Bahria University,

Karachi Campus

COURSE: DATA MINING
TERM: FALL 2023, CLASS: BSE- 6 (B)

PROJECT NAME:

AMAZON PRODUCT ANALYSIS

BY:

BARIRAH BAKHTIAR (02-131212-083)
MAHGUL WAHID (02-131212-055)

Dr.Hina Shakir
/ Engr. Hamza

Signed Remarks: Score:

HEADINGS:

1. TABLE OF CONTENT
2. INTRODUCTION & PROBLEM
3. TECHNOLOGY
4. FUNCTIONALITIES
5. MODULE DISTRIBUTION
6. CODE
7. INTERFACES
8. CONCLUSION

INTRODUCTION

The rapid growth of e-commerce has led to an explosion of online product reviews, which serve as a crucial resource for both consumers and businesses. Consumers rely on these reviews to make informed purchasing decisions, while businesses use them to gauge customer satisfaction and improve their products and services. The dataset at hand is a collection of Amazon product reviews, focusing on Amazon devices. It includes detailed information about each review, such as the review text, ratings, and various product metadata.

This project aims to leverage this dataset to gain insights into customer sentiments, analyze product ratings, and provide product recommendations. By utilizing natural language processing and machine learning techniques, we can transform raw review data into actionable insights that benefit both consumers and businesses.

PROBLEM STATEMENT

The primary goal of this project is to perform a comprehensive analysis of Amazon product reviews to understand customer sentiments, analyze product ratings, and implement a product recommendation system. Specifically, the project aims to address the following problems:

1. Sentiment Analysis:

Objective: To analyze the sentiment of the reviews and categorize them as positive, negative, or neutral.

Benefit: Helps businesses understand overall customer satisfaction and identify areas for improvement.

2. Word Cloud Visualization:

Objective: To generate word clouds for positive and negative reviews to identify common words and themes.

Benefit: Provides a visual representation of frequent terms in reviews, highlighting key aspects that customers like or dislike.

3. Rating Analysis:

Objective: To visualize the distribution of ratings and calculate the average rating for each product category.

Benefit: Offers insights into the general feedback on products and helps identify high and low-performing categories.

4. Product Recommendation System:

Objective: To implement a recommendation system that suggests similar products based on the review text.

Benefit: Enhances the customer shopping experience by providing personalized product recommendations.

TECHNOLOGY:

To achieve the objectives outlined in this project, a variety of technologies and tools were employed. These include libraries for data processing, visualization, natural language processing, and machine learning. The key technologies used in this project are:

1. Python

- **Description**: Python is a versatile programming language that is widely used for data analysis, machine learning, and web development.
- **Usage**: The core programming language used for data processing, analysis, and building the recommendation system.

2. Pandas

- **Description**: A powerful data manipulation and analysis library for Python.
- **Usage**: Used for loading, cleaning, and manipulating the dataset.

3. NumPy

- **Description**: A library for numerical operations in Python.
- **Usage**: Used for numerical computations and handling arrays.

4. Matplotlib

- **Description**: A plotting library for Python that provides a flexible way to create static, animated, and interactive visualizations.
- **Usage**: Used for plotting the distribution of sentiments and ratings.

5. WordCloud

- **Description**: A library for generating word cloud images from text data.
- **Usage**: Used to create visual representations of the most frequent words in positive and negative reviews.

6. TextBlob

- **Description**: A simple library for processing textual data, providing a consistent API for diving into common natural language processing (NLP) tasks.
- **Usage**: Used for performing sentiment analysis on the review texts.

7. Scikit-learn

- **Description**: A machine learning library for Python that provides simple and efficient tools for data mining and data analysis.
- **Usage**: Used for transforming text data into vectors using TF-IDF Vectorizer and calculating cosine similarity for the recommendation system.

8. Streamlit

- **Description**: An open-source app framework for creating and sharing data science and machine learning web apps.
- **Usage**: Used to build an interactive web application for data visualization and interaction.

9. Seaborn

- **Description**: A Python visualization library based on Matplotlib that provides a high-level interface for drawing attractive statistical graphics.
- **Usage**: (Optional) Can be used for creating more complex and visually appealing plots, though not essential in this particular project.

10. Natural Language Toolkit (nltk)

- **Description**: A platform for building Python programs to work with human language data.
- **Usage**: Used to download and use stopwords for text preprocessing.

11. TensorFlow and Keras

- **Description**: Open-source libraries for machine learning and deep learning.
- **Usage**: (Mentioned in the notebook, but not necessarily used in the final implementation) Could be used for advanced sentiment analysis or other machine learning tasks if required.

FUNCTIONALITIES:

1. Data Loading and Cleaning

- **File Upload**:
  - Users can upload a CSV file containing the product reviews.
- **Data Inspection**:
  - Display the first few rows of the dataset to understand its structure.
  - Identify and display missing values in each column.
- **Data Cleaning**:
  - Drop rows with missing values.
  - Convert price columns from string to numerical values (if applicable).

