Assignment -3
Priyanshu
2020106

I have downloaded the dataset from amazon site and removed the useless information from the dataset on an external machine so that i can pickle the dataset on my pc

```python
import pickle
# Load the pickle file
def load():
    with open('data.pickle', 'rb') as f:
        loaded_data = pickle.load(f)
        return loaded_data

df = load()
```

✓ 9.7s   Python

```python
import pickle
# Load the pickle file
def load():
    with open('metadata.pkl', 'rb') as f:
        loaded_data = pickle.load(f)
        return loaded_data

meta_df = load()
```

✓ 2.2s   Python

Filtered the dataset only for headphones category

```python
meta_df['category'] = meta_df['category'].apply(lambda x: [category.lower().strip() for category in x])

# Filter rows where "headphones" is in the categories list
meta_df = meta_df[meta_df['category'].apply(lambda x: 'headphones' in x)]

# Reset the index after filtering
meta_df.reset_index(drop=True, inplace=True)

print(meta_df)
```

[133] ✓ 4.0s

```
             asin                                              title  \
0      0132492776  Wireless Bluetooth Headphones Earbuds with Mic...
1      0558835155  Polaroid Pbm2200 PC / Gaming Stereo Headphones...
2      0594478162                Official Nook Audio Ie250 Earphones
3      0646531158  In Search of Tom Bowen and the Therapy He Insp...
4      0684873176  The Mutineer: Rants, Ravings, and Missives fro...
...           ...                                                ...
31110  B01HJ8E11E  Bluetooth Headphones,Yostyle Wireless Headphon...
31111  B01HJAVYDU  MAXROCK Noise Isolating Sleeping Headphones Ea...
31112  B01HJAPNHI  Wireless Bluetooth Headset, HandsFree Wireless...
31113  B01HJAVZI4  MAXROCK Wired Headphones In-ear Headphone Spor...
31114  B01HJAVXMC  MAXROCK Noise Isolating Sleeping Headphones Ea...

                 brand                                    category
0      Enter The Arena  [electronics, headphones, earbud headphones]
1             Polaroid                    [electronics, headphones]
2                 Nook  [electronics, headphones, earbud headphones]
3          Fitquipment                    [electronics, headphones]
4      Enter The Arena  [electronics, headphones, earbud headphones]
...                ...                                          ...
31110          Yostyle  [electronics, headphones, earbud headphones]
31111          MAXROCK  [electronics, headphones, earbud headphones]
31112          snorain  [electronics, headphones, earbud headphones]
31113          MAXROCK  [electronics, headphones, earbud headphones]
31114          MAXROCK  [electronics, headphones, earbud headphones]

[31115 rows x 4 columns]
```

Answer 4th

```python
num_reviews = len(df)
print(f"Number of reviews for headphones: {num_reviews}")

avg_rating_score = df['overall'].mean()
print(f"Average rating score for headphones: {avg_rating_score:.2f}")

num_unique_products = len(asin_set)
print(f"Number of unique products for headphones: {num_unique_products}")

num_good_ratings = len(df[df['overall'] >= 3])
print(f"Number of good ra (type alias) df: Any {num_good_ratings}")

num_bad_ratings = len(df[df['overall'] < 3])
print(f"Number of bad ratings for headphones: {num_bad_ratings}")


reviews_per_rating = df['overall'].value_counts().sort_index()
print(f"Reviews per rating for headphones:\n{reviews_per_rating}")
```

[145] ✓ 0.0s

```
Number of reviews for headphones: 372167
Average rating score for headphones: 4.01
Number of unique products for headphones: 30471
Number of good ratings for headphones: 312041
Number of bad ratings for headphones: 60126
Reviews per rating for headphones:
overall
1.0     31616
2.0     28510
3.0     41427
4.0     75024
5.0    195590
Name: count, dtype: int64
```

