**BSD2513: ARTIFICIAL INTELLIGENCE**
GROUP PROJECT

TITLE:

# ELECTRONIC PRODUCT FROM AMAZON RECOMMENDATION

MADE FOR:
**DR. KU MUHAMMAD NAIM KU KHALIF**

MADE BY:

| NAME | MATRIC ID |
|---|---|
| NOR FARAWAHIDA BT ABDULLAH | SD21010 |
| HANIS SHAHIRA BINTI SHAHIMI | SD21016 |
| DHARUMASHAN A/L BATHIBAN | SD21032 |
| AQILA SOFEA BINTI ABD MANAF | SD21006 |
| RABIATUL SOFIA BINTI ADNAN | SD21007 |

**[GROUP E]**

DUE DATE:
**20TH JUNE 2023**

## PHOTOS OF MEMBERS

NOR FARAWAHIDA


HANIS SHAHIRA

DHARUMASHAN A/L BATHIBAN

AQILA SOFEA

RABIATUL SOFIA

**TABLE OF CONTENTS**

# 1.0 INTRODUCTION

In today's highly competitive market, making informed decisions about product recommendations is crucial for businesses to stay ahead. Companies require effective techniques to suggest items that engage their customer's needs and improve their entire shopping experience because there are so many options available from other competitors and customer preferences are constantly changing. Over time, Amazon has seen a considerable increase in the quantity of reviews. Customers who have made purchases on Amazon give evaluations by giving the item a star rating between 1 and 5 and sharing a short summary of their thoughts on the item. Thus, from that, we can generate a recommendation system for the users to know which products are the best for them.

## 1.1 Project Description

Amazon currently uses item-item collaborative filtering, which scales to massive datasets and produces high quality recommendation systems in real time. This system aims to forecast the "rating" or preferences that a user would be interested in. It is a type of information filtering system.

The goal of this project is to improve the platform's capacity to provide consumers with personalised and pertinent product recommendations. The system will make use of collaborative filtering's ability to make precise predictions and assist users in finding products that are in line with their interests, thereby increasing user satisfaction and engagement on the site.

## 1.2 Problem Statement

One of the biggest e-commerce platforms, Amazon, seeks to enhance its electronic product recommendation system. The existing recommendation system makes product suggestions based on a user's browsing and purchasing history, but it is not sufficiently personalised to take into account the interests of specific users. Through the provision of more precise and pertinent product recommendations, Amazon wants to boost the user experience.

The problems that must be overcome in order to create an effective Amazon electronic product suggestion system include the enormous amount of data that must be processed. Every day, millions of product reviews are sent to Amazon. These reviews must be evaluated in order to create a model that can provide precise suggestions. Subsequently, the data's lack of density. Even users who do leave product ratings occasionally review a limited number of items. As a result, it can be challenging to create a model that can use this data to generate reliable recommendations. Last but not least, the demand for customization. Different user preferences must be able to be taken into consideration by the system when providing recommendations.

### 1.3 Basic Description Of The Data

The Datafiniti Amazon Electronic Product Recommendation dataset is associated with a list of over 1,500 customer reviews for various Amazon products, including the Kindle, Fire TV Stick and more. For each product, the dataset contains basic details about the item as well as rating, review content, and other information. Some of the features of the dataset is Product information. Products' names, IDs, URLs, categories, prices, and brands are all included in this. The next feature is review information and it consists of the reviewer metadata (such as the reviewer's name, location, and age), as well as the review date, review rating, review title, and review content. There are also other qualities such as URL for the product image, the description of the product, and the overall sentiment (positive, negative, or neutral) of the product reviews.

The data contains 5000 rows and 24 columns of information on the electronic products sold on Amazon. These columns include:

- **id**: A unique code assigned to each review.
- **dateAdded**: The date on which the review was included into the dataset.
- **dateUpdated**: The most recent update date for the review.
- **name:** The name of the product
- **asins**: The Amazon Standard Identification Number for the product. This is a unique identifier for each product on Amazon, consisting of 10 letters and/or numbers.

- **brand**: The product's name brand. This is the business that produced the item, which in this case is Amazon.
- **categories:** This is a category that the product belongs to, such as Computers,Electronics Features,Tablets,Electronics,iPad & Tablets,Kindle E-readers,iPad Accessories,Used:Tablets,E-Readers,E-Readers & Accessories,Computers/Tablets & Networking,Used:Computers Accessories,iPads Tablets,All Tablets,Tablets & E-readers,Computers & Tablets,Amazon,Tablets & eBook Readers
- **primaryCategories**: This is where the product's main category, such as "Electronics", "Office Supplies", "Hardware" or "Media" is listed.
- **imageURLs**: The list of URLs leading to the product's images. Various websites, including Amazon.com, may have provided the images.
- **keys**: The Amazon Product Advertising API keys used to collect the data.
- **manufacturer**: The company that created the product.
- **manufacturerNumber**: A special alphanumeric code that the manufacturer assigns to distinguish one particular product or component from another. To uniquely identify a product, it is frequently used in conjunction with the productID column.
- **reviews.date**: The date the review was published. This is the date that the review was submitted to Amazon.
- **reviews.dateAdded**: The datetime field that indicates the date and time when the product review was added to the datasets
- **reviews.id**: The ID of the reviewer. This is a unique identifier for each reviewer on Amazon.
- **reviews.numHelpful:** The number of helpful votes for the product. This is the number of times that users have voted that a review was helpful.
- **reviews.rating**: The rating of the product, on a scale of 1 to 5 stars. This is the average rating of the product based on user reviews.
- **reviews.text**: The text of the review. This is the text that the reviewer wrote about the product.
- **reviews.title**: The title of the product. This is a short description of the product

## 2.0 SUMMARY OF PROJECT

### 2.1 Project Objectives

- To design a user-friendly interface for interaction and viewing recommendations
- To enhance the user experience and increase customer satisfaction on the Amazon platform.
- To suggest the customers based on their user prediction to give a high rating

### 2.2 Project Question

- Why is a user-friendly interface important for the customers?
- Why was this project made?
- How does this project work?

### 2.3 Project Content

This project's goal is to make Amazon electronic goods recommendations based on a predetermined dataset. The collection probably includes data about different electrical devices that are sold on Amazon, including their categories, features, reviews, ratings, and pricing. Personalised suggestions for users based on their likes and interests are what we want to achieve by creating a recommendation system that can analyse this dataset.

## 3.0 METHODOLOGY

### 3.1 Data Collection

The first step is to collect the required data for the project. The data is collected from the Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv file, which contains information about Amazon consumer reviews of various products.

