Project Report

PERSONALIZED RECOMMENDATION SYSTEM FOR CUSTOMERS OF FOOD DELIVERY APP IN INDIA

IE7275

Data Mining in Engineering

Submitted by:

Group No 6

Chandini Nekkanti (NUId: 002642616)

Konda Susheel Kumar Yadav (NUId:002674007)

Professor:

Soumya Janardhanan

Context:

In the modern era of technology and convenience, the restaurant industry has embraced digital platforms to connect with customers and enhance their dining experiences. Online restaurant discovery and food delivery services have become an integral part of urban lifestyles. Users often face the challenge of choosing from a multitude of dining options available to them. A restaurant recommendation system aims to address this challenge by providing personalized and relevant restaurant and food suggestions to users based on their preferences, behaviour, and the characteristics of the restaurants and dishes.

Domain:

The problem falls within the domain of recommender systems and data science applied to the restaurant industry. Recommender systems play a crucial role in enhancing user experience by providing tailored recommendations that match users' preferences. The restaurant industry is highly competitive, and businesses are constantly seeking innovative ways to attract and retain customers. Effective restaurant recommendation systems can contribute to increased customer satisfaction, engagement, and business growth.

Application:

The restaurant recommendation system described in the code is designed to operate within the context of a restaurant aggregator platform, leveraging data from Zomato, a popular restaurant discovery and food delivery platform. The system aims to provide users with personalized restaurant and food recommendations based on collaborative filtering and content-based filtering techniques.

Challenges:

a) **Data Integration:** The system must integrate data from multiple sources, including user profiles, restaurant information, menu items, orders, and sales data. Ensuring data consistency, accuracy, and relevance can be challenging.
b) **Data Sparsity:** In a large dataset, users may have interacted with only a small fraction of available restaurants and dishes. This data sparsity can hinder the effectiveness of collaborative filtering techniques.

c) **Cold Start Problem:** New users or restaurants with limited historical data may pose a challenge, as collaborative filtering may struggle to make accurate recommendations for them.
d) **Personalization:** Providing personalized recommendations that accurately reflect a user's tastes, preferences, and dietary restrictions is essential for user satisfaction. Balancing between popularity and personalization is a challenge.
e) **Feature Engineering:** Extracting meaningful features from user profiles and restaurant data for content-based filtering requires careful consideration and domain knowledge.
f) **Evaluation:** Assessing the performance of the recommendation system and choosing appropriate evaluation metrics to measure user satisfaction and engagement can be complex.
g) **Scalability:** As the user base and restaurant database grow, the recommendation system must efficiently handle the increasing volume of data and user interactions.
h) **Hybrid Approaches:** Developing effective hybrid recommendation techniques that combine collaborative filtering and content-based filtering while addressing their respective limitations.

Problem Definition: Enhancing Restaurant Recommendation System Using Data Analytics.

Within the context of the restaurant recommendation system in the given problem setting, the specific problem being addressed is to develop an effective and personalized restaurant and food recommendation system that leverages data analytics techniques. The goal is to provide users with accurate, relevant, and engaging recommendations based on their preferences and behaviour, ultimately enhancing their dining experience and increasing user engagement on the restaurant aggregator platform.

Objectives:

The objective of this project is to develop a accurate, personalized recommendation system for a food delivery app that can recommend dishes and restaurants to a user based on a variety of parameters such as their previous orders, restaurant ratings, preferred cuisine type, frequency of orders, location, and the restaurant's popular dishes.

The primary problems being addressed are: Can we accurately predict user preferences based on their previous order data and restaurant ratings? How can we successfully use different data features to improve the precision of our recommendations?

Ultimately, the project aims to offer users a highly personalized food ordering experience by putting forward dishes and restaurants that are ideally aligned with their tastes, using the provided parameters and previous

Data Source: Kaggle (www.kaggle.com)

Dataset Title: Zomato Restaurants Data

Description:

The dataset was sourced from Kaggle, a platform for data science and machine learning enthusiasts. The dataset titled "Zomato Restaurants Data" contains information about various restaurants listed on the Zomato platform. Zomato is a popular online platform that provides information about restaurants, their menus, reviews, and ratings.

Citation: Kaggle. (https://www.kaggle.com/datasets/anas123siddiqui/zomato-database?select=users.csv). Zomato Restaurants Data. Retrieved from URL of the dataset.

Dataset Description: Zomato Restaurant and User Data

The "Zomato Restaurant and User Data" dataset is a comprehensive collection of information pertaining to restaurant items, sales transactions, and user demographics sourced from the Zomato platform. This dataset offers valuable insights into the restaurant industry, user preferences, and consumption trends, making it a valuable resource for various analytical and business-oriented purposes.

Dataset Details:

- Entries (Rows): The dataset contains a total of 1,195,056 entries, each representing a distinct observation or transaction.

- Columns (Features): It encompasses 26 columns, each providing distinct information to facilitate in-depth analysis.

