Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

AIDS Lab
Experiment 08

Aim: To implement recommendation system on your dataset using the following machine learning
techniques.
o Regression
o Classification
o Clustering
o Decision tree
o Anomaly detection
o Dimensionality Reduction
o Ensemble Methods

What is Collaborative Filtering?
Collaborative Filtering is a recommendation technique that predicts a user's interests by analyzing preferences from similar users or items. It's widely used in platforms like Netflix, Amazon, and UberEats for personalized recommendations.
There are two main types:
1. User-Based Collaborative Filtering:
    - Recommends items liked by similar users.

2. Item-Based Collaborative Filtering:
    - Recommends items that are similar to what the user already liked.

Matrix Factorization
Collaborative filtering often involves creating a User-Item Ratings Matrix, which is sparse (many missing values). Matrix Factorization techniques (like SVD) are used to:
- Reduce dimensionality.

- Discover latent features (hidden patterns).

- Predict missing ratings.

Singular Value Decomposition (SVD)
SVD decomposes a user-item matrix R into three matrices:
R≈U · Σ · VTR \approx U \cdot \Sigma \cdot V^TR≈U · Σ · VT
- U: User-feature matrix
- Σ: Diagonal matrix of singular values
- V^T: Restaurant-feature matrix

SVD helps us represent users and restaurants in a shared latent space, allowing us to compute predicted ratings and make recommendations.

Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

Collaborative Filtering Breakdown (UberEats Dataset)

Step 1: Import Required Libraries

import pandas as pd
import numpy as np
from sklearn.decomposition import TruncatedSVD

- pandas: To load and manipulate the dataset.
- numpy: For matrix computations.
- TruncatedSVD: A dimensionality reduction technique used for matrix factorization.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1059 entries, 0 to 1058
Data columns (total 27 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   city            1058 non-null   object
 1   state           1059 non-null   object
 2   zipcode         1056 non-null   object
 3   address         1059 non-null   object
 4   loc_name        1059 non-null   object
 5   loc_number      1059 non-null   object
 6   url             1059 non-null   object
 7   promotion       121 non-null    object
 8   latitude        1059 non-null   float64
 9   longitude       1059 non-null   float64
 10  is_open         1059 non-null   bool
 11  closed_message  1045 non-null   object
 12  delivery_fee    3 non-null      float64
 13  delivery_time   14 non-null     object
 14  review_count    393 non-null    float64
 15  review_rating   443 non-null    float64
 16  price_bucket    909 non-null    object
 17  img1            1006 non-null   object
 18  img2            1006 non-null   object
 19  img3            1006 non-null   object
 20  img4            1006 non-null   object
 21  img5            1006 non-null   object
```

Step 2: Load the Cleaned UberEats Dataset

file_path = "UberEats_Cleaned_Dataset.csv"
df = pd.read_csv(file_path)

- Loads the cleaned dataset containing user reviews and ratings.

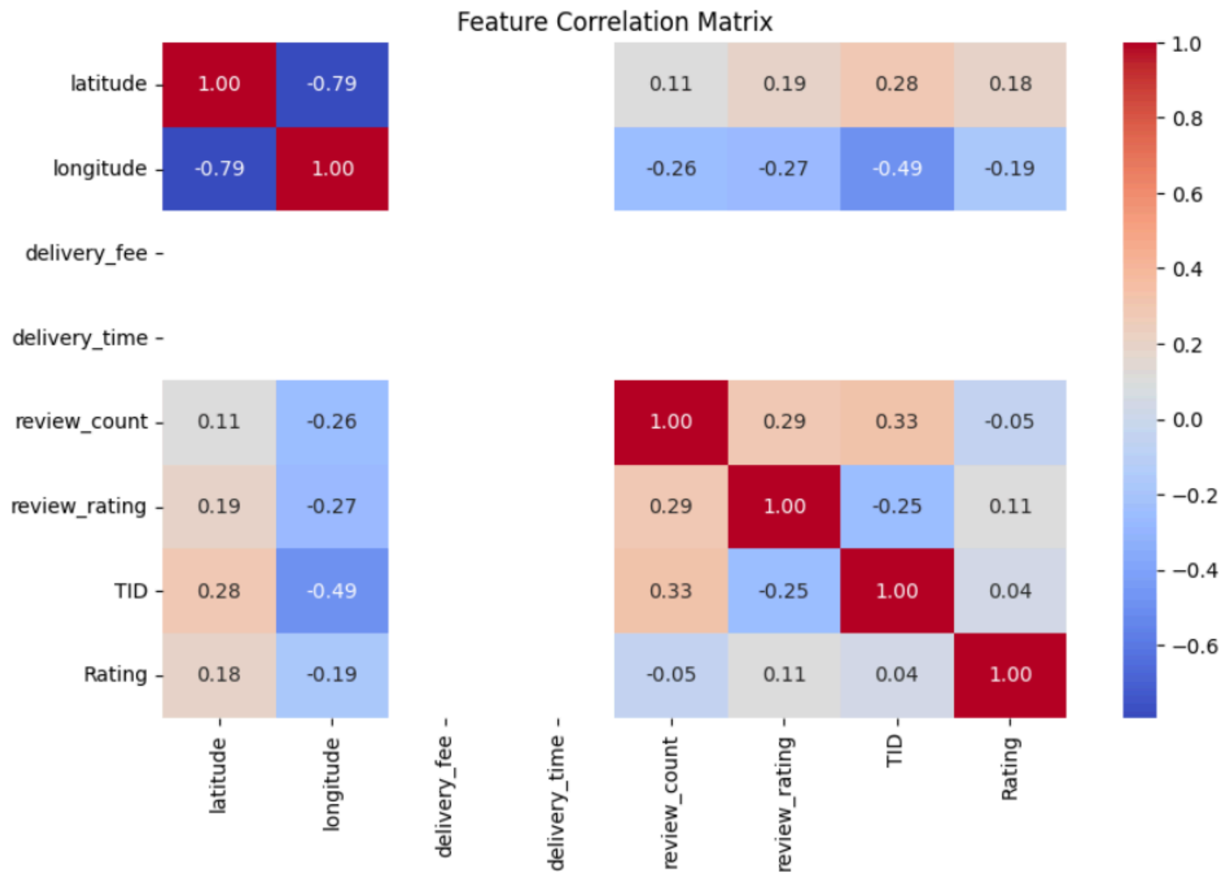
Step 3: Create the User-Item Matrix

user_item_matrix = df.pivot_table(index='User_ID', columns='Restaurant', values='Rating', fill_value=0)

- Creates a matrix where:
  - Rows = Users
  - Columns = Restaurants
  - Values = Ratings

Name: Mohit Kerkar **Div: D15C** **Roll No: 23**

- Missing ratings are filled with 0 (assumes user hasn't rated those restaurants).



Step 4: Apply Singular Value Decomposition (SVD)