Preprocessing steps

```python
# Function to remove HTML tags
def remove_html_tags(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

# Function to remove accented characters
def remove_accented_chars(text):
    text = unicodedata.normalize('NFKD', text).encode('ascii', 'ignore').decode('utf-8', 'ignore')
    return text

# Function to expand acronyms
def expand_acronyms(text):
    # Define acronym dictionary (add more as needed)
    acronyms = {
        "lol": "laugh out loud",
        "brb": "be right back"
        # Add more acronyms and their expansions here
    }
    # Replace acronyms with expansions
    for acronym, expansion in acronyms.items():
        text = re.sub(r'\b' + re.escape(acronym) + r'\b', expansion, text)
    return text

# Function to remove special characters
def remove_special_characters(text):
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    return text

# Function for lemmatization
def lemmatize_text(text):
    lemma_words = []
    doc = nlp(text)
    for token in doc:
        lemma_words.append(token.lemma_)
    return ' '.join(lemma_words)

# Function for text normalization
def normalize_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'\s+', ' ', text).strip()  # Remove extra whitespaces
    return text
```

6 a

```python
import pandas as pd
merged_df = pd.merge(meta_df, df, on='asin', how='inner')


# Step 2: Count the occurrences of each brand
brand_counts = merged_df['brand'].value_counts()

# Step 3: Sort the counts in descending order
sorted_brand_counts = brand_counts.sort_values(ascending=False)

# Step 4: Extract the top 20 brands
top_20_brands = sorted_brand_counts.head(20)

# Display the top 20 most reviewed brands in the headphones category
print("Top 20 most reviewed brands in the headphones category:")
print(top_20_brands)
```

[148] ✓ 0.2s

```
Top 20 most reviewed brands in the headphones category:
brand
Sony              32955
Sennheiser        21516
Bose               9582
Plantronics        8340
Skullcandy         8316
JLAB               7731
JVC                7692
Audio-Technica     6791
Philips            6527
Panasonic          6053
Koss               5784
LG                 5624
Samsung            5604
Mpow               5480
Bluedio            5132
MEE audio          4644
Anker              4290
Symphonized        4284
TaoTronics         4059
Klipsch            4050
Name: count, dtype: int64
```

6b

```python
merged_df = pd.merge(meta_df, df, on='asin', how='inner')

# Step 2: Count the occurrences of each brand
brand_counts = meta_df['brand'].value_counts()

# Step 3: Sort the counts in descending order
sorted_brand_counts = brand_counts.sort_values(ascending=True)

# Step 4: Extract the top 20 brands
top_20_brands = sorted_brand_counts.head(20)

# Display the top 20 most reviewed brands in the headphones category
print("Top 20 least reviewed brands in the headphones category:")
print(top_20_brands)
```

[149] ✓ 0.2s

```
Top 20 least reviewed brands in the headphones category:
brand
kathy ireland CONNECT     1
BBK                       1
Better Products Global    1
Minidi                    1
HARD CORE TECH            1
veniam                    1
TONESOUL                  1
zhuoyue                   1
aoda                      1
Fantronics                1
CalorMixs                 1
RTPWireless               1
Nameo                     1
CYNDIE Wedding Favor      1
GPX                       1
KEKU                      1
ART                       1
fFLAT5                    1
Santagada Music           1
Lorida                    1
Name: count, dtype: int64
```

6c

```python
# Step 2: Group the data by product (title) and calculate the sum of positive class ratings
positive_ratings_sum = merged_df.groupby('title')['overall'].apply(lambda x: (x >= 3).sum())

# Step 3: Find the headphone with the highest sum of positive class ratings
most_positively_reviewed_headphone = positive_ratings_sum.idxmax()

brand_of_most_positively_reviewed = merged_df.loc[merged_df['title'] == most_positively_reviewed_headphone, 'brand'].iloc[0]
# Display the most positively reviewed headphone
print(f"The most positively reviewed headphone is: {most_positively_reviewed_headphone} provided by {brand_of_most_positively_reviewed}")
```

[150] ✓ 1.0s

```
The most positively reviewed headphone is: Sony MDRZX100 Headphones (Black) provided by Sony
```