Descriptions:

Column Name	Data Type	Description
f_id	Object	Unique identifier for data traceability.
item	Object	Name of the restaurant item offered for sale.
veg_or_non_veg	Object	Indicates if the item is vegetarian or non-vegetarian.
menu_id	Object	Identifier for the menu to which the item belongs.
r_id	Int64	Integer-based identifier for the restaurant.
cuisine_x	Object	Cuisine type associated with the restaurant item.
price	Object	Price of the restaurant item.
order_date	Object	Date when the order for the item was placed.

sales_qty	Int64	Quantity of the item sold in the transaction.
sales_amount	Int64	Total sales amount generated by the item sale.
user_id	Int64	Identifier for the user associated with the transaction.
name_x	Object	Name of the user who placed the order.
city	Object	City where the restaurant is located.
rating	Object	Rating given to the restaurant.
rating_count	Object	Count of ratings received by the restaurant.
cost	Object	Cost of producing the restaurant item.
cuisine_y	Object	Cuisine type associated with the restaurant.

address	Object	Address of the restaurant.
name_y	Object	Name of the restaurant.
Age	Int64	Age of the user associated with the transaction.
Gender	Object	Gender of the user.
Marital Status	Object	Marital status of the user.
Occupation	Object	Occupation of the user.
Monthly Income	Object	Monthly income of the user.
Educational Qualifications	Object	Educational qualifications of the user.
Family size	Int64	Size of the user's family.

Data Analysis Opportunities:

This dataset presents opportunities for insightful analyses, including:

Sales Trends: Understanding popular items, their sales quantities, and revenues generated.

User Preferences: Exploring user demographics, preferences, and ordering behaviors.

Restaurant Performance: Evaluating restaurant ratings, counts, and their correlation with sales.

Cost Efficiency: Analyzing production costs and their impact on pricing strategies.

This "Zomato Restaurant and User Data" dataset offers a valuable glimpse into the world of restaurant operations and user interactions, enabling data-driven decision-making for businesses and researchers alike.

In our project, we have undertaken various data mining tasks to develop a restaurant and food item recommendation system based on user demographics and past orders. Here's a detailed breakdown of the data mining tasks we have performed:

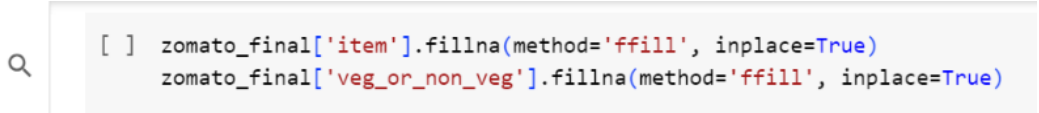
1. Data Loading and Preprocessing:

Loading Data from CSV Files: We imported necessary libraries and to access our dataset and used pandas to load data from CSV files (food.csv, menu.csv, etc.) into dataframes.

Merging Datasets: Combined multiple dataframes using inner joins based on specific columns (f_id, user_id and r_id). This consolidation ensures that relevant information from different sources is linked together.

Column Renaming: Renamed columns, such as changing id to r_id, to provide more meaningful and consistent column names across datasets.

Handling Missing Data: Addressed missing data in our dataset by applying forward fill (method='ffill') to fill missing values based on the previous non-null value in the same column. Since the missing values are from veg_or_non_veg column we cannot impute the mean values and since these column plays an important role in modelling we cannot drop these values.



String Replacement: Some of the columns of the dataset like cost column has Indian Rupee symbol '₹', so it is reading the entire column as a string variable so In order to replace it with the integer we will replace the symbol with the empty space. Similarly like for other columns like r_id, f_id the strings are converted into the integers

```
print(zomato_final['cost'].unique())
```

```
['₹ 200' '₹ 100' '₹ 250' '₹ 300' '₹ 400' '₹ 399' '₹ 299' '₹ 150' '₹ 280'
 '₹ 350' '₹ 500' '₹ 149' '₹ 850' '₹ 450' '₹ 900' '₹ 600' '₹ 220' '₹ 550'
 '₹ 120' '₹ 199' '₹ 240' '₹ 110' '₹ 370' '₹ 1000' '₹ 20' '₹ 800' '₹ 90'
 '₹ 99' '₹ 1500' '₹ 25' '₹ 750' '₹ 700' '₹ 499' '₹ 1200' '₹ 1100' '₹ 80'
 '₹ 50' '₹ 60' '₹ 5' '₹ 249' '₹ 2' '₹ 650' '₹ 160' '₹ 180' '₹ 178' '₹ 230'
 '₹ 260' '₹ 349' '₹ 599' '₹ 330' '₹ 126' '₹ 310' '₹ 140' '₹ 325' '₹ 70'
 '₹ 290' '₹ 175' '₹ 125' '₹ 8' '₹ 375' '₹ 275' '₹ 225' '₹ 210' '₹ 352'
 '₹ 170' nan '₹ 130' '₹ 10' '₹ 1800' '₹ 159' '₹ 193' '₹ 188' '₹ 320'
 '₹ 268' '₹ 421' '₹ 259' '₹ 380' '₹ 425' '₹ 1600' '₹ 148' '₹ 164' '₹ 449'
 '₹ 257' '₹ 189' '₹ 219' '₹ 85' '₹ 710' '₹ 75' '₹ 410' '₹ 1300' '₹ 251'
 '₹ 245' '₹ 129' '₹ 270' '₹ 176' '₹ 59' '₹ 40' '₹ 1900' '₹ 1250' '₹ 14'
 '₹ 298' '₹ 197' '₹ 360' '₹ 30' '₹ 336' '₹ 497' '₹ 699' '₹ 137' '₹ 252'
 '₹ 1245' '₹ 198' '₹ 999' '₹ 55' '₹ 255' '₹ 540' '₹ 235' '₹ 2000' '₹ 460'
 '₹ 158' '₹ 239' '₹ 190' '₹ 340' '₹ 510' '₹ 171' '₹ 1400' '₹ 49' '₹ 990'
 '₹ 179' '₹ 1050' '₹ 248' '₹ 3999']
```

```
zomato_final['cost'] = zomato_final['cost'].str.replace('₹', '')
zomato_final['cost'] = pd.to_numeric(zomato_final['cost'], errors='coerce')
```

```
cost_mean = zomato_final['cost'].mean()
zomato_final['cost'].fillna(cost_mean, inplace=True)
```

