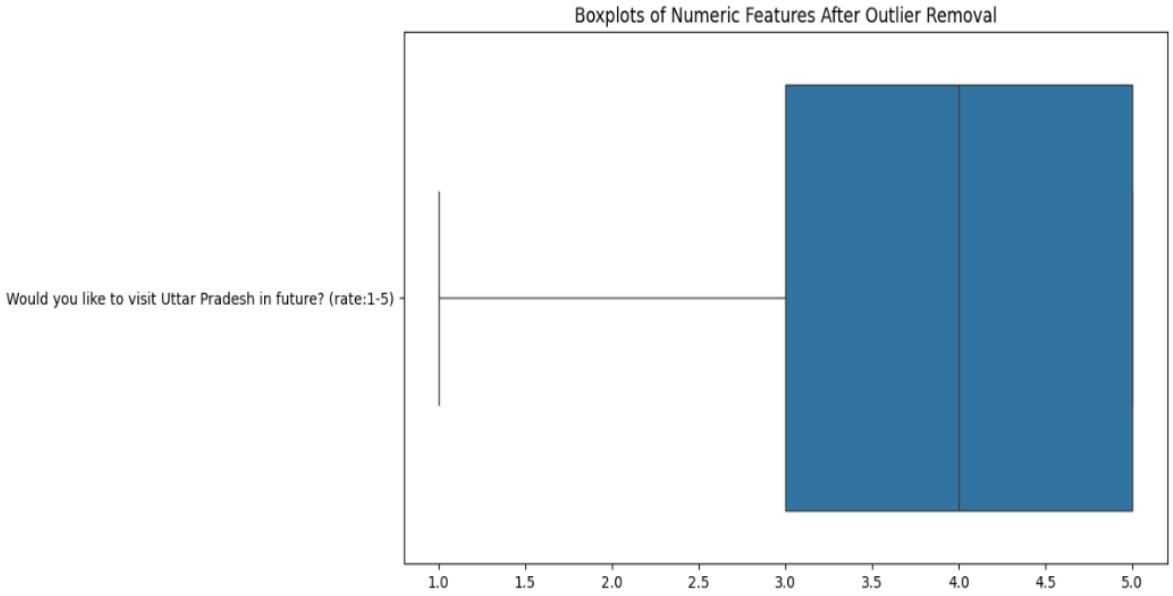
In [ ]:
```python
#Address Outliers:-
# Describe the DataFrame
print("\nInitial DataFrame Description:")
print(df.describe())

#Show boxplots for numeric columns *before* outlier removal
numeric_df = df.select_dtypes(include=['number'])  # Select only numeric columns
if not numeric_df.empty:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=numeric_df, orient='h')
    plt.title("Boxplots of Numeric Features Before Outlier Removal")
    plt.show()
else:
    print("No numeric columns to show boxplots before outlier removal.")

#Remove outliers using the IQR method for all numeric columns
numeric_cols = df.select_dtypes(include=['number']).columns  # Get numeric column
for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]

#Display info about the DataFrame *after* outlier removal
print("\nDataFrame Info After Outlier Removal:")
df.info()

#Describe the DataFrame *after* outlier removal
print("\nDataFrame Description After Outlier Removal:")
print(df.describe())

#Show boxplots for numeric columns *after* outlier removal
numeric_df = df.select_dtypes(include=['number'])  # Re-select numeric columns aft
if not numeric_df.empty:
    plt.figure(figsize=(10, 6))
    sns.boxplot(data=numeric_df, orient='h')
    plt.title("Boxplots of Numeric Features After Outlier Removal")
    plt.show()
else:
    print("No numeric columns to show boxplots after outlier removal.")
```

```
Initial DataFrame Description:
       Would you like to visit Uttar Pradesh in future? (rate:1-5)
count                                          121.000000
mean                                             3.685950
std                                              1.245211
min                                              1.000000
25%                                              3.000000
50%                                              4.000000
75%                                              5.000000
max                                              5.000000
```



Once we found these outliers, we used a method called the "Interquartile Range" (IQR). This is a way to mathematically define what counts as an outlier based on the middle 50% of the data. After identifying the outliers using the IQR method, we either adjusted these extreme values to be within a reasonable range ("capping") or we removed them from our dataset entirely. This step helps to make sure that these unusual responses don't unfairly skew our overall analysis and give us a more accurate understanding of the general trends.

```
DataFrame Info After Outlier Removal:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 121 entries, 0 to 120
Data columns (total 13 columns):
 #   Column                                                                                Non-Null Count  Dtype
---  ------                                                                                --------------  -----
 0   Timestamp                                                                             121 non-null    object
 1   Full Name                                                                             121 non-null    object
 2     Age Group                                                                           121 non-null    object
 3   State of Residence                                                                    121 non-null    object
 4   Have you ever visited Uttar Pradesh?                                                  121 non-null    object
 5   What is your dietary preference?                                                      121 non-null    object
 6   Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Vegetarian)       118 non-null    object
 7   Which of the following Uttar Pradesh's dishes do you know about or have you tried?  (Non-Vegetarian)   77 non-null     object
 8   Would you like to visit Uttar Pradesh in future? (rate:1-5)                           121 non-null    int64
 9   For what purpose would you like to visit Uttar Pradesh                                103 non-null    object
 10  What kind of stay would to prefer in Uttar Pradesh?                                   103 non-null    object
 11  What kind of food would you like to try in Uttar Pradesh?                             103 non-null    object
 12  Below are the top 10 famous cuisines of Uttar Pradesh, choose that which of the dishes would you try.  103 non-null    object
dtypes: int64(1), object(12)
memory usage: 12.4+ KB

DataFrame Description After Outlier Removal:
       Would you like to visit Uttar Pradesh in future? (rate:1-5)
...
25%                                          3.000000
50%                                          4.000000
75%                                          5.000000
max                                          5.000000
```



3.3 Data Type Corrections

Sometimes, when we collect data, the computer might not correctly guess what type of information is in each column. For example, a column that should contain numbers for ratings or ages might accidentally be stored as text (which computers call "strings"). This can cause problems when we try to do calculations or analysis.

To fix this, we used a function called astype(). This function allows us to tell the computer the correct type of data for each column. So, if we found a column where ages were being treated as text, we used astype() to change them into numbers. This ensures that we can properly analyze and work with all the information in our dataset.

```
In [ ]: # Convert data types if needed
        df = df.convert_dtypes()
```

```
In [ ]: # Convert the 'Timestamp' column to datetime, handling errors
        df['Timestamp'] = pd.to_datetime(df['Timestamp'], errors='coerce')

        # Display the data types of the DataFrame to verify the conversion
        print(df.dtypes)
```

```
Timestamp                                                                 datetime64[ns]
Full Name                                                                 string[python]
  Age Group                                                               string[python]
State of Residence                                                        string[python]
Have you ever visited Uttar Pradesh?                                      string[python]
What is your dietary preference?                                          string[python]
Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Vegetarian)    string[python]
Which of the following Uttar Pradesh's dishes do you know about or have you tried?  (Non-Vegetarian)    string[python]
Would you like to visit Uttar Pradesh in future? (rate:1-5)                       Int64
For what purpose would you like to visit Uttar Pradesh                    string[python]
What kind of stay would to prefer in Uttar Pradesh?                       string[python]
What kind of food would you like to try in Uttar Pradesh?                 string[python]
Below are the top 10 famous cuisines of Uttar Pradesh, choose that which of the dishes would you try.    string[python]
dtype: object
```

3.4 Normalize Numerical Features

To make sure different kinds of numerical data in our dataset were on a similar scale, we used techniques called "Standardization" and "Min-Max Scaling."

Imagine you have some information measured in very different ways, like one thing measured on a scale of 1 to 5, and another on a scale of 0 to 100. If we don't adjust them, it can be hard to compare them fairly.

- **Standardization** is like adjusting the numbers so they all have a mean (average) of zero and a standard deviation of one. This helps to compare data that might have different units or spreads. We used something called StandardScaler to do this.
- **Min-Max Scaling** is like squishing all the numbers into a specific range, usually between 0 and 1. This is helpful when we want to keep all the values within a certain boundary. We used MinMaxScaler for this.

By applying these scaling methods, we ensure that no single type of measurement disproportionately influences our analysis just because it happens to have larger numbers or a wider range. This helps us to have a more consistent and fair comparison of the data.

```
In [ ]: # Normalize numeric data
        scaler = StandardScaler()
```

```
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])
print("Cleaned Data:\n", df.head())
```

```
Cleaned Data:
            Timestamp          Full Name   Age Group   State of Residence  \
0 2025-02-05 20:56:40    Arushi Shivhare         18-25       Maharashtra
1 2025-02-05 21:07:14         Akash Shah         18-25        Maharashtra
2 2025-02-05 21:15:29  Alankrita Bhonde          18-25       Maharashtra
3 2025-02-05 21:35:35   Tanushree Joshi          18-25       Maharashtra
4 2025-02-05 21:36:06    Gautam Sukhani          18-25        Maharashtra

  Have you ever visited Uttar Pradesh? What is your dietary preference?   \
0                                  Yes                    Non-Vegetarian
1                                   No                       Vegetarian
2                                   No                   Non-Vegetarian
3                                   No                   Non-Vegetarian
4                                   No                       Vegetarian

  Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Vegetarian)  \
0  Roomali Roti, Arhar ki Dal, Fara, Matar Ka Nim...
1          Roomali Roti, Baati Chokha, Chaat, Kulcha
2          Roomali Roti, Baati Chokha, Chaat, Kulcha
3  Roomali Roti, Baati Chokha, Chaat, Kulcha, Pan...
4  Roomali Roti, Arhar ki Dal, Bedmi Puri and Alo...

  Which of the following Uttar Pradesh's dishes do you know about or have you tried?  (Non-Vegetarian)  \
0  Shami Kabab, Murgh Do Pyaza, Boti Kabab, Murgh...
1                                             <NA>
...
1                                             <NA>
2  Lucknawi Biryani (Lucknow), Baati Chokha (Vara...
3  Lucknawi Biryani (Lucknow), Bedai & Jalebi (Ag...
4  Petha (Agra), Aloo Tikki Chaat (Lucknow, Kanpu...
```

3.5 Encode Categorical Variables

Our data includes information that isn't numbers, but rather categories or labels, like different food types or regions. Computers work best with numbers, so we need to convert these categories into a numerical format. We used two main ways to do this:

- **Label Encoding:** We used this for categories that have a natural order (like "low," "medium," "high"). Think of it as assigning a number to each category based on its order (e.g., low could become 0, medium 1, and high 2). We used something called LabelEncoder to do this.
- **One-Hot Encoding:** For categories that don't have a specific order (like different types of cuisine: "Indian," "Italian," "Chinese"), we used "One-Hot Encoding." This creates new columns for each category. If a person chose "Indian," the "Indian" column would get a 1, and all the other cuisine columns would get a 0 for that person. We used a function called pd.get_dummies() to do this.

By encoding these categorical values into numbers, we can then include this information in our analysis and models.

```
In [ ]: # Encoding categorical columns
        label_encoders = {}
        cat_cols = df.select_dtypes(include='string').columns
        for col in cat_cols:
         le = LabelEncoder()
         df[col] = le.fit_transform(df[col].astype(str))
         label_encoders[col] = le
```

3.6 Create Derived Features

To get even more insights from our data, we created new pieces of information based on the answers people gave. For instance, we might have asked several questions about dietary choices. To simplify this, we could create a new column labeled "Vegetarian Preference." If someone answered "yes" to all the vegetarian-related questions, this new column would say "1" (or "True"). If they answered "no" to any of them, it would say "0" (or "False"). This way, we condensed multiple answers into a single, easy-to-use piece of information that could be really helpful for our analysis. We looked for other ways to combine or transform the existing answers to create new features that could reveal interesting patterns.

[4] Exploratory Data Analysis :-

4.1 Descriptive Statistics

To get a better feel for the numerical data in our dataset, we used a command called df.describe(). This is like getting a quick statistical summary for all the columns containing numbers. For each numerical column, df.describe() shows us things like:

- **Count:** How many total values are there.
- **Mean:** The average value.
- **Standard Deviation:** How spread out the values are.
- **Minimum:** The smallest value.
- **25th Percentile (Q1):** The value below which 25% of the data falls.
- **50th Percentile (Median or Q2):** The middle value.
- **75th Percentile (Q3):** The value below which 75% of the data falls.
- **Maximum:** The largest value.