6d

```python
# Assuming headphones_df is the DataFrame containing headphones data

# Step 1: Convert the reviewTime column to datetime format
df['reviewTime'] = pd.to_datetime(df['reviewTime'])

# Step 2: Extract the year from the reviewTime column
df['year'] = df['reviewTime'].dt.year

# Step 4: Group the data by year and count the ratings for each year
ratings_count_by_year = df.groupby('year')['overall'].count()

year_with_max_count = ratings_count_by_year.idxmax()

# Step 5: Display the count of ratings for each of the 5 consecutive years
print("Count of ratings for 'headphones' over 5 consecutive years:")
print(ratings_count_by_year)

print()
print()
```

```
Count of ratings for 'headphones' over 5 consecutive years:
year
2000       24
2001       45
2002       75
2003      123
2004      333
2005      893
2006     1545
2007     3049
2008     4837
2009     6808
2010     7327
2011    10447
2012    15092
2013    31878
2014    54989
2015    88899
2016    85594
2017    43527
2018    16682
Name: overall, dtype: int64
```

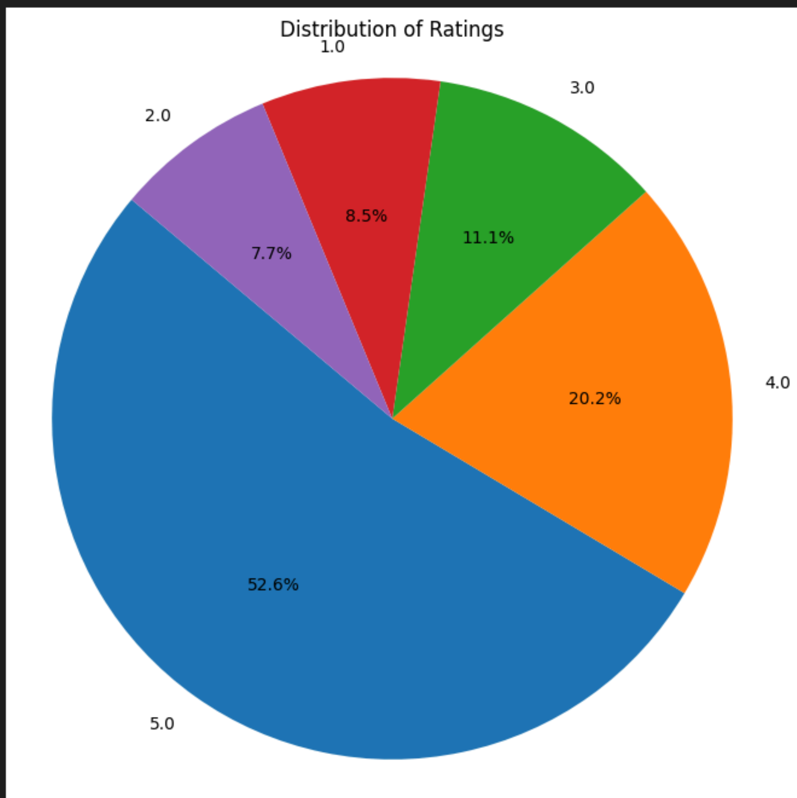
6f

```python
good_wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(good_word_freq)
bad_wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(bad_word_freq)

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.imshow(good_wordcloud, interpolation='bilinear')
plt.title('Word Cloud for Good Ratings')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(bad_wordcloud, interpolation='bilinear')
plt.title('Word Cloud for Bad Ratings')
plt.axis('off')

plt.show()
```

Most commons words are shown in large size in above images.

6g

[153] ✓ 0.1s




```python
# Assuming your DataFrame is named 'df'
# If the 'reviewTime' column is not already in datetime format, convert it
df['reviewTime'] = pd.to_datetime(df['reviewTime'])

# Group by year and count unique reviewerIDs
customer_count_per_year = df.groupby(df['reviewTime'].dt.year)['reviewerID'].nunique()

# Find the year with the highest number of customers
year_with_most_customers = customer_count_per_year.idxmax()
num_customers_highest_year = customer_count_per_year.max()

print(f"The year with the highest number of customers is {year_with_most_customers} with {num_customers_highest_year} customers.")
```

[132] ✓ 0.0s

```
The year with the highest number of customers is 2015 with 1870 customers.
```