### 3.2 Data Pre-processing

The application that we used in this project is Jupyter Notebook. There are several libraries that were used in our coding. The libraries used are:

```
In [1]: #import the reqired libraries
        from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = "all"
        import numpy as np
        import pandas as pd
        import math
        import json
        import time
        import matplotlib.pyplot as plt
        import seaborn as sns
```

- NumPy is used to provide support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.
- Pandas are used to provide data structures and functions for efficiently handling structured data, such as data frames.
- The maths library provides mathematical functions and constants. It includes functions like logarithmic, trigonometric, and exponential functions.
- The json library is used for encoding and decoding JSON (JavaScript Object Notation) data. It allows reading and writing data in JSON format.
- The time library provides functions for working with time-related tasks. It is used to measure the execution time of code, add delays, and handle timestamps.
- Matplotlib is used to visualize data. The pyplot module provides a simple interface to create types of plots like line plots, bar charts, histograms, and scatter plots.
- Seaborn is used to provide a higher-level interface for creating visually appealing and informative statistical graphics.

```
In [11]: oldDt=pd.read_csv("C:/Users/User/Downloads/Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv")
         oldDt.head(5)
```

Out[11]:

| | id | dateAdded | dateUpdated | name | asins | brand | categories | primaryCategories | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | AVqVGZNvQMlgsOJE6eUY | 2017-03-03T16:56:05Z | 2018-10-25T16:36:31Z | Amazon Kindle E-Reader 6" Wifi (8th Generation... | B00ZV9PXP2 | Amazon | Computers,Electronics Features,Tablets,Electro... | Electronics | https://pisces.bbystatic |
| 1 | AVqVGZNvQMlgsOJE6eUY | 2017-03-03T16:56:05Z | 2018-10-25T16:36:31Z | Amazon Kindle E-Reader 6" Wifi (8th Generation... | B00ZV9PXP2 | Amazon | Computers,Electronics Features,Tablets,Electro... | Electronics | https://pisces.bbystatic |
| 2 | AVqVGZNvQMlgsOJE6eUY | 2017-03-03T16:56:05Z | 2018-10-25T16:36:31Z | Amazon Kindle E-Reader 6" Wifi (8th Generation... | B00ZV9PXP2 | Amazon | Computers,Electronics Features,Tablets,Electro... | Electronics | https://pisces.bbystatic |
| 3 | AVqVGZNvQMlgsOJE6eUY | 2017-03-03T16:56:05Z | 2018-10-25T16:36:31Z | Amazon Kindle E-Reader 6" | B00ZV9PXP2 | Amazon | Computers,Electronics Features,Tablets,Electro... | Electronics | https://pisces.bbystatic |

This code reads the CSV file and stores the data in a pandas DataFrame called oldDt. The head() function is used to display the first 5 rows of the DataFrame.

```
In [12]: products={
             'B00IOY8XWQ':'Kindle Voyage 4GB Wifi',
             'B00IOYAM4I':'Kindle Voyage 4GB Wifi+Cellular',
             'B00QFQRELG':'Powerfast Charger',
             'B00REQKWGA':'Kindle + Charging Cover Wifi',
             'B00VINDBJK':'Kindle + Charging Cover Wifi',
             'B00ZV9PXP2':'Kindle 8 Wifi',
             'B010CEHQTG':'Bluetooth Speaker',
             'B017JG41PC':'Kindle Wifi',
             'B0189XYY0Q':'Tablet 10 Wifi 8GB',
             'B018Y224PY':'Tablet with Alexa 16GB',
             'B018Y225IA':'Kindle 16GB Wifi',
             'B018Y22BI4':'Tablet Wifi 16gb',
             'B018Y22C2Y':'Kids Tablet',
             'B018Y23MNM':'Kids Tablet',
             'B01ACEKAJY':'Tablet 8 Wifi 32GB',
             'B01AHB9CN2':'Tablet 8 Wifi 16GB',
             'B01AHB9C1E':'Tablet with Alexa 32GB',
             'B01AHB9CYG':'TV',
             'B01AHBBG04':'Tablet 8 Wifi 16GB',
             'B01AHBDCKQ':'Tablet 8 Wifi 32GB',
             'B01BH83OOM':'Bluetooth Speaker',
             'B01J24C0TI':'Bluetooth Speaker',
             'B01N32NCPM':'TV',
             'B06XB29FPF':'Echo-Plus'
         }
         oldDt.asins=oldDt.asins.replace(products)
         oldDt
```

The dictionary product is defined to map the product codes ('asins') to their respective names. The asins column in the oldDt DataFrame is then replaced with the corresponding product names using the replace() function.

```
In [13]: dt = oldDt[['reviews.username','asins','reviews.rating']]
         dt
```

Out[13]:

| | reviews.username | asins | reviews.rating |
|---|---|---|---|
| 0 | llyyue | Kindle 8 Wifi | 3 |
| 1 | Charmi | Kindle 8 Wifi | 5 |
| 2 | johnnyjojojo | Kindle 8 Wifi | 4 |
| 3 | Kdperry | Kindle 8 Wifi | 5 |
| 4 | Johnnyblack | Kindle 8 Wifi | 5 |
| ... | ... | ... | ... |
| 4995 | litle | Tablet with Alexa 16GB | 5 |
| 4996 | gracie | Tablet with Alexa 16GB | 5 |
| 4997 | Hawk | Tablet with Alexa 16GB | 4 |
| 4998 | Mrbilly | Tablet with Alexa 16GB | 5 |
| 4999 | tabman | Tablet with Alexa 16GB | 5 |

5000 rows × 3 columns

```
In [14]: dt.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   reviews.username  5000 non-null   object
 1   asins             5000 non-null   object
 2   reviews.rating    5000 non-null   int64 
dtypes: int64(1), object(2)
memory usage: 117.3+ KB

In [15]: dt.shape
Out[15]: (5000, 3)
```

The DataFrame dt is created by selecting only the 'reviews.username', 'asins', and 'reviews.rating' columns from oldDt. dt.info() provides information about the DataFrame,

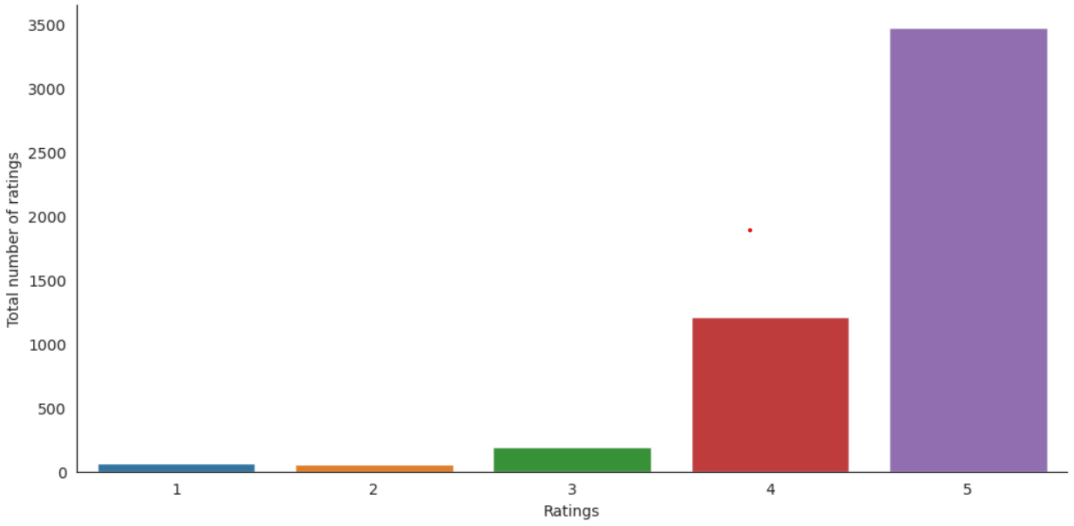
such as the column data types and missing values. dt.shape returns the number of rows and columns in the DataFrame.

```
In [17]: dt.isnull().sum()

Out[17]: reviews.username    0
         asins               0
         reviews.rating      0
         dtype: int64
```

This code checks for missing values in the dt DataFrame and returns the sum of missing values for each column.