2. Sentiment Analysis

- **Text Preprocessing**:
  - Convert review text to lowercase.
  - Remove punctuation and stopwords.
- **Sentiment Classification**:
  - Use TextBlob to calculate the polarity of each review.
  - Classify reviews into positive, negative, or neutral based on polarity scores.
- **Visualization**:
  - Plot the distribution of sentiments across the reviews.

3. Word Cloud Generation

- **Positive Reviews**:
  - Generate and display a word cloud for positive reviews to highlight common themes and keywords.

- **Negative Reviews**:
  - Generate and display a word cloud for negative reviews to identify frequent complaints and issues.

4. Rating Analysis

- **Distribution of Ratings**:
  - Plot the distribution of ratings to understand general customer feedback.
- **Average Rating by Category**:
  - Calculate and display the average rating for each product category to identify high and low-performing categories.

5. Product Recommendation System

- **TF-IDF Vectorization**:
  - Transform review texts into TF-IDF vectors to quantify the importance of words in the reviews.
- **Cosine Similarity**:
  - Calculate cosine similarity between products based on their review vectors.
- **Recommendation Generation**:
  - Allow users to input a product ID (ASIN).
  - Generate and display a list of similar products based on the cosine similarity scores.

6. Interactive Web Application

- **User Interface**:
  - Built using Streamlit to provide an interactive and user-friendly experience.
- **Dynamic Data Handling**:
  - Real-time data loading, processing, and visualization based on user inputs and uploaded files.

MODULE DISTRIBUTION

DATA LOADING AND DATA CLEANING, PRODUCT ANALYSIS, IMAGE PROCESSING VISUALIZATION AND MODEL IMPLEMENTATION	**BARIRAH BAKHTIAR (02-131212-083)**
DATA LOADING, CLEANING, VISUALIZATION, RECOMMENTATION SYSTEM, SENTIMENT ANALYSIS	**MAHGUL WAHID (02-131212-055)**

CODE:

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
from io import BytesIO
import requests
import numpy as np
import cv2
from sklearn.cluster import KMeans
from wordcloud import WordCloud
from collections import Counter
import re
import nltk
from nltk.corpus import stopwords

import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input, decode_predictions
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

# Initialize the MobileNetV2 model
model = MobileNetV2(weights='imagenet')

nltk.download('stopwords')

# Title of the app
st.title('Data Processing and Visualization')

# Load the dataset
uploaded_file = st.file_uploader("Choose a CSV file", type="csv")
if uploaded_file is not None:
    try:
        data = pd.read_csv(uploaded_file)

        st.write("First few rows of the dataset:")
        st.write(data.head())

        # Display missing values
        st.write("Missing values in each column:")
        missing_values = data.isnull().sum()
        st.write(missing_values)

        # Drop rows with missing values
        data = data.dropna()
        st.write("Data after dropping missing values:")
        st.write(data.head())

        # Convert price columns to numerical data
        def convert_price(price):
            return float(price.replace('₹', '').replace(',', '').strip())

        data['discount_price'] = data['discount_price'].apply(convert_price)
        data['actual_price'] = data['actual_price'].apply(convert_price)
```

```
st.write("Data after converting price columns:")
st.write(data.head())

# Check unique values in the ratings column to identify non-numeric entries
unique_ratings = data['ratings'].unique()
st.write("Unique values in the ratings column:")
st.write(unique_ratings)

# Remove rows with invalid ratings
valid_ratings_mask = data['ratings'].apply(lambda x: x.replace('.', '', 1).isdigit())
data = data[valid_ratings_mask]
st.write("Data after removing invalid ratings:")
st.write(data.head())

# Convert the ratings column to a numerical type
data['ratings'] = data['ratings'].astype(float)
st.write("Data after converting ratings to float:")
st.write(data.head())

# Convert the no_of_ratings column to an integer type
data['no_of_ratings'] = data['no_of_ratings'].apply(lambda x: int(x.replace(',', '').strip()))
st.write("Data after converting no_of_ratings to int:")
st.write(data.head())

# Ensure the relevant columns are numeric
data['discount_price'] = data['discount_price'].replace('[\₹,]', '', regex=True).astype(float)
data['actual_price'] = data['actual_price'].replace('[\₹,]', '', regex=True).astype(float)
data['ratings'] = pd.to_numeric(data['ratings'], errors='coerce')

# Clean and convert no_of_ratings to numeric, handling any non-string values appropriately
data['no_of_ratings'] = data['no_of_ratings'].astype(str)
data['no_of_ratings'] = data['no_of_ratings'].apply(lambda x: x if x.isdigit() else np.nan)
data['no_of_ratings'] = pd.to_numeric(data['no_of_ratings'], errors='coerce').fillna(0).astype(int)

# Handle NaN values in price columns
data['discount_price'] = data['discount_price'].fillna(0)
data['actual_price'] = data['actual_price'].fillna(0)

# Check if sales column exists, if not create a dummy sales column (for example purposes)
if 'sales' not in data.columns:
    np.random.seed(42)
    data['sales'] = np.random.randint(100, 1000, size=len(data))