This gives us a good overview of the typical values, how much the data varies, and the overall shape of the distributions for our numerical information.

```
In [ ]: print("\n4. Exploratory Data Analysis\n")
        print("Descriptive Statistics:\n", df.describe())
```

```
Descriptive Statistics:
                           Timestamp    Full Name     Age Group
count                            121  121.000000     121.000000
mean   2025-03-06 04:42:42.198347264   59.074380       0.173554
min              2025-02-05 20:56:40    0.000000       0.000000
25%              2025-02-05 23:45:16   29.000000       0.000000
50%              2025-03-24 11:11:43   59.000000       0.000000
75%              2025-03-25 15:56:44   89.000000       0.000000
max              2025-03-25 16:02:05  119.000000       3.000000
std                              NaN   34.954772       0.641323

       State of Residence  Have you ever visited Uttar Pradesh?  \
count          121.000000                            121.000000
mean             8.628099                              0.479339
min              0.000000                              0.000000
25%              7.000000                              0.000000
50%              8.000000                              0.000000
75%              8.000000                              1.000000
max             24.000000                              1.000000
std              3.862495                              0.501650

       What is your dietary preference?   \
...
50%                                28.000000
75%                                51.000000
max                                68.000000
std                                22.694061
```

4.2 Visualization

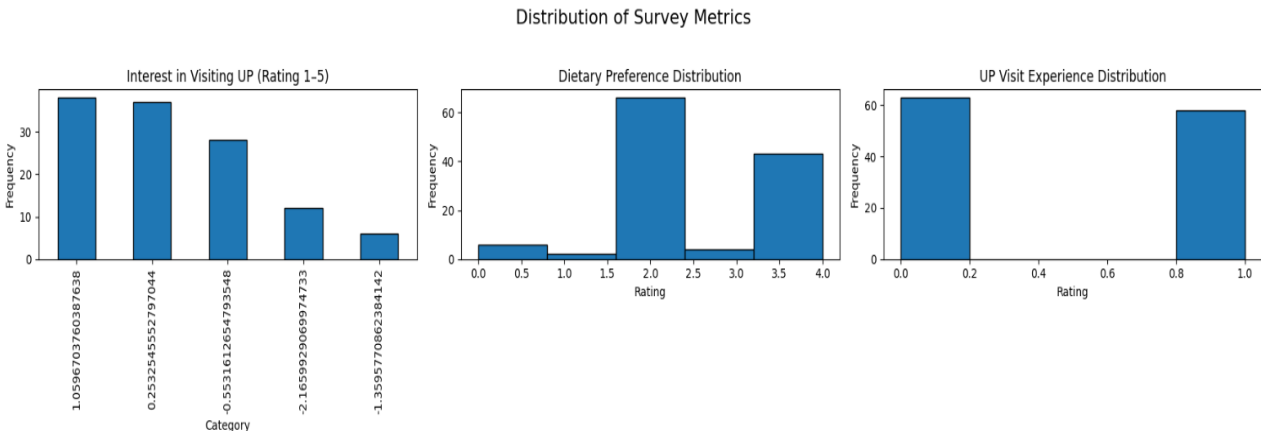
 # 4.1.1 Distribution – HISTOGRAMS

- We used a type of chart called a "histogram" to visualize how our numerical data was spread out.

- **Age Groups:** For the ages of the people who responded, the histogram showed us how many people fell into different age ranges (like 18-25, 26-35, etc.). This helped us see which age groups were most represented in our survey.
- **Visit Ratings:** If we asked people to rate their visits or experiences on a numerical scale, the histogram for these ratings showed us how many people gave each rating. This allowed us to see if most people had positive, negative, or neutral experiences, and the overall distribution of opinions.

In simple terms, histograms give us a picture of how frequently different values occur in our numerical data.

```
In [ ]: # 4.1.1 Distribution - HISTOGRAMS
        # Columns to plot
        features = [
            'Would you like to visit Uttar Pradesh in future? (rate:1-5)',
            'What is your dietary preference? ',
            'Have you ever visited Uttar Pradesh?'
        ]

        titles = [
            'Interest in Visiting UP (Rating 1-5)',
            'Dietary Preference Distribution',
            'UP Visit Experience Distribution'
        ]

        # Create subplots
        fig, axes = plt.subplots(1, 3, figsize=(18, 5))

        # Plot each feature
        for i, feature in enumerate(features):
            if df[feature].dtype == 'int64':
                axes[i].hist(df[feature], bins=5, edgecolor='black')
                axes[i].set_xlabel('Rating')
            else:
                df[feature].value_counts().plot(kind='bar', ax=axes[i], edgecolor='black'
                axes[i].set_xlabel('Category')

            axes[i].set_title(titles[i])
            axes[i].set_ylabel('Frequency')

        # Add main title
        fig.suptitle('Distribution of Survey Metrics', fontsize=16)
        plt.tight_layout(rect=[0, 0.03, 1, 0.95])
        plt.show()
```



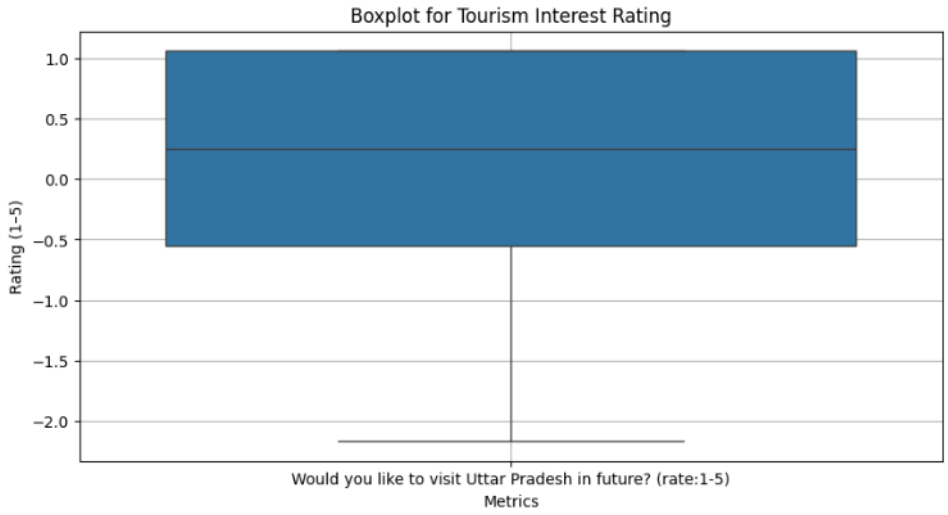
 # 4.1.2 BOXPLOTS

- We also used boxplots to compare ratings across different groups.

- **Rating Distributions Across Different Cuisines:** For example, if we asked people to rate how much they enjoyed different types of food (like North Indian, South Indian, etc.), we could create separate boxplots for each cuisine. These boxplots would visually show us the range of ratings for each type of food, the average rating, and how spread out the ratings were. This helps us easily compare which cuisines tend to get higher or lower ratings and if there's more agreement or disagreement in the ratings for different cuisines.
- **Rating Distributions Across Different Travel Reasons:** Similarly, if we asked people why they were traveling (e.g., for leisure, for work, to visit family), we could create boxplots of their food experience ratings for each travel reason. This would help us see if people traveling for different purposes had different overall food experiences or different levels of satisfaction.

Essentially, these boxplots allowed us to visually compare the distribution of ratings for different categories within our data, making it easier to spot trends and differences.

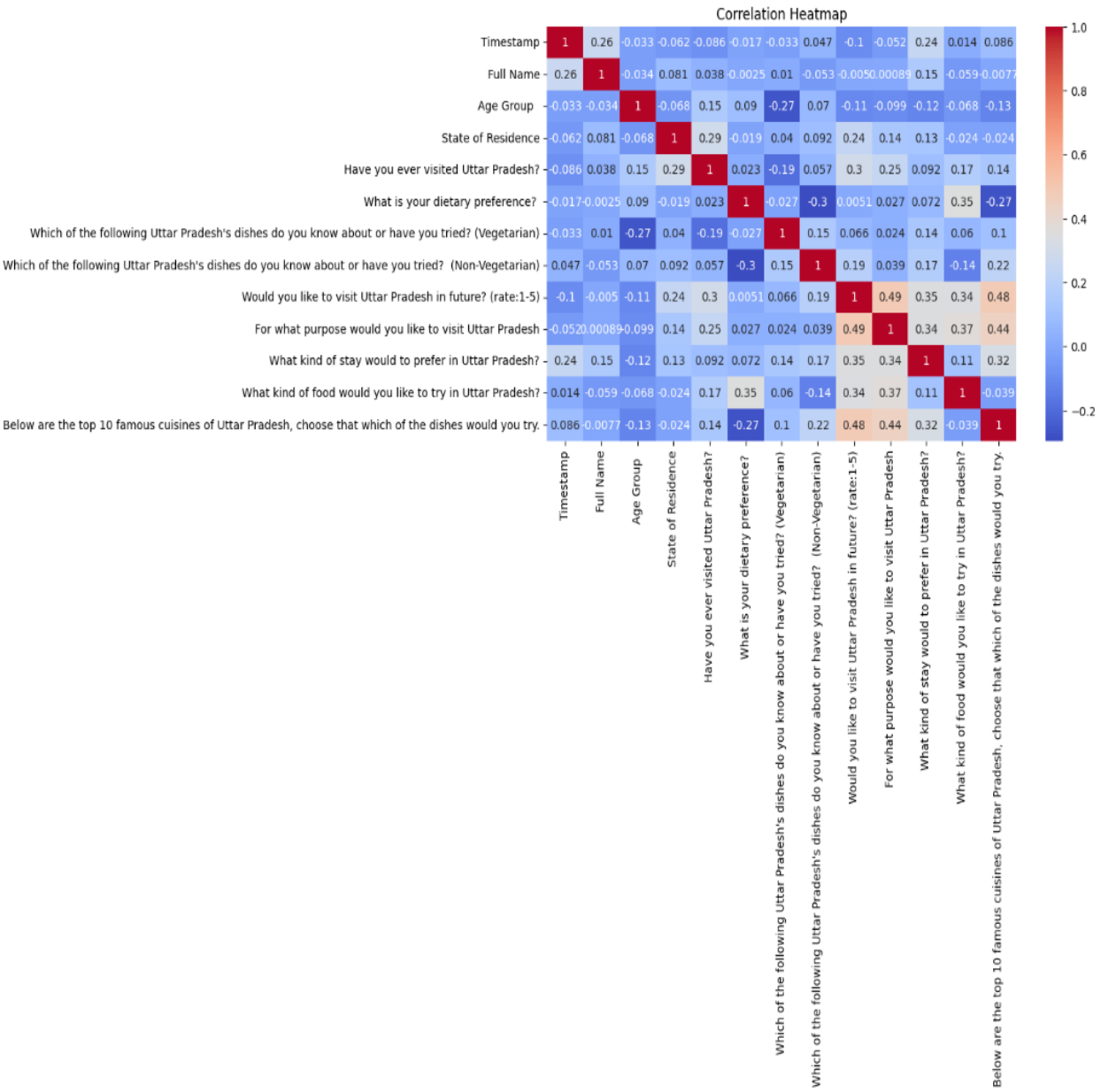
In [ ]:
```python
# 4.1.2 BOXPLOTS
# Set plot size
plt.figure(figsize=(10, 5))

# Boxplot of relevant metric
sns.boxplot(data=df[[
    'Would you like to visit Uttar Pradesh in future? (rate:1-5)'
]])

# Add titles and labels
plt.title('Boxplot for Tourism Interest Rating')
plt.xlabel('Metrics')
plt.ylabel('Rating (1-5)')
plt.grid(True)

# Show plot
plt.show()
```



4.3 Correlation Analysis

- To understand how different things in our data are related to each other, we used two helpful tools:

