Recommender System

A1882311 Harsh Rathi

The University of Adelaide

4433_COMP_SCI_7306 Mining Big Data

Lecturer: Dr. Alfred Krzywicki

Table of Contents

1. Executive Summary

This report outlines the development and implementation of a recommendation system for a practical problem using pattern mining and collaborative filtering techniques. The objective of this project was to create a scalable solution capable of generating personalized recommendations based on historical transaction data. Focus is to create a recommendation system which will give a recommendation of a product to a customer using Python, but we're especially focusing on two things, making sure it can handle a lot of customer data and their purchase history without any problems and checking how well it recommends to each of our customers. The expected outcome of the project is a functional recommendation system capable of providing accurate and relevant recommendations to users.

In my evaluation, I have compared two approaches to give item recommendations for our customers. First was user-based recommendation approach and second was item-based recommendation approach. These approaches utilize different strategies to provide recommendations to users. The user-based approach considers similarities between users to recommend items, while the item-based approach recommends items based on similarities between items.

My evaluation results showed valuable insights into the effectiveness of these recommendation approaches. For my user-based recommendation approach, I received an overall average precision of approximately 16.24% which may sound low as I could have done much better in pre-processing of the given data, but my prime focus was on successfully implementing this model. However, according to the precision I received it says 16.24% of recommended items were relevant to users' preferences. The average recall, which means measuring the proportion of relevant items successfully recommended, stood at around 7.39% and the F1-score which is a balanced measure of precision and recall, was approximately 7.92%. In comparison, the item-based recommendation method exhibited slightly lower performance metrics. The average precision was around 13.36%, with an average recall of approximately 7.65%. The F1-score for the item-based approach was about 7.58%.

2. Introduction

In the growth of retail and electronic commerce, the need for personalizing the shopping experience and customer satisfaction has a critical focus. To contribute technology in this growth have inspired me to dive into automatic recommendation systems, Customers frequently encounter problems in decision making when they are exposed to innumerable options in retails resulting in suboptimal purchasing choices and low satisfaction. Acknowledging this challenges, my project endeavors to utilize sophisticated recommendation systems to elevate the shopping experience for patrons of a grocery store.

Association rule/pattern mining for recommender

In our modern digital era, recommendation systems have become vital aids in helping users navigate through the vast area of information available online. These systems utilize advanced algorithms and data analysis techniques to predict and suggest content, products, or services that are tailored to individual preferences, thereby enriching user experiences which leads to an increase in customer base of a company. The push for better recommendation systems isn't just limited to monopolies like YouTube, Meta or Amazon, even smaller e-commerce and customer service companies are striving to enhance their recommendation systems.
Inspired by monopolies like amazon my primary objective for this project is to develop a recommendation system capable of analyzing customers' purchase histories and suggesting relevant products based on their preferences. By harnessing the power of data analytics and coding with python libraries, I aimed to provide personalized recommendations tailored to each individual customer's needs and preferences.

The expected business benefits of implementing such a recommendation system are enormous. Firstly, by offering personalized item recommendations, the grocery store can improve customer satisfaction and loyalty, leading to increased repeat purchases and higher customer lifetime value. Additionally, by guiding customers towards products they are likely to purchase, the recommendation system can help optimize inventory management and reduce wastage. Overall, the implementation of an effective recommendation system has the potential to drive revenue growth, enhance operational efficiency, and establish the grocery store as a leader in customer-centric retailing.

However, there are some potential hurdles which affects recommendation systems and due to which I have faced problems with my precision and f1 scores were:

Data Quality and Quantity: One limitation of my project is the dependency on the quality and quantity of available data set. As with any data-driven methodology, the accuracy and comprehensiveness of the dataset directly strikes the performance of the recommendation system. I could have given more focus on pre-processing the input data by augmenting the dataset with additional relevant variables or maybe improving data collection processes to enhance the system's effectiveness.