### 3.3 Exploratory Data Analysis (EDA)

```
In [18]: with sns.axes_style('white'):
             g = sns.catplot(x="reviews.rating", data=dt, aspect=2.0, kind='count')
             g.set_axis_labels("Ratings", "Total number of ratings")

Out[18]: <seaborn.axisgrid.FacetGrid at 0x213b7d9b520>
```




The code creates a bar plot to show the count of different ratings in the dataset. The x-axis represents the ratings, while the y-axis represents the total number of ratings for each rating value. It helps us understand how many ratings are given for each rating value and gives an overview of the distribution of ratings in the dataset.

```
In [19]: # Number of unique user id  in the data
         print('Number of unique users in Raw data = ', dt['reviews.username'].nunique())
         # Number of unique product id  in the data
         print('Number of unique product in Raw data = ', dt['asins'].nunique())

         Number of unique users in Raw data =  3815
         Number of unique product in Raw data =  17

In [20]: unique_products = dt['asins'].unique()
         print(unique_products)

         ['Kindle 8 Wifi' 'Bluetooth Speaker' 'TV' 'Echo-Plus' 'Powerfast Charger'
          'Kids Tablet' 'Kindle Wifi' 'Tablet Wifi 16gb' 'Tablet 8 Wifi 16GB'
          'Tablet 8 Wifi 32GB' 'Tablet 10 Wifi 8GB' 'Kindle + Charging Cover Wifi'
          'Kindle Voyage 4GB Wifi' 'Kindle 16GB Wifi'
          'Kindle Voyage 4GB Wifi+Cellular' 'Tablet with Alexa 32GB'
          'Tablet with Alexa 16GB']

In [21]: most_rated=dt.groupby('reviews.username').size().sort_values(ascending=False)[:10]
         print('Top 10 users based on ratings: \n',most_rated)

         Top 10 users based on ratings:
          reviews.username
         Mike      26
         Chris     14
         Dave      13
         Nick      13
         John      13
         Rick      13
         Bill      12
         Robert    12
         Tony      10
         Steve     10
         dtype: int64
```

```
In [87]: counts=dt['reviews.username'].value_counts()
         dt1=dt[dt['reviews.username'].isin(counts[counts>5].index)]
         dt1
```

Out[87]:

| | reviews.username | asins | reviews.rating |
|---|---|---|---|
| 16 | John | Kindle 8 Wifi | 5 |
| 17 | Bill | Kindle 8 Wifi | 4 |
| 26 | Debbie | Kindle 8 Wifi | 5 |
| 30 | Jeff | Kindle 8 Wifi | 4 |
| 49 | Mark | Kindle 8 Wifi | 4 |
| ... | ... | ... | ... |
| 4937 | Robert | Tablet with Alexa 16GB | 5 |
| 4944 | Alex | Tablet with Alexa 16GB | 5 |
| 4952 | Dave | Tablet with Alexa 16GB | 5 |
| 4953 | Tablet | Tablet with Alexa 16GB | 1 |
| 4968 | Anonymous | Tablet with Alexa 16GB | 5 |

302 rows × 3 columns

The code carries out a number of data modeling-related operations. It begins by locating and displaying the dataset's unique products. The top 10 users are then displayed after the top users are determined based on the number of ratings they have provided. It then creates a new dataset called "dt1" by filtering the data to only include users who have evaluated three or more items. The text-based columns in the filtered dataset are then described in detail by the code. Additionally, it prints the total number of unique users and unique products in the final filtered dataset as well as the number of users who have rated five or more distinct items. The code then generates a new table called "dt1_agg" and organises the results according to the average ratings each user gave various products. The

understanding of the dataset's characteristics and the data modelling process are both aided by these operations.

```
In [88]: dt1.describe(include='object')
```

Out[88]:

| | reviews.username | asins |
|---|---|---|
| count | 302 | 302 |
| unique | 32 | 17 |
| top | Mike | Bluetooth Speaker |
| freq | 26 | 62 |

```
In [89]: print('Number of users who have rated 3 or more items =', len(dt1))
         print('Number of unique users in the final data = ', dt1['reviews.username'].nunique())
         print('Number of unique products in the final data = ', dt1['asins'].nunique())
```

```
Number of users who have rated 3 or more items = 302
Number of unique users in the final data =  32
Number of unique products in the final data =  17
```

```
In [90]: dt1_agg = dt1.groupby(['reviews.username', 'asins'])['reviews.rating'].mean().unstack(fill_value=0)
         dt1_agg = dt1_agg.round(decimals=1)  # Round to 1 decimal places

         userRates= dt1_agg.rename_axis(None, axis=1)
         userRates.shape
```

```
Out[90]: (32, 17)
```

```
In [91]: userRates
```

The code carries out the following tasks: it computes and displays the number of users who have rated three or more items, the number of unique users, and the number of unique products in the dataset. It also provides descriptive statistics for columns with object data types in the 'dt1' DataFrame. The DataFrame is divided into user and product groups, the mean rating is computed for each group, and a new DataFrame is made with the mean ratings unstacked. Next, one decimal place is added to the resulting DataFrame. The shape of the DataFrame, which represents the number of distinct users and distinct products in the dataset, is then obtained.

## 3.4 Data Modelling

```
In [92]: #Split the data randomnly into train and test datasets into 70:30 ratio
trainDt, testDt = train_test_split(dt1, test_size = 0.3, random_state=0)
trainDt.head()
testDt.head()
```

Out[92]:

| | reviews.username | asins | reviews.rating |
|---|---|---|---|
| 2219 | 1234 | Kids Tablet | 3 |
| 1596 | Jeff | Kindle Wifi | 4 |
| 4812 | Lucy | Tablet 8 Wifi 32GB | 5 |
| 687 | Steve | Bluetooth Speaker | 5 |
| 2221 | Terry | Kids Tablet | 5 |

Out[92]:

| | reviews.username | asins | reviews.rating |
|---|---|---|---|
| 3786 | Mike | Tablet 8 Wifi 16GB | 3 |
| 2574 | Greg | Tablet Wifi 16gb | 5 |
| 3816 | James | Tablet 8 Wifi 16GB | 5 |
| 3401 | Jeff | Kindle Voyage 4GB Wifi | 4 |
| 838 | Mike | Bluetooth Speaker | 5 |

```
In [93]: print('Shape of training data: ',trainDt.shape)
print('Shape of testing data: ',testDt.shape)
```

```
Shape of training data:  (211, 3)
Shape of testing data:  (91, 3)
```

```
In [94]: #Count of user_id for each unique product as recommendation score
trainDt_grouped = trainDt.groupby('asins').agg({'reviews.username': 'count'}).reset_index()
trainDt_grouped.rename(columns = {'reviews.username': 'score'},inplace=True)
trainDt_grouped=trainDt_grouped.sort_values(by='score',ascending=False)
trainDt_grouped.shape
trainDt_grouped
```