Data type Conversion: data type conversion is a fundamental step in data preprocessing that ensures data integrity, compatibility, and optimal usage throughout the data analysis and modeling pipeline.

```
zomato_final['f_id'].unique()
```

```
array(['fd0', 'fd1', 'fd2', ..., 'fd1043766', 'fd1043767', 'fd1043768'],
      dtype=object)
```

```
zomato_final['f_id'] = zomato_final['f_id'].str.replace('fd', '').astype(int)
```

Label Encoding: Label encoding is a method of converting categorical data (data that consists of categories or labels) into numerical values. In label encoding, each unique category is assigned a unique integer value. This is done to represent categorical data in a format that can be used by machine learning algorithms, which often require numerical inputs.

Label encoding is used when you have categorical data with an inherent ordinal relationship, where the order of categories matters. It is not suitable for nominal data (categories without any inherent order) because it might introduce unintended ordinal relationships where none exist.

```
[ ] from sklearn.preprocessing import LabelEncoder
    columns_to_encode = ['veg_or_non_veg', 'Gender', 'Marital Status', 'Occupation', 'Monthly Income','Educational Qualifications']

    # Perform label encoding
    le = LabelEncoder()
    for column in columns_to_encode:
        zomato_final[column] = le.fit_transform(zomato_final[column])
```

Dropping Variables: If a variable has a large amount of missing data and imputation is not appropriate, it might be better to drop the variable to avoid introducing bias. Since some variables are not contributing that good in the extraction of the model we can drop the variables

```
[ ] zomato1.columns

    Index(['Unnamed: 0_x_x', 'f_id', 'item', 'veg_or_non_veg', 'Unnamed: 0_y_x',
           'menu_id', 'r_id', 'cuisine_x', 'price', 'Unnamed: 0_x_y', 'order_date',
           'sales_qty', 'sales_amount', 'currency', 'user_id', 'Unnamed: 0_y_y',
           'name_x', 'city', 'rating', 'rating_count', 'cost', 'cuisine_y',
           'lic_no', 'link', 'address', 'menu', 'Unnamed: 0', 'name_y', 'email',
           'password', 'Age', 'Gender', 'Marital Status', 'Occupation',
           'Monthly Income', 'Educational Qualifications', 'Family size'],
          dtype='object')

[ ] zomato_final = zomato1.drop(['Unnamed: 0_x_x', 'Unnamed: 0_y_x', 'Unnamed: 0_x_y', 'currency', 'Unnamed: 0_y_y', 'Unnamed: 0', 'menu', 'email', 'passu
```

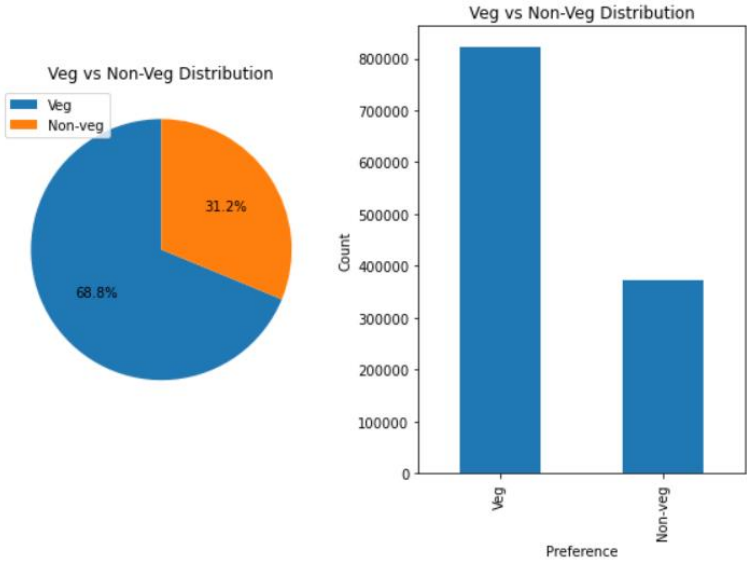
DATA EXPLORATION

The exploration phase of our project served as the bedrock for extracting insights, revealing underlying patterns, and connecting preferences tied to demographics. As we waded through the vast expanses of our data, we established some essential findings that eventually became the pillars for our recommendation system. Let's delve into the depths of our exploration:

1. Veg vs. Non-Veg Distribution:

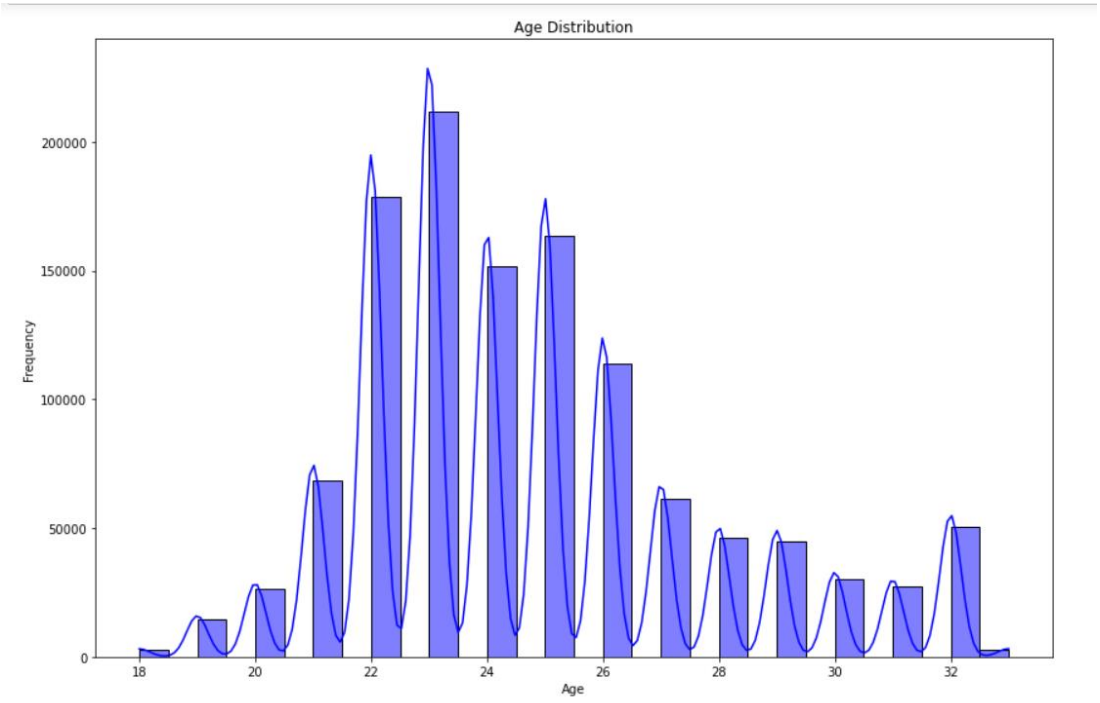
Our initial dive into the dataset revealed a significant inclination towards vegetarianism. Through a pie chart that distinctly captured the difference, we found that a substantial 68% of

the users preferred vegetarian food, while the remaining 32% had a penchant for non-vegetarian dishes. This proportion provides a foundational understanding, pivotal for curating personalized recommendations in the subsequent stages.



2. Age Distribution:

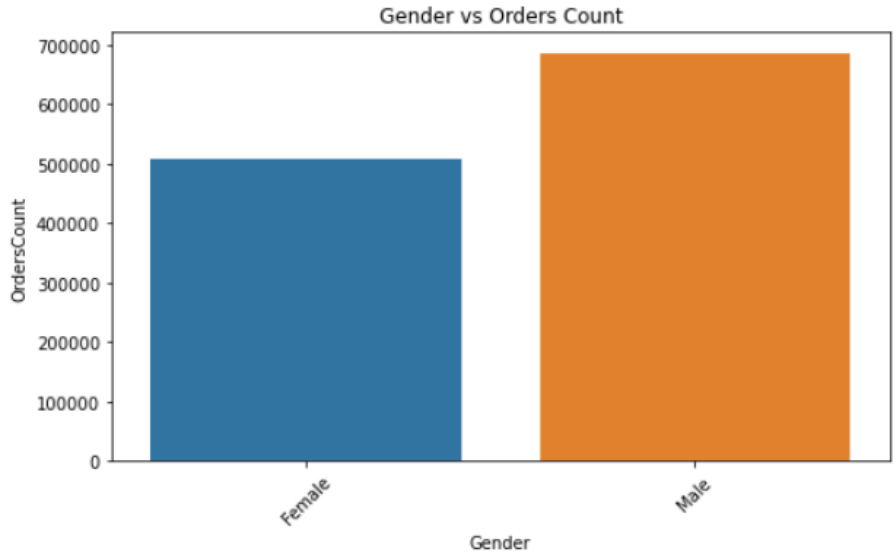
Using a frequency plot, we mapped out the age distribution across the dataset. A notable discovery was the concentration of users between the age brackets of 18-32 years, with a peak concentration in the 22-26 years range. This finding underlines the fact that younger individuals, possibly millennials and Gen Z, are more inclined to use the platform, shaping their food preferences and ordering patterns.



