**Swiggy’s Restaurant Recommendation System using Streamlit**

***Process:***

1. Data Collection: Gathered a dataset of restaurants including information like name, cuisine, city, rating, cost, etc.
2. Data Preprocessing: Cleaned the data by removing missing values and encoding categorical features (cuisine, city) using one-hot encoding.
3. Feature Scaling: Standardized the encoded data using StandardScaler to ensure all features have similar scales.
4. Dimensionality Reduction: Applied PCA to reduce the number of features while retaining the variance in the data.
5. Similarity Calculation: Used cosine similarity to find the similarity between input restaurant data and all other restaurants in the dataset.
6. Recommendation System: Developed a function to recommend top N similar restaurants based on cuisine and city, filtered by a similarity threshold.
7. Streamlit App: Created an interactive web app to allow users to select cuisine and city, and receive personalized restaurant recommendations.

***Needs:***

* **Data**: A clean, encoded dataset of restaurants with attributes like name, city, cuisine, rating, and cost.
* **Libraries**: Python libraries like pandas, scikit-learn, streamlit for data handling, modeling, and web app development.
* **Computational Tools**: Access to a machine learning environment for running PCA, similarity calculations, and building the recommendation system.

***STEP 1*: IMPORTING NEEDED LIBRARIES**

**Explanation of Libraries Used in the Project:**

**pandas**: Used for data manipulation and analysis. It handles operations such as reading datasets (CSV files), cleaning the data, and converting categorical features into suitable formats for machine learning.

**numpy**: A core library for numerical computing in Python. It is used for handling arrays and performing mathematical operations, especially for matrix operations needed in PCA and cosine similarity calculations.

**pickle**: Used for saving and loading Python objects, such as trained models (e.g., KMeans, PCA, etc.), so that the model can be reused in different sessions without needing to retrain it.

**OneHotEncoder**: A preprocessing tool from sklearn used to convert categorical variables (like cuisine and city) into a binary matrix of 0s and 1s. This encoding is essential for the machine learning models to process categorical data.

**KMeans**: A clustering algorithm from sklearn used for grouping similar restaurants based on their features. This can help identify patterns and clusters within the dataset for better recommendations.

**cosine\_similarity**: A metric from sklearn to measure the cosine of the angle between two vectors. This is used in the recommendation system to calculate similarity between a user's selected restaurant and other restaurants in the dataset.

**PCA (Principal Component Analysis)**: A dimensionality reduction technique used to reduce the number of features in the dataset while retaining the variance. PCA helps in speeding up computations for similarity calculations and is useful in handling high-dimensional data.

**StandardScaler**: A preprocessing tool to standardize the data (scale it) so that each feature has a mean of 0 and a standard deviation of 1. This is crucial for PCA and similarity calculations, ensuring that features contribute equally to the analysis.

These libraries work together to clean, process, and analyze the data, create the recommendation system, and allow for efficient computation of similarity-based recommendations.

***STEP 2:* Data Preprocessing and Encoding for Swiggy Restaurant Dataset.**

**Explanation of the Code for Data Cleaning and Preprocessing:**

**Loading Raw Dataset**:

* + The dataset (swiggy.csv) is loaded using pd.read\_csv() to initiate data preprocessing.

**Handling Missing Values**:

* + The unique values in the critical columns (city, cuisine, rating, cost, name) are printed to inspect and identify missing data.
  + Rows with missing values in essential columns (city, cuisine, name) are dropped using dropna() to ensure data integrity.

**Handling Invalid Numeric Data**:

* + The rating column is converted to numeric values using pd.to\_numeric(), coercing any non-numeric values to NaN.
  + The cost column is cleaned by removing the currency symbol (₹), commas, and extra spaces, and is then converted to numeric values.

**Dropping Rows with Invalid Ratings or Cost**:

* + Rows with NaN values in the rating or cost columns are dropped to retain valid entries only.

**Standardizing Text Columns**:

* + The city and cuisine columns are standardized by stripping extra spaces and capitalizing the first letter of each word.

**One-Hot Encoding for Categorical Columns**:

* + The city and cuisine columns are encoded using One-Hot Encoding to convert categorical data into numerical format that can be used for machine learning models.
  + The OneHotEncoder is applied, and the encoded features are stored in a DataFrame.

**Saving the Encoder**:

* + The trained OneHotEncoder is saved to a pickle file (encoder.pkl) to reuse later during encoding new data.

**Concatenating Encoded Features**:

* + The encoded features are concatenated with the original dataset to form a complete dataset with both original and encoded information.

**Saving Cleaned and Encoded Data**:

* + The cleaned data, which includes both original and encoded features, is saved as cleaned\_data.csv.
  + The encoded features are separately saved as encoded\_data.csv for further use.

This process ensures that the dataset is cleaned, missing values are handled, categorical variables are encoded, and the data is ready for further analysis or model building.