Out[94]: (17, 2)

Out[94]:

| | asins | score |
|---|---|---|
| 0 | Bluetooth Speaker | 46 |
| 1 | Echo-Plus | 30 |
| 12 | Tablet 8 Wifi 16GB | 27 |
| 2 | Kids Tablet | 23 |
| 4 | Kindle 16GB Wifi | 20 |
| 14 | Tablet Wifi 16gb | 13 |
| 13 | Tablet 8 Wifi 32GB | 8 |
| 8 | Kindle Wifi | 8 |
| 3 | Kindle + Charging Cover Wifi | 7 |
| 15 | Tablet with Alexa 16GB | 7 |
| 5 | Kindle 8 Wifi | 6 |
| 11 | Tablet 10 Wifi 8GB | 4 |
| 6 | Kindle Voyage 4GB Wifi | 4 |
| 10 | TV | 3 |
| 16 | Tablet with Alexa 32GB | 3 |
| 7 | Kindle Voyage 4GB Wifi+Cellular | 1 |

The code splits the 'dt1' dataset randomly into training and testing datasets in a 70:30 ratio. It displays the first few rows of the training and testing datasets and prints the shape of both datasets to indicate the number of rows and columns. The code then groups the training dataset by product ('asins') and calculates the count of unique user IDs

('reviews.username') for each product, assigning it as the 'score' column in the resulting DataFrame. The DataFrame is sorted in descending order based on the 'score' column. The shape of the grouped DataFrame is displayed, indicating the number of unique products and their corresponding scores.

In [95]:
```
#Sort the products on recommendation score
trainDt_sort = trainDt_grouped.sort_values(['score', 'asins'], ascending = [0,1])

#Generate a recommendation rank based upon score
trainDt_sort['rank'] = trainDt_sort['score'].rank(ascending=0, method='first')

#Get the top 10 recommendations
popRec= trainDt_sort.head()
popRec
```

Out[95]:

| | asins | score | rank |
|---|---|---|---|
| 0 | Bluetooth Speaker | 46 | 1.0 |
| 1 | Echo-Plus | 30 | 2.0 |
| 12 | Tablet 8 Wifi 16GB | 27 | 3.0 |
| 2 | Kids Tablet | 23 | 4.0 |
| 4 | Kindle 16GB Wifi | 20 | 5.0 |

the product ranked first is the most popular product.

The code uses the recommendation score in descending order to sort the 'trainDt_grouped' DataFrame, which holds the count of user ratings for each distinct product. Products are arranged in the 'trainDt_sort' DataFrame in ascending order by product ID ('asins') and descending order by recommendation score ('score'). The 'trainDt_sort' DataFrame now includes a column called 'rank' that gives each product a recommendation rank based on its score. Higher scores result in lower ranks, which are determined in descending order. The code then chooses the top 10 recommendations by transferring the first 10 rows from the 'trainDt_sort' DataFrame to the 'popRec' DataFrame.

In [96]:
```
# Use popularity based recommender model to make predictions
def recommend(user_id):
    userRec = popRec

    #Add user_id column for which the recommendations are being generated
    userRec['userID'] =user_id

    #Bring user_id column to the front
    cols = userRec.columns.tolist()
    cols = cols[-1:] + cols[:-1]
    userRec = userRec[cols]

    return userRec
```

In [97]:
```
find_recom = [10,100,150]   # This list is user choice.
for i in find_recom:
    print("The list of recommendations for the userId: %d\n" %(i))
    print(recommend(i))
    print("\n")
```

```
The list of recommendations for the userId: 10

    userID              asins  score  rank
0       10   Bluetooth Speaker     46   1.0
1       10           Echo-Plus     30   2.0
12      10  Tablet 8 Wifi 16GB     27   3.0
2       10         Kids Tablet     23   4.0
4       10    Kindle 16GB Wifi     20   5.0


The list of recommendations for the userId: 100

    userID              asins  score  rank
0      100   Bluetooth Speaker     46   1.0
1      100           Echo-Plus     30   2.0
12     100  Tablet 8 Wifi 16GB     27   3.0
2      100         Kids Tablet     23   4.0
4      100    Kindle 16GB Wifi     20   5.0


The list of recommendations for the userId: 150

    userID              asins  score  rank
0      150   Bluetooth Speaker     46   1.0
1      150           Echo-Plus     30   2.0
12     150  Tablet 8 Wifi 16GB     27   3.0
2      150         Kids Tablet     23   4.0
4      150    Kindle 16GB Wifi     20   5.0
```

The code defines a 'recommend' function that generates recommendations for a given user ID based on the popularity model. It takes a user ID as input, adds it to the 'popRec' DataFrame, and returns the modified DataFrame. The loop iterates over a list of user IDs and prints the recommendations for each user.

**collaborative filtering**

In [98]:
```
dt_CF = pd.concat([trainDt, testDt]).reset_index()
dt_CF.head()
```

Out[98]:

| | index | reviews.username | asins | reviews.rating |
|---|---|---|---|---|
| 0 | 2219 | 1234 | Kids Tablet | 3 |
| 1 | 1596 | Jeff | Kindle Wifi | 4 |
| 2 | 4812 | Lucy | Tablet 8 Wifi 32GB | 5 |
| 3 | 687 | Steve | Bluetooth Speaker | 5 |
| 4 | 2221 | Terry | Kids Tablet | 5 |

In [99]:
```
import pandas as pd

pivotDt = round(pd.pivot_table(dt_CF, index='reviews.username', columns='asins', values='reviews.rating', fill_value=0),1)
pivotDt
```

```
In [102]: pivotDt.set_index(['user_index'], inplace=True)

          # Actual ratings given by users
          pivotDt
```

Out[102]:

| asins | Bluetooth Speaker | Echo-Plus | Kids Tablet | Kindle + Charging Cover Wifi | Kindle 16GB Wifi | Kindle 8 Wifi | Kindle Voyage 4GB Wifi | Kindle Voyage 4GB Wifi+Cellular | Kindle Wifi | Powerfast Charger | TV | Tablet 10 Wifi 8GB | Tablet 8 Wifi 16GB | Tablet 8 Wifi 32GB | Tablet Wifi 16gb | Tablet with Alexa 16GB | Tablet with Alexa 32GB |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_index | | | | | | | | | | | | | | | | | |
| 0 | 4.5 | 0.0 | 3.0 | 0 | 5.0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 0 | 5 | 5 | 0 |
| 1 | 0.0 | 0.0 | 4.0 | 0 | 5.0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 4.7 | 0 | 4 | 5 | 0 |
| 2 | 5.0 | 4.5 | 0.0 | 0 | 4.0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 4.5 | 0 | 4 | 5 | 0 |
| 3 | 0.0 | 3.5 | 0.0 | 0 | 5.0 | 4 | 0 | 5 | 4.3 | 0 | 0 | 0 | 5.0 | 0 | 4 | 0 | 0 |
| 4 | 5.0 | 5.0 | 4.5 | 4 | 5.0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 0 | 5 | 0 | 0 |
| 5 | 0.0 | 0.0 | 5.0 | 0 | 4.0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 3 | 4.0 | 5 | 5 | 4 | 5 |
| 6 | 5.0 | 5.0 | 5.0 | 5 | 0.0 | 3 | 0 | 0 | 3.0 | 0 | 0 | 5 | 0.0 | 0 | 5 | 0 | 0 |
| 7 | 0.0 | 5.0 | 5.0 | 0 | 5.0 | 0 | 5 | 0 | 0.0 | 0 | 0 | 0 | 4.3 | 0 | 5 | 5 | 0 |
| 8 | 5.0 | 5.0 | 5.0 | 0 | 0.0 | 0 | 0 | 0 | 0.0 | 0 | 0 | 0 | 4.3 | 0 | 0 | 0 | 0 |

