Untitled12

April 12, 2024

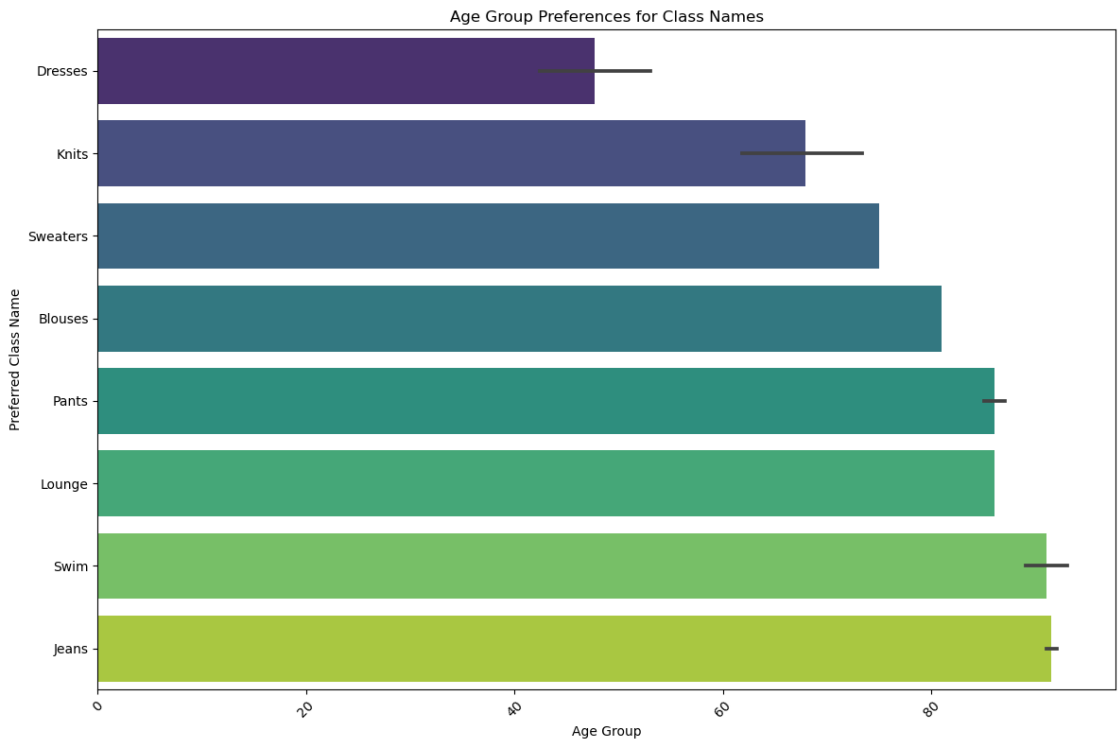
[1]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
csv_url = '/Users/riteshchandra/Desktop/Womens Clothing E-Commerce Reviews.csv'
df = pd.read_csv(csv_url)
df
```

[1]:
```
       Unnamed: 0  Clothing ID  Age  \
0               0          767   33
1               1         1080   34
2               2         1077   60
3               3         1049   50
4               4          847   47
…             …            …  …
23481       23481         1104   34
23482       23482          862   48
23483       23483         1104   31
23484       23484         1084   28
23485       23485         1104   52

                                                   Title  \
0                                                    NaN
1                                                    NaN
2                                 Some major design flaws
3                                        My favorite buy!
4                                        Flattering shirt
…                                                      …
23481                      Great dress for many occasions
23482                          Wish it was made of cotton
23483                               Cute, but see through
23484  Very cute dress, perfect for summer parties an…
23485                     Please make more like this one!