# Extract category information
data['category'] = data['main_category'] + ' - ' + data['sub_category']

# Drop rows with missing sales data
data = data.dropna(subset=['sales'])

# Check for any remaining NaN values in the features
st.write("Remaining NaN values in the features:")
st.write(data[['discount_price', 'actual_price', 'ratings', 'no_of_ratings']].isna().sum())

# Drop rows with NaN values in the features
```

```
            data = data.dropna(subset=['discount_price', 'actual_price', 'ratings', 'no_of_ratings'])

            # Select features and target variable
            features = data[['discount_price', 'actual_price', 'ratings', 'no_of_ratings']]
            target = data['sales']

            # Split the data into training and testing sets
            X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

            # Initialize the model
            model = RandomForestRegressor(n_estimators=100, random_state=42)

            # Train the model
            model.fit(X_train, y_train)

            # Make predictions on the test set
            predictions = model.predict(X_test)

            # Calculate the mean squared error
            mse = mean_squared_error(y_test, predictions)
            st.write(f'Mean Squared Error: {mse}')

            # Calculate the R^2 score for the model
            r2_score = model.score(X_test, y_test)
            st.write(f'R^2 Score: {r2_score}')

            # Create a DataFrame for all the data, not just the test set
            all_predictions = model.predict(features)
            results_all = pd.DataFrame({
                'Product': data['name'],
                'Actual': target,
                'Predicted': all_predictions,
                'Category': data['category']
            })

            # Plot actual vs predicted sales for each product
            st.subheader('Actual vs Predicted Sales for Each Product')
            fig, ax = plt.subplots(figsize=(14, 8))
            sns.scatterplot(x='Actual', y='Predicted', hue='Category', data=results_all, ax=ax)
            ax.plot([results_all['Actual'].min(), results_all['Actual'].max()], [results_all['Actual'].min(), results_all['Actual'].max()],
color='red', lw=2)
            ax.set_title('Actual vs Predicted Sales for Each Product')
            ax.set_xlabel('Actual Sales')
            ax.set_ylabel('Predicted Sales')
            ax.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
            st.pyplot(fig)

            # Visualize sales for each product in a bar chart
            st.subheader('Actual Sales for Top 20 Products')
            fig, ax = plt.subplots(figsize=(14, 8))
            results_all_sorted = results_all.sort_values(by='Actual', ascending=False)
            sns.barplot(x='Actual', y='Product', data=results_all_sorted.head(20), palette='viridis', ax=ax)
            ax.set_title('Actual Sales for Top 20 Products')
            ax.set_xlabel('Actual Sales')
ax.set_ylabel('Product')
st.pyplot(fig)

st.subheader('Predicted Sales for Top 20 Products')
fig, ax = plt.subplots(figsize=(14, 8))
sns.barplot(x='Predicted', y='Product', data=results_all_sorted.head(20), palette='viridis', ax=ax)
ax.set_title('Predicted Sales for Top 20 Products')
ax.set_xlabel('Predicted Sales')
ax.set_ylabel('Product')
st.pyplot(fig)

# Price Distribution Analysis
st.header('Price Distribution Analysis')

# Distribution of Product Ratings
st.subheader('Distribution of Product Ratings')
fig, ax = plt.subplots(figsize=(10, 6))
data['ratings'].hist(bins=20, edgecolor='black', ax=ax)
ax.set_title('Distribution of Product Ratings')
ax.set_xlabel('Ratings')
ax.set_ylabel('Frequency')
ax.grid(False)
st.pyplot(fig)

# Distribution of Discounted Product Prices
st.subheader('Distribution of Discounted Product Prices')
fig, ax = plt.subplots(figsize=(10, 6))
data['discount_price'].hist(bins=20, edgecolor='black', ax=ax)
ax.set_title('Distribution of Discounted Product Prices')
ax.set_xlabel('Discounted Price (₹)')
ax.set_ylabel('Frequency')
ax.grid(False)
st.pyplot(fig)

# Distribution of Actual Product Prices
st.subheader('Distribution of Actual Product Prices')
fig, ax = plt.subplots(figsize=(10, 6))
data['actual_price'].hist(bins=20, edgecolor='black', ax=ax)
ax.set_title('Distribution of Actual Product Prices')
ax.set_xlabel('Actual Price (₹)')
ax.set_ylabel('Frequency')
ax.grid(False)
st.pyplot(fig)

# Ratings and Reviews Analysis
st.header('Ratings and Reviews Analysis')

# Average Rating
average_rating = data['ratings'].mean()
st.write(f"Average Rating of Amazon Products: {average_rating:.2f}")

# Plot distribution of ratings
st.subheader('Distribution of Ratings for Amazon Products')
fig, ax = plt.subplots(figsize=(10, 6))
```

```
data['ratings'].hist(bins=20, edgecolor='black', ax=ax)
ax.set_title('Distribution of Ratings for Amazon Products')
ax.set_xlabel('Ratings')
ax.set_ylabel('Frequency')
ax.grid(False)
st.pyplot(fig)