The 'pd.concat' function is used to first combine the 'trainDt' and 'testDt' DataFrames into a single DataFrame called 'dt_CF'. Then, 'pivotDt' is created from 'dt_CF', a pivot table where the rows stand in for users, the columns for products ('asins'), and the values for the user ratings for each product. The pivot table is rounded to one decimal place, and any missing values are replaced with a value of 0. The 'user_index' column has been added to give each user a special index. Finally, the new index for "pivotDt" is set to be the "user_index" column.

```
In [103]: import scipy.sparse as sp

          is_sparse = sp.issparse(pivotDt)

          if is_sparse:
              print("The DataFrame is a sparse matrix.")
          else:
              print("The DataFrame is not a sparse matrix.")

          The DataFrame is not a sparse matrix.
```

***sparse matrix=number of zero in data frame is 2/3 more than number of nonzeros. Since it is not a sparse matrix, svd (singular value decomposition will be applied using numpy library.***

```
In [104]: # Convert DataFrame to NumPy array
          pvArray = pivotDt.values

          # Perform singular value decomposition (a factorization of a real or complex matrix)
          U, sigma, Vt = np.linalg.svd(pvArray, full_matrices=False)

In [105]: print('Left singular matrix: \n',U)
```

```
In [106]: print('Sigma: \n', sigma)

          Sigma:
           [45.86746973 18.15312633 16.93956227 15.67330206 13.91878706 12.89713661
           11.93315399  9.90776722  9.0571973   8.10163739  6.92427515  6.61795148
            5.78590877  4.74845311  4.62460998  3.58752128  2.88811059]

In [107]: sigma = np.diag(sigma)
          print('Diagonal matrix: \n',sigma)
```

If the 'pivotDt' DataFrame is sparse, the code determines this and prints the appropriate message. The DataFrame is then changed into a NumPy array. The matrices 'U','sigma', and 'Vt' are obtained using the array and singular value decomposition (SVD). The code

outputs the singular values ('sigma'), builds a diagonal matrix using those values, and then outputs the correct singular matrix ('Vt'). This code applies SVD to extract latent features and make predictions as part of data modelling.

```
In [110]: # Recommend the items with the highest predicted ratings

def recommend_items(userID, pivotDt, predsDt, num_recommendations):
    # index starts at 0
    user_idx = userID-1

    # Get and sort the user's ratings
    sorted_user_ratings = pivotDt.iloc[user_idx].sort_values(ascending=False)

    #sorted_user_ratings
    sorted_user_predictions = predsDt.iloc[user_idx].sort_values(ascending=False)

    #sorted_user_predictions
    temp = pd.concat([sorted_user_ratings, sorted_user_predictions], axis=1)
    temp.index.name = 'Recommended Items'
    temp.columns = ['user_ratings', 'user_predictions']
    temp = temp.loc[temp.user_ratings == 0]
    temp = temp.sort_values('user_predictions', ascending=False)

    print('\nBelow are the recommended items for user(user_id = {}):\n'.format(userID))
    print(temp.head(num_recommendations))

userID = 7 num_recommendations = 5 recommend_items(userID, pivotDt, predsDt, num_recommendations)


userID = 8 num_recommendations = 5 recommend_items(userID, pivotDt, predsDt, num_recommendations)

In [111]: userRates.head()
```

The recommendation_items function, which is defined in the code, accepts the user ID, the pivotDt DataFrame (which contains actual ratings), the predsDt DataFrame (which contains predicted ratings), and the desired number of recommendations. To align the user ID with the index, which starts at 0, it begins by deducting 1 from the user ID. The user's actual ratings and predicted ratings are then retrieved and sorted. The code combines these two series, names the columns, filters out items that have already been rated by the user, and then sorts the remaining items in descending order according to the predicted ratings. The top 'num_recommendations' items from the sorted DataFrame are then shown to print the recommended items for the specified user ID.

```
In [111]: userRates.head()
```

Out[111]:

| reviews.username | Bluetooth Speaker | Echo-Plus | Kids Tablet | Kindle + Charging Cover Wifi | Kindle 16GB Wifi | Kindle 8 Wifi | Kindle Voyage 4GB Wifi | Kindle Voyage 4GB Wifi+Cellular | Kindle Wifi | Powerfast Charger | TV | Tablet 10 Wifi 8GB | Tablet 8 Wifi 16GB | Tablet 8 Wifi 32GB | Tablet Wifi 16gb | Tablet with Alexa 16GB | Table witl Alex 32GI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1234 | 4.5 | 0.0 | 3.0 | 0.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 | 5.0 | 0. |
| Alex | 0.0 | 0.0 | 4.0 | 0.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.7 | 0.0 | 4.0 | 5.0 | 0. |
| Anonymous | 5.0 | 4.5 | 0.0 | 0.0 | 4.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 4.5 | 0.0 | 4.0 | 5.0 | 0. |
| Bill | 0.0 | 3.5 | 0.0 | 0.0 | 5.0 | 4.0 | 0.0 | 5.0 | 4.3 | 0.0 | 0.0 | 0.0 | 5.0 | 0.0 | 4.0 | 0.0 | 0. |
| Bobby | 5.0 | 5.0 | 4.5 | 4.0 | 5.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 5.0 | 0.0 | 0. |

```
In [112]: predsDt.head()
```

Out[112]:

| asins | Bluetooth Speaker | Echo-Plus | Kids Tablet | Kindle + Charging Cover Wifi | Kindle 16GB Wifi | Kindle 8 Wifi | Kindle Voyage 4GB Wifi | Kindle Voyage 4GB Wifi+Cellular | Kindle Wifi | Powerfast Charger | TV | Tabl Wifi |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4.500000e+00 | 2.160891e-14 | 3.000000e+00 | 1.179161e-14 | 5.0 | 3.555413e-16 | 1.569930e-14 | -3.705295e-15 | -1.930856e-15 | 2.667947e-15 | 7.026510e-15 | 4.5477 |
| 1 | -3.585290e-15 | 1.423709e-14 | 4.000000e+00 | -4.860361e-15 | 5.0 | 5.998164e-15 | 8.425640e-16 | 2.291222e-15 | 6.403211e-15 | 3.751543e-16 | 1.307396e-15 | -2.440( |
| 2 | 5.000000e+00 | 4.500000e+00 | 5.516834e-15 | 2.363110e-15 | 4.0 | -5.337771e-15 | 4.509379e-15 | 1.589677e-15 | -1.663701e-15 | 1.115863e-15 | 2.175351e-15 | -2.174( |
| 3 | 6.034693e-16 | 3.500000e+00 | 6.138838e-15 | -1.657935e-14 | 5.0 | 4.000000e+00 | -3.302424e-15 | 5.000000e+00 | 4.300000e+00 | -1.209157e-15 | -1.255398e-15 | -1.1094 |
| 4 | 5.000000e+00 | 5.000000e+00 | 4.500000e+00 | 4.000000e+00 | 5.0 | -5.824411e-15 | 1.391057e-14 | -3.008089e-15 | -6.497674e-15 | 5.444939e-15 | 4.618183e-15 | -6.496( |

The userRates DataFrame, which contains the average user ratings provided for various products, is displayed in its first few rows by the userRates.head() code. It gives a summary of the typical ratings given by each user. Predicted ratings for users and products are displayed in the first few rows of the predsDt DataFrame using the predsDt.head() function. It offers a preview of the predicted ratings that a recommendation model produces.