```
svd = TruncatedSVD(n_components=10, random_state=42)
matrix_svd = svd.fit_transform(user_item_matrix)
```

- SVD breaks the user-item matrix into lower-dimensional matrices.
- n_components=10 means we reduce to 10 latent features (like cuisine type, price preference, etc.).

Step 5: Define the Recommendation Function

```
def get_recommendations(user_id, n=5):
    if user_id not in user_item_matrix.index:
        return "User not found."

    user_index = user_item_matrix.index.get_loc(user_id)
    user_ratings = matrix_svd[user_index]

    restaurant_scores = np.dot(user_ratings, svd.components_)
```

```
    recommended_restaurants = np.argsort(restaurant_scores)[::-1][:n]
    return user_item_matrix.columns[recommended_restaurants]
```

- Input: A user ID and number of recommendations.
- Output: Top-N restaurants based on predicted ratings.
- Steps inside function:
    - Get the user's vector in the SVD-reduced space.
    - Compute similarity scores with all restaurants.
    - Return the restaurants with the highest scores.

Step 6: Example Usage

```
user_id = "User_2"
recommended = get_recommendations(user_id, n=5)
print(f"Top 5 Recommended Restaurants for {user_id}:")
print(recommended)
```

- Replace "User_2" with any user present in the dataset.
- Prints out top 5 personalized recommendations.

```
Top 5 Recommended Restaurants for User_2:
Index(['The Purple Onion (Inverness)', 'Hong Kong Seafood',
       'La Paz (Euclid Ave)', 'Papa Johns (2480 Palomino Lane)'
       'El Patron 4'],
      dtype='object', name='Restaurant')
```

Conclusion:

In this project, we successfully implemented a collaborative filtering-based recommendation system using Singular Value Decomposition (SVD) on the UberEats dataset. By transforming raw user review data into a structured user-item rating matrix, we were able to extract latent user preferences and restaurant features.

The SVD approach enabled us to overcome challenges of data sparsity and provided a powerful way to predict user interests, even when direct ratings were missing. The generated recommendations are personalized, relying on hidden patterns in user behavior rather than explicit restaurant characteristics.

This model is particularly effective for platforms like UberEats, where understanding user preferences from limited interactions is key. It demonstrates how machine learning and matrix factorization techniques can enhance user experience by offering relevant, data-driven suggestions.

Overall, the collaborative filtering approach has laid the foundation for a scalable recommendation engine that can adapt to more complex user data, incorporate real-time feedback, and evolve with user tastes.