# Correlation Between Rating and Price
st.subheader('Correlation Between Discount Price and Ratings')
fig, ax = plt.subplots(figsize=(10, 6))
sns.scatterplot(x='discount_price', y='ratings', data=data, alpha=0.5, ax=ax)
ax.set_title('Correlation Between Discount Price and Ratings')
ax.set_xlabel('Discount Price (₹)')
ax.set_ylabel('Ratings')
ax.grid(True)
st.pyplot(fig)

# Review Count Analysis
st.header('Review Count Analysis')

# Distribution of review counts
st.subheader('Distribution of Review Counts for Amazon Products')
fig, ax = plt.subplots(figsize=(10, 6))
data['no_of_ratings'].hist(bins=20, edgecolor='black', ax=ax)
ax.set_title('Distribution of Review Counts for Amazon Products')
ax.set_xlabel('Number of Reviews')
ax.set_ylabel('Frequency')
ax.grid(False)
st.pyplot(fig)

# Relationship between number of reviews and ratings
st.subheader('Relationship Between Number of Reviews and Ratings')
fig, ax = plt.subplots(figsize=(10, 6))
sns.scatterplot(x='no_of_ratings', y='ratings', data=data, alpha=0.5, ax=ax)
ax.set_title('Relationship Between Number of Reviews and Ratings')
ax.set_xlabel('Number of Reviews')
ax.set_ylabel('Ratings')
ax.grid(True)
st.pyplot(fig)

# Relationship between number of reviews and discount price
st.subheader('Relationship Between Number of Reviews and Discount Price')
fig, ax = plt.subplots(figsize=(10, 6))
sns.scatterplot(x='no_of_ratings', y='discount_price', data=data, alpha=0.5, ax=ax)
ax.set_title('Relationship Between Number of Reviews and Discount Price')
ax.set_xlabel('Number of Reviews')
ax.set_ylabel('Discount Price (₹)')
ax.grid(True)
st.pyplot(fig)

# Product Popularity and Visibility
st.header('Product Popularity and Visibility')

# Frequency of Product Listings by Main Category
```

```
listing_frequency_main_category = data['main_category'].value_counts()
st.subheader('Frequency of Product Listings by Main Category')
fig, ax = plt.subplots(figsize=(10, 6))
listing_frequency_main_category.plot(kind='bar', ax=ax)
ax.set_title('Frequency of Product Listings by Main Category')
ax.set_xlabel('Main Category')
ax.set_ylabel('Number of Listings')
ax.grid(False)
st.pyplot(fig)

# Frequency of Product Listings by Sub-Category
listing_frequency_sub_category = data['sub_category'].value_counts()
st.subheader('Frequency of Product Listings by Sub-Category')
fig, ax = plt.subplots(figsize=(10, 6))
listing_frequency_sub_category.plot(kind='bar', ax=ax)
ax.set_title('Frequency of Product Listings by Sub-Category')
ax.set_xlabel('Sub-Category')
ax.set_ylabel('Number of Listings')
ax.grid(False)
st.pyplot(fig)

# Display the listing frequencies
st.write("Listing Frequency by Main Category:")
st.write(listing_frequency_main_category)
st.write("\nListing Frequency by Sub-Category:")
st.write(listing_frequency_sub_category)

# Image Analysis Example
st.header('Image Analysis Example')
image_urls = data['image'].dropna().unique()
if len(image_urls) > 0:
    def download_and_process_image(url):
        try:
            response = requests.get(url)
            img = Image.open(BytesIO(response.content))

            # Convert image to a format suitable for analysis
            img = img.convert("RGB")
            img_np = np.array(img)

            # Example: Use OpenCV to convert image to grayscale
            img_gray = cv2.cvtColor(img_np, cv2.COLOR_RGB2GRAY)

            # Example: Use KMeans to cluster colors in the image
            img_reshape = img_np.reshape((-1, 3))
            kmeans = KMeans(n_clusters=5)
            kmeans.fit(img_reshape)
            dominant_colors = kmeans.cluster_centers_

            # Display the original and grayscale images
            fig, axs = plt.subplots(1, 2, figsize=(12, 6))

            axs[0].imshow(img)
            axs[0].set_title('Original Image')
            axs[1].imshow(img_gray, cmap='gray')
            axs[1].set_title('Grayscale Image')

            for ax in axs:
                ax.axis('off')

            st.pyplot(fig)

            st.write("Dominant colors in the image:")
            st.write(dominant_colors)

        except Exception as e:
            st.error(f"Error processing image from URL {url}: {e}")

    # Select a sample image URL for demonstration
    sample_image_url = st.selectbox('Select an image URL for analysis', image_urls)
    download_and_process_image(sample_image_url)
else:
    st.write("No image URLs found in the dataset.")