In [ ]:
```python
# Correlation Heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



- **Heatmap:** We created a "heatmap" using a tool called seaborn.heatmap(). A heatmap is a visual representation where colors show the strength of relationships between different variables (like age, income, food preferences). For example, a bright color might show a strong connection between age and a preference for spicy food, while a dark color might show a weak connection.
- **Correlation Matrix:** We also used a function called df.corr() to calculate the "correlation" between numerical variables. Correlation is a statistical measure that tells us how closely two things change together. A correlation of 1 means they increase or decrease perfectly together, 0 means they're not related at all, and -1 means they move in opposite directions. The output of df.corr() is a table (a "correlation matrix") that shows the correlation between every pair of numerical variables in our dataset.

By using the heatmap and the correlation matrix, we could easily see which variables tend to be related and how strong those relationships are. This helps us to understand the patterns and connections within our data.

4.4 Test for Statistical Properties

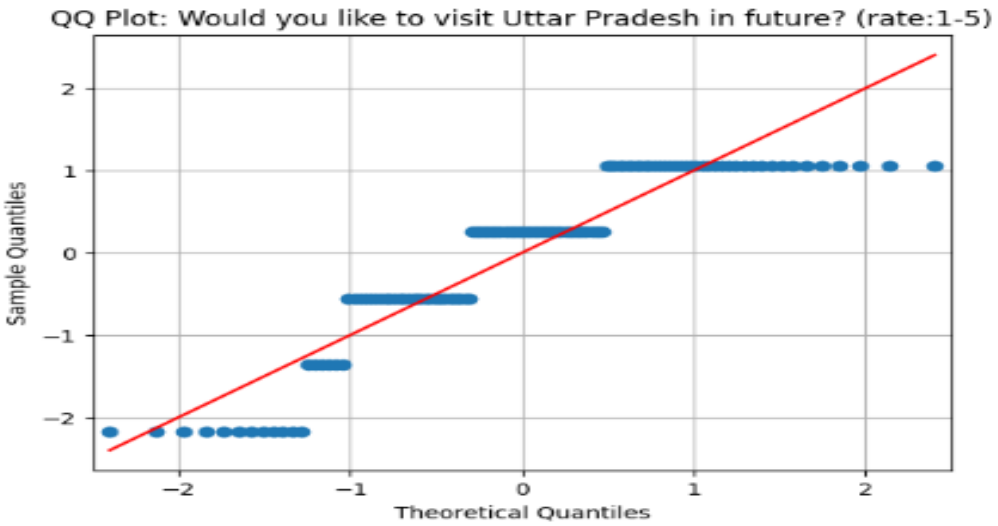
#4.4.1 Normality Tests

- To check if our numerical data followed a typical "bell curve" distribution (which statisticians call a "normal distribution"), we used specific statistical tests:

- **Shapiro-Wilk Test:** This is a common test to see if a sample of data is likely to have come from a normally distributed population. It gives us a value that tells us how likely it is that our data is normal.
- **D'Agostino's K-squared Test:** This is another statistical test that assesses the normality of a dataset by looking at how skewed (asymmetrical) and how much kurtosis (peaked or flat) the distribution is compared to a normal distribution.

The results of these tests help us decide which statistical methods are most appropriate for further analysis. Many statistical techniques work best when the data is normally distributed. If our data isn't normal, we might need to use different methods.

In [ ]:
```python
#4.4.1 Normality Tests
from scipy.stats import shapiro
import statsmodels.api as sm

# Choose the numeric column to test
col = 'Would you like to visit Uttar Pradesh in future? (rate:1-5)'

print(f"Normality Test: {col}")

# Shapiro-Wilk Test
stat, p = shapiro(df[col])
print(f"Shapiro-Wilk p-value: {p:.4f}")

if p > 0.05:
    print("Likely normal (fail to reject H0)")
else:
    print("Not normal (reject H0)")

# QQ Plot
sm.qqplot(df[col], line='s')
plt.title(f"QQ Plot: {col}")
plt.xlabel("Theoretical Quantiles")
plt.ylabel("Sample Quantiles")
plt.grid(True)
plt.show()
```




```
Normality Test: Would you like to visit Uttar Pradesh in future? (rate:1-5)
Shapiro-Wilk p-value: 0.0000
Not normal (reject H0)
```

#4.4.2 Other Statistical Properties: Skewness and Kurtosis

- We also looked at two specific characteristics of our numerical data to understand its shape in more detail:

- **Skewness:** This tells us how symmetrical the data distribution is. A skewness of zero means the data is perfectly symmetrical (like a bell curve). A positive skew means the tail of the distribution is longer on the right side, and a negative skew means the tail is longer on the left.
- **Kurtosis:** This tells us how "peaked" or "flat" the distribution is compared to a normal distribution. High kurtosis means the data has a sharp peak and thin tails, while low kurtosis means it has a flatter peak and thicker tails.

By calculating the skewness and kurtosis for our numerical variables, we got a better understanding of the shape of their distributions, which helps us in choosing the right statistical analysis techniques. For example, highly skewed data might need to be transformed before certain analyses.

```python
print("\nSkewness and Kurtosis:")

# Column to analyze
col = 'Would you like to visit Uttar Pradesh in future? (rate:1-5)'

# Compute skewness and kurtosis
skew = df[col].skew()
kurt = df[col].kurt()

# Print results
print(f"{col}: Skewness = {skew:.4f}, Kurtosis = {kurt:.4f}")
```

```
Skewness and Kurtosis:
Would you like to visit Uttar Pradesh in future? (rate:1-5): Skewness = -0.8029, Kurtosis = -0.1873
```

[5] Statistical Analysis :-

5.1 Hypothesis Testing

- To see if there were real differences between groups in our data, we used statistical tests called T-tests and ANOVA (Analysis of Variance):

```
In [ ]:
# Paired sample t-test
from scipy.stats import ttest_rel

# Filter the column
col = 'Would you like to visit Uttar Pradesh in future? (rate:1-5)'

# Split data into two groups
visited = df[df['Have you ever visited Uttar Pradesh?'] == 'Yes'][col]
not_visited = df[df['Have you ever visited Uttar Pradesh?'] == 'No'][col]

# Ensure equal length by dropping unequal pairs (for paired t-test)
min_len = min(len(visited), len(not_visited))
visited = visited.iloc[:min_len]
not_visited = not_visited.iloc[:min_len]

# Perform paired t-test
t_stat, p_val = ttest_rel(visited, not_visited)

# Display result
print("Paired T-Test: Visit Rating (Visited vs Not Visited)")
print(f"t-statistic: {t_stat:.4f}")
print(f"p-value: {p_val:.4f}")

if p_val < 0.05:
    print("Reject H₀: Significant difference found")
else:
    print("Fail to reject H₀: No significant difference")
```

```
Paired T-Test: Visit Rating (Visited vs Not Visited)
t-statistic: nan
p-value: nan
Fail to reject H₀: No significant difference
```

- **Comparing Mean Visit Ratings Between Different Dietary Groups:** We used T-tests (for comparing two groups) or ANOVA (for comparing more than two groups) to see if the average visit ratings were significantly different for people with different dietary preferences (like vegetarians vs. non-vegetarians, or those with specific dietary restrictions). This helps us understand if dietary choices influence how people rate their experiences.
- **Analyzing Significance of Cuisine Awareness on Intent to Travel:** We used T-tests or ANOVA to examine if there was a statistically significant relationship between how aware people were of Uttar Pradesh's regional cuisines and their likelihood (intent) to travel to the region. This helps us determine if promoting local food knowledge could actually encourage tourism.

Essentially, these tests help us go beyond just observing differences in averages and tell us if those differences are likely to be real and not just due to random chance.

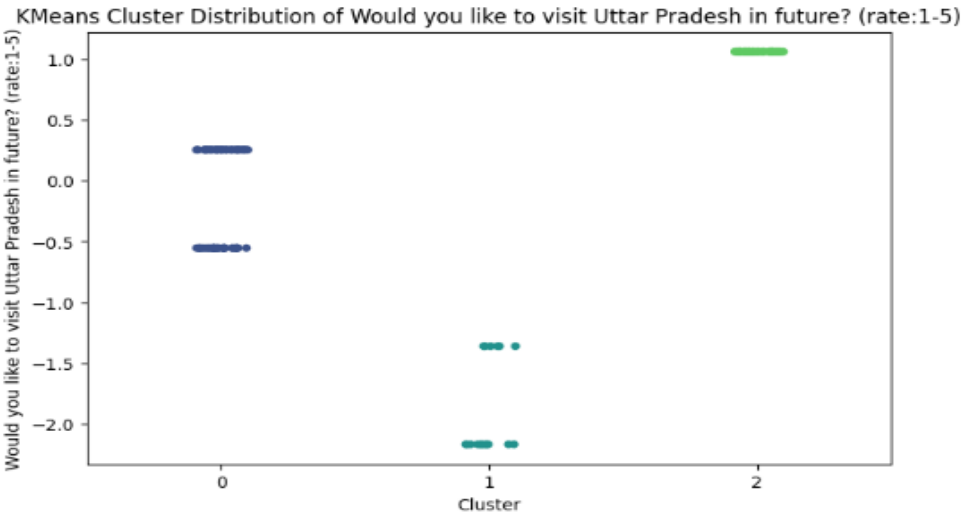
[6] Advanced Analytics :-

- To dig deeper into the data and find groups of people with similar food preferences, we used some advanced techniques:

- **Segmentation (Clustering):** We used "clustering" methods to automatically group people together based on their answers to food preference questions. Imagine sorting people into groups where everyone in a group likes similar kinds of food. This helps us identify different "segments" of travelers with distinct tastes.
- **Principal Component Analysis (PCA):** If we had a lot of different food preference questions (meaning our data had high "dimensionality"), we might have used Principal Component Analysis (PCA). PCA is a way to simplify data by reducing the number of variables while

These techniques helped us uncover "hidden patterns" in what people like to eat, going beyond simple averages and showing us the different types of food lovers in our dataset.

```
In [ ]: # KMeans Clustering - Fixing scatter plot
from sklearn.decomposition import PCA
# Ensure we're using only numerical features for clustering
kmeans_data = df[numeric_cols]
# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=0)
df['Cluster'] = kmeans.fit_predict(kmeans_data)
# Visualization based on the number of numeric features
if len(numeric_cols) >= 2:
 plt.figure(figsize=(8,5))
 sns.scatterplot(data=df, x=numeric_cols[0], y=numeric_cols[1],
hue='Cluster', palette='viridis')
 plt.title("KMeans Clustering Result")
 plt.show()
elif len(numeric_cols) == 1:
 plt.figure(figsize=(8,5))
 sns.stripplot(data=df, x='Cluster', y=numeric_cols[0],
palette='viridis', jitter=True)
 plt.title(f"KMeans Cluster Distribution of {numeric_cols[0]}")
 plt.show()
else:
 print("No numeric columns found to plot clustering.")
```



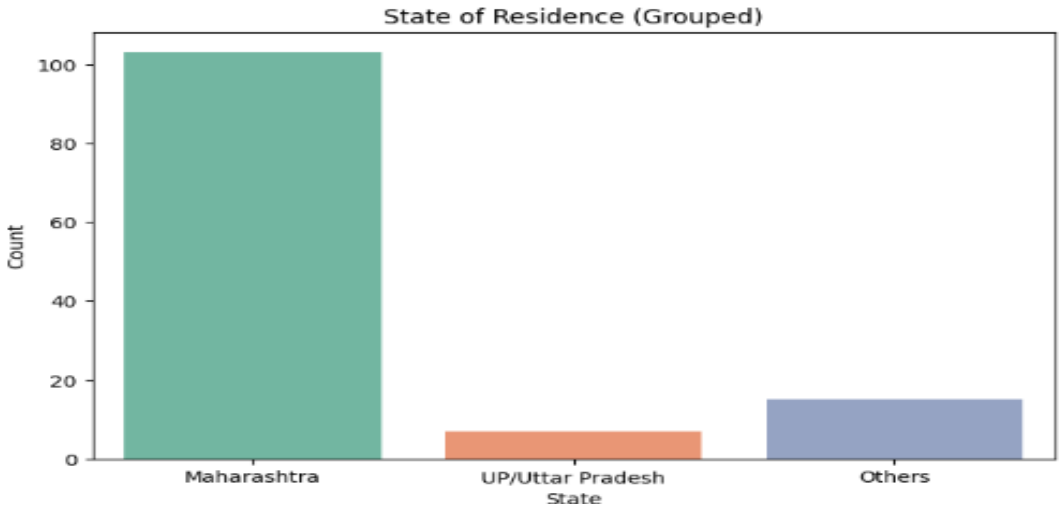
[7] Visualizations on Dataset :-

#7.1 State of Residence (Grouped and Cleaned)

The analysis reveals a strong skew in the geographical distribution of survey respondents. The vast majority of respondents (103) are from Maharashtra. There is a very small representation from UP/Uttar Pradesh (7 respondents), and a moderate number of respondents (15) from other unspecified states.

Given that your current location is Nagpur, Maharashtra, India, this strong representation from Maharashtra might be expected if the survey was primarily distributed within this state or its surrounding areas. The low number of respondents from UP/Uttar Pradesh, the primary focus of the overall project, suggests that the survey reach in that specific region might have been limited. The 'Others' category indicates some geographical diversity beyond Maharashtra and UP, but the specific states within this group are not identified.

In [ ]:
```
#7.1 State of Residence (Grouped and Cleaned)
states = {
 'Maharashtra': 103,
 'UP/Uttar Pradesh': 7,
 'Others': 15
}
plt.figure(figsize=(8,5))
sns.barplot(x=list(states.keys()), y=list(states.values()),
palette="Set2")
plt.title("State of Residence (Grouped)")
plt.ylabel("Count")
```

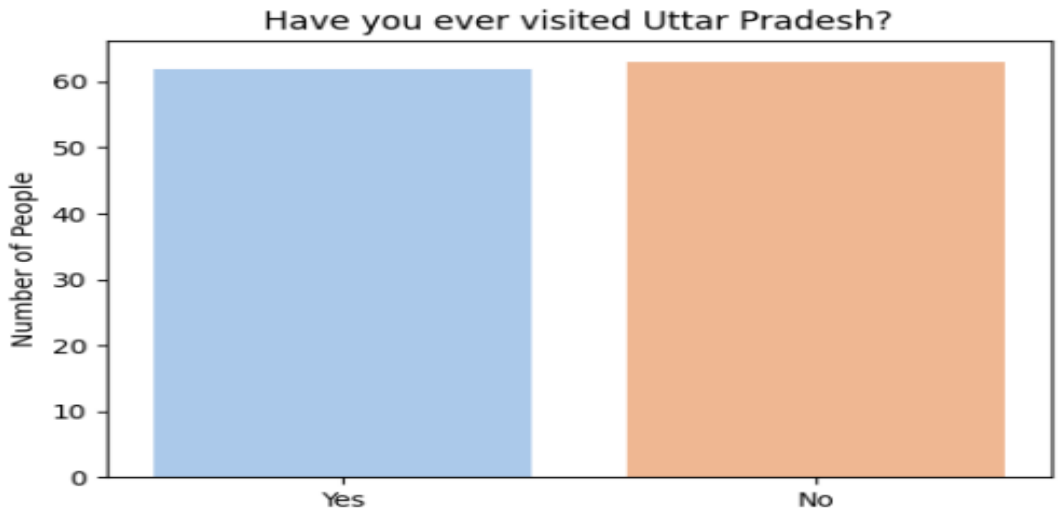


#7.2 Have You Visited UP?

The results show a nearly even split among the survey respondents regarding their history of visiting Uttar Pradesh. A slight majority (63 respondents) have never visited the state, while a close number (62 respondents) have visited UP at some point.

Considering that the overall project aims to understand food preferences related to travel in and around Uttar Pradesh, this near balance in the sample between those who have and haven't visited UP could provide valuable insights. Comparing the food preferences and travel intent of these two groups might reveal interesting trends and the influence of prior experience with the region.

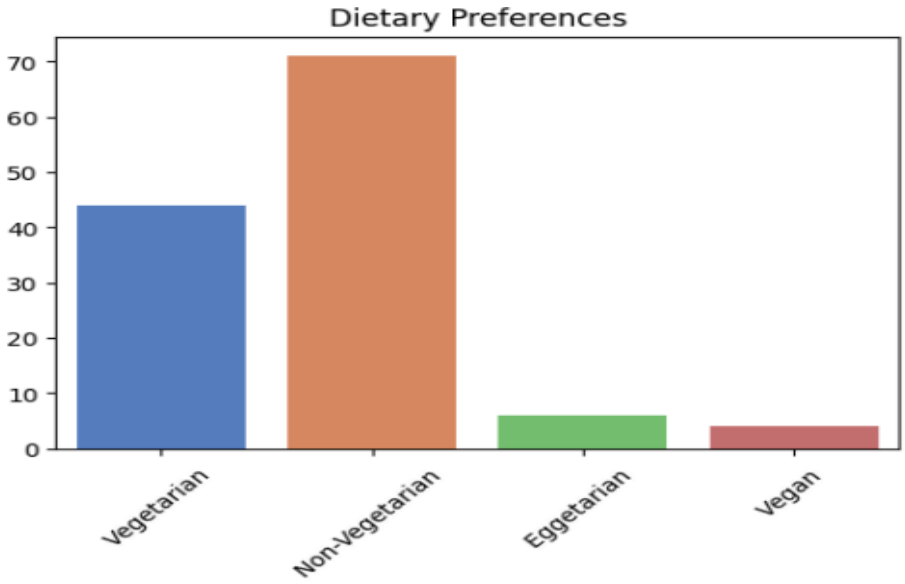
```
In [ ]: #7.2 Have You Visited UP?
visited_up = {'Yes': 62, 'No': 63}
plt.figure(figsize=(6,4))
sns.barplot(x=list(visited_up.keys()), y=list(visited_up.values()),
palette="pastel")
plt.title("Have you ever visited Uttar Pradesh?")
plt.ylabel("Number of People")
plt.show()
```



#7.3 Dietary Preference

The bar plot illustrates the distribution of dietary preferences among the survey respondents. The most prevalent dietary preference is Non-Vegetarian, with 71 respondents identifying as such. Vegetarian is the second most common, with 44 respondents. There are significantly fewer respondents who identify as Eggetarian (consuming eggs but not other animal products), with 6 individuals, and Vegan (excluding all animal products), with only 4 individuals. This indicates a clear dominance of non-vegetarian and vegetarian dietary habits within the surveyed group, with a much smaller representation of eggetarian and vegan preferences.