                                          Review Text  Rating  \
0      Absolutely wonderful - silky and sexy and comf…       4
1      Love this dress!  it's sooo pretty.  i happene…       5
2      I had such high hopes for this dress and reall…       3
3      I love, love, love this jumpsuit. it's fun, fl…       5
```

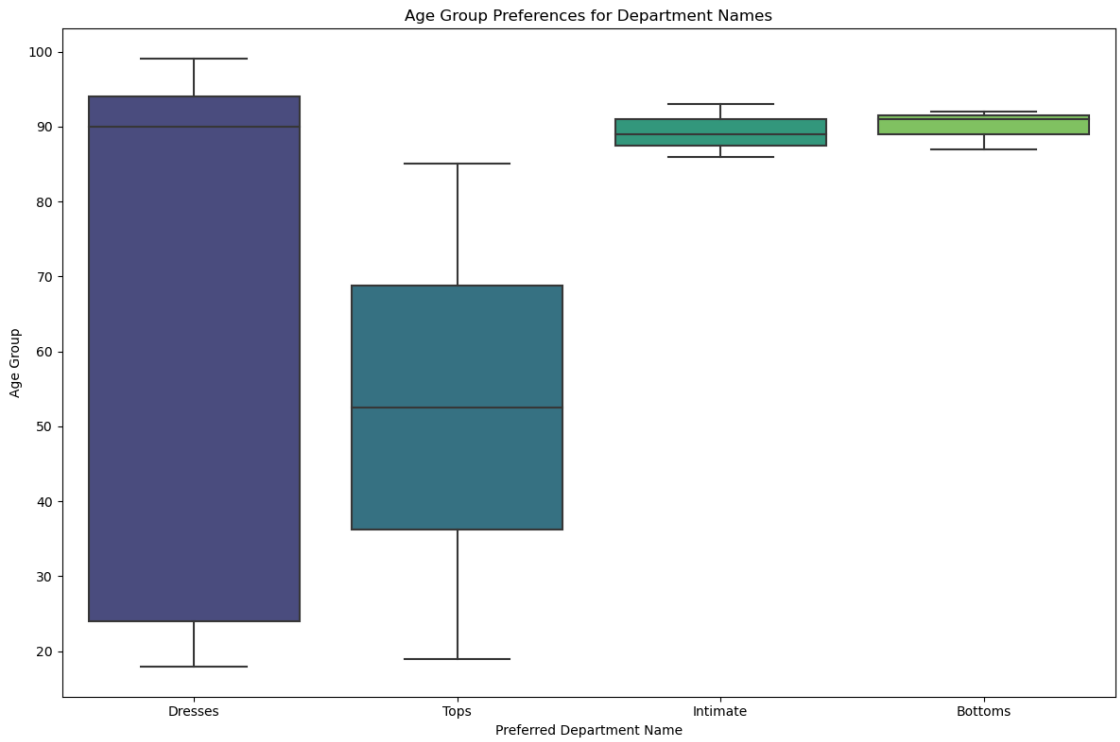
```
4      This shirt is very flattering to all due to th…       5
…                                                   …     …
23481  I was very happy to snag this dress at such a …       5
23482  It reminds me of maternity clothes. soft, stre…       3
23483  This fit well, but the top was very see throug…       3
23484  I bought this dress for a wedding i have this …       3
23485  This dress in a lovely platinum is feminine an…       5

       Recommended IND  Positive Feedback Count   Division Name  \
0                    1                        0       Initmates
1                    1                        4         General
2                    0                        0         General
3                    1                        0  General Petite
4                    1                        6         General
…                  …                        …              …
23481                1                        0  General Petite
23482                1                        0  General Petite
23483                0                        1  General Petite
23484                1                        2         General
23485                1                       22  General Petite

      Department Name Class Name
0            Intimate  Intimates
1             Dresses    Dresses
2             Dresses    Dresses
3             Bottoms      Pants
4                Tops    Blouses
…                  …        …
23481         Dresses    Dresses
23482            Tops      Knits
23483         Dresses    Dresses
23484         Dresses    Dresses
23485         Dresses    Dresses