3. Demographics vs. Order Count:

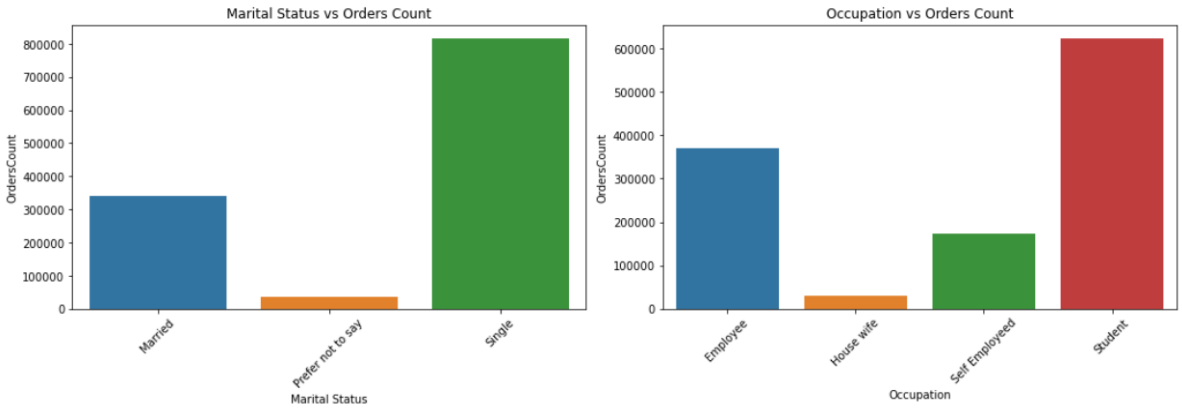
A deeper dive into demographics revealed several intriguing patterns:

Gender: Men were predominant, accounting for 60% of the orders, while women comprised the remaining 40%.



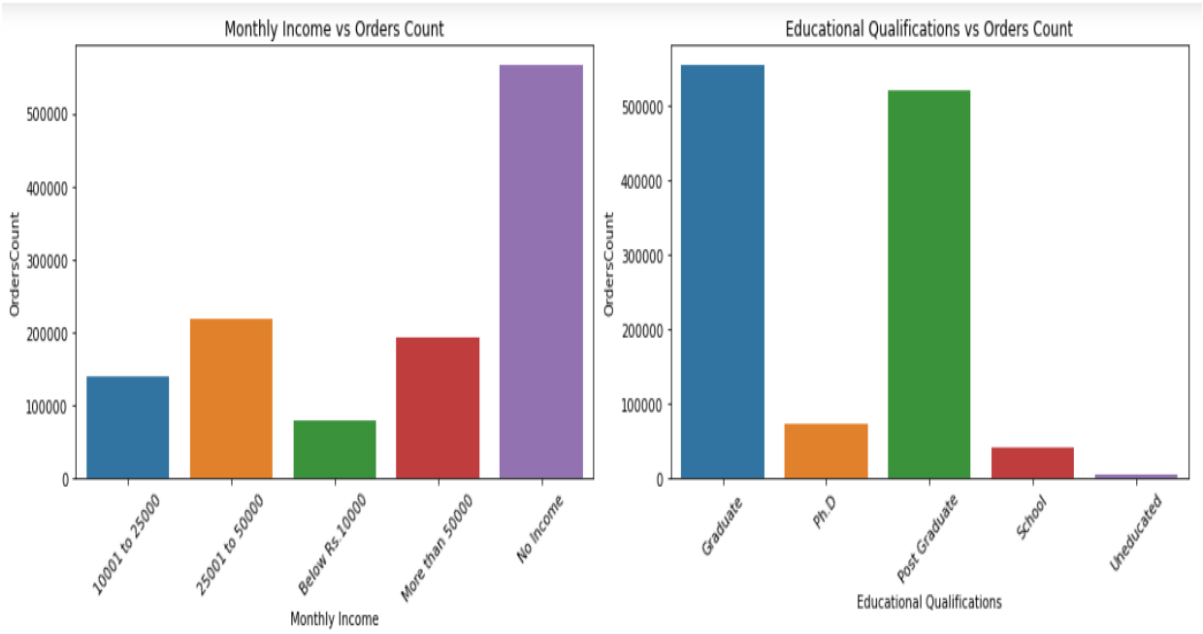
Marital Status: The platform seems to resonate more with singles, who contribute to over two-thirds of the orders.

Occupation: Students, possibly owing to their flexible schedules and varied food habits, lead the pack with 50% of orders, followed by employees at 40%. Housewives, on the other hand, represent a negligible percentage.

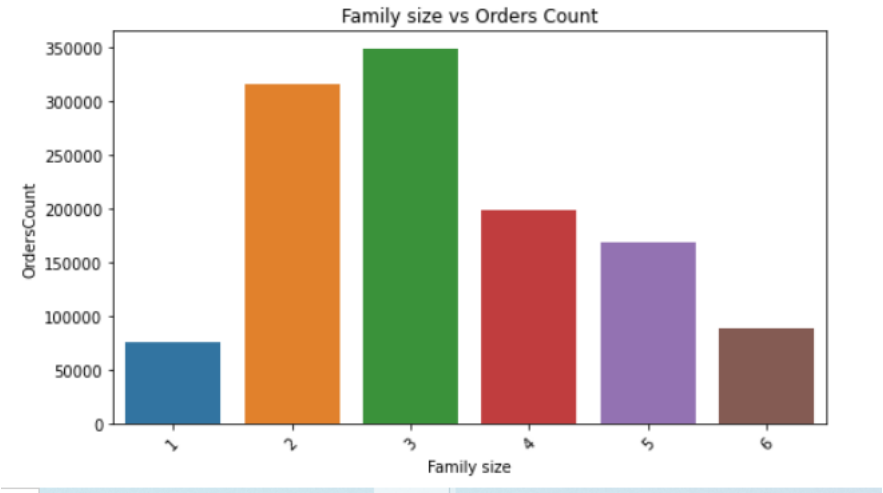


Income: A surprising revelation was that a vast section of users, almost 45%, did not disclose their income, bringing an interesting dynamic to the data interpretation.