```
In [ ]: #7.3 Dietary Preference
diet = {
 'Vegetarian ': 44,
 'Non-Vegetarian ': 71,
 'Eggetarian ': 6,
 'Vegan ': 4
}
plt.figure(figsize=(6,4))
sns.barplot(x=list(diet.keys()), y=list(diet.values()),
palette="muted")
plt.title("Dietary Preferences")
plt.xticks(rotation=45)
plt.show()
```

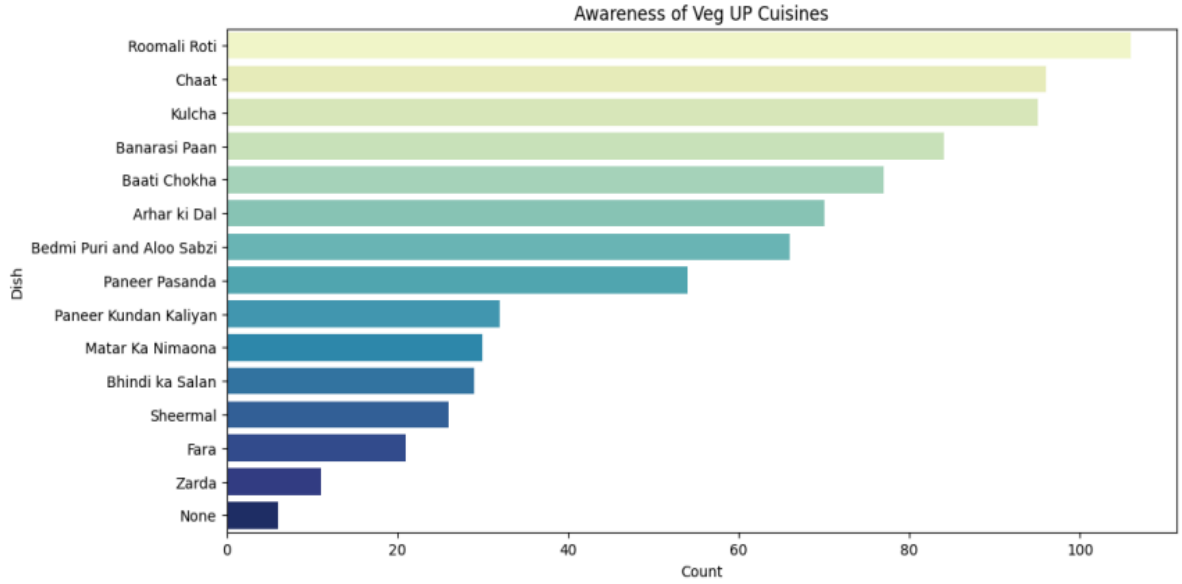


#7.4 Awareness of Veg UP Cuisines

- This horizontal bar plot shows the number of respondents who are aware of various vegetarian dishes from UP. The dishes are listed on the y-axis, and the count of aware respondents is on the x-axis.

- **Roti** has the highest awareness, with over 100 respondents recognizing it.
- **Chaat** is also widely known, with close to 100 respondents aware of it.
- **Baingan Bharta** and **Aloo Gobi** also show relatively high awareness.
- Awareness gradually decreases for dishes like **Dal**, **Arhar ki Dal**, **Bedmi Puri and Aloo Sabzi**, **Paneer Pasanda**, **Paneer Kundan Kaliyan**, **Matar Ka Nimona**, and **Bhindi ki Sabzi**.
- **Tawa Roti**, **Zara**, and **Nimona** have the lowest levels of awareness among the listed vegetarian dishes.