# Keyword Analysis
st.header('Keyword Analysis')
text_data = data['name'].astype(str) + ' ' + data['sub_category'].astype(str)

def clean_text(text):
    text = text.lower()
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

text_data = text_data.apply(clean_text)
stop_words = set(stopwords.words('english'))
text_data = text_data.apply(lambda x: ' '.join([word for word in x.split() if word not in stop_words]))

all_text = ' '.join(text_data)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_text)

st.subheader('Word Cloud of Product Titles and Descriptions')
fig, ax = plt.subplots(figsize=(10, 5))
ax.imshow(wordcloud, interpolation='bilinear')
ax.axis('off')
ax.set_title('Word Cloud of Product Titles and Descriptions')
st.pyplot(fig)

word_counts = Counter(all_text.split())
common_keywords = word_counts.most_common(20)

st.subheader('Most Common Keywords in Product Titles and Descriptions')
st.write(common_keywords)

# Competitive Analysis
st.header('Competitive Analysis')
```

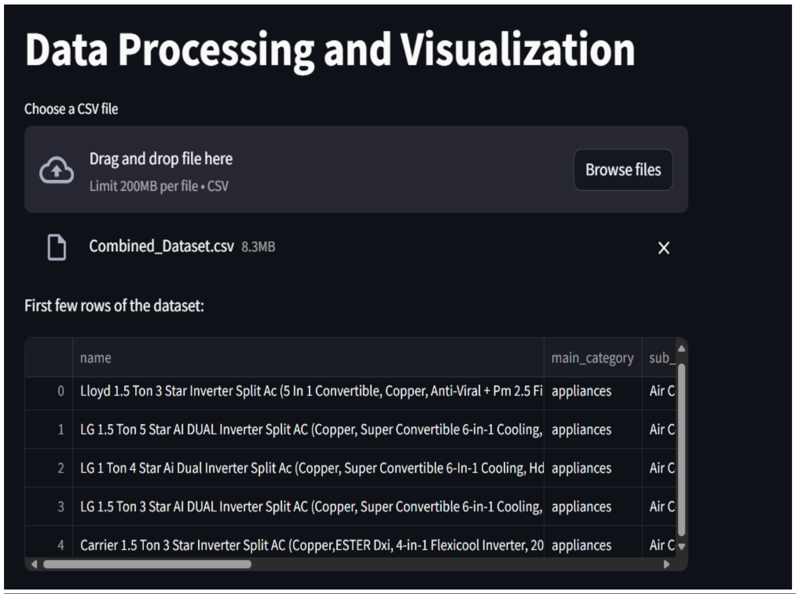
```
# Brand Performance Analysis
data['brand'] = data['name'].str.split().str[0]
data = data.dropna(subset=['ratings', 'no_of_ratings'])
brand_performance = data.groupby('brand').agg({
    'ratings': 'mean',
    'no_of_ratings': 'sum',
    'discount_price': 'mean',
    'actual_price': 'mean'
}).reset_index()

brand_performance['average_discount_percentage'] = ((brand_performance['actual_price'] -
brand_performance['discount_price']) / brand_performance['actual_price']) * 100
st.subheader('Brand Performance Analysis')
st.write(brand_performance.head())

# Product Comparison Analysis
threshold_rating = 4.0
threshold_reviews = 100
highly_rated_products = data[(data['ratings'] >= threshold_rating) & (data['no_of_ratings'] >= threshold_reviews)]
product_comparison = highly_rated_products.groupby(['sub_category', 'brand']).agg({
    'ratings': 'mean',
    'no_of_ratings': 'sum',
    'discount_price': 'mean',
    'actual_price': 'mean'
}).reset_index()

product_comparison['average_discount_percentage'] = ((product_comparison['actual_price'] -
product_comparison['discount_price']) / product_comparison['actual_price']) * 100
st.subheader('Product Comparison Analysis')
st.write(product_comparison.head())

# Visualize Brand Performance: Average Rating
top_brands = brand_performance.sort_values(by='ratings', ascending=False).head(20)
st.subheader('Top 20 Brands Performance: Average Rating')
fig, ax = plt.subplots(figsize=(12, 6))
ax.bar(top_brands['brand'], top_brands['ratings'], color='skyblue')
ax.set_xlabel('Brand')
ax.set_ylabel('Average Rating')
ax.set_title('Top 20 Brands Performance: Average Rating')
ax.set_xticklabels(top_brands['brand'], rotation=45, ha='right')
st.pyplot(fig)

# Visualize Brand Performance: Total Number of Ratings
top_brands_by_ratings = brand_performance.sort_values(by='no_of_ratings', ascending=False).head(20)
st.subheader('Top 20 Brands Performance: Total Number of Ratings')
fig, ax = plt.subplots(figsize=(12, 6))
ax.bar(top_brands_by_ratings['brand'], top_brands_by_ratings['no_of_ratings'], color='lightgreen')
ax.set_xlabel('Brand')
ax.set_ylabel('Total Number of Ratings')
ax.set_title('Top 20 Brands Performance: Total Number of Ratings')
ax.set_xticklabels(top_brands_by_ratings['brand'], rotation=45, ha='right')
st.pyplot(fig)
```