[23486 rows x 11 columns]
```

[2]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Group by age groups and calculate preferences for class names
age_class_preferences = df.groupby('Age')['Class Name'].agg(lambda x: x.
 ↪value_counts().index[0]).reset_index()

# Plotting preferences for class names by age group using a bar plot
plt.figure(figsize=(12, 8))
```

```
sns.barplot(data=age_class_preferences, x='Age', y='Class Name',␣
↪palette='viridis')
plt.title('Age Group Preferences for Class Names')
plt.xlabel('Age Group')
plt.ylabel('Preferred Class Name')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[3]:
```
# Group by age groups and calculate preferences for class names
age_class_preferences = df.groupby('Age')['Department Name'].agg(lambda x: x.
↪value_counts().index[0]).reset_index()

# Plotting preferences for class names by age group using a box plot
plt.figure(figsize=(12, 8))
sns.boxplot(data=age_class_preferences, x='Department Name', y='Age',␣
↪palette='viridis')
plt.title('Age Group Preferences for Department Names')
plt.xlabel('Preferred Department Name')
plt.ylabel('Age Group')
plt.tight_layout()
plt.show()
```



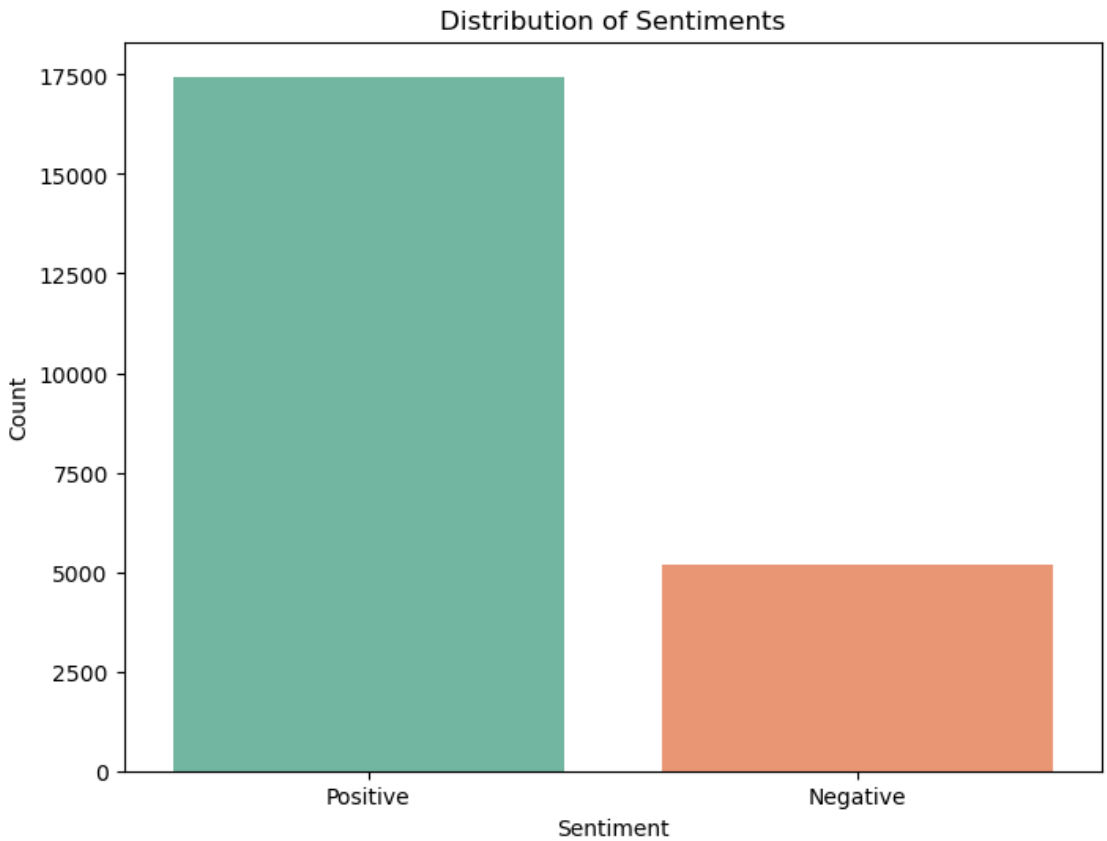
[4]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Drop rows with missing 'Review Text'
df.dropna(subset=['Review Text'], inplace=True)

# Convert sentiment into binary classification (recommended or not)
df['Recommended'] = df['Recommended IND']

# Select features (independent variables) and target variable
X = df['Review Text']
y = df['Recommended']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Convert text data into TF-IDF vectors
tfidf_vectorizer = TfidfVectorizer(max_features=1000, stop_words='english')
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
```