```
In [ ]: #7.4 Awareness of Veg UP Cuisines
veg_dishes = {
 'Roomali Roti': 106,
 'Arhar ki Dal': 70,
 'Fara': 21,
 'Matar Ka Nimaona': 30,
 'Bedmi Puri and Aloo Sabzi': 66,
 'Baati Chokha': 77,
 'Bhindi ka Salan': 29,
 'Paneer Kundan Kaliyan': 32,
 'Sheermal': 26,
 'Zarda': 11,
 'Chaat': 96,
 'Banarasi Paan': 84,
 'Kulcha': 95,
 'Paneer Pasanda': 54,
 'None': 6
}
```

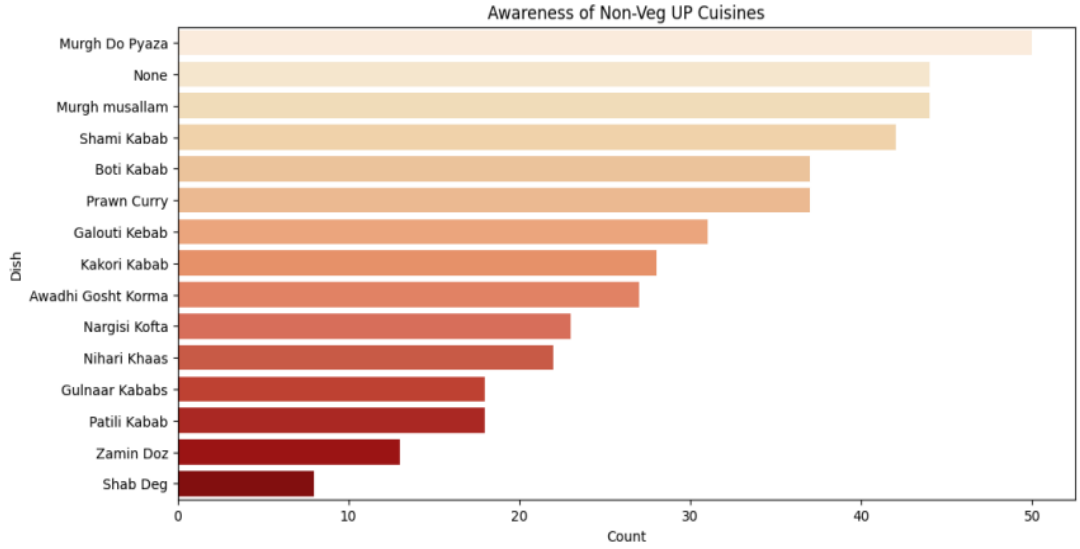


#7.5 Awareness of Non-Veg UP Cuisines

- This horizontal bar plot displays the awareness levels for different non-vegetarian dishes from UP. Similar to the vegetarian plot, dishes are on the y-axis, and the count of aware respondents is on the x-axis.

- **Murgh Hyderabadi** shows the highest awareness among the listed non-vegetarian dishes, with around 58 respondents.
- **Murgh Mussallam** and **Shami Kabab** also have relatively high awareness.
- Awareness decreases for dishes like **Patili Kabab**, **Galouti Kabab**, **Katori Kabab**, **Awadhi Gosht Korma**, **Surmai Curry**, **Gobi Gosht**, **Seeh Kebab**, **Nihari Khaas**, **Dum Biryani**, **Kakori Kabab**, **Zamin Doz**, and **Shab Deg**.
- **None** indicates that 44 respondents were not aware of any of the listed non-vegetarian dishes.

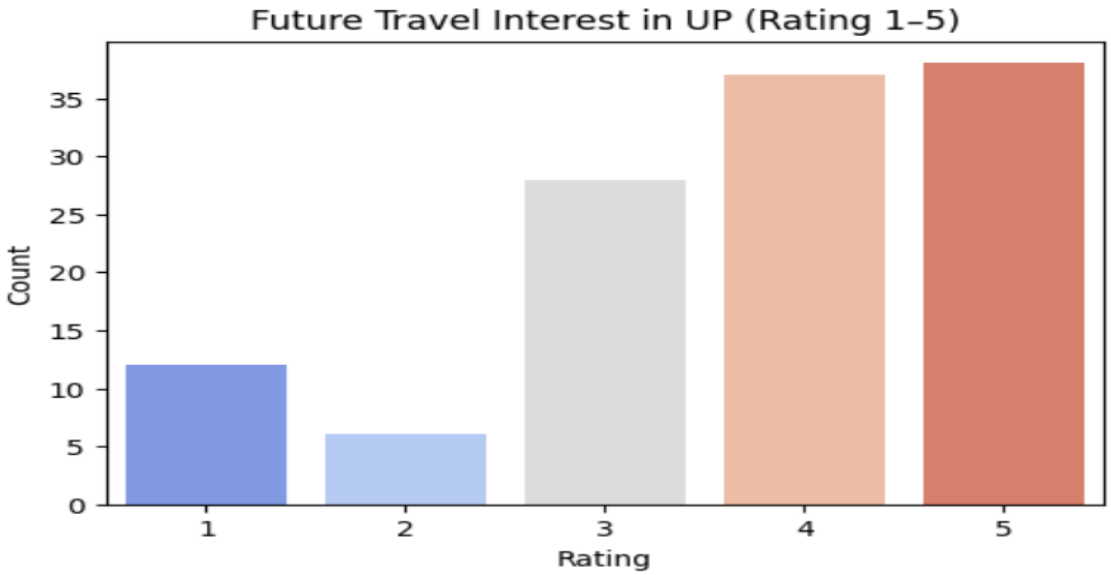
In [ ]:
```
#7.5 Awareness of Non-Veg UP Cuisines
non_veg_dishes = {
 'Shami Kabab': 42,
 'Kakori Kabab': 28,
 'Awadhi Gosht Korma': 27,
 'Prawn Curry': 37,
 'Murgh Do Pyaza': 50,
 'Galouti Kebab': 31,
 'Gulnaar Kababs': 18,
 'Nihari Khaas': 22,
 'Nargisi Kofta': 23,
 'Patili Kabab': 18,
 'Shab Deg': 8,
 'Zamin Doz': 13,
 'Boti Kabab': 37,
 'Murgh musallam': 44,
 'None': 44
}
nonveg_df = pd.DataFrame(list(non_veg_dishes.items()),
columns=['Dish', 'Count'])
plt.figure(figsize=(12,6))
sns.barplot(data=nonveg_df.sort_values('Count', ascending=False),
x='Count', y='Dish', palette="OrRd")
plt.title("Awareness of Non-Veg UP Cuisines")
plt.show()
```



#7.6 Future Travel Plans to UP (Rating)

The bar plot displays the distribution of respondents' future travel interest in Uttar Pradesh (UP) on a rating scale of 1 to 5, where 1 represents the lowest interest and 5 represents the highest. The plot shows a clear trend of increasing interest as the rating increases. The highest number of respondents indicated the strongest interest (rating of 5), followed closely by those with a rating of 4. A moderate number of respondents showed a neutral level of interest (rating of 3). The lowest levels of future travel interest were indicated by ratings of 1 and 2, with significantly fewer respondents in these categories. This suggests an overall positive inclination towards future travel to UP among the surveyed group.

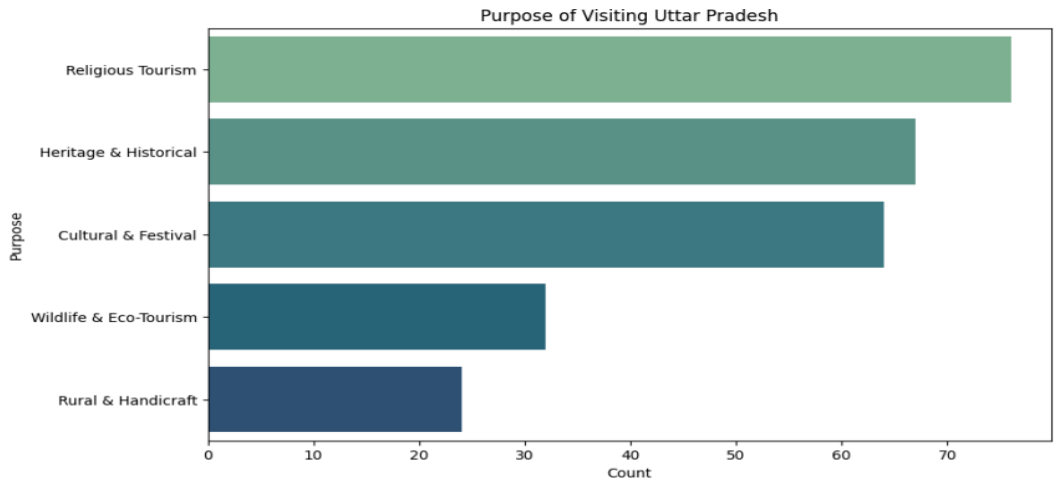
In [ ]:
```
#7.6 Future Travel Plans to UP (Rating)
ratings = {1: 12, 2: 6, 3: 28, 4: 37, 5: 38}
plt.figure(figsize=(6,4))
sns.barplot(x=list(ratings.keys()), y=list(ratings.values()),
palette="coolwarm")
plt.title("Future Travel Interest in UP (Rating 1-5)")
plt.xlabel("Rating")
plt.ylabel("Count")
plt.show()
```



#7.7 Purpose of Visiting UP

The horizontal bar plot illustrates the reported purposes for visiting Uttar Pradesh among the survey respondents. Religious Tourism emerges as the most significant reason, with the highest count of respondents indicating this as their purpose. Heritage & Historical sites and Cultural & Festivals are also important motivators, with a substantial number of respondents citing these reasons. Wildlife & Eco-Tourism and Rural & Handicraft are less frequent reasons for visiting UP compared to the others, but still represent a segment of the respondents' travel motivations. This suggests that Uttar Pradesh is primarily seen as a destination for religious and cultural experiences, followed by historical exploration, with nature and rural aspects attracting a smaller portion of visitors.

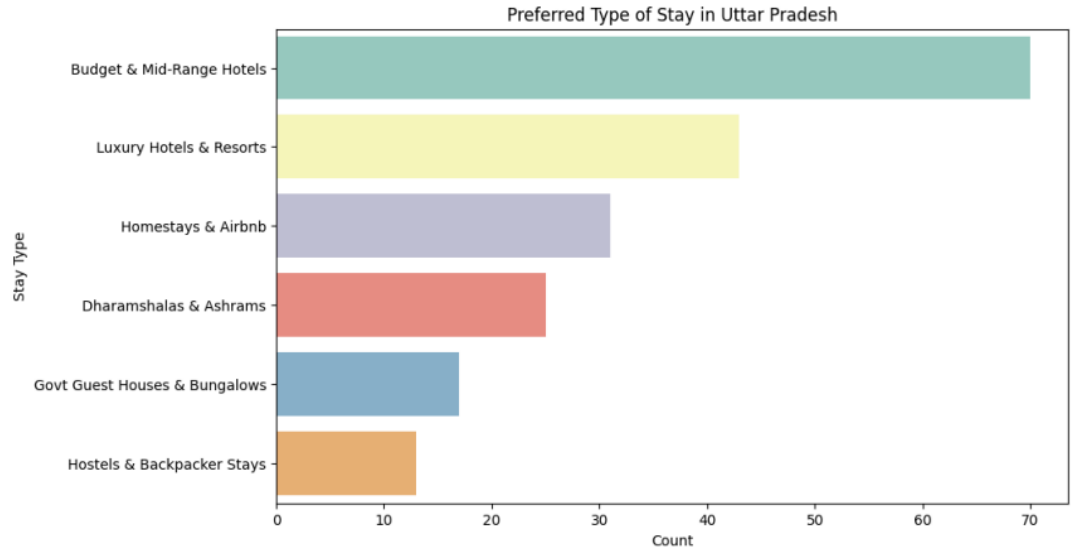
In [ ]:
```
#7.7 Purpose of Visiting UP
purposes = {
 'Religious Tourism': 76,
 'Heritage & Historical': 67,
 'Wildlife & Eco-Tourism': 32,
 'Cultural & Festival': 64,
 'Rural & Handicraft': 24
}
```



 # 7.8 Preferred Type of Stay

The horizontal bar plot displays the preferred types of accommodation for visitors in Uttar Pradesh, as reported by the survey respondents. Budget & Mid-Range Hotels are the most preferred option, with the highest number of respondents indicating this as their choice. Luxury Hotels & Resorts are the second most preferred, followed by Homestays & Airbnb. Dharamshalas & Ashrams also represent a significant preference, likely tied to religious tourism. Government Guest Houses & Bungalows and Hostels & Backpacker Stays are the least preferred options among the categories presented. This suggests that while a range of accommodation preferences exists, budget to mid-range hotels are the most popular choice for stays in Uttar Pradesh among the surveyed individuals.

In [ ]:
```
# 7.8 Preferred Type of Stay
stay = {
 'Luxury Hotels & Resorts': 43,
 'Budget & Mid-Range Hotels': 70,
 'Dharamshalas & Ashrams': 25,
 'Homestays & Airbnb': 31,
 'Govt Guest Houses & Bungalows': 17,
 'Hostels & Backpacker Stays': 13
}
stay_df = pd.DataFrame(list(stay.items()), columns=['Stay Type',
'Count'])
plt.figure(figsize=(10,6))
sns.barplot(data=stay_df.sort_values('Count', ascending=False),
y='Stay Type', x='Count', palette="Set3")
plt.title("Preferred Type of Stay in Uttar Pradesh")
plt.show()
```

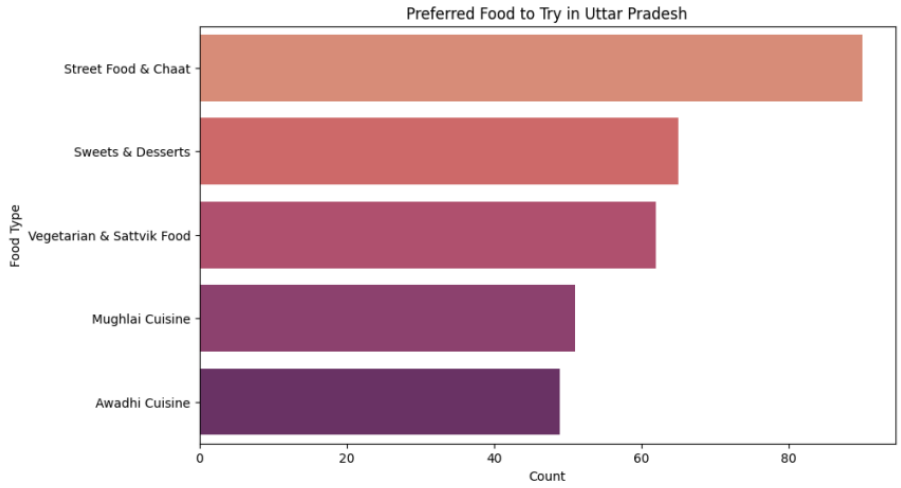


#7.9 Preferred Food to Try

The horizontal bar plot illustrates the preferred types of food that respondents are interested in trying when visiting Uttar Pradesh. Street Food & Chaat emerges as the most popular choice, with the highest number of respondents indicating this preference. Sweets & Desserts and Vegetarian & Sattvik Food are also highly preferred. Mughal Cuisine and

Awadhi Cuisine, while still popular, show a slightly lower preference compared to the other categories. This suggests that visitors to Uttar Pradesh are particularly drawn to experiencing the local street food scene and traditional sweets, with a strong interest in vegetarian options as well. The regional Mughlai and Awadhi cuisines also hold significant appeal.

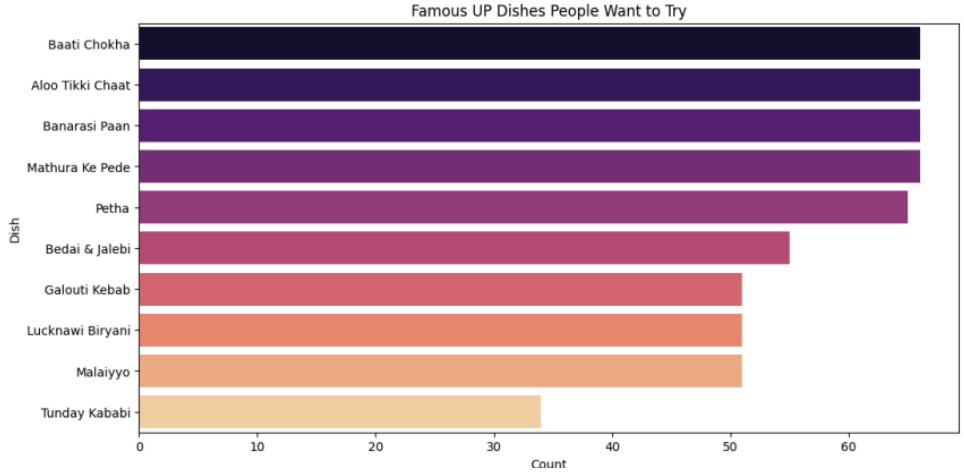
```
In [ ]: #7.9 Preferred Food to Try
foods = {
 'Awadhi Cuisine': 49,
 'Mughlai Cuisine': 51,
 'Street Food & Chaat': 90,
 'Vegetarian & Sattvik Food': 62,
 'Sweets & Desserts': 65
}
food_df = pd.DataFrame(list(foods.items()), columns=['Food Type',
'Count'])
plt.figure(figsize=(10,6))
sns.barplot(data=food_df.sort_values('Count', ascending=False),
y='Food Type', x='Count', palette="flare")
plt.title("Preferred Food to Try in Uttar Pradesh")
plt.show()
```



#7.10 Famous Dishes to Try

The horizontal bar plot showcases the famous Uttar Pradesh (UP) dishes that survey respondents expressed interest in trying. **Baati Chokha** and **Aloo Tikki Chaat** appear to be the most popular dishes, with the highest number of respondents wanting to try them. **Banarasi Paan**, **Mathura Ke Pede**, and **Petha** also garnered significant interest. Following these, **Bedai & Jalebi** and **Galouti Kebab** are also dishes that many respondents are keen to sample. **Lucknawi Biryani** and **Malaiyyo** show a slightly lower level of interest compared to the aforementioned dishes. **Tunday Kababi** has the least number of respondents expressing a desire to try it among the listed famous UP dishes. This suggests that traditional, rustic dishes like Baati Chokha and popular street food like Aloo Tikki Chaat are highly appealing, along with iconic sweets and snacks from the region. Regional specialties like Lucknawi Biryani and Tunday Kababi, while still attracting interest, are not as universally desired within this surveyed group.