Education: The platform appears to be a favorite among the highly educated, indicating their busier lifestyles and preference for convenience.

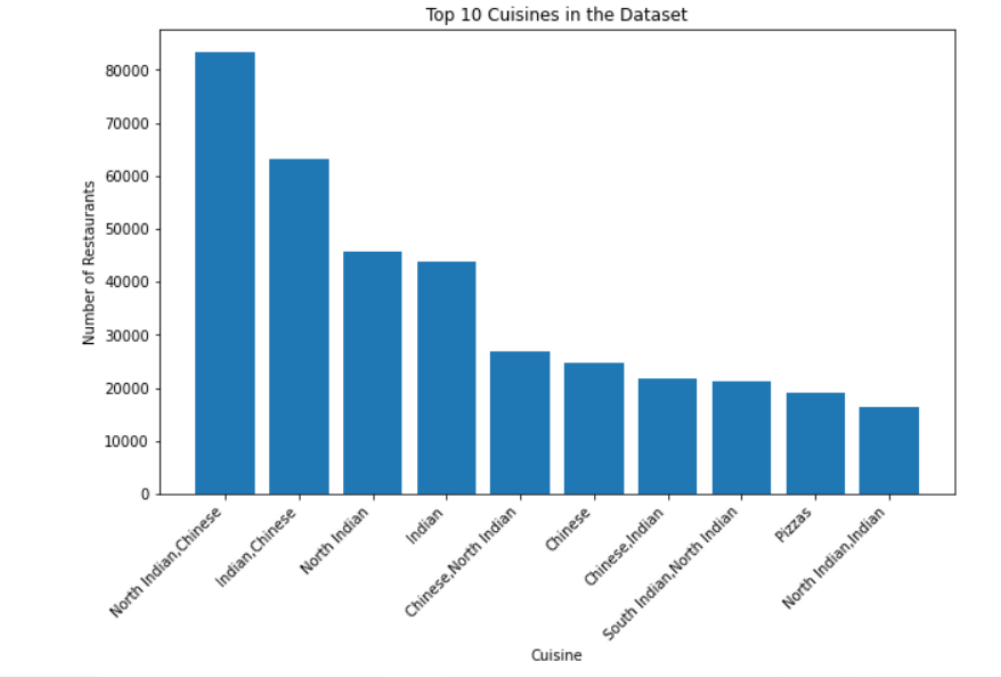


Family Size: Most orders come from households with a family size of 2 or 3, underlining the demographic of young couples or nuclear families.



4. Top Cuisines:

Our data also brought to light the top cuisines that found favor among the users. Recognizing these top cuisines not only provides insights into current food trends but also aids in strategizing promotions for specific dishes or introducing new menu items in alignment with these preferences.



Our comprehensive data exploration painted a vivid picture of our user base, their preferences, and the demographic attributes that influence their choices. These insights, gleaned from meticulous analysis, will now guide our recommendation engine, ensuring that it is both precise and relevant to the user's palate.

MODEL FUNCTIONING

1. Hybrid Recommendation System: In recommendation system literature, hybrid recommendation systems blend various input sources and recommendation algorithms to produce a final list of recommendations. They are particularly beneficial because they combine the strengths of both collaborative filtering and content-based filtering.

Collaborative Filtering: This approach uses user behavior (e.g., ratings, views) to find similarities between users or items. Once these similarities are found, it recommends items based on what similar users liked or interacted with. In the provided code, this is the part where we look for users similar to our target user and consider their preferences.

Content-based Filtering: Here, the focus is on the attributes of items and a profile of the user's preferences. In the context of the code, when the user's personal historical preferences for food type (veg/non-veg) and cuisine are used to filter restaurants, this is a content-based approach.

The `recommend_dishes` function recommends three restaurants to a user, based on their past preferences and the preferences of users similar to them. For each restaurant, it also recommends the top dishes. This function is a hybrid recommendation system blending both user-specific and collaborative filtering techniques.