### 3.5 Data Evaluation

```
In [113]: meanRate=userRates.mean()

In [114]: meanPred=predsDt.mean()

In [115]: rmse_df = pd.concat([meanRate, meanPred], axis=1)
          rmse_df.columns = ['Avg_actual_ratings', 'Avg_predicted_ratings']
          print(rmse_df.shape)

          rmse_df['item_index'] = np.arange(0, rmse_df.shape[0], 1)
          rmse_df

          (17, 2)
```

Out[115]:

| | Avg_actual_ratings | Avg_predicted_ratings | item_index |
|---|---|---|---|
| **Bluetooth Speaker** | 3.750000 | 3.750000 | 0 |
| **Echo-Plus** | 3.165625 | 3.165625 | 1 |
| **Kids Tablet** | 2.765625 | 2.765625 | 2 |
| **Kindle + Charging Cover Wifi** | 0.593750 | 0.593750 | 3 |
| **Kindle 16GB Wifi** | 2.975000 | 2.975000 | 4 |
| **Kindle 8 Wifi** | 0.937500 | 0.937500 | 5 |
| **Kindle Voyage 4GB Wifi** | 0.843750 | 0.843750 | 6 |
| **Kindle Voyage 4GB Wifi+Cellular** | 0.312500 | 0.312500 | 7 |
| **Kindle Wifi** | 1.103125 | 1.103125 | 8 |
| **Powerfast Charger** | 0.156250 | 0.156250 | 9 |
| **TV** | 0.593750 | 0.593750 | 10 |
| **Tablet 10 Wifi 8GB** | 0.843750 | 0.843750 | 11 |
| **Tablet 8 Wifi 16GB** | 2.906250 | 2.906250 | 12 |

The code figures out what users and products are averaged for in terms of actual and predicted ratings. These averages are combined into a DataFrame with the columns 'Avg_actual_ratings' and 'Avg_predicted_ratings', which is called rmse_df. The number of rows and columns is displayed using the rmse_df shape. The 'item_index' column, which has a sequential range of values from 0 to the number of rows in the DataFrame, is also added to the rmse_df table. This code makes it easier to evaluate and analyse the mean ratings and determine how accurately the predictions were made.

```
In [116]: #RMSE=Root Mean Square Error (measure differences between values predicted by a model or an estimator )
          from IPython.display import Image
          Image("RMSE1.jpg")
```

Out[116]:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}}$$

```
In [117]: #apply formula
          RMSE = round((((rmse_df.Avg_predicted_ratings - rmse_df.Avg_actual_ratings) ** 2).mean()) ** 0.5, 2)
          print('\nRMSE SVD Model = {} \n'.format(RMSE))

          RMSE SVD Model = 0.0
```

The code determines the root mean square error (RMSE) to gauge the variations between the ratings that were predicted and those that were received. The average squared differences between average actual ratings and average predicted ratings are calculated using the RMSE formula. The mean of these squared differences is then taken, and the square root of the mean is calculated. Two decimal places are added to the calculated

RMSE value. This metric evaluates the Singular Value Decomposition (SVD) model's overall performance in predicting ratings.

