Data Science Bootcamp

Capstone Project

Project Title: Recommender (Build intelligence to help customers discover products they may like and most likely purchase)

Problem Statement
In the e-commerce industry, providing personalized product recommendations to customers is crucial for enhancing user experience, increasing customer engagement, and boosting sales. Building an effective shopping recommender system can significantly improve customer satisfaction and drive repeat purchases. As a data scientist, your task is to develop a shopping recommender system that suggests relevant products to customers based on their preferences and browsing history, thereby increasing the likelihood of purchase and overall customer retention.

Description: In the project, developed a system that will recommend the user various products based on their shopping patterns. Which will highly help them to decide what to buy with what and how to buy. For that, Collaborative, Content and hybrid filtering methods have been used for the recommendation purposes.
Table 1: Customer.csv
Columns:
customer_Id: Tells the unique customer Identification number
DOB: Date of birth of the customers
Gender: Gender of the customer
City_code: Tells the unique code of each customer cities

Table 2: prod_cat_info
Columns:
prod_cat_code: Tells the code for each product category
prod_cat: Tells the category of the products
prod_sub_cat_code: Code of subcategory of each product
prod_subcat: Segregate the category of products in subcategories

Table 3: Transaction.csv

Columns:
transaction_id:Describe the transaction id of the customer
cust_id: Tells the unique customer Identification number
tran_date: Date of the transaction
prod_subcat_code: Code of subcategory of each product

prod_cat_code: Tells the code for each product category

Qty: Number of items purchased by the customer
Rate: Price of the item
Tax: Tax applied on the item purchased
Total_amt: describe the total money spent by the customer and calculated as :
= (Rate*Qty)+Tax
Store_type: Type of the store the purchased has been made from by the customer
Which are e-shop
Flagship store
MBR
Teleshop

Steps to Follow to download Anaconda Navigator Installation:
Links to download the anaconda navigators
For Mac: https://docs.anaconda.com/free/anaconda/install/mac-os/
For Windows: https://docs.anaconda.com/free/anaconda/install/windows/
For Linux: https://docs.anaconda.com/free/anaconda/install/linux/

Steps to follow to work on Jupyter Notebook:

- Open the Anaconda Navigator and click on Jupyter Notebook to launch.
- In the Jupyter Notebook interface, click on the "New" button and select a kernel(e.g., Pyhton) for your notebook.
    1. Jupyter Notebook consist of cells. There are two main types of cells: Code and Markdown Cells.
       Code cells: used to write codes and to run that cell one can press Shift + Enter or click the "Run" button.

<u>Steps for the project to follow:</u>

- Follow the above steps to create a new folder and named Capstone.
- The new folder has been created on Jupyter notebook terminal click on that and follow these steps:
  >New > Upload>Select the file(csv,excel etc) from your system<upload
- Follow the above step and upload the 3 csv files which are:
  Customer.csv
  prod_cat_info.csv
  Transaction.csv
- Now click to the New>Python 3 (ipykernel)> Rename the file > Save it
- For the project, there are four python files has been created to show the step by step process for data cleaning and visualizing, and to show how different algorithms are work.

- The four python files are:

  Capstone_Recommender_Data_Cleaning.ipynb
  Capstone_Recommender_Collaborative_Filtering.ipynb
  Capstone_Recommender_Content_Filtering.ipynb
  Capstone_Recommender_Hybrid_Filtering.ipynb

Steps to follow in the jupyter notebook:

1. Import the important libraries :

   >import numpy as np
   >import pandas as pd
   >import seaborn as sns
   >import matplotlib.pyplot as plt
   >from datetime import date
   >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

2. Load the datasets and create the DataFrame:

   >customer = pd.read_csv("Customer.csv")
   >prod_cat_info = pd.read_csv("prod_cat_info.csv")
   >transactions=pd.read_csv("Transactions.csv")