```
            # Visualize Brand Performance: Average Discount Percentage
            top_brands_by_discount = brand_performance.sort_values(by='average_discount_percentage',
ascending=False).head(20)
            st.subheader('Top 20 Brands Performance: Average Discount Percentage')
            fig, ax = plt.subplots(figsize=(12, 6))
            ax.bar(top_brands_by_discount['brand'], top_brands_by_discount['average_discount_percentage'], color='salmon')
            ax.set_xlabel('Brand')
            ax.set_ylabel('Average Discount Percentage')
            ax.set_title('Top 20 Brands Performance: Average Discount Percentage')
            ax.set_xticklabels(top_brands_by_discount['brand'], rotation=45, ha='right')
            st.pyplot(fig)

            # Visualize Product Comparison: Average Rating
            top_product_comparison = product_comparison.sort_values(by='ratings', ascending=False).head(20)
            st.subheader('Top 20 Product Comparison: Average Rating')
            fig, ax = plt.subplots(figsize=(12, 8))
            ax.barh(top_product_comparison['sub_category'] + ' - ' + top_product_comparison['brand'],
                top_product_comparison['ratings'], color='skyblue')
            ax.set_xlabel('Average Rating')
            ax.set_ylabel('Sub-Category and Brand')
            ax.set_title('Top 20 Product Comparison: Average Rating')
            ax.invert_yaxis()  # Invert y-axis to have the highest rating on top
            st.pyplot(fig)

            # Visualize Product Comparison: Total Number of Ratings
            top_product_comparison_ratings = product_comparison.sort_values(by='no_of_ratings', ascending=False).head(20)
            st.subheader('Top 20 Product Comparison: Total Number of Ratings')
            fig, ax = plt.subplots(figsize=(12, 8))
            ax.barh(top_product_comparison_ratings['sub_category'] + ' - ' + top_product_comparison_ratings['brand'],
                top_product_comparison_ratings['no_of_ratings'], color='lightgreen')
            ax.set_xlabel('Total Number of Ratings')
            ax.set_ylabel('Sub-Category and Brand')
            ax.set_title('Top 20 Product Comparison: Total Number of Ratings')
            ax.invert_yaxis()  # Invert y-axis to have the highest rating on top
            st.pyplot(fig)

            # Visualize Product Comparison: Average Discount Percentage
            top_product_comparison_discount = product_comparison.sort_values(by='average_discount_percentage',
ascending=False).head(20)
            st.subheader('Top 20 Product Comparison: Average Discount Percentage')
            fig, ax = plt.subplots(figsize=(12, 8))
            ax.barh(top_product_comparison_discount['sub_category'] + ' - ' + top_product_comparison_discount['brand'],
                top_product_comparison_discount['average_discount_percentage'], color='salmon')
            ax.set_xlabel('Average Discount Percentage')
            ax.set_ylabel('Sub-Category and Brand')
            ax.set_title('Top 20 Product Comparison: Average Discount Percentage')
            ax.invert_yaxis()  # Invert y-axis to have the highest discount on top
            st.pyplot(fig)



            # Competitive Analysis: K-means clustering
            def clean_and_convert(column):
```

```
        # Feature extraction and K-means clustering
            features = products[['ratings', 'no_of_ratings', 'discount_price', 'actual_price']]
            scaler = StandardScaler()
            scaled_features = scaler.fit_transform(features)

            kmeans = KMeans(n_clusters=5, random_state=42)
            products['cluster'] = kmeans.fit_predict(scaled_features)

            cluster_centers = scaler.inverse_transform(kmeans.cluster_centers_)
            cluster_centers_df = pd.DataFrame(cluster_centers, columns=features.columns)

            st.write("Cluster Centers:")
            st.dataframe(cluster_centers_df)

            st.write("Cluster Counts:")
            st.bar_chart(products['cluster'].value_counts().sort_index())

            # Visualize the clusters
            fig, ax = plt.subplots(figsize=(10, 6))
            sns.scatterplot(data=products, x='discount_price', y='actual_price', hue='cluster', palette='viridis', alpha=0.6, ax=ax)
            ax.set_title('Clusters of Products based on Price')
            ax.set_xlabel('Discount Price')
            ax.set_ylabel('Actual Price')
            ax.legend(title='Cluster')
            st.pyplot(fig)

            fig, ax = plt.subplots(figsize=(10, 6))
            sns.scatterplot(data=products, x='ratings', y='no_of_ratings', hue='cluster', palette='viridis', alpha=0.6, ax=ax)
            ax.set_title('Clusters of Products based on Ratings')
            ax.set_xlabel('Ratings')
            ax.set_ylabel('Number of Ratings')
            ax.legend(title='Cluster')
            st.pyplot(fig)


    except Exception as e:
        st.error(f"Error loading file: {e}")
else:
    st.warning("Please upload a CSV file.")
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from textblob import TextBlob

# Load the data
@st.cache_data
def load_data(file):
    data = pd.read_csv(file)
    return data
```

```
# Preprocess the data
def preprocess_data(data):
    data.dropna(subset=['reviews.text'], inplace=True)
    data['reviews.text'] = data['reviews.text'].apply(lambda x: ' '.join(x.lower() for x in x.split()))
    data['reviews.text'] = data['reviews.text'].str.replace('[^\w\s]', '')
    return data