1. Find Similar Users:

It calls an external function `find_similar_users()` to get a list of users that are similar to the input user. The details of how similarity is determined are not provided, but it's likely based on some distance metric or similarity score using historical user behavior or demographics.

```
def find_similar_users(zomato_final, user_id, n=10):
    # if the input user_id exists in the dataset. If it exists, use that user's data, otherwise, selecting a random user.
    if user_id in zomato_final['user_id'].values:
        user = zomato_final[zomato_final['user_id'] == user_id].iloc[0]
    else:
        user = zomato_final.sample(1).iloc[0]
    others = zomato_final[zomato_final['user_id'] != user_id] # Filtering the original user to get all other users.
    demo = ['Age', 'Gender', 'Marital Status', 'Occupation', 'Monthly Income', 'Educational Qualifications', 'Family size']
    vector = user[demo].values   # Get the attributes of the selected user in a vector.
    others['similarity'] = others.apply(lambda x: np.sum((x[demo].values - vector)**2), axis=1)# For every other user, calculating
    similar_users = others.sort_values('similarity').head(n)['user_id'].unique().tolist()# Sorting the other users by their simila
    return similar_users
```

2. Filtering by City:

The dataset is filtered based on the given city to ensure that the recommendations are relevant to the user's current location.

```
similar_users = find_similar_users(zomato_final, user_id)
zomato_final = zomato_final[zomato_final['city'] == city]  # Filtering the dataset by the input city.
```

3. Fetch User Data & Similar Users' Data:

```
user_data = zomato_final[zomato_final['user_id'] == user_id]
similar_users_data = zomato_final[zomato_final['user_id'].isin(similar_users)]
```

Individual datasets for the given user and the similar users are created for easy reference.

4. User's Preference:

If there's historical data for the user, the function determines their preferred type of food (veg/non-veg) and cuisine. Based on these preferences, the dataset is filtered to find relevant restaurants.

```
# If we have the data for the user, find out their preferred type of food (veg/non-veg) and cuisine and filt
if not user_data.empty:
    user_pref_veg = user_data['veg_or_non_veg'].values[0]
    user_pref_cuisine = user_data['cuisine_x'].values[0]
    user_restaurants = set(zomato_final[(zomato_final['veg_or_non_veg'] == user_pref_veg) &
                                        (zomato_final['cuisine_x'] == user_pref_cuisine)]['r_id'].unique())
```

5. Similar Users' Preferences:

```
# For the similar users, find out the most common food type and cuisine they prefer and filtering the data.
if not similar_users_data.empty:
    similar_pref_veg = similar_users_data['veg_or_non_veg'].mode().iloc[0]
    similar_pref_cuisine = similar_users_data['cuisine_x'].mode().iloc[0]
    similar_users_restaurants = set(zomato_final[(zomato_final['veg_or_non_veg'] == similar_pref_veg) &
                                                 (zomato_final['cuisine_x'] == similar_pref_cuisine)]['r_id'].unique())
```

For the group of similar users, the function identifies the most common (mode) food type and cuisine preferred by this group. Then, it filters the dataset to get restaurants that cater to these common preferences.

6. Combine Recommendations:

Recommendations from both the user's data and similar users' data are merged. This combination creates a set of restaurant recommendations tailored for the user but also influenced by the broader preferences of similar users.

```
combined_restaurants = list(user_restaurants | similar_users_restaurants)# Combining the two restuarant names
if combined_restaurants: #arranging in asceding order according to sales quantity sum
    combined_restaurants = zomato_final[zomato_final['r_id'].isin(combined_restaurants)].groupby('r_id').sum()['sales_qty'].sort_values(ascending=False).index.tolist()
```

7. Prioritize by Sales Quantity:

The restaurants in the combined list are sorted based on their sales quantities, implying that restaurants with higher sales are prioritized. This introduces a popularity-based recommendation layer.

```
    top_restaurants = zomato_final.groupby('r_id').sum()['sales_qty'].sort_values(ascending=False).head(3 - len(combined_restaurants)).index.tolist()
    combined_restaurants.extend(top_restaurants)
combined_restaurants = combined_restaurants[:3]
top_dishes_per_restaurant = {r_id: get_top_dishes_for_restaurant(zomato_final, r_id) for r_id in combined_restaurants}
```

8. Supplement Recommendations:

If there are fewer than three restaurants in the combined list, the function fetches the top-selling restaurants in the city to supplement the list. This ensures a minimum of three recommendations.

```
# The loop when there are less than 3 in the reccomenations.
if len(combined_restaurants) < 3:
    top_restaurants = zomato_final.groupby('r_id').sum()['sales_qty'].sort_values(ascending=False).head(3 - len(combined_restaurants)).index.tolist()
    combined_restaurants.extend(top_restaurants)
combined_restaurants = combined_restaurants[:3]
top_dishes_per_restaurant = {r_id: get_top_dishes_for_restaurant(zomato_final, r_id) for r_id in combined_restaurants}
```

9. Get Top Dishes:

For each of the recommended restaurants, the function retrieves the top dishes, most likely based on their sales or popularity.

```
def get_top_dishes_for_restaurant(zomato_final, r_id):
    top_dishes = zomato_final[zomato_final['r_id'] == r_id].groupby('f_id').sum()['sales_qty'].sort_values(ascending=False).head(3).index.tolist()
    return top_dishes if len(top_dishes) == 3 else top_dishes
```

10. Convert IDs to Names:

In the final step, the function replaces restaurant and dish IDs with their actual names for easy interpretation by users. This transformation makes the output user-friendly.

```
# Replace restaurant IDs with their names
restaurant_names = zomato_final.drop_duplicates(subset='r_id').set_index('r_id')['name_x'].to_dict()
dish_names = zomato_final.drop_duplicates(subset='f_id').set_index('f_id')['item'].to_dict()

named_output = {
    'Top Restaurants and their Top Dishes': {
        restaurant_names[r_id]: [dish_names[f_id] for f_id in dishes]
        for r_id, dishes in top_dishes_per_restaurant.items()
    }
}
```

This hybrid system, merging personal, collaborative, and popularity-based recommendations, aims to provide a well-rounded list of suggestions that are both tailored to the individual and influenced by broader dining trends.