Cold start problem: It happens when our recommendation system finds it difficult to give good suggestions to new or less frequent customers who haven't been very active with the purchase in the given retail. To solve this problem, we need to come up with clever ideas. One way is to use information about the customer, like their age or where they live and form a cluster in our prediction systems. These groups of people will make our recommendation system better to make better suggestions. Another way is to bring in data from other places outside of our system, like social media or online reviews, to understand what new customers might like. By doing this, we can make recommendations that are more tailored to each new customer's tastes and preferences, or which follow the genre of current trends.

Relying on just one method for recommendations can also lead to limitations in our system. For example, if we only use collaborative filtering, which suggests items-based similarities between users or items, we might miss out on other important factors that could improve our recommendations. This could include things like item popularity, customer demographics, or item

characteristics. By sticking to just one method, we might not capture the full complexity of customer preferences, which could result in less accurate or diverse recommendations. Therefore, I focused on contributing my ideas of implementing two methods which were user-based recommendation system and item-based recommendation system. Which would give a comprehensive experience of how these methods work on a given dataset.

Extractive Summarization: The report highlights the development and implementation of a recommendation system aimed at providing personalized product recommendations to customers based on their purchase history. It compares two approaches: user-based recommendation and item-based recommendation, evaluating their effectiveness in terms of average precision, recall, and F1-score. The user-based approach achieved higher precision metrics compared to the item-based approach. The introduction outlines the significance of recommendation systems in enhancing customer satisfaction and loyalty in the retail and e-commerce sector. Potential challenges such as data quality and the cold start problem are identified, along with suggestions for overcoming them. Additionally, relying solely on one recommendation method is acknowledged as a limitation, prompting the exploration of two different approaches to ensure comprehensive recommendation coverage.

3. Exploratory Analysis

3.1. Data Overview

The datasets collected for this project contain transactional data from a grocery store which means data having customer-ids showing their purchased product and on what date they have purchased by what quantity and providing their bill no and the date of transaction and also price of a single item and Cost (total cost of the transaction). We have two sets of data as,

basket_data_by_date_train.csv: This dataset is intended for training and tuning purposes. It contains transactional data organized by date, with each row representing a single transaction made by a customer. The dataset includes the following columns:
Bill No: Unique identifier for each transaction.
Item name: Name of the item purchased.
Quantity: Quantity of the item purchased in this transaction.
Date: Date and time of the transaction.
Price: Price of a single item.
Customer-ID: Unique identifier for each customer.
Cost: Total cost of the transaction.

basket_data_by_date_test.csv: This dataset is meant for obtaining the final results. Similar to the training dataset, it contains transactional data with the same columns as mentioned above. However, the transactions in this dataset occur at later dates than those in the training dataset, allowing for testing of "future" recommendations.

According to my Analysis of the given data, for favorable accurate recommendations I have focused on three main important columns present in data set that are ‘Customer-Id(representing

Customers)' 'Quantity(by what size they have purchased the item)' and 'Item Name'.

3.2. Data Analysis:

```
# Load the datasets
train_df = pd.read_csv('basket_data_by_date_train.csv')
print(train_df.info())
train_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40000 entries, 0 to 39999
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   BillNo      40000 non-null  int64  
 1   Itemname    40000 non-null  object 
 2   Quantity    40000 non-null  int64  
 3   Date        40000 non-null  object 
 4   Price       40000 non-null  float64
 5   CustomerID  40000 non-null  int64  
 6   cost        40000 non-null  float64
dtypes: float64(2), int64(3), object(2)
memory usage: 2.1+ MB
None
```

	BillNo	Itemname	Quantity	Date	Price	CustomerID	cost
0	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	1/12/2010 8:26	3.39	17850	20.34
1	536365	GLASS STAR FROSTED T-LIGHT HOLDER	6	1/12/2010 8:26	4.25	17850	25.50
2	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	1/12/2010 8:26	2.55	17850	15.30
3	536365	RED WOOLLY HOTTIE WHITE HEART.	6	1/12/2010 8:26	3.39	17850	20.34
4	536365	SET 7 BABUSHKA NESTING BOXES	2	1/12/2010 8:26	7.65	17850	15.30

