Python Libraries Used in the Project:

1) **Numpy:** Used for numerical operations
   Syntax: import numpy as np

2) **Pandas:** Used for data manipulation and analysis.
   Syntax: import pandas as pd

3) **Matplotlib :** Used for plotting to create interactive visualization
   Syntax: import matplotlib.pyplot as plt

4) **Seaborn:** Used for statistical data visualization built on top of matplotlib
   Syntax: import seaborn as sns

5) **Scikit-learn:** It is a machine learning library which is used to provide tools for data mining and data analysis.
   Syntax: from sklearn import <algorithms>

6) **Statsmodels:** Libraray used for estimating and testing statistical models.
   Syntax: import statsmodels.api as sm

7) **SciPy:**Library built on Numpy and provides additional functionality for scienticfic amd technical computing.
   Syntax: import scipy

8) **math:** It is built-in library that provides mathematical functions.
   Syntax: import math

9) **datetime:** datetime module is a part of standard library and provides classes for working with date and time.
   Syntax: from datetime import datetime as dt
   from datetime import date
   from datetime import time

Where as np, pd, sns, sm, dt are the alias names for the respective libraries

Functions used in the project

1) print(): Used to display text and values on the console.

2) len(): Used to returns the length of an object.

3) max(),min() : Used to return the maximum and minimum value from the sequence.

4) sum(): returns the sum of all the elements.

5) abs(): returns the absolute value of a number.

6) unique(): returns the unique value from the sequence.

7) pd.read_csv: used to read the csv file.

8) df.head(): used to view the top protion of the dataframes.

9) df.describe(): used to calculate the descriptive statistics on numerical columns.

10) df.info(): tells the information about the dataframe.

11) df.shape: tells the rows and columns about the dataframes.

12) df.groupby: used to group data based on one or more columns and to perform aggregations.

13) Pd.merge(df1,df2,on='Name')= used to merge two dataframes and also includes on which columns and how to join the two dataframes.

14) Isnull(): to know whether there is null values in the dataframes.

15) df.columns: tells the name of the columns that are there in the dataframes.

16) dropna(): used to drop the null values.

17) fillna(): used to fill the null values with the specified values.

18) rename(): used to rename the column name to the new one.

19) reset.index(): used to reset the index of Dataframe.

20) plt(): used to plot the dataset

21) plt.show(): used to display the plot on the console.

22) plt.xlabel(): used to name the x-axis on the plot.S

23) plt.ylabel(): used to name the y-axis on the plot.

24) plt.title(): used to display the title.

Modules from scikit-learn Used in the project

1) Cosine Similarity: from sklearn.metrics.pairwise import cosine_similarity

Used to calculate the cosine similarity between two non-zero vectors of an inner product space that measures the cosine of the angke between them.

2) LabelEncoder: from sklearn.preprocessing import LabelEncoder

Used for label for encodinf categorical labels wirth numerical values which is commonly used when working with machine learning models that require numerical inputs.

3) TfidVectorizer: from sklearn.feature_extraction.text import TfidfVectorizer

Used for conerting a collection of raw documents into a matrix of TF-IDF(Term Frequency-Inverse Document FGrequency) features. It is common technique in NLP to represent text data numerically.

4) linear_kernel:from sklearn.metrics.pairwise import linear_kernel

Used to compute the linear kernel between two metrics. In the context of text data, it can be used to calculate the linear similarity between sets of features, often used in document similarity tasks

Plots Used in Project:

1) catplot: used to create a box plot.
2) Barplot: used to create the baroplot which display the mean or another statistic for numeric variable grouped by categories.

Algorithms Used in the Project

Collaborative Filtering: Technique uses the idea that users who agreed the past trend must agree the future trend which allows to make predictions by collecting preferences from many users.
Types of Collaborative Filtering:

- User-Based Collaborative Filtering: Identifies the similar preferences, behavioural patterns and recommends items likes similar users.

- Item-based Collaborative Filtering: Recommends the products to the user which he liked or interacted the most.

Content-Based Filtering: Recommends items to the users based on the features of the items and preferences expressed by the user which take consideration of the attributes or characteristics of the items and recommends items similar to those the user likes before.

Hybrid Filtering: Combines the collaborative and content-based filtering to leverage the strengths of both the approaches. The Idea is to provide more accurate and diverse recommendations by combining the information from user behaviour patterns and item attributes.
Libraries:

```
>from sklearn.feature_extraction.text import TfidfVectorizer
>from sklearn.metrics.pairwise import linear_kernel
>from sklearn.metrics.pairwise import cosine_similarity
>from sklearn.preprocessing import LabelEncoder
>from sklearn.model_selection import train_test_split
```

Evaluation Matrix

Precision_Score: It measures the proportion of positively predicted labels that are actually correct. Precision is also known as the positive predictive value. Precision score is used in conjunction with the recall score to trade-off false positives and false negatives. Precision is mainly used when we need to predict the positive class and there is a greater cost associated with false positives than with false negatives such as in medical diagnosis or spam filtering. For example, if a model is 99% accurate but only has 50% precision, that means that half of the time when it predicts an email is a spam, it is actually not spam. The following is the formula for how to calculate precision score:
Precision Score = True Positives/ (False Positives + True Positives)

Recall score: in machine learning represents the model's ability to correctly predict the positives out of actual positives. This is unlike precision which measures how many predictions made by models are actually positive out of all positive predictions made. For example: if your model is trying to identify positive reviews, the recall score would be what percent of those positive reviews did it correctly predict as a positive. Recall is also known as sensitivity or the true positive rate.
Recall Score = True Positives / (False Negatives + True Positives)

F1 score: is harmonic mean of precision and recall score. It gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy thereby making it an alternative to Accuracy metrics (it doesn't require us to know the total number of observations). It's Optimizing for recall helps with minimizing the chance of not detecting a malignant cancer. However, this comes at the cost of predicting malignant cancer in patients although the patients are healthy (a high number of FP).
The following is the formula of F1 Score:
F1 Score = 2* Precision Score * Recall Score/ (Precision Score + Recall Score)

Accuracy score: is the metrics that is defined as the ratio of true positives and true negatives to all positive and negative observations. learning models make on new data we haven't seen before. The following is the formula of the accuracy score.

Accuracy Score = (TP + TN)/ (TP + FN + TN + FP)

Where as TP: True Positive

TN: True Negative

FN:False Negative

FP:False positive

Libraries:

>from sklearn.metrics import precision_score, recall_score, f1_score, average_precision_score