# Sentiment analysis
def sentiment_analysis(data):
    data['polarity'] = data['reviews.text'].apply(lambda x: TextBlob(x).sentiment.polarity)
    data['sentiment'] = data['polarity'].apply(lambda x: 'positive' if x > 0 else ('negative' if x < 0 else 'neutral'))
    return data

# Recommendation system
def recommend_products(data, product_id, num_recommendations=5):
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(data['reviews.text'])
    cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)
    idx = data.index[data['asins'] == product_id][0]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:num_recommendations+1]
    product_indices = [i[0] for i in sim_scores]
    return data.iloc[product_indices]

# Visualizations
def plot_sentiment_distribution(data):
    sentiment_counts = data['sentiment'].value_counts()
    fig, ax = plt.subplots()
    sentiment_counts.plot(kind='bar', ax=ax)
    ax.set_title('Sentiment Distribution')
    ax.set_xlabel('Sentiment')
    ax.set_ylabel('Number of Reviews')
    st.pyplot(fig)

def plot_wordcloud(data, sentiment):
    reviews = ' '.join(data[data['sentiment'] == sentiment]['reviews.text'])
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(reviews)
    fig, ax = plt.subplots()
    ax.imshow(wordcloud, interpolation='bilinear')
    ax.axis('off')
    st.pyplot(fig)

def plot_rating_distribution(data):
    rating_counts = data['reviews.rating'].value_counts()
    fig, ax = plt.subplots()
    rating_counts.plot(kind='bar', ax=ax)
    ax.set_title('Rating Distribution')
    ax.set_xlabel('Rating')
    ax.set_ylabel('Number of Reviews')
    st.pyplot(fig)

def plot_average_rating_by_category(data):
    avg_rating = data.groupby('categories')['reviews.rating'].mean().sort_values()
    fig, ax = plt.subplots()
    avg_rating.plot(kind='barh', ax=ax)
    ax.set_title('Average Rating by Category')
    ax.set_xlabel('Average Rating')
    ax.set_ylabel('Category')
    st.pyplot(fig)

# Streamlit app
st.title('Product Reviews Sentiment Analysis and Recommendation System')

uploaded_file = st.file_uploader("Choose a CSV file", type="csv", key="file_uploader")
if uploaded_file is not None:
    data = load_data(uploaded_file)
    data = preprocess_data(data)
    data = sentiment_analysis(data)

    st.header('Sentiment Analysis')
    plot_sentiment_distribution(data)

    st.header('Word Clouds')
    st.subheader('Positive Reviews')
    plot_wordcloud(data, 'positive')
    st.subheader('Negative Reviews')
    plot_wordcloud(data, 'negative')

    st.header('Rating Analysis')
    plot_rating_distribution(data)

    st.header('Average Rating by Category')
    plot_average_rating_by_category(data)