The dataset consists of 40,000 entries and 7 columns:

Bill No: An integer column representing the bill number for each transaction.
Item name: A categorical column containing the name of the purchased item.
Quantity: An integer column indicating the quantity of each item purchased in the transaction.
Date: A string column representing the date and time of each transaction.
Price: A floating-point column indicating the price of a single item.
Customer-ID: An integer column representing the unique identifier for each customer.
Cost: A floating-point column indicating the total cost of the transaction.

```
# Check for null values in the training data
print("Null values in Training Data:")
print(train_df.isnull().sum())
```

```
Null values in Training Data:
BillNo        0
Itemname      0
Quantity      0
Date          0
Price         0
CustomerID    0
cost          0
dtype: int64
```

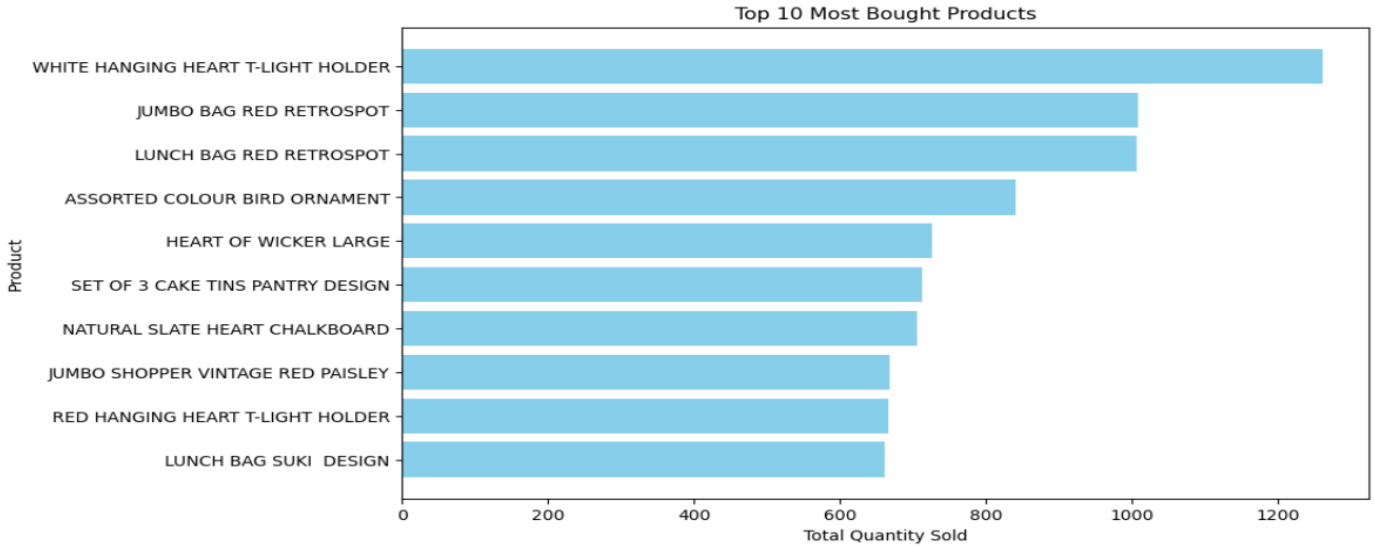
```
# Check for null values in the training data
print("Null values in Training Data:")
print(test_df.isnull().sum())
```

```
Null values in Training Data:
BillNo        30000
Itemname      30000
Quantity      30000
Date          30000
Price         30000
CustomerID    30000
cost          30000
dtype: int64
```

The memory usage of the Data Frame is approximately 2.1+ MB which makes it suitable for a jupyter notebook environment. The training dataset does not contain any null values in any of the

columns, however testing dataset contained 30,000 null values which could have negatively impacted our model's performance. Null Values are nothing but fillers for missing values present in our data set which are better to be removed for accurate implementation of our method.