***STEP 3:* City Name Cleaning and Standardization for Swiggy Dataset.**

**Explanation of the Code for Cleaning and Extracting Unique City Names:**

**Loading the Dataset**:

* + The dataset (swiggy.csv) is loaded using pd.read\_csv() from the specified file path (file\_path).

**Extracting the 'city' Column**:

* + The city column from the dataset is extracted to focus on city names for cleaning and standardization.

**Defining a Cleaning Function for City Names**:

* + A function clean\_city\_name(city) is defined to clean and standardize city names.
  + The function:
    - Splits the city name on commas (,) and keeps only the main city name.
    - Removes unwanted terms like &, phase, sector, campus, etc., by checking if any of these terms appear in the city name and stripping them if found.
    - The function ensures the city name is consistently formatted in lowercase and without unnecessary terms.

**Applying the Cleaning Function**:

* + The clean\_city\_name() function is applied to the city column using apply() to clean each city name.

**Removing Empty City Names**:

* + Any city names that are empty strings ('') after cleaning are removed from the list to ensure valid data.

**Getting Unique Cities**:

* + The unique city names are extracted using .unique() to ensure each city appears only once.

**Sorting the Unique Cities**:

* + The unique cities are sorted alphabetically to present the data in a more organized manner.

**Printing the Results**:

* + The unique, cleaned, and sorted city names are printed for inspection.

This process ensures that the city names are standardized, irrelevant terms are removed, and the dataset is cleaned for further analysis.

***STEP 4:* Restaurant Recommendation System Using PCA and Cosine Similarity.**

**Explanation of the Code for Restaurant Recommendation System:**

**Loading the Cleaned and Encoded Data**:

* + The cleaned data (cleaned\_data.csv) and encoded data (encoded\_data.csv) are loaded from CSV files for further processing.

**Checking for Missing Values**:

* + Missing values in the encoded data are checked using isnull().sum() to ensure no data is missing before proceeding with the analysis.

**Standardizing the Data**:

* + The encoded data is standardized using StandardScaler to normalize the features, which is important before applying Principal Component Analysis (PCA).

**Applying PCA for Dimensionality Reduction**:

* + PCA is applied to reduce the number of features in the encoded data, preserving the most significant variance. Here, 10 components are chosen (n\_components=10), but this can be adjusted based on the dataset's needs.
  + The explained variance ratio of the PCA components is printed to show how much of the original variance is retained.

**Defining the Restaurant Recommendation Function**:

* + The recommend\_restaurant() function is created to recommend similar restaurants based on user input (cuisine and city).
  + The function:
    - Filters the dataset to find a restaurant matching the given input\_cuisine and input\_city.
    - If a match is found, the restaurant's encoded vector is extracted and standardized.
    - PCA is applied to the input vector to project it into the lower-dimensional space.
    - Cosine similarity is calculated between the input restaurant and all others in the dataset.
    - Restaurants with similarity scores above the threshold (similarity\_threshold) are recommended. The top top\_n similar restaurants are returned.

**Example Usage**:

* + An example is provided with the input Indian cuisine and Abohar city.
  + The function returns the recommended restaurants and their corresponding similarity scores.

**Displaying Results**:

* + The recommended restaurants and their similarity scores are printed.
  + A sample of the transformed PCA features is displayed to show the reduced-dimensional representation.

This code helps build a restaurant recommendation system that suggests similar restaurants based on cosine similarity and PCA-transformed features, improving the efficiency and accuracy of recommendations.

***STEP 5:*** Streamlit Restaurant Recommendation System

**Purpose**: This Streamlit app allows users to get restaurant recommendations based on their selected cuisine and city. It leverages a preprocessed dataset and applies PCA (Principal Component Analysis) for dimensionality reduction and cosine similarity for finding similar restaurants.

**Steps**:

1. **Data Loading**: The preprocessed and encoded data (encoded\_data.csv and cleaned\_data.csv) is loaded into the app.
2. **Standardization & PCA**: The encoded data is standardized using StandardScaler, followed by PCA for dimensionality reduction to retain the most significant features.
3. **Recommendation Function**: The recommend\_restaurant() function filters restaurants based on user inputs (cuisine and city), calculates cosine similarity between the selected restaurant and others, and returns the top similar restaurants.
4. **Streamlit UI**:
   * The user is presented with two dropdown menus to select a cuisine and city.
   * Upon clicking the "Get Recommendations" button, the app displays the top recommended restaurants (up to 10) or a message if no matching data is found.

**Key Features**:

* **Cuisine and City Selection**: Users can choose their preferences.
* **Recommendations**: The system suggests the top similar restaurants based on input criteria, displayed as a table.

This app provides an easy-to-use interface for restaurant recommendations using machine learning techniques such as PCA and cosine similarity

***Conclusion:***

The project successfully implements a restaurant recommendation system that provides users with personalized suggestions based on their preferences for cuisine and city. Using PCA for dimensionality reduction and cosine similarity for recommendations ensures the system is computationally efficient. The Streamlit app adds a user-friendly interface for easy interaction, making it a valuable tool for restaurant discovery.