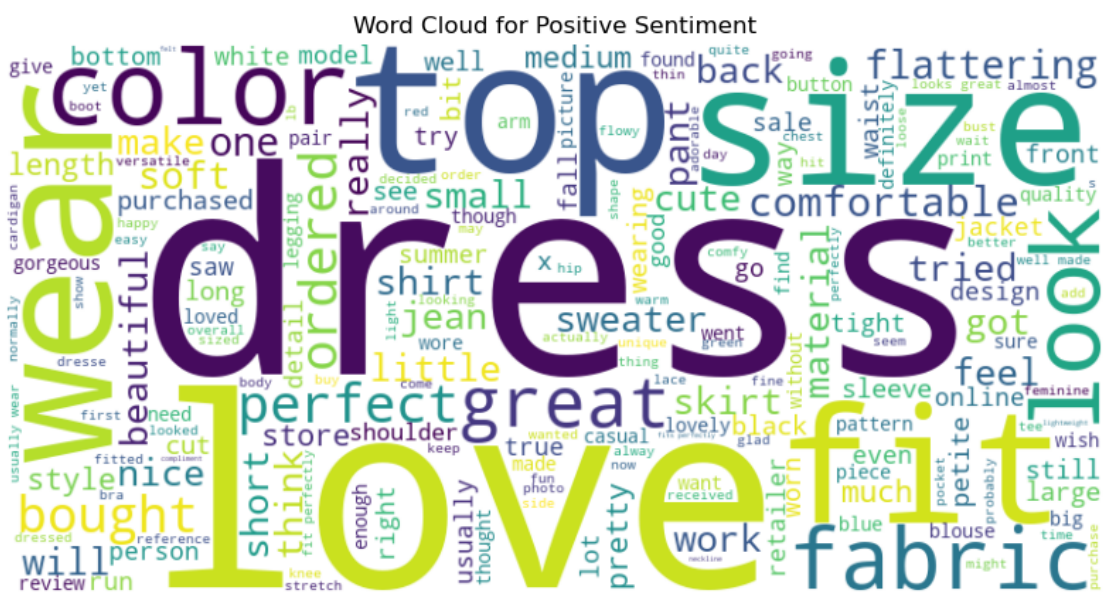
```python
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# Initialize and fit logistic regression model
model = LogisticRegression()
model.fit(X_train_tfidf, y_train)

# Predict recommendation on the test set
y_pred = model.predict(X_test_tfidf)

# Calculate accuracy score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Extracting feature names (words) from TF-IDF vectorizer
feature_names = tfidf_vectorizer.get_feature_names_out()

# Getting coefficients associated with each feature
coefficients = model.coef_[0]

# Combine feature names and coefficients into a DataFrame
feature_coefficients_df = pd.DataFrame({'Feature': feature_names, 'Coefficient':
 ↪ coefficients})

# Sorting the DataFrame by coefficient values to identify important features
important_features = feature_coefficients_df.sort_values(by='Coefficient',␣
 ↪ascending=False).head(10)
print("Top 10 important features:")
print(important_features)
```

```
Accuracy: 0.8860675645837933
Top 10 important features:
         Feature  Coefficient
599      perfect     4.555310
495         love     4.443361
177  comfortable     4.060183
184  compliments     4.054834
479       little     3.908177
359        great     3.715823
313         fits     3.594701
348         glad     3.167830
774         soft     2.995775
601    perfectly     2.946056
```