Answer 10th

Model = LogisticRegression

```python
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

tfidf_vectorizer = TfidfVectorizer(max_features=50000)
X = tfidf_vectorizer.fit_transform(df['reviewText'])

df['rating_class'] = df['overall'].apply(lambda x: 'Good' if x > 3 else ('Average' if x == 3 else 'Bad'))
y = df['rating_class']

# Step 4: Split the data into training and testing sets (75:25 ratio)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Step 5: Train a machine learning model on the training data
model = LogisticRegression(max_iter=1000)  # You can choose a different classification model
model.fit(X_train, y_train)

# Step 6: Evaluate the model on the testing data
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

[11] ✓ 42.9s

```
Accuracy: 0.8256378839663808
Classification Report:
              precision    recall  f1-score   support

     Average       0.45      0.16      0.23     10256
         Bad       0.72      0.67      0.69     14985
        Good       0.86      0.96      0.91     67801

    accuracy                           0.83     93042
   macro avg       0.68      0.60      0.61     93042
weighted avg       0.79      0.83      0.80     93042
```

Model = RandomForestClassifier

```python
# Step 5: Train a machine learning model on the training data
model = RandomForestClassifier(n_estimators=100, random_state=42,n_jobs = -1)  # You can adjust the number of estimators as needed
model.fit(X_train, y_train)

# Step 6: Evaluate the model on the testing data
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

[115]

```
Accuracy: 0.8153414586960728
Classification Report:
              precision    recall  f1-score   support

     Average       0.74      0.12      0.20     10256
         Bad       0.80      0.53      0.63     14985
        Good       0.82      0.99      0.89     67801

    accuracy                           0.82     93042
   macro avg       0.79      0.54      0.58     93042
weighted avg       0.81      0.82      0.78     93042
```

Model = DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report


# Step 5: Train a machine learning model on the training data
model = DecisionTreeClassifier(random_state=42)  # You can adjust other parameters as needed
model.fit(X_train, y_train)

# Step 6: Evaluate the model on the testing data
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

[118]

```
Accuracy: 0.7495539648760775
Classification Report:
              precision    recall  f1-score   support

     Average       0.29      0.25      0.27     10256
         Bad       0.55      0.54      0.54     14985
        Good       0.85      0.87      0.86     67801

    accuracy                           0.75     93042
   macro avg       0.56      0.56      0.56     93042
weighted avg       0.74      0.75      0.75     93042
```

Model = KneighborsClassifiers

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer


# Step 5: Train a machine learning model on the training data
model = KNeighborsClassifier(n_neighbors=5, n_jobs = -1)  # You can adjust the number of neighbors as needed
model.fit(X_train, y_train)

# Step 6: Evaluate the model on the testing data
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

[116]

```
Accuracy: 0.7289073751639045
Classification Report:
              precision    recall  f1-score   support

     Average       0.27      0.07      0.12     10256
         Bad       0.48      0.20      0.28     14985
        Good       0.76      0.95      0.84     67801

    accuracy                           0.73     93042
   macro avg       0.50      0.41      0.41     93042
weighted avg       0.66      0.73      0.67     93042
```

Model = MultinomialNB

```python
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# Step 5: Train a machine learning model on the training data
model = MultinomialNB()
model.fit(X_train, y_train)

# Step 6: Evaluate the model on the testing data
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.778981535220653
Classification Report:
              precision    recall  f1-score   support

     Average       0.51      0.01      0.01     10256
         Bad       0.82      0.34      0.48     14985
        Good       0.78      0.99      0.87     67801

    accuracy                           0.78     93042
   macro avg       0.70      0.45      0.45     93042
weighted avg       0.75      0.78      0.71     93042
```

To compare the performance of the five machine learning models (Logical Regression, Random Forest, Decision Tree, K-Nearest Neighbors, Multinomial Naive Bayes), let's analyze their precision, recall, F1-score, and support for each of the three target classes (Average, Bad, Good).