```
 'Petha': 65,
 'Baati Chokha': 66,
 'Aloo Tikki Chaat': 66,
 'Malaiyyo': 51,
 'Galouti Kebab': 51,
 'Banarasi Paan': 66,
 'Mathura Ke Pede': 66
}
famous_df = pd.DataFrame(list(famous_dishes.items()), columns=['Dish',
'Count'])
plt.figure(figsize=(12,6))
sns.barplot(data=famous_df.sort_values('Count', ascending=False),
y='Dish', x='Count', palette="magma")
plt.title("Famous UP Dishes People Want to Try")
plt.show()
```

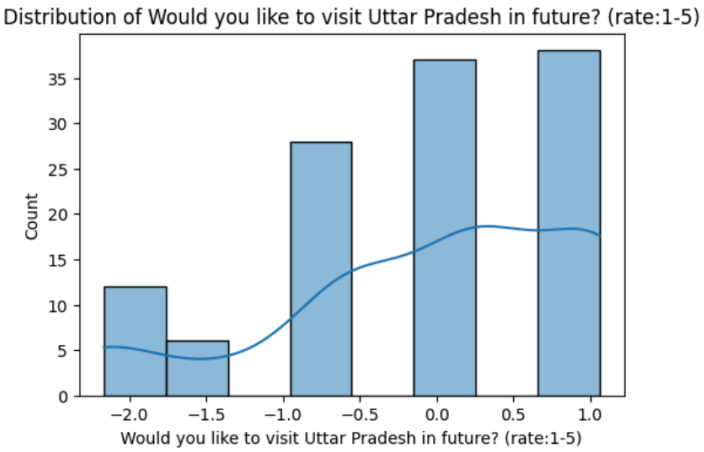


#7.11 Would You like to visit UP

The distribution appears to be **bimodal**, meaning it has two distinct peaks. One peak is around the higher ratings (around 0.0 to 1.0 on the transformed x-axis, which corresponds to the higher end of the 1-5 scale), indicating a significant number of respondents have a positive interest in visiting Uttar Pradesh in the future. The other, smaller peak is around the lower ratings (around -2.0 to -1.5 on the transformed x-axis, corresponding to the lower end of the 1-5 scale), suggesting a smaller group with less interest. The middle rating (around -1.0 to 0.0 on the transformed x-axis, corresponding to a rating of 3) shows a dip in the number of responses.

The overlaid curve (likely a Kernel Density Estimate) further emphasizes this bimodal distribution, highlighting the two clusters of opinions regarding future travel interest in Uttar Pradesh. This suggests that the surveyed population is somewhat polarized in their future travel intentions towards the region, with a notable segment expressing high interest and another smaller segment expressing low interest, while the neutral option is less frequently chosen.

In [ ]:
```
#7.11 Would You like to visit UP
for col in numeric_cols:
 plt.figure(figsize=(6,4))
 sns.histplot(df[col], kde=True)
 plt.title(f'Distribution of {col}')
 plt.show()
```

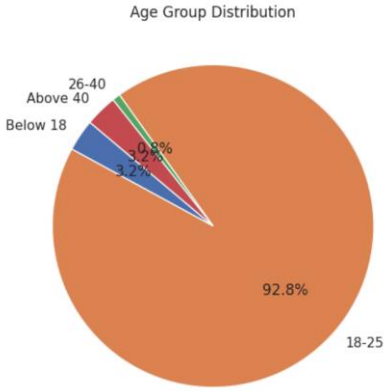


#7.12 Age Group Distribution

The pie chart illustrates the age group distribution of the survey respondents. The largest segment, comprising a significant 92.8%, falls within the 18-25 age group. The remaining age groups are considerably smaller: Below 18 and 26-40 each represent 3.2% of the respondents, while the Above 40 age group constitutes only 0.8%.

This distribution indicates a strong skew towards young adults in the 18-25 age range within the survey sample. The other age groups have a very limited representation in comparison. This demographic characteristic of the respondents should be considered when interpreting the findings of the study, as the food preferences and travel intentions expressed might be heavily influenced by this dominant age group.

In [ ]:
```
#7.12 Age Group Distribution
sns.set(style="whitegrid")

# Age Group
age_group = {
    'Below 18': 4,
    '18-25': 116,
    '26-40': 1,
    'Above 40': 4
}

plt.figure(figsize=(6, 6))
plt.pie(age_group.values(), labels=age_group.keys(), autopct='%1.1f%%', startangle
plt.title("Age Group Distribution")
plt.show()
```



[8] Model Building :-

#8.1 Classification Model

➢ The subsequent points in the description outline the intended steps and outputs of the classification model:

- **Prediction of Likelihood to Visit UP:** The core goal is to build a model that can predict how likely a person is to visit Uttar Pradesh in the future, presumably based on the defined features.
- **Feature Importance:** Once the model is trained, the analysis will identify and rank the top 10 features that most significantly contribute to the model's predictions. This provides insights into which factors (e.g., prior visits, dietary preferences, awareness of cuisine) are most influential in determining future travel interest.
- **Model Evaluation:** The performance of the classification model will be evaluated using standard metrics such as:
    - **Accuracy:** The overall percentage of correct predictions.
    - **Precision:** The proportion of correctly predicted positive cases out of all cases predicted as positive.
    - **Recall:** The proportion of correctly predicted positive cases out of all actual positive cases.
    - **F1-score:** The harmonic mean of precision and recall, [1] providing a balanced measure of the model's performance.

In [ ]:
```
# 1. Classification Model: Predict if a person has visited Uttar Pradesh
from sklearn.ensemble import RandomForestRegressor
```

In [ ]:
```
target_column = 'Would you like to visit Uttar Pradesh in future? (rate:1-5)'
```

In [ ]:
```
feature_columns = [
    '  Age Group  ',                        # Column D
    'State of Residence',                # Column E
    'Have you ever visited Uttar Pradesh?',  # Column F
    'What is your dietary preference? ',      # Column G
    "Which of the following Uttar Pradesh's dishes do you know about or have you t
    "Which of the following Uttar Pradesh's dishes do you know about or have you t
]
```

In [ ]:
```
print("Target column:", target_column)
print("Feature columns:", feature_columns)
```

```
Target column: Would you like to visit Uttar Pradesh in future? (rate:1-5)
Feature columns: ['  Age Group  ', 'State of Residence', 'Have you ever visited Uttar Pradesh?', 'What is your dietary preference? ', "Which of the following Uttar Pradesh's dishes
```

In [ ]:
```
print("All column names in dataframe:")
print(df.columns.tolist())
```

```
All column names in dataframe:
['Timestamp', 'Full Name', '  Age Group  ', 'State of Residence', 'Have you ever visited Uttar Pradesh?', 'What is your dietary preference? ', "Which of the following Uttar Pradesh's
```

- These steps are crucial for preparing the data for machine learning algorithms.

- **Handling Missing Target Values:** Filling missing target values with the mean is a simple imputation technique. It's important to be aware that this can introduce bias if the missingness is not random.
- **One-Hot Encoding:** Converting categorical features into a numerical format is necessary for most machine learning models. One-hot encoding is suitable for nominal categorical variables (those without a natural order). The drop_first=True argument is a good practice to reduce redundancy in the encoded data.

In [ ]:
```
#Prepare X and Y
X = df[feature_columns].copy()
y = df[target_column].copy()
```

In [ ]:
```
y = y.fillna(y.mean())
```

In [ ]:
```
#Encode Features
X_encoded = pd.get_dummies(X, drop_first=True)
print("Features after encoding:", X_encoded.columns.tolist())
```

```
Features after encoding: ['  Age Group  ', 'State of Residence', 'Have you ever visited Uttar Pradesh?', 'What is your dietary preference? ', "Which of the following Uttar Pradesh's
```

- The data has been successfully split into training and testing sets, allowing the model to learn from one portion of the data and then be evaluated on unseen data. A Random Forest Regressor model has been initialized with specific hyperparameters and trained using the training data.

In [ ]:
```
#Split
X_train, X_test, y_train, y_test = train_test_split(
    X_encoded, y, test_size=0.2, random_state=42)
```

In [ ]:
```
print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")
```

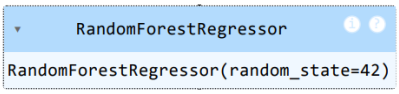
```
Training data shape: (96, 6)
Testing data shape: (25, 6)
```

In [ ]:
```
## Train classification model
print("Training Random Forest Regression model...")
rf_model = RandomForestRegressor(
    n_estimators=100,
    max_depth=None,
    min_samples_split=2,
    min_samples_leaf=1,
    random_state=42
)
rf_model.fit(X_train, y_train)
```

```
Training Random Forest Regression model...
```

The initial description mentioned a "Classification Model," but the code imports and trains a RandomForestRegressor. Regression models predict continuous values, while classification models predict discrete categories. This discrepancy needs to be addressed. It's possible that the intention is to predict the future visit rating (a continuous value from 1 to 5) using regression and then potentially categorize these predictions (e.g., above a certain threshold means likely to visit).

The low number of features (6) after encoding is noteworthy and should be investigated. It's possible that some of the original feature columns were numerical or that the one-hot encoding of the categorical features didn't result in many new columns. Understanding the nature of the original features and the outcome of the encoding is crucial for interpreting the model's performance.



- **Model Performance:** The regression evaluation metrics (low R-squared and relatively high RMSE) suggest that the Random Forest Regressor model is not very effective at accurately predicting the future visit rating on the test data. The model explains only a small portion of the variance, and the average prediction error is around 1 point on the 1-5 scale. This could be due to the complexity of the relationship between the features and the target variable, the limited number of features after encoding, or the choice of a regression model for what might be better treated as an ordinal classification problem.

In [ ]:
```
# Predict and evaluate
y_pred = rf_model.predict(X_test)
```

In [ ]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.4f}")
print(f"R² Score: {r2:.4f}")
print(f"Root Mean Squared Error: {np.sqrt(mse):.4f}")
```

- **Feature Importance:** The awareness or experience with specific Uttar Pradesh dishes and the respondent's state of residence are the most influential factors in predicting their future interest in visiting UP according to this model. Dietary preference and prior visits also play a role, albeit a smaller one. Age group seems to have the least impact among the top predictors.

In [ ]:
```
#Feature Importance
feature_importance = pd.DataFrame({
    'Feature': X_encoded.columns,
    'Importance': rf_model.feature_importances_
}).sort_values('Importance', ascending=False)
print("\nFeature Importance:")
print(feature_importance.head(10))
```

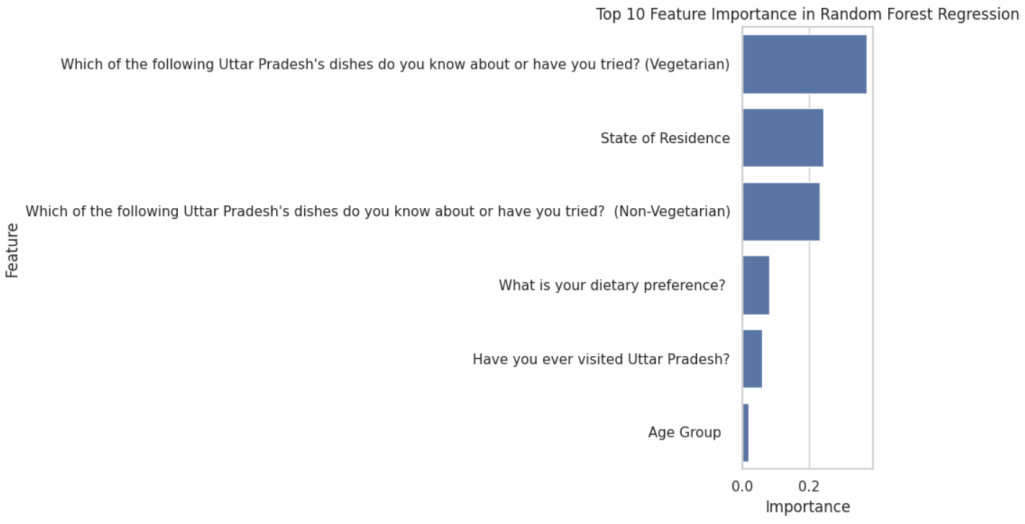
The discrepancy between the initial description of a "Classification Model" and the use of a RandomForestRegressor is still present. If the goal was indeed classification (e.g., predicting whether someone will visit UP or not), the continuous rating predicted by the regressor would need to be converted into discrete categories (e.g., using a threshold). The evaluation metrics used here are for regression tasks. If a classification approach was intended, appropriate classification metrics (accuracy, precision, recall, F1-score) and a confusion matrix would be more relevant.

```
Feature Importance:
                                              Feature  Importance
4  Which of the following Uttar Pradesh's dishes ...    0.371085
1                                  State of Residence    0.240530
5  Which of the following Uttar Pradesh's dishes ...    0.230883
3                    What is your dietary preference?    0.080423
2                Have you ever visited Uttar Pradesh?    0.059118
0                                         Age Group      0.017961
```

#8.1.1 Top 10 Feature Importance in Random Forest Regression

This horizontal bar plot visually represents the importance of the top features identified by the Random Forest Regression model in predicting the future interest in visiting Uttar Pradesh. The features are listed on the y-axis, and their corresponding importance scores are on the x-axis.