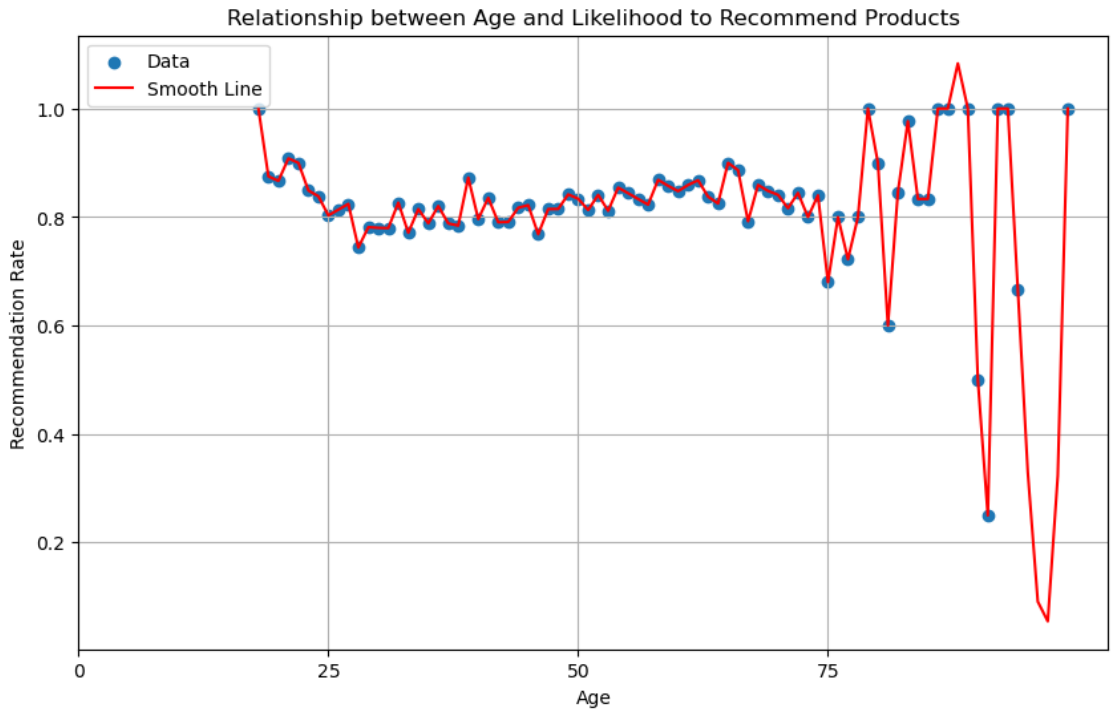
[5]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
```

```python
# Drop rows with missing 'Review Text'
df.dropna(subset=['Review Text'], inplace=True)

# Define function to extract sentiment from ratings
def get_sentiment(rating):
    if rating >= 4:
        return 'Positive'
    else:
        return 'Negative'

# Apply sentiment function to create a new column
df['Sentiment'] = df['Rating'].apply(get_sentiment)

# Plot distribution of sentiments
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Sentiment', palette='Set2')
plt.title('Distribution of Sentiments')
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.show()