3. Changing column label to similar in all tables
   >customer.rename(columns = {'customer_Id':'cust_id'}, inplace = True)
4. removing row if customer_id/prod_sub_cat_code is null from customer and prod_cat_info dataframe
   >customer.dropna(subset=['cust_id'],inplace=True)
   >prod_cat_info.dropna(subset=['prod_sub_cat_code'],inplace=True)

5. Display top 5 rows of customer data
   >customer.head(5)

6. display top 5 rows of prod_cat_info data
   >prod_cat_info.head(5)

7. Display top 5 rows of transactions data
   >transactions.head(5)

8. Display rows and columns of customer data

>customer.shape

9. display rows and columns of prod_cat_info data
   >prod_cat_info.shape

10. display rows and columns of transactions data
    >transactions.shape

11. Creating super table by joining customer and prod_cat_info to transactions
    >df1=transactions.merge(customer,how='left',on='cust_id')
    >transaction_master_bi=pd.merge(left=df1, right=prod_cat_info,how='left',left_on=['prod_cat_code','prod_subcat_code'],right_on=['prod_cat_code','prod_sub_cat_code'])
    >transaction_master_bi.drop(columns='prod_sub_cat_code',axis=1,inplace=True)

12. Displaying the head rows of super table
    > transaction_master_bi.head(10)

13. Display the shape of super table
    > transaction_master_bi.shape

14. Display the name of the columns of the super table
    > transaction_master_bi.columns

15. Display the information of the super table
    > transaction_master_bi.info()

16. Display the statistics of the super table.
    > transaction_master_bi.describe()

17. Display the sum of the null values if any
    > transaction_master_bi.isnull().sum()

18. Correceting data type of numeric and date columns

    >transaction_master_bi['city_code']=pd.to_numeric(transaction_master_bi['city_code'],downcast='integer').fillna(1).astype(int)
    >transaction_master_bi['tran_date'] = pd.to_datetime(transaction_master_bi['tran_date'])
    >transaction_master_bi['DOB'] = pd.to_datetime(transaction_master_bi['DOB'])

>transaction_master_bi['Qty']=abs(transaction_master_bi['Qty'])
>transaction_master_bi['Rate']=abs(transaction_master_bi['Rate'])
>transaction_master_bi['total_amt']=abs(transaction_master_bi['total_amt'])

19. handling null values
>transaction_master_bi.dropna(subset=['transaction_id','cust_id','prod_subcat_code','prod_cat_code','tran_date'],inplace=True)

20. Filling null values
> transaction_master_bi['Gender'].fillna('F',inplace=True)
>transaction_master_bi['Store_type'].fillna('NA',inplace=True)
>transaction_master_bi['city_code'].fillna(-1,inplace=True)
>transaction_master_bi['DOB'].fillna(pd.to_datetime('1900-01-01'),inplace=True)

21. Plotting the graphs by using following codes and obtain different graphs that are mentioned in the visual folder:
Plot the bar chart
<u>Plot Gender Count</u>
>number_of_male_female=transaction_master_bi['Gender'].value_counts()
>a = number_of_male_female.plot(kind='bar', color='b')
>plt.xticks(rotation=360)
>plt.xlabel('Gender')
>plt.ylabel('Count')
>plt.title('Distribution of Gender')

Add count labels above each bar
>for i, count in enumerate(number_of_male_female):
    plt.text(i, count + 0.1, str(count), ha='center', va='bottom')

>plt.show()

<u>Plot for Age and Gender</u>
plt.figure(figsize=(15,10))
sns.catplot(x='Age',data=transaction_master_bi,kind='count',hue="Gender",width=0.7)

# bins = [30, 35, 40, 45, 50,55] # Example bins
labels = ['30-34', '35-39', '40-44', '45-49','<50']
transaction_master_bi['Age_Bin'] = pd.cut(transaction_master_bi['Age'], bins=bins, labels=labels, right=False)

```
# Create countplot
plt.figure(figsize=(10, 6))
sns.countplot(data=transaction_master_bi, x='Age_Bin', hue='Gender')
plt.title('Countplot of Age Bins by Gender')
plt.xlabel('Age Bins')
plt.ylabel('Count')
plt.xticks(rotation=360)  # Rotate x-axis labels for better visibility
plt.legend(title='Gender')
plt.tight_layout()
plt.show()
```