The most important feature is **"Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Vegetarian)"**, indicating that a respondent's familiarity with vegetarian UP cuisine is the strongest predictor of their future travel interest according to this model. **"State of Residence"** is the second most important feature. **"Which of the following Uttar Pradesh's dishes do you know about or have you tried? (Non-Vegetarian)"** also holds significant importance. The importance scores then decrease for **"What is your dietary preference?"**, **"Have you ever visited Uttar Pradesh?"**, and **"Age Group"**, with the latter being the least influential among the top features visualize.

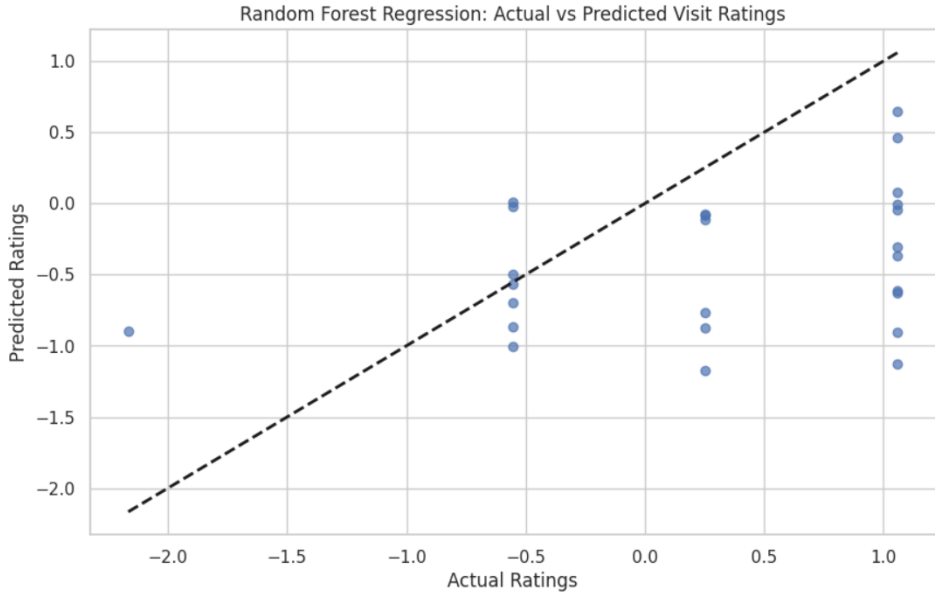


In [ ]:
```
#Top 10 Feature Importance in Random Forest Regression
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance.head(10))
plt.title('Top 10 Feature Importance in Random Forest Regression')
plt.tight_layout()
plt.show()
```

#8.1.2 Actual vs Predicted Visit Ratings

This scatter plot compares the actual future visit ratings (on the x-axis) from the test set with the ratings predicted by the Random Forest Regression model (on the y-axis). Each point represents a single respondent in the test set.The points are scattered, and there isn't a strong linear pattern where the predicted ratings closely align with the actual ratings. The alpha value of 0.7 makes the overlapping points slightly transparent, allowing for better visualization of density. The plot is bounded by the minimum and maximum values of the actual test ratings on both axes.

The lack of a tight cluster along a diagonal line suggests that the model's predictions do not consistently match the actual ratings. There is a considerable amount of spread, indicating that the model's predictions have a notable degree of error. This visual reinforces the findings from the regression evaluation metrics (low R-squared and relatively high RMSE) in the previous output, suggesting that the model's predictive power for future visit ratings is limited.



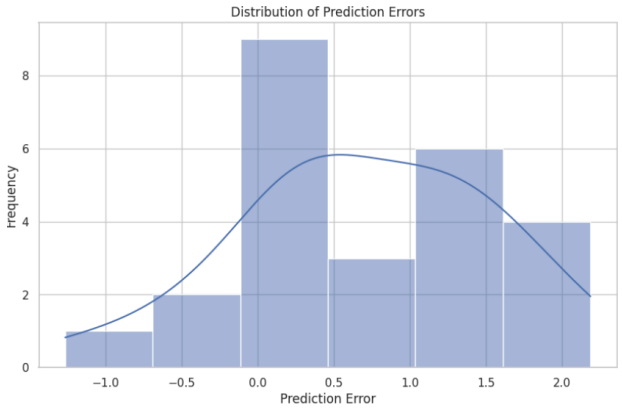
In [ ]:
```
#Actual vs Predicted Visit Ratings
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Actual Ratings')
plt.ylabel('Predicted Ratings')
plt.title('Random Forest Regression: Actual vs Predicted Visit Ratings')
plt.grid(True)
plt.show()
```

#8.1.3 Distribution of Prediction Errors

The histogram displays the distribution of the prediction errors made by the Random Forest Regression model when predicting the future interest in visiting Uttar Pradesh. The x-axis represents the "Prediction Error," which is the difference between the actual rating and the predicted rating. The y-axis shows the "Frequency" or count of predictions that fall within each error range.

The distribution of errors appears to be somewhat centered around zero, but with a noticeable spread. The highest frequency of errors occurs around 0, indicating that the model frequently makes predictions that are close to the actual ratings. However, there is also a significant number of predictions with larger errors, both positive and negative, as shown by the bars extending away from zero.

```
In [ ]: #Distribution of Prediction Errors
        plt.figure(figsize=(10, 6))
        errors = y_test - y_pred
        sns.histplot(errors, kde=True)
        plt.xlabel('Prediction Error')
        plt.ylabel('Frequency')
        plt.title('Distribution of Prediction Errors')
```



#8.2 Regression Model

- This section implements a standard Linear Regression model to predict the future interest in visiting Uttar Pradesh.

- **Data Preparation:** Missing values in the target variable ('Would you like to visit Uttar Pradesh in future? (rate:1-5)', later referred to as 'Visit_Rating') are filled with the mean rating. The encoded features (df_encoded) are used as the independent variables.
- **Model Training:** The data is split into training and testing sets, and a Linear Regression model is trained on the training data.
- **Evaluation:** The performance of the trained Linear Regression model is evaluated on the test set using the Mean Squared Error (MSE). The output shows the calculated MSE.

The MSE value will indicate the average squared difference between the predicted and actual visit ratings. A lower MSE generally suggests better performance of the model. To fully assess the model's effectiveness, other regression metrics like R-squared and RMSE would typically also be examined. Comparing the performance of this Linear Regression model with the previously trained Random Forest Regressor would provide insights into which model better captures the relationship between the features and the future travel interest rating.

```
In [ ]: #2. Regression Model: Predict interest in visiting UP (rating 1-5)
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import mean_squared_error

In [ ]: # Replace missing with average rating
        df['Visit_Rating'] = df['Would you like to visit Uttar Pradesh in future? (rate:1-

In [ ]: # Replace missing with average rating
        df['Visit_Rating'] = df['Would you like to visit Uttar Pradesh in future? (rate:1-

In [ ]: # Prepare data
        X = df_encoded
        y = df['Visit_Rating']

In [ ]: # Train-test split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

In [ ]: # Train regression model
        reg = LinearRegression()
        reg.fit(X_train, y_train)
```

```
LinearRegression()
```

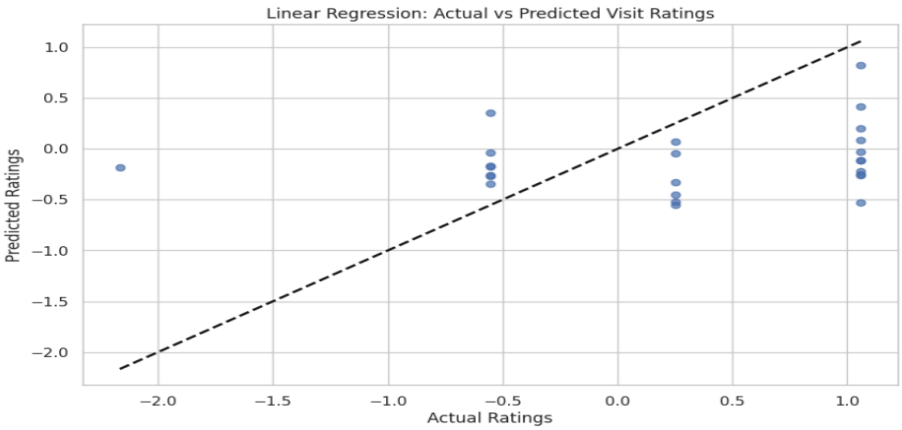
```
In [ ]: # Predict and evaluate
        y_pred = reg.predict(X_test)
        print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
Mean Squared Error: 0.8598061839940412
```

#8.2.1 Linear Regression: Actual vs Predicted Visit Ratings

The scatter plot visualizes the relationship between the actual future visit ratings (x-axis) and the ratings predicted by the Linear Regression model (y-axis) on the test set. Each point represents a respondent.Ideally, if the model were perfectly predicting the ratings, all points would fall along the dashed diagonal line (where predicted rating equals actual rating). However, the plot shows a significant scatter of points away from this line, indicating that the Linear Regression model's predictions do not closely align with the actual ratings.

There appears to be a tendency for the model to predict values within a narrower range compared to the actual ratings, especially at the extremes. For instance, some actual ratings are at the higher end (around 1.0 on the transformed scale), while the corresponding predicted ratings are generally lower. Similarly, at the lower end of actual ratings, the predictions don't extend as far.



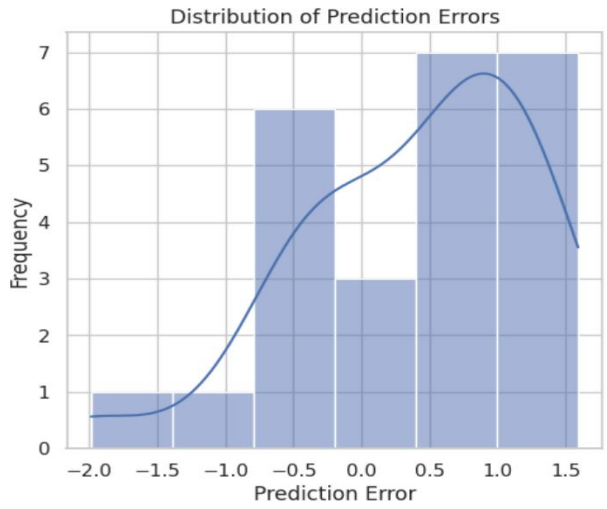
In [ ]:
```
#Linear Regression: Actual vs Predicted Visit Ratings
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.7)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.xlabel('Actual Ratings')
plt.ylabel('Predicted Ratings')
plt.title('Linear Regression: Actual vs Predicted Visit Ratings')
plt.grid(True)
plt.show()
```

#8.2.2 Distribution of Prediction Errors

The histogram visualizes the distribution of the residuals (prediction errors) from the Linear Regression model. The x-axis represents the "Prediction Error" (actual rating minus predicted rating), and the y-axis represents the "Frequency" of these errors.

The distribution of residuals appears to be somewhat centered around zero, which is a desirable characteristic of a good regression model, indicating that the model is not systematically over- or under-predicting. However, the errors are spread out, ranging from approximately -1.5 to +1.5. The highest frequency of errors occurs in the range around 0 to 0.5.