Challenges Faced:

1. **Creation of user item matrix and user restaurant matrix:** While creating the user item matrix we were unable to create the matrix because of the huge number of rows in the dataset (1.2 million). While mapping it is creating 1.2*1.2 million values which is leading to the computational errors and the system is also crashing so unable to create the user item matrix and the same is happening with the user restaurant matrix.
2. **Incorporation of Hybrid Algorithm:** While incorporating the hybrid algorithm because of lack of domain knowledge we were unable to compute the weights and do the calculations because of larger dataset which again was a huge problem so we ended up creating a set to do the hybrid algorithm and display the output
3. **Assigning Parameters:** Since each and every column plays an important role in feature extraction due to lack of domain knowledge in the beginning we were unable to decide which parameters has more weightage in terms of well performance of the model. Later on based on null values and cardinality we were able to decide the important parameters required for the best performance of the filtering.
4. **Computational Problems:** Because of considering the realtime dataset which containing many parameters and 1.2 million rows we were unable to compute many model because the system and even google colab is getting crashed and because of this issue we were unable to do svd and unable to create pivot table, cosine matrix, user item matrix and user restaurant matrix. Then we tried considering the sample dataset which is 0.1% of the given dataset but we found the results were disastrous so we again decided to consider full dataset and because of its large size even the current model is also taking much more time to compute.
5. **Identify loop holes:** There are many loop holes if we miss considering any of the parameters. For example in india each state has its own traditional and most liked food so location based analysis is one of the major loophole in this project, so we considered giving it as input in order to avoid that issue. Even after that the filtering is giving restaurants based on the sales quantity only which we identified and inorder to avoid it created the sets and displayed the outputs.

Future Scope: We can elaborate this project in the future since there are many factors which can be considered to yield the very best results.

1. **Extensive demographic mapping:** In demographic mapping many other parameters can be considered such as price, ratings, item count etc.., which were not present in our present dataset. By considering other demographic information like this we can even yield the best recommendation results. The more the number of features we extract the best the recommendations will be.
2. **Based on ratings (Sentimental Analysis):** Since the ratings will be in the text format we can perform the sentiment analysis and extract the emotion of the user(Positive or Negative emotion) and use that as the target variable for knowing restaurant popularity instead of sales quantity can yield the best results.
3. **Time Series Forecasting:** We can do the recommendations based on the weather, season and the timing of the day or night. Because different people prefer different types of food for different seasons, climates and timings. So we can consider these parameters and do the time seriues forecasting and yield the better recommendation systems.
4. **Advanced Matrix Factorization:** we can create matrix like svd and cosine matrix and derive the recommendation systems and create a hybrid algorithm along with the collaborative and content based filtering which will yield best recommendation system when combining these three different filtering techniques
5. **Location based Recommendations:** In India because of varied diversity and varied cultures and regional foods, most of the people from one region prefer authentic dishes from their region. So can develop a recommendation system based on the regional analysis for each and every particular region.
6. **Contextual Recommendations:** Based on the occasions like festivals, marriages and other functions people prefer different food items for different occasions so we can consider this parameter and do the filtering which can give good recommendations.

Conclusion:

```
result = pd.DataFrame(data= recommendations)
result
```

	Top Restaurants and their Top Dishes
AB FOODS POINT	[Aloo Tikki Burger, Kit K at Shake, Nutela S...
CHAWLA SAAB THE JUICE MASTER	[Aloo Tikki Burger, Cold Sandwich, Achari Maggi]
Shri Balaji fast food and Variety store	[Aloo Tikki Burger, Special King Size Burger, ...

The journey to develop a personalized recommendation system for restaurant dishes, as embarked upon in this project, has been both challenging and rewarding. Through an in-depth analysis and application of hybrid recommendation techniques, we have achieved a system that blends the insights of both collaborative filtering and content-based filtering.

Our recommendation system successfully harnesses the power of user behavior and historical preferences. By identifying similarities among users and their dish preferences, we offer recommendations that align with both the personal tastes of individual users and the preferences of their peers. This ensures a heightened likelihood of users appreciating and acting upon our recommendations.

Additionally, the project addressed inherent challenges in recommendation systems, such as the "Cold Start" problem. By incorporating strategies like using popular city-based restaurants in the absence of user history, the system provides valuable recommendations even for new users with no prior interactions.

However, like all models, ours is not without limitations. The potential popularity bias might limit the discovery of lesser-known culinary gems, and there's always room to improve diversity and serendipity in our recommendations. Future work could delve into refining the algorithm, incorporating more sophisticated models, and perhaps adding a layer of machine learning to enhance the adaptability and accuracy of our recommendations.

In conclusion, this project has taken significant strides in the domain of personalized culinary recommendations. The resulting system is poised to enhance the dining experiences of its users, directing them towards choices that resonate with their tastes while also offering delightful culinary discoveries.