Logical Regression Model:
Accuracy: 0.8256
Classification Report:
Average: Precision=0.45, Recall=0.16, F1-score=0.23, Support=10256
Bad: Precision=0.72, Recall=0.67, F1-score=0.69, Support=14985
Good: Precision=0.86, Recall=0.96, F1-score=0.91, Support=67801
Random Forest Model:
Accuracy: 0.8153
Classification Report:
Average: Precision=0.74, Recall=0.12, F1-score=0.20, Support=10256
Bad: Precision=0.80, Recall=0.53, F1-score=0.63, Support=14985
Good: Precision=0.82, Recall=0.99, F1-score=0.89, Support=67801
Decision Tree Model:
Accuracy: 0.7496
Classification Report:
Average: Precision=0.29, Recall=0.25, F1-score=0.27, Support=10256
Bad: Precision=0.55, Recall=0.54, F1-score=0.54, Support=14985

Good: Precision=0.85, Recall=0.87, F1-score=0.86, Support=67801
K-Nearest Neighbors Model:
Accuracy: 0.7289
Classification Report:
Average: Precision=0.27, Recall=0.07, F1-score=0.12, Support=10256
Bad: Precision=0.48, Recall=0.20, F1-score=0.28, Support=14985
Good: Precision=0.76, Recall=0.95, F1-score=0.84, Support=67801
Multinomial Naive Bayes Model:
Accuracy: 0.7790
Classification Report:
Average: Precision=0.51, Recall=0.01, F1-score=0.01, Support=10256
Bad: Precision=0.82, Recall=0.34, F1-score=0.48, Support=14985
Good: Precision=0.78, Recall=0.99, F1-score=0.87, Support=67801

Comparison:
Precision: Logical Regression > Random Forest > Decision Tree > K-Nearest Neighbors > Multinomial Naive Bayes
Recall: Logical Regression > Random Forest > Decision Tree > K-Nearest Neighbors > Multinomial Naive Bayes
F1-score: Logical Regression > Random Forest > Decision Tree > K-Nearest Neighbors > Multinomial Naive Bayes
Based on the comparison, Logical Regression has the best performance overall followed closely by Random Forest. However, depending on specific requirements and trade-offs, other models may also be suitable.

Answer 11 th

For this I have reduced the datasize on the basis that if a user has rated less than 6 items or a item has less than 6 reviews

```python
df = df.groupby('asin').filter(lambda x: len(x) >= 6)

df = df.reset_index()
print(df)
```

[157] ✓ 0.1s

```
       level_0   index  overall reviewTime      reviewerID        asin  \
0            6     149      4.0 2018-01-22   ALZJMBRRKUEON  B00000JBHP
1            7     151      5.0 2017-12-18  A2RQ0AT4XZUIXL  B00000JBHP
2            8     155      4.0 2017-11-29  A35W3JQYP0M655  B00000JBHP
3            9     156      5.0 2017-11-25  A306NASGVUDFKF  B00000JBHP
4           10     185      5.0 2015-03-16  A3KM3OWNFODAL0  B00000JBHP
...        ...     ...      ...        ...             ...         ...
19020    27054  371831      5.0 2017-06-05  A1BKJNAWJT2TG2  B01GHOMA6E
19021    27055  371833      5.0 2017-05-24  A16QODENBJVUI1  B01GHOMA6E
19022    27062  371983      3.0 2018-01-08   ACIDLSVWLDNBF  B01H2NDPGI
19023    27063  371988      5.0 2018-08-29  A3PTRCMBQ8ZRDX  B01H2VDRX6
19024    27066  372018      5.0 2018-09-26   A8AJDWU9K26E5  B01H4CFXZ8

                                              reviewText  year
0      comfortable light come mm plug adapter fit mp ...  2018
1      first issue warning large knob end cord come r...  2017
2      always enjoy old style ear cover headphone lik...  2017
3      headphone scatter throughout house work wirele...  2017
4      light weight headphonesexcellent sound quality...  2015
...                                                  ...   ...
19020  use six year old granddaughter tablet work wel...  2017
19021  audiophile quality headphone absolutely perfec...  2017
19022  great build quality solid comfortable wear nic...  2018
19023  like sound profile nice bass high volume great...  2018
19024  good purchase experience design headphone suit...  2018

[19025 rows x 8 columns]
```

This is my user_item matrix

```python
user_item_matrix = pd.pivot_table(df, values='overall', index='reviewerID', columns='asin', fill_value=0)

# Step 2: Print the user-item rating matrix
print(user_item_matrix)
```