# Visualize the most common words associated with positive sentiment
positive_reviews = ' '.join(df[df['Sentiment'] == 'Positive']['Review Text'].
 ↪tolist())
wordcloud = WordCloud(width=800, height=400, background_color='white').
 ↪generate(positive_reviews)

plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud for Positive Sentiment')
plt.axis('off')
plt.show()
```


Distribution of Sentiments
17500
15000
12500
10000
7500
5000
2500
0
Count
Positive
Negative
Sentiment


Word Cloud for Positive Sentiment
dress
love
fit
top
size
wear
color
look
fabric
great
perfect
comfortable
flattering
bought
nice
beautiful
tried
feel
sweater
jean
shirt
small
cute
medium
back
well
model
white
bottom
give
length
make
one
soft
purchased
ordered
really
try
see
summer
little
skirt
material
sleeve
online
pattern
even
much
still
large
wish
petite
blouse
work
pretty
lot
usually
right
enough
short
think
store
shoulder
true
casual
black
retailer
worn
tight
got
design
jacket
print
front
waist
sale
pant
fall
picture
arm
button
gorgeous
saw
long
loved
cut
style
will
person
review
run
need
first
beautiful
detail
legging
though
go
find
wearing
good
way
definitely
looks great
almost
quality
sure
feminine
lightweight
tee
time
might
purchase
probably
pocket
blue
big
piece
want
received
made
fun
photo
side
lace
fine
without
perfectly
comfy
better
well made
add
shape
warm
actually
unique
thing
go
hip
x
wore
looking
light
overall
sized
fitted
compliment
bra
reference
dressed
knee
stretch
neckline
fit perfectly
glad
now
always
wanted
keep
come
body
buy
dresse
cardigan
normally
show
usually wear
happy
easy
say
versatile
pair
red
thin
found
quite
going
chest
hit
day
adorable
flowy
bust
wait
loose
decided
order
around
may
boot
yet
lb
fall
bit
thought
thin
tall

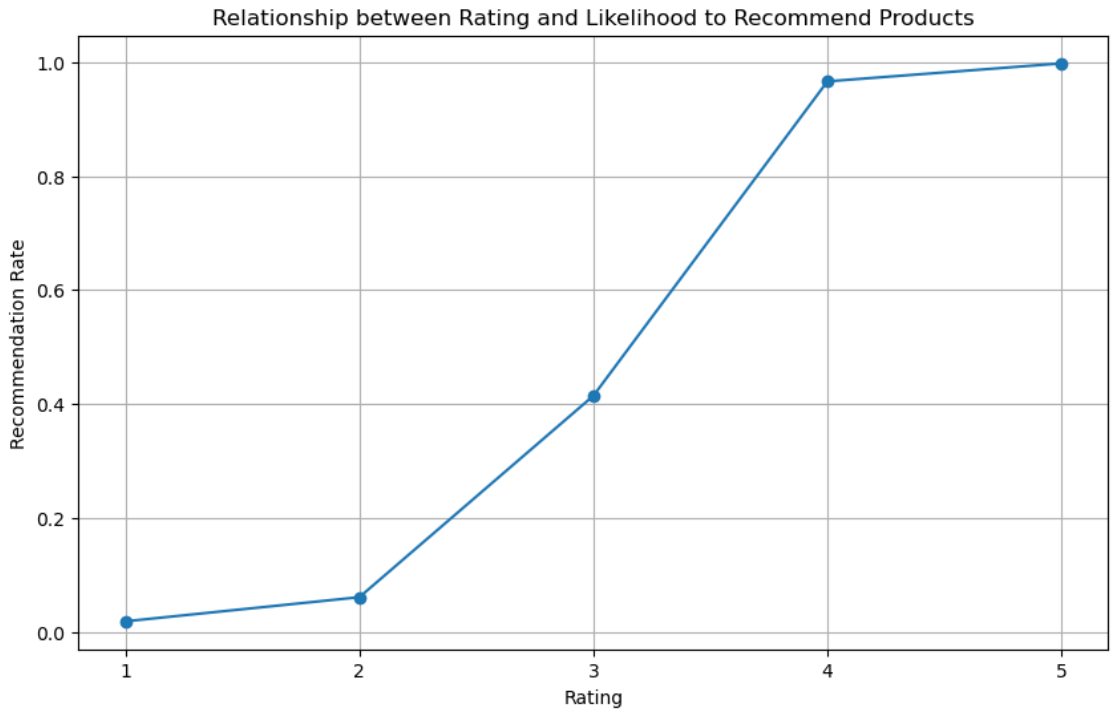
[6]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import make_interp_spline

# Assuming df is your DataFrame containing the dataset

# Group the data by age and calculate the average recommendation rate
age_recommendation = df.groupby('Age')['Recommended IND'].mean().reset_index()

# Interpolate the data for smoother curve
age_smooth = make_interp_spline(age_recommendation['Age'],␣
 ↪age_recommendation['Recommended IND'])

# Plotting the scatter plot and the smooth line
plt.figure(figsize=(10, 6))

# Scatter plot
plt.scatter(age_recommendation['Age'], age_recommendation['Recommended IND'],␣
 ↪label='Data')

# Smooth line
age_range = range(age_recommendation['Age'].min(), age_recommendation['Age'].
 ↪max() + 1)
plt.plot(age_range, age_smooth(age_range), linestyle='-', color='red',␣
 ↪label='Smooth Line')

plt.title('Relationship between Age and Likelihood to Recommend Products')
plt.xlabel('Age')
plt.ylabel('Recommendation Rate')
plt.xticks(range(0, age_recommendation['Age'].max() + 1, 25))  # Setting x-axis␣
 ↪ticks
plt.legend()
plt.grid(True)  # Adding grid lines
plt.show()
```