I have also implemented a bar plot to visualize the top 10 most bought products based on the total quantity sold. This analysis helps in understanding which products are the most popular or in-demand among customers. By examining the top-selling products, businesses can gain insights into customer preferences, identify trends, and make data-driven decisions related to inventory management, marketing strategies, and product offerings.



Analize customer activity data to identify the top 10 customers who purchased the highest quantity of products. It's a common task in sales analysis to understand customer behavior and identify key customers for targeted marketing or personalized services.

	CustomerID	Quantity
1290	17850	1553
695	15311	1435
113	12748	1230
1286	17841	1127
630	15039	1036
533	14606	998
188	13089	714
403	14085	663
804	15808	634
219	13174	630

4. Implementation and Testing

4.1. User matrix:

It creates a table to transform the original Data Frame train_df into a user-item matrix suitable for user-based collaborative filtering. Each row represents a Customer-ID, each column represents Item Name, and the values in the matrix represent the quantity of each item purchased by each customer. We used sparce matrix after this to eleminate zero values accord in our user matrix. Zero value occurs when a customer-id didn't buy a particular item.

```
# Create pivot tables and convert to sparse CSR matrix for user-based collaborative filtering
user_matrix = train_df.pivot_table(index='CustomerID', columns='Itemname', values='Quantity', aggfunc='sum', fill_value=0)

user_matrix.head()
```

Itemname	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 DAISY PEGS IN WOOD BOX	12 EGG HOUSE PAINTED WOOD	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	...	YULETIDE IMAGES S/6 PAPER BOXES	ZINC FINISH 15CM PLANTER POTS
CustomerID													
12347	0	0	0	0	0	0	0	0	0	0	...	0	0
12350	0	0	0	0	0	0	0	0	0	0	...	0	0
12352	0	0	0	0	0	0	0	0	0	0	...	0	0
12356	0	0	0	0	0	0	0	0	0	0	...	0	0
12362	0	0	0	0	0	0	0	0	0	0	...	0	0

5 rows × 2523 columns

4.2. Sparce matrix:

Sparse matrices are represented in a way that only the values which are non-zero are considered with their row and column indices, rather than storing the entire matrix. This efficient storage format reduces memory usage and speeds up computations, especially for operations involving large matrices. Therefore, with this we can reduce are memory which could be beneficial for efficient implementation of our model

4.3. Item matrix:

It is a transpose of user matrix suitable for item based collaborative filtering.

```
# Transpose for item-based
item_matrix = train_df.pivot_table(index='Itemname', columns='CustomerID', values='Quantity', aggfunc='sum', fill_value=0)
item_matrix.head()
```

CustomerID	12347	12350	12352	12356	12362	12370	12373	12377	12383	12386	...	18230	18233	18239	18245	18250	18257	18259	18260	18269	182
Itemname																					
10 COLOUR SPACEBOY PEN	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
12 COLOURED PARTY BALLOONS	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
12 DAISY PEGS IN WOOD BOX	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
12 EGG HOUSE PAINTED WOOD	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	
12 IVORY ROSE PEG PLACE SETTINGS	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	

4.4. Singular Value Decomposition (SVD):

Singular Value Decomposition (SVD) is a mathematical method that helps us analyze and simplify large and sparse user-item interaction matrices commonly found in recommendation systems. These matrices represent the interactions between users and items, such as purchases or ratings. SVD breaks down these matrices into three smaller matrices: one for users, one for items, and one for singular values. Each matrix captures different aspects of the original data. The user matrix represents user preferences or characteristics, the item matrix represents item features or attributes, and the singular values matrix represents the strength of the relationships between users and items. This decomposition process allows us to identify hidden patterns or latent factors within the data.

These factors could include user preferences, item characteristics, or underlying trends. By reducing the dimensionality of the original matrix, SVD helps us extract the most important information while discarding noise or irrelevant details.