[158] ✓ 0.1s

```
asin            B00000JBHP  B00001P4XA  B00001P4XH  B00001P4ZH  B00001P505  \
reviewerID
A1004703RC79J9         0.0         0.0         0.0         0.0         0.0
A100UD67AHFODS         0.0         0.0         0.0         0.0         0.0
A100WO06OQR8BQ         0.0         0.0         0.0         0.0         0.0
A1053FVPAZUKMF         0.0         0.0         0.0         0.0         0.0
A10AFVU66A79Y1         0.0         0.0         0.0         0.0         0.0
...                    ...         ...         ...         ...         ...
AZSHQNI2TOQG4          0.0         0.0         0.0         0.0         0.0
AZW10G02DNJI4          0.0         0.0         0.0         0.0         0.0
AZXFS8GCTSQ5R          0.0         0.0         0.0         0.0         0.0
AZXV98EONIU4S          0.0         0.0         0.0         0.0         0.0
AZZYW4YOE1B6E          0.0         0.0         0.0         0.0         0.0

asin            B00001W0DI  B00001WRSJ  B00004SY4H  B00004T8R2  B00004Z0BN  \
reviewerID
A1004703RC79J9         0.0         0.0         0.0         0.0         0.0
A100UD67AHFODS         0.0         0.0         0.0         0.0         0.0
A100WO06OQR8BQ         0.0         0.0         0.0         0.0         0.0
A1053FVPAZUKMF         0.0         0.0         0.0         0.0         0.0
A10AFVU66A79Y1         0.0         0.0         0.0         0.0         0.0
...                    ...         ...         ...         ...         ...
AZSHQNI2TOQG4          0.0         0.0         0.0         0.0         0.0
AZW10G02DNJI4          0.0         0.0         0.0         0.0         0.0
AZXFS8GCTSQ5R          0.0         0.0         0.0         0.0         0.0
...
AZXV98EONIU4S          0.0
AZZYW4YOE1B6E          0.0

[3099 rows x 1277 columns]
```

Steps for min max scalar

```
from sklearn.preprocessing import MinMaxScaler

# Assuming pivot_table is the DataFrame containing the pivot table with user's reviews

# Create a MinMaxScaler object
scaler = MinMaxScaler()

# Apply min-max scaling to normalize the ratings
pivot_table_normalized = user_item_matrix.copy()  # Create a copy to preserve the original pivot table
pivot_table_normalized[:] = scaler.fit_transform(pivot_table_normalized.values)

# Display the normalized pivot table
print(pivot_table_normalized)
```

[69] ✓ 0.0s

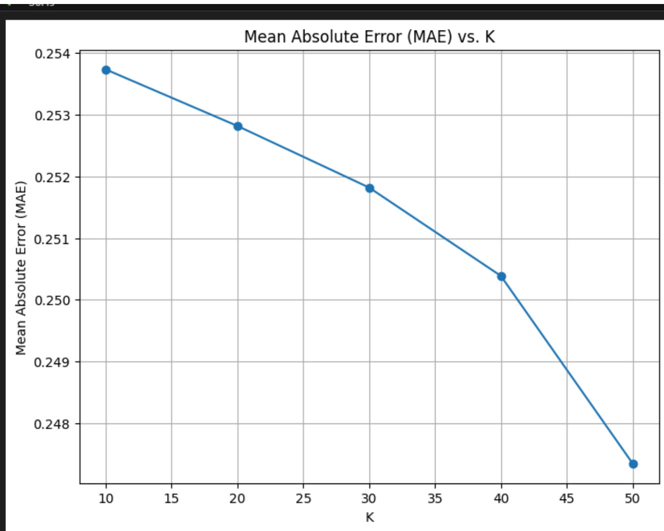
This is my cosine similarity matrix

```
import numpy as np
# Convert the pivot table to a numpy array
matrix = user_item_matrix.values

# Calculate the dot product of the matrix with its transpose
dot_product = np.dot(matrix, matrix.T)

# Calculate the norms of the rows of the matrix
norms = np.linalg.norm(matrix, axis=1)

# Calculate the cosine similarity matrix
cosine_similarity_matrix = dot_product / np.outer(norms, norms)

print(cosine_similarity_matrix)
print(cosine_similarity_matrix.shape)
```