[7]:
```
import pandas as pd
import plotly.graph_objs as go

# Assuming df is your DataFrame containing the dataset

# Group the data by age and calculate the average recommendation rate
age_recommendation = df.groupby('Age')['Recommended IND'].mean().reset_index()

# Create an interactive line plot using Plotly
fig = go.Figure()

# Add the line trace
fig.add_trace(go.Scatter(x=age_recommendation['Age'],␣
 ↪y=age_recommendation['Recommended IND'], mode='lines', name='Recommendation␣
 ↪Rate'))

# Update layout
fig.update_layout(title='Relationship between Age and Likelihood to Recommend␣
 ↪Products',
                  xaxis_title='Age',
                  yaxis_title='Recommendation Rate',
                  xaxis=dict(type='category'),  # Ensure x-axis treats ages as␣
 ↪categories
```

```
                  yaxis=dict(tickformat='.2f'),  # Format y-axis ticks to two␣
 ↪decimal places
                  template='plotly_white'  # Set plot style
                  )

# Show the interactive plot
fig.show()
```

[8]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming df is your DataFrame containing the dataset

# Group the data by rating and calculate the average recommendation rate
rating_recommendation = df.groupby('Rating')['Recommended IND'].mean().
 ↪reset_index()

# Plotting the relationship between rating and recommendation rate
plt.figure(figsize=(10, 6))
plt.plot(rating_recommendation['Rating'], rating_recommendation['Recommended␣
 ↪IND'], marker='o', linestyle='-')
plt.title('Relationship between Rating and Likelihood to Recommend Products')
plt.xlabel('Rating')
plt.ylabel('Recommendation Rate')
plt.xticks(range(1, 6))  # Setting x-axis ticks to range from 1 to 5
plt.yticks([0, 0.2, 0.4, 0.6, 0.8, 1.0])  # Setting y-axis ticks
plt.grid(True)  # Adding grid lines
plt.show()
```



[9]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Assuming df is your DataFrame containing the dataset
# Selecting relevant features for the logistic regression model
X = df[['Rating', 'Age']]
y = df['Recommended IND']

# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)

# Initialize and fit logistic regression model
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)

# Making predictions on the testing set
y_pred = logistic_model.predict(X_test)

# Evaluating the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```python
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.9346434091410908
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.93      0.84       812
           1       0.98      0.94      0.96      3717

    accuracy                           0.93      4529
   macro avg       0.87      0.93      0.90      4529
weighted avg       0.94      0.93      0.94      4529
```

[ ]:

```python

```