SVDS gives three matrices as its output which will not be suitable for our CF model so it is important to have dot product of U(user) and Sigma(diagonal) and then having dot product of that to Vt(item).

4.5. Cosine similarity matrix:

Cosine similarity is a metric used to measure the similarity between two vectors in an inner product space. It calculates the cosine of the angle between these vectors, indicating how close they are. In the context of recommendation systems, cosine similarity is often referred to assess the similarity between user's or item's vector, representing preferences or characteristics. Higher cosine similarity scores suggest greater similarity, indicating that users or items have similar preferences or characteristics.

4.6. Item based recommendation system:

The recommend_items_item_based function is designed to generate item-based recommendations for a given user. Firstly, in Input Parameters. The function takes two input parameters: user_id and num_recommendations. user_id is the ID of the customer for whom recommendations are generated. The variable num_recommendations are the number of recommendations to be returned, with a default value of 5(This can be any number depends on how many recommendations we need). Before proceeding with recommendation generation, my function verifies if the provided user_id is present in our training data set. If the user is not present, it immediately returns an empty list, indicating that no recommendations can be made. Next, the function retrieves the item interaction vector for the specified user from the user_matrix. It then calculates item scores for all items based on their similarity with the user's interaction vector. This computation involves performing a dot product between the item_similarity_matrix and the user's item interaction vector. Once item scores are computed, the function identifies the indices of the top num_recommendations items based on these scores. It retrieves the names of these recommended items along with their corresponding scores for further processing. Finally, the function constructs and returns a list of tuples containing recommended items and their scores.

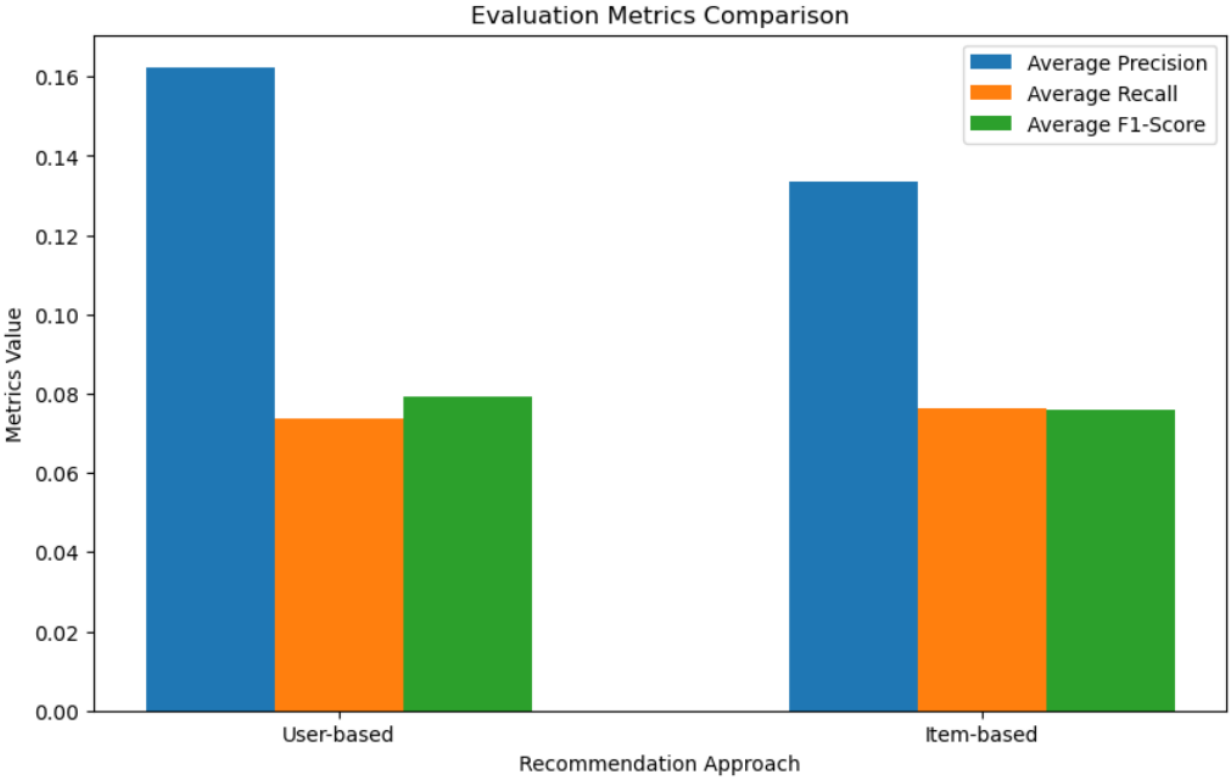
4.7. User based recommendation system:

The user-based recommendation approach focuses on identifying similar users based on their past interactions or preferences. This code snippet (recommend_items_user_based) focuses on user-based recommendation. It identifies similar users to the target user based on their past interactions. It aggregates item scores from similar users and recommends items that are popular among those similar users. The recommendations are based on items that similar users have interacted with and liked.

The overall results indicate the performance of the recommendation systems when evaluated against the test data.

Figure 1. System Architecture for …. system.

2. Discussion of Results




```
Overall Results:
User-based Metrics: {'Average Precision': 0.16240000000000002, 'Average Recall': 0.07387374122571547, 'Average F1-Score': 0.079
15921259525953}
Item-based Metrics: {'Average Precision': 0.1336, 'Average Recall': 0.07645660650394528, 'Average F1-Score': 0.0758152156015199
6}
```

User-based Recommendation Metrics:
Average Precision: Approximately 16.24%
Average Recall: Approximately 7.39%
Average F1-Score: Approximately 7.92%
These metrics suggest that the user-based recommendation approach achieved a moderate level of precision, indicating that around 16.24% of the recommended items were relevant to users' preferences. However, the recall and F1-score are relatively lower, indicating that a smaller proportion of relevant items were successfully recommended.

Item-based Recommendation Metrics:
Average Precision: Approximately 13.36%
Average Recall: Approximately 7.65%
Average F1-Score: Approximately 7.58%
The item-based recommendation approach exhibited slightly lower performance metrics compared to the user-based approach. While the average precision is similar, the recall and F1-score are slightly lower, indicating a comparable but slightly less effective performance in recommending relevant items.

Overall, both recommendation approaches show potential in providing personalized recommendations to users based on their purchase history. However, there is room for improvement in enhancing the precision, recall, and overall effectiveness of the recommendation systems to better meet user preferences and improve the shopping experience.

Table 1. Performance metrics for Extractive summaries

Method	Average Precision	Average Recall	F1-Score		
Item based recommendation system:	13.36%	7.65%	7.58%		
User based recommendation system:	16.24%	7.39	7.29		

3. References:

librarysearch.adelaide.edu.au. (n.d.). https://librarysearch.adelaide.edu.au/discovery/fulldisplay?&context=PC&vid=61ADELAIDE_INST:UOFA&search_scope=all&tab=Everything&docid=cdi_scopus_primary_2_s2_0_33750702813. [online] Available at: https://librarysearch.adelaide.edu.au/permalink/61ADELAIDE_INST/1eubam4/cdi_scopus_primary_2_s2_0_33750702813 [Accessed 16 Apr. 2024].

librarysearch.adelaide.edu.au. (n.d.). https://librarysearch.adelaide.edu.au/discovery/fulldisplay?&context=PC&vid=61ADELAIDE_INST:UOFA&search_scope=all&tab=Everything&docid=cdi_proquest_miscellaneous_1448716877. [online] Available at: https://librarysearch.adelaide.edu.au/permalink/61ADELAIDE_INST/1eubam4/cdi_proquest_miscellaneous_1448716877 [Accessed 16 Apr. 2024].

Association rule/pattern mining for recommender

-