[17] ✓ 0.2s

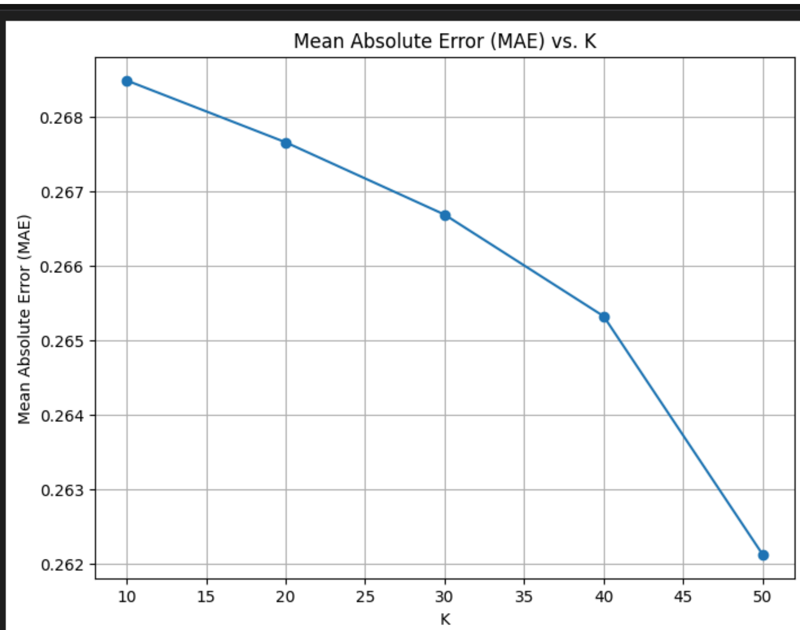
```
[[1. 0. 0. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 1. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]
 [0. 0. 0. ... 0. 0. 1.]]
(3099, 3099)
```

Final Plot of user-user recommender system




```
[0.2537312099520779, 0.25281563043219774, 0.251817549103528, 0.25038705558067814, 0.24735077795555752]
```

Final plot of item-item recommender system




```
[0.26848293196492434, 0.267653168389171, 0.2666826110582889, 0.2653196694422307, 0.26212407625373785]
```

Answer 12th

```
Top 10 Users with Maximum Sum of Ratings for Products:
asin            B00000JBHP  B00001P4XA  B00001P4XH  B00001P4ZH  B00001P505  \
reviewerID
A2DKQQIZ793AV5         0.0         0.0         0.0         0.0         0.0
AIFLY2HF8NS8U          0.0         0.0         0.0         0.0         0.0
A6FIAB28IS79           0.0         5.0         0.0         3.0         0.0
A2XX2A4OJCDNLZ         0.0         0.0         0.0         0.0         0.0
A23GFTVIETX7DS         0.0         0.0         0.0         0.0         0.0
A2XXBZPQT5EXHV         0.0         0.0         0.0         0.0         0.0
A979ON3H1O593          0.0         0.0         1.0         0.0         0.0
A1UQBFCERIP7VJ         0.0         0.0         0.0         0.0         0.0
A3077MQTAKOVFZ         0.0         0.0         0.0         5.0         0.0
A1JUKS0DSO2XZG         0.0         0.0         0.0         0.0         0.0

asin            B00001W0DI  B00001WRSJ  B00004SY4H  B00004T8R2  B00004Z0BN  \
reviewerID
A2DKQQIZ793AV5         0.0         0.0         0.0         0.0         0.0
AIFLY2HF8NS8U          0.0         0.0         0.0         0.0         0.0
A6FIAB28IS79           5.0         5.0         0.0         0.0         0.0
A2XX2A4OJCDNLZ         0.0         0.0         0.0         0.0         0.0
A23GFTVIETX7DS         0.0         0.0         0.0         0.0         0.0
A2XXBZPQT5EXHV         0.0         0.0         0.0         0.0         0.0
A979ON3H1O593          0.0         5.0         0.0         0.0         0.0
A1UQBFCERIP7VJ         0.0         0.0         0.0         0.0         0.0
A3077MQTAKOVFZ         0.0         0.0         5.0         0.0         0.0
...
A3077MQTAKOVFZ         143.0
A1JUKS0DSO2XZG         141.0

[10 rows x 1278 columns]
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...