## 4.0 RESULTS & DISCUSSION

### Main Page

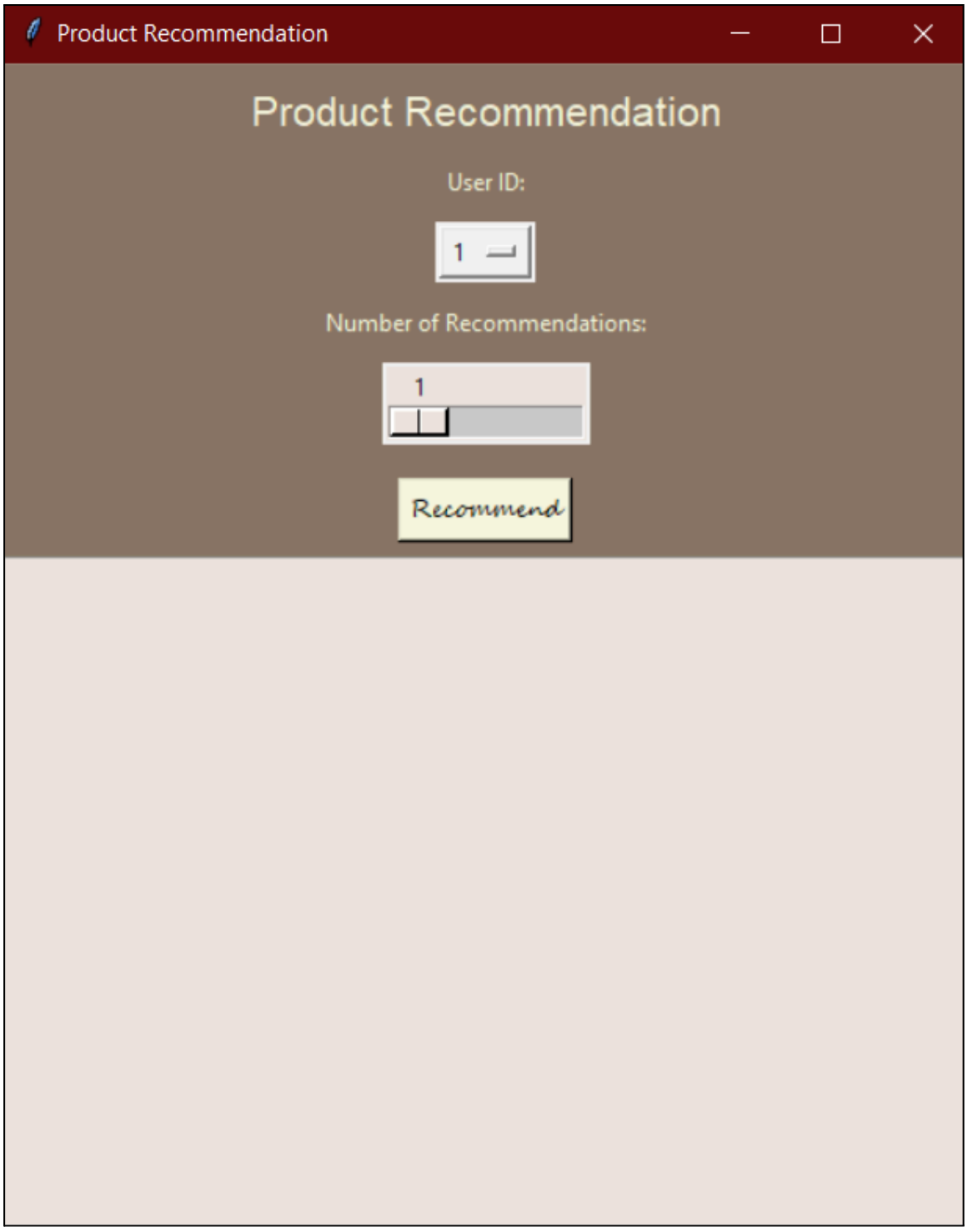
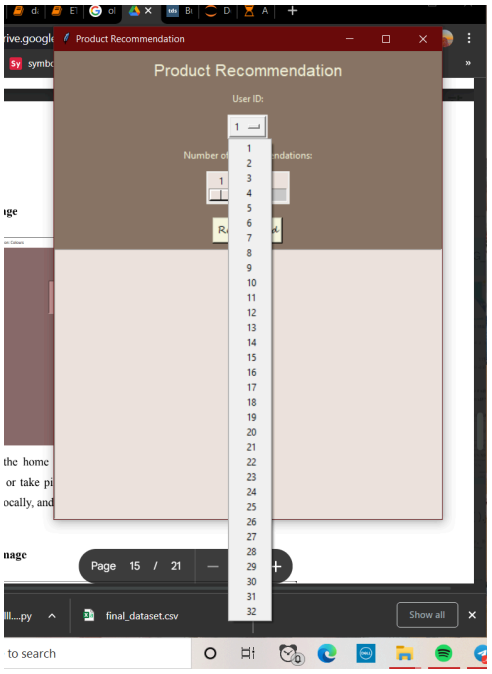
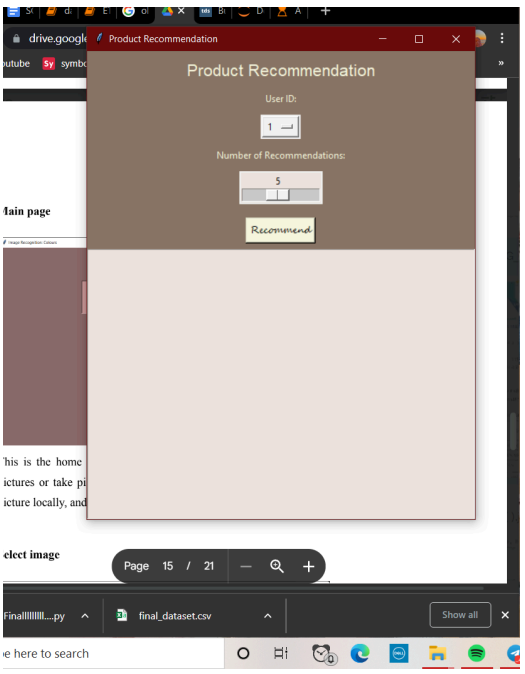



*Figure 4.1 : Main Page*

Figure above is the home page of the program. On this page, users need to choose their user ID and number of recommendations that they want. The user ID button using the dropdown button while the number of recommendations using the slide button.




After the user input the data needed, the user needs to click on the Recommend button and the recommended item will appear at the bottom based on the number they input.

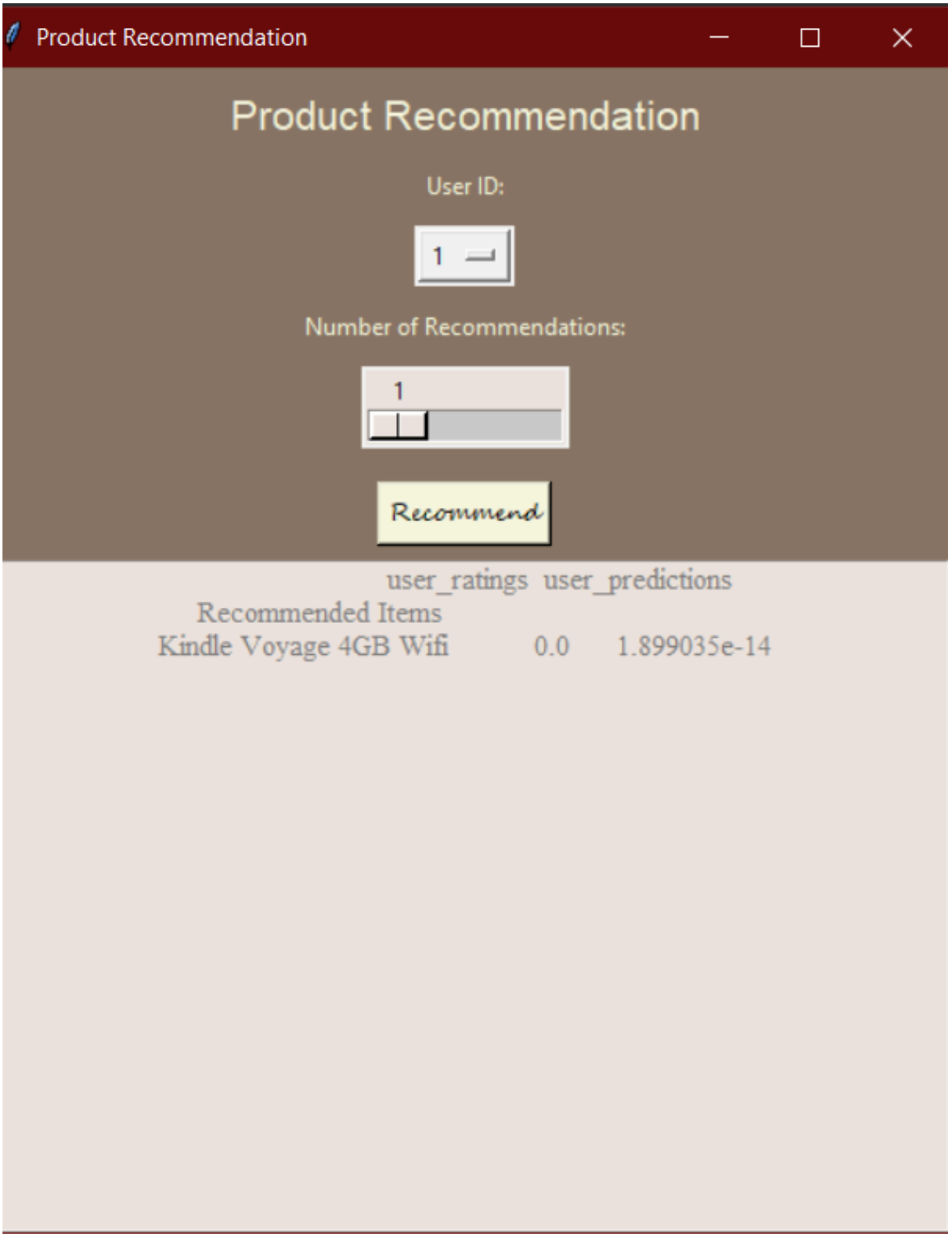



*Figure 4.2: Final Page*

From the figure above, we can see that the recommended item for this user is Kindle Voyage 4GB Wifi with the user_predictions 1.899035e-14.