Plot for Unique Products

First find the unique products by using the group by functions

```
>unique_products=transaction_master_bi.groupby('prod_cat')['prod_cat
'].unique()
> plt.figure(figsize=(12, 8))
sns.countplot(x='prod_cat', data=transaction_master_bi,
palette='viridis')
plt.title('Unique Products Count for Each Category')
plt.xlabel('Product Category')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')  # Adjust rotation for better
readability

plt.show()
```

Plot for Unique Store Types:

```
ax=transaction_master_bi['Store_type'].value_counts().plot(kind='bar')
for bars in ax.containers:
    ax.bar_label(bars)
plt.xticks(rotation=45)
plt.show()
```

Plot for the product category:

```
# Plotting
plt.figure(figsize=(10, 6))  # Setting figure size
plt.bar(grouped_df_prod_cat['prod_cat'],
grouped_df_prod_cat['total_amt'], color='skyblue')  # Creating bar plot

# Adding titles and labels
```

```
plt.title('Total Price by Product')
plt.xlabel('Product')
plt.ylabel('Total Price')

# Displaying plot
plt.tight_layout()  # Adjust layout to prevent clipping of labels
plt.show()
```

22. Codes for filtering Method

Collaborative Filtering

```
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
transaction_master_bi['prod_code'] =
label_encoder.fit_transform(transaction_master_bi['product_code'])
transaction_master_bi['sex'] =
label_encoder.fit_transform(transaction_master_bi['Gender'])
transaction_master_bi['age_split'] =
label_encoder.fit_transform(transaction_master_bi['age_range'])
transaction_master_bi['city'] =
label_encoder.fit_transform(transaction_master_bi['city_code'])
edf=transaction_master_bi[['product_code','prod_code']].drop_duplicate
s()
# Create user-item matrix with additional features
user_item_matrix = pd.pivot_table(transaction_master_bi, values='Qty',
index=['cust_id','sex', 'age_split',
'city'],columns='prod_code',aggfunc="sum", fill_value=0,)
user_item_matrix.fillna(0, inplace=True)  # Fill missing values

# Calculate cosine similarity between users
user_similarity = cosine_similarity(user_item_matrix)
cust_ids = user_item_matrix.index.get_level_values('cust_id').unique()
user_similarity_df = pd.DataFrame(user_similarity, index=cust_ids,
columns=cust_ids)
```

Content-Based Filtering:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
Product_Master_bi['content'] = Product_Master_bi['product_code']
# Create TF-IDF matrix
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
```

```
tfidf_matrix =
tfidf_vectorizer.fit_transform(Product_Master_bi['content'])

# Compute the cosine similarity
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
product_mapping = {}
for idx, row in Product_Master_bi.iterrows():
    product_mapping[(row['product_code'])] = idx
```

Evaluation Matrix:

Content Filtering:

```
Mean Precision: 0.8041
Mean Recall: 0.7109
Mean F1-Score: 0.7421
Mean Average Precision: 1.0000
```

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

precision_score:

```
> precision = precision_score(true_labels, predicted_scores)
```

recall_score:

```
> recall = recall_score(true_labels, predicted_scores)
```

f1_score:

```
f1 = f1_score(true_labels, predicted_scores)
```

average_precision_score:

```
> average_precision = average_precision_score(true_labels,
predicted_scores)
```


```
Mean Precision: 0.8041
Mean Recall: 0.7109
Mean F1-Score: 0.7421
Mean Average Precision: 1.0000
```


Recommendation:

After performing all the necessary filtering’s and algorithms, it is advisable that Hybrid Filtering method is the best among the three filtering methods Hybrid recommendation systems aim to combine the strengths of both content-based and collaborative filtering methods, addressing some of the limitations associated with end approach.

- Improved Recommendation Accuracy
- Addressing reducing Cold-Start Problem
- Handling Sparsity and Scalability
- Adaptability to User references
- Increased Robustness