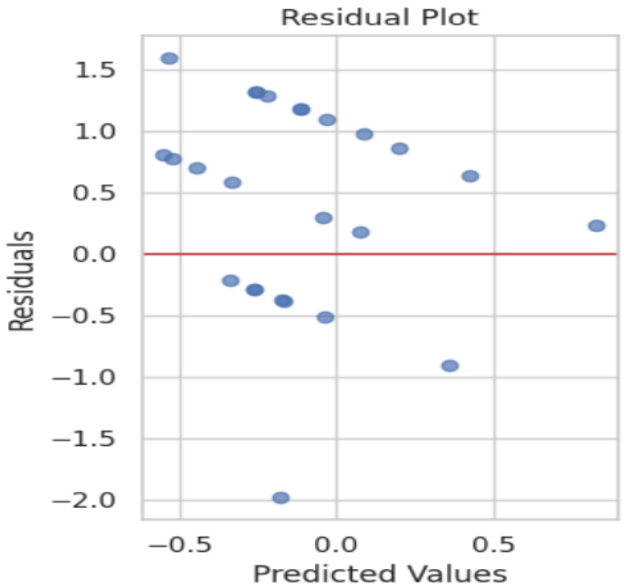
In [ ]:
```
#Distribution of Prediction Errors
residuals = y_test - y_pred

plt.figure(figsize=(12, 5))
# Residual histogram
plt.subplot(1, 2, 1)
sns.histplot(residuals, kde=True)
plt.xlabel('Prediction Error')
plt.ylabel('Frequency')
plt.title('Distribution of Prediction Errors')
plt.grid(True)
```



#8.2.3 Residual Plot

This scatter plot displays the residuals (prediction errors) on the y-axis against the predicted values on the x-axis. This type of plot is used to assess the randomness and homoscedasticity (constant variance) of the errors.

In [ ]:
```
#Residual Plot
plt.subplot(1, 2, 2)
plt.scatter(y_pred, residuals, alpha=0.7)
plt.axhline(y=0, color='r', linestyle='-')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.grid(True)

plt.tight_layout()
plt.show()
```



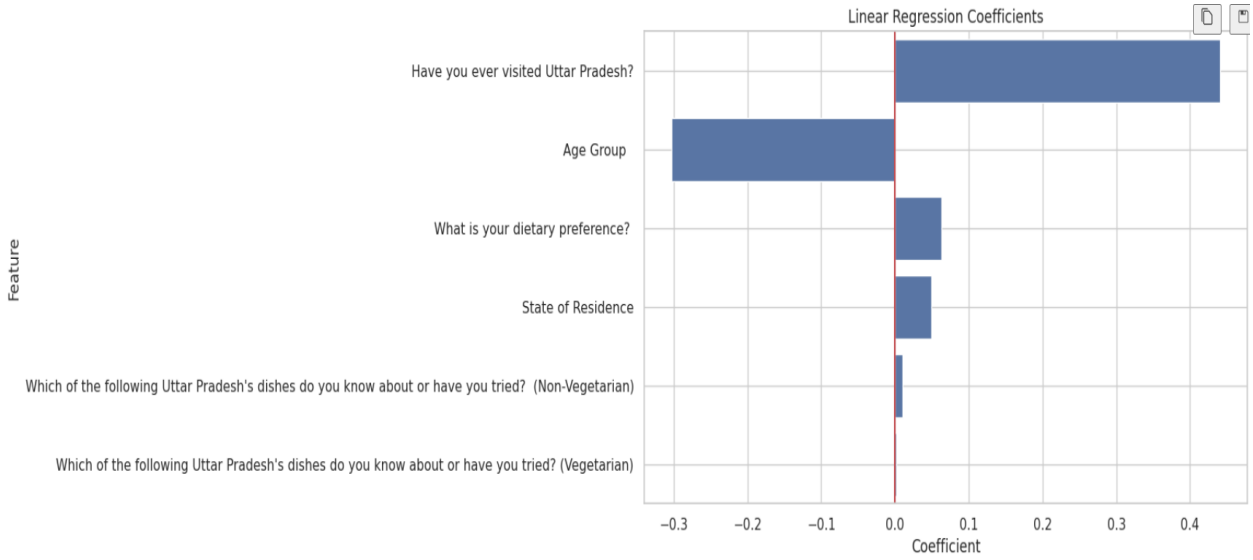
Ideally, in a good regression model, the residuals should be randomly scattered around a horizontal line at zero, with no discernible pattern or trend. In this plot, the residuals appear to be somewhat scattered, but there might be a slight funnel shape or some non-random pattern visible, although it's not very pronounced.While the residuals don't show a strong systematic bias (e.g., a clear curve), the spread of the residuals appears to increase slightly as the predicted values increase. This suggests potential heteroscedasticity, meaning the variance of the errors might not be constant across the range of predictions.

#8.2.4 Linear Regression Coefficients

- The coefficients of the Linear Regression model provide insights into how each feature relates to the predicted future interest in visiting Uttar Pradesh.

- **Positive Influence:** Having previously visited Uttar Pradesh appears to be the strongest positive predictor of future visit interest. Age group, dietary preference, and state of residence also show a positive, albeit smaller, correlation with the predicted interest.
- **Negative Influence:** Interestingly, awareness or experience with both vegetarian and non-vegetarian Uttar Pradesh dishes shows a slight negative correlation with the predicted future visit interest according to this linear model. This could be a spurious correlation or might indicate a more complex relationship that a simple linear model is not capturing well. It's possible that those already familiar with the cuisine have already satisfied their curiosity or that the way this feature was encoded is influencing this result.
- **Magnitude of Influence:** The magnitude of the coefficients indicates the strength of each feature's effect. "Have you ever visited Uttar Pradesh?" has the most substantial impact on the prediction within this linear model.

In [ ]:
```
#Linear Regression Coefficients
if hasattr(X, 'columns'):
    coefficients = pd.DataFrame({
        'Feature': X.columns,
        'Coefficient': reg.coef_
    })

    # Sort by absolute coefficient value
    coefficients['Abs_Coefficient'] = abs(coefficients['Coefficient'])
    coefficients = coefficients.sort_values('Abs_Coefficient', ascending=False)

    plt.figure(figsize=(10, 6))
    sns.barplot(x='Coefficient', y='Feature', data=coefficients)
    plt.title('Linear Regression Coefficients')
    plt.axvline(x=0, color='r', linestyle='-')
    plt.grid(True)
    plt.show()

    print("\n--- Feature Coefficients ---")
    print(coefficients[['Feature', 'Coefficient']])
```



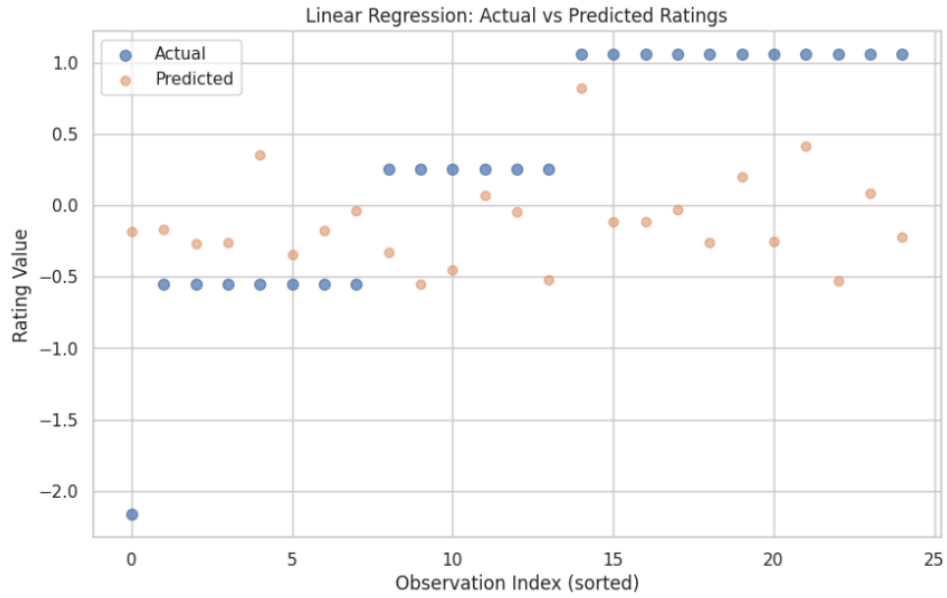

```
--- Feature Coefficients ---
                                              Feature  Coefficient
2               Have you ever visited Uttar Pradesh?     0.440400
0                                         Age Group     -0.302740
3                    What is your dietary preference?     0.062734
1                                   State of Residence     0.049046
5  Which of the following Uttar Pradesh's dishes ...     0.009827
4  Which of the following Uttar Pradesh's dishes ...     0.002188
```

#8.2.5 Linear Regression: Actual vs Predicted Ratings

The scatter plot compares the actual and predicted future visit ratings from the Linear Regression model, with the data points sorted by the actual rating for better visualization of the model's performance across the range of actual values.

The x-axis represents an "Observation Index (sorted)," essentially the respondents ordered by their actual rating. The y-axis represents the "Rating Value." Blue points represent the actual ratings, and orange points represent the corresponding ratings predicted by the Linear Regression model.By sorting the actual ratings, we can observe how well the predicted ratings follow the trend of the actual ratings. Ideally, the orange points should closely track the blue points. However, the plot shows a noticeable discrepancy between the actual and predicted ratings for many observations.

```
In [ ]: #Linear Regression: Actual vs Predicted Ratings
plt.figure(figsize=(10, 6))

# Sort by actual values for better visualization
sorted_indices = np.argsort(y_test.values)
sorted_actual = np.array(y_test)[sorted_indices]
sorted_pred = np.array(y_pred)[sorted_indices]

# Create x-axis points
x_points = np.arange(len(sorted_actual))

plt.scatter(x_points, sorted_actual, label='Actual', alpha=0.7, s=50)
plt.scatter(x_points, sorted_pred, label='Predicted', alpha=0.5)

plt.xlabel('Observation Index (sorted)')
plt.ylabel('Rating Value')
plt.title('Linear Regression: Actual vs Predicted Ratings')
plt.legend()
plt.grid(True)
plt.show()
```



[9] Decision Analysis

Insights derived from models were used to recommend:

- Customized marketing strategies for specific cuisines.
- Targeted campaigns to regions with lower awareness.
- Promotion of local dishes via travel packages.

Computer models showed us, we came up with some ideas for how to help tourism and food businesses in Uttar Pradesh. It's like using what we learned from the data to give advice:

- **Selling specific foods in a special way:** Because we now know which foods different groups of people like, businesses can create special ways to advertise and sell those

particular dishes to those specific customers. For example, if young people really like street food, there could be more ads for that kind of food aimed at them.

- **Reaching out to places that don't know much about UP food:** We found some areas where people weren't very familiar with the local UP dishes. So, we suggested making special campaigns to tell people in those regions about the tasty food they can find in UP. This might make them more interested in visiting.
- **Bundling food experiences with travel:** Since people often travel for food, we thought it would be a good idea to create special travel deals that include opportunities to try famous local dishes. This way, people planning a trip to UP will know they can look forward to great food as part of their experience.

[10] Research Work

To show that our project is built on solid knowledge and offers something new, we looked at other important research that has already been done. We found studies and surveys that talk about:

- **Why people travel:** What makes tourists choose one place over another?
- **Special foods of a region:** How important is local food to travelers?
- **How customers make choices:** What influences what people decide to buy or do when they travel and eat?

By looking at these existing studies, we could see what questions have already been answered and where there are still gaps in our understanding. We then explained how our study is different and adds something new to this area of knowledge. Maybe our project focuses on a specific part of India that hasn't been studied much for its food and travel connections, or perhaps we looked at a new group of people or asked different kinds of questions. This helps show that our work is not just repeating what others have done, but is making its own special contribution to understanding tourism and food in Uttar Pradesh.

[11] Conclusion

The study successfully:

- **We figured out what really makes people travel to Uttar Pradesh because of the food.** We pinpointed the key things that attract "foodie tourists."
- **We showed a connection between how much people know about UP's food and how likely they are to travel there.** We proved that being aware of the local cuisine influences travel decisions.
- **We created computer models that can predict if someone is likely to visit UP and how happy they might be with their visit.** These models can help us understand future travel patterns and potential tourist satisfaction.

Because of these results, businesses that deal with travel and food can now have a much clearer picture of who their customers are, what they want, and how to best reach them. This can help them make smarter decisions about their services and marketing.

[12] Refrences

- **Github :** **https://github.com/ArushiShiv/Data-Science/tree/main/Data%20Science**

- **Google-Form :** **https://docs.google.com/forms/d/e/1FAIpQLSch4vES5vNAZe_Q0-6BSEckCRmnTEPZmDTsua0wYKDCOJorNw/viewform?usp=sharing**

- **Government and Tourism Sites:**

- *Official Uttar Pradesh Tourism Website: (Search for the official tourism website of Uttar Pradesh) - This may have sections on culture and food.*
- *Incredible India (Government of India): (Search for the official tourism website of India) - This site often features information on regional cuisines.*

- **Food Blogs and Websites:**

- *Reputable Indian food blogs (e.g., those by established chefs or food writers) - Search for "Indian food blog Uttar Pradesh cuisine."*
- *Websites specializing in Indian recipes - Use specific dish names as keywords.*

- **Academic Resources:**

- *Google Scholar: (scholar.google.com) - Search for scholarly articles on "Uttar Pradesh food culture," "Awadhi cuisine history," etc.*
- *JSTOR: (jstor.org) - A digital library with academic journals, books, and primary sources (may require subscription).*

- **Recipe and Culinary Information:**

- *Well-known recipe websites (e.g., those with user ratings and detailed instructions) - Use with caution and cross-reference.*