This is another example to show that our program is running without problem.

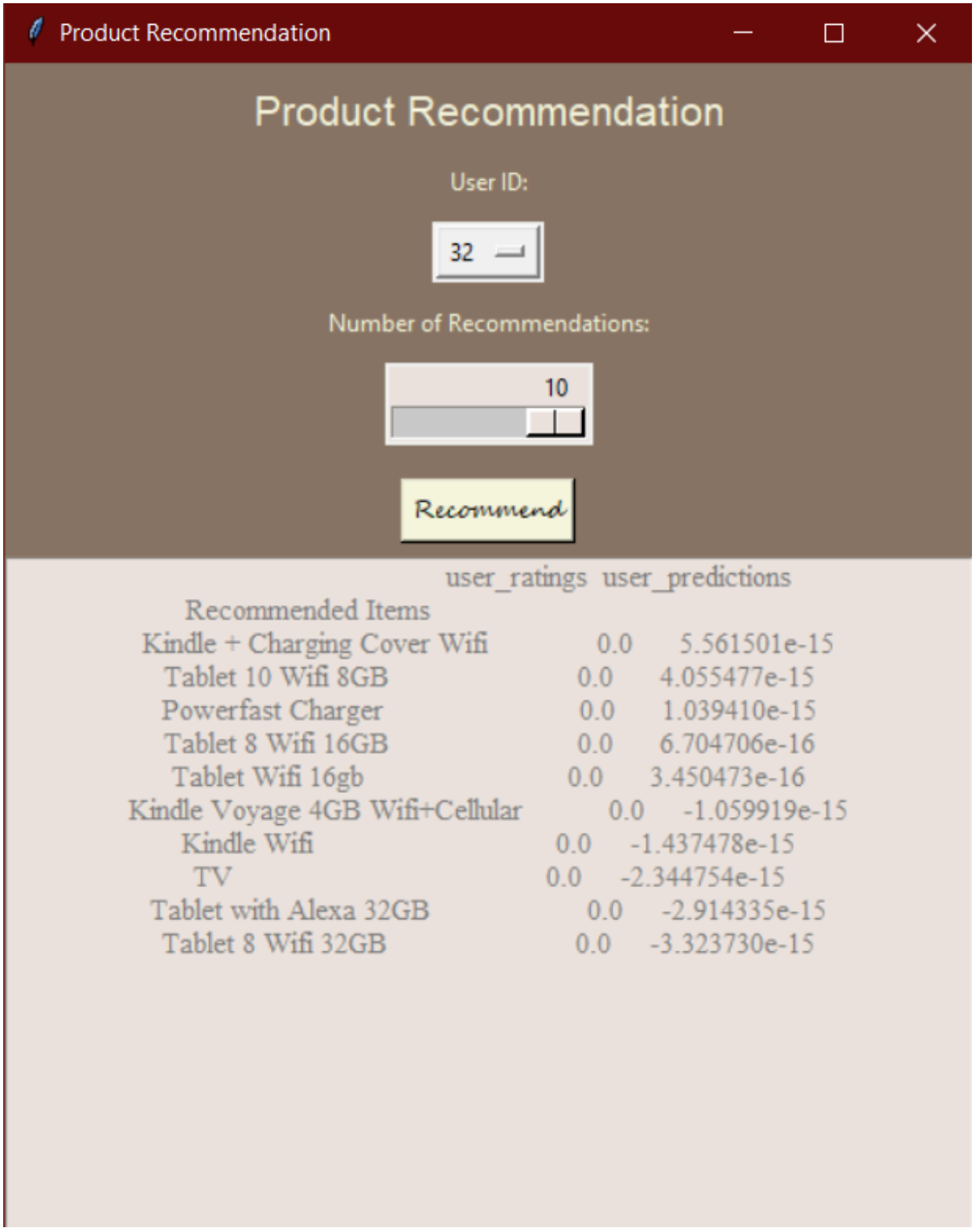



*Figure 4.3: Final Page*

## 5.0 CONCLUSION

In conclusion, the main goal of this project is to create a recommendation algorithm for Amazon's electrical items with the intention of providing consumers with tailored suggestions. The project intends to improve the user experience and boost customer satisfaction on the Amazon platform by using the dataset and applying various approaches including data analysis, user preference modelling, collaborative filtering, and content-based filtering. The benefits of this study reach beyond the narrow field of recommendation algorithms and affect society, the environment, and industrial systems more broadly. First off, the algorithm's personalised recommendations greatly enhance the user experience on Amazon by providing pertinent and customised advice. This makes it simple for people to choose goods that suit their interests and tastes, making their shopping experience more pleasurable and gratifying.

The algorithm's capacity to lessen information overload is also notable. It streamlines the decision-making process and saves consumers time and effort when looking for suitable items among the numerous possibilities accessible on Amazon by filtering and providing them with relevant product recommendations. The algorithm's recommendation method helps consumers make more educated purchases based on their tastes while still being resource-efficient. As a result, the possibility of returns or exchanges decreases, resulting in less waste and resource consumption from needless product handling and transportation.

From a financial standpoint, Amazon's marketing and sales initiatives may benefit from the built recommendation system. The organisation may create discounts and targeted advertising campaigns that are tailored to user preferences and behaviour, improving conversion rates and overall business success. Amazon benefits from the project's data analysis component's insightful analyses. With a greater understanding of consumer behaviour, product trends, and market demand, the corporation will be better able to manage inventories and make wise business decisions.

Additionally, the created recommendation algorithm's scalability and flexibility are important contributions. Beyond electrical devices, the basic ideas and methodologies may be applied to other businesses and domains, such as recommending television shows, books, music, or personalised content on various web platforms. The project's advantages and possible uses are expanded as a result. The influence of this effort goes beyond the

immediate realm of recommendation algorithms. It helps to improve customer happiness, resource efficiency, and user experience while also giving businesses useful information. The initiative aims to provide a win-win situation for customers, the business, and the greater online retail ecosystem by utilising the power of recommendation algorithms.

## REFERENCES

*Amazon-Product-Recommendation/Recommendation System.ipynb ...* (n.d.). GitHub. Retrieved June 20, 2023, from https://github.com/LaxmiChaudhary/Amzon-Product-Recommendation/blob/master/Recommendation%20System.ipynb

*A Complete Study of Amazon's Recommendation System*. (n.d.). Argoid. Retrieved June 20, 2023, from https://www.argoid.ai/blog/decoding-amazons-recommendation-system

Sainato, L. (n.d.). *(DOC) Strategy Recommendation Project Final | James Bellew*. Academia.edu. Retrieved June 20, 2023, from https://www.academia.edu/7398635/Strategy_Recommendation_Project_Final

Valdata, G. (2022, November 25). *Building a Recommender System for Amazon Products with Python*. Towards Data Science. Retrieved June 20, 2023, from https://towardsdatascience.com/building-a-recommender-system-for-amazon-products-with-python-8e0010ec772c