    st.header('Product Recommendation System')
    product_id = st.text_input('Enter Product ID (ASIN) for Recommendations')
    if product_id:
        recommendations = recommend_products(data, product_id)
        st.write('Top Recommendations:')
        st.write(recommendations)
```

INTERFACES

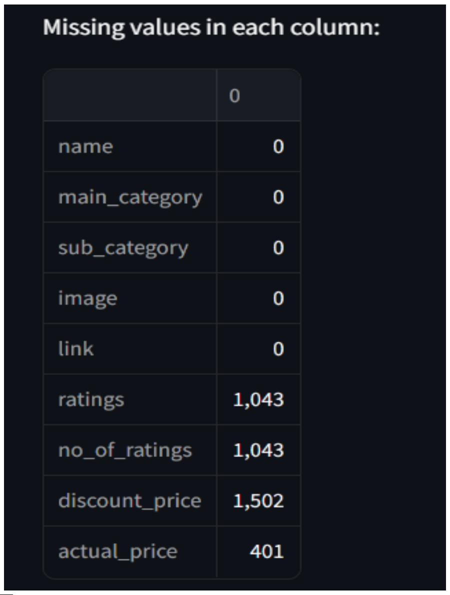


Data Processing and Visualization

Choose a CSV file

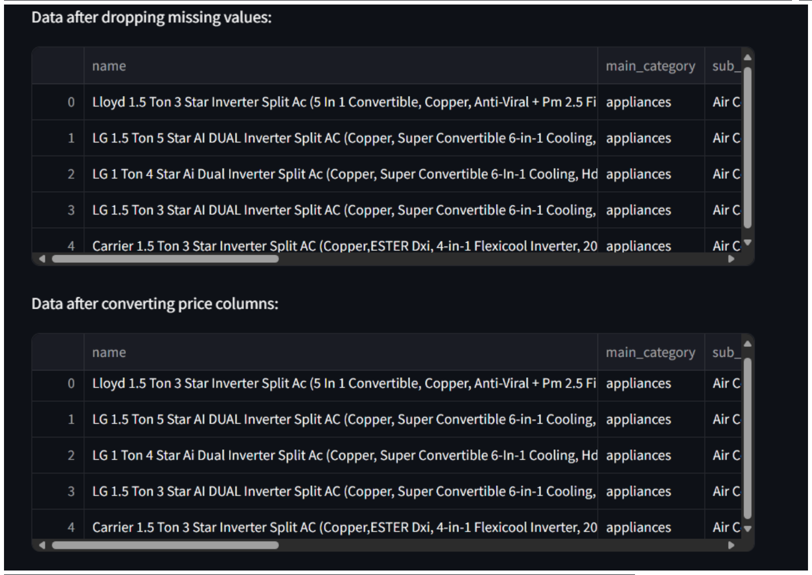
First few rows of the dataset:

	name	main_category	sub_
0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Fi	appliances	Air C
1	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
2	LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd	appliances	Air C
3	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
4	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Dxi, 4-in-1 Flexicool Inverter, 20	appliances	Air C



Missing values in each column:

	0
name	0
main_category	0
sub_category	0
image	0
link	0
ratings	1,043
no_of_ratings	1,043
discount_price	1,502
actual_price	401

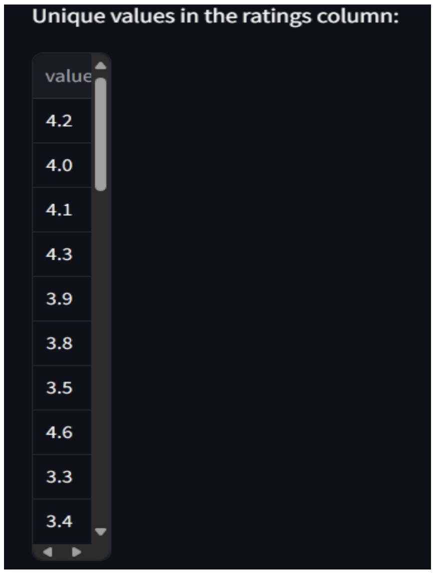


Data after dropping missing values:

	name	main_category	sub_
0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Fi	appliances	Air C
1	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
2	LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd	appliances	Air C
3	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
4	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Dxi, 4-in-1 Flexicool Inverter, 20	appliances	Air C

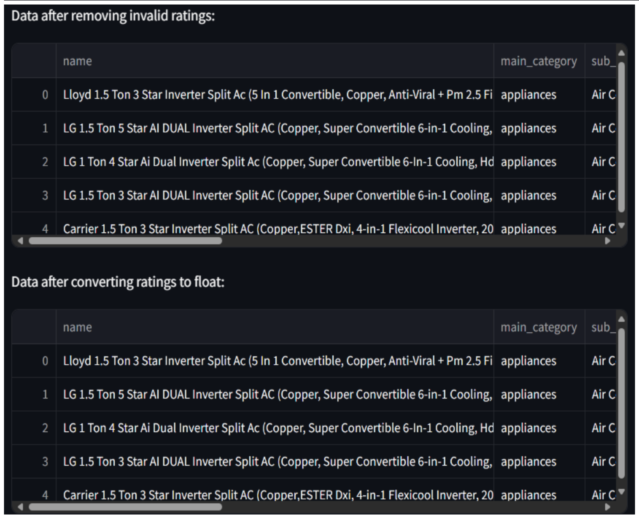
Data after converting price columns:

	name	main_category	sub_
0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Fi	appliances	Air C
1	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
2	LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd	appliances	Air C
3	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
4	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Dxi, 4-in-1 Flexicool Inverter, 20	appliances	Air C



Unique values in the ratings column:

value
4.2
4.0
4.1
4.3
3.9
3.8
3.5
4.6
3.3
3.4

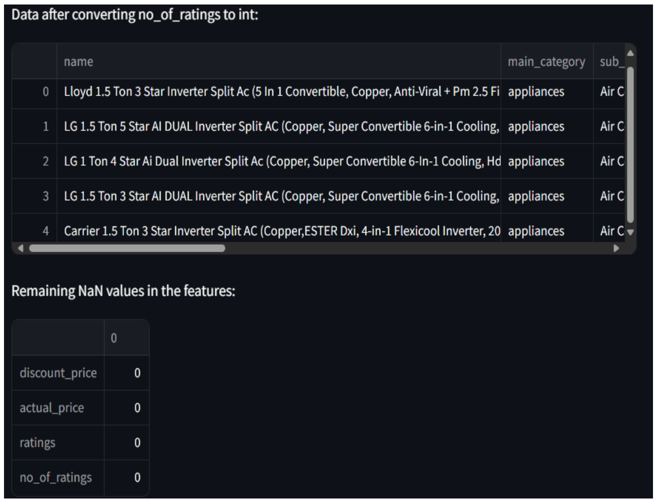


Data after removing invalid ratings:

	name	main_category	sub_
0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Fi	appliances	Air C
1	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
2	LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd	appliances	Air C
3	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
4	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Dxi, 4-in-1 Flexicool Inverter, 20	appliances	Air C

Data after converting ratings to float:

	name	main_category	sub_
0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Fi	appliances	Air C
1	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
2	LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd	appliances	Air C
3	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
4	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Dxi, 4-in-1 Flexicool Inverter, 20	appliances	Air C



Data after converting no_of_ratings to int:

	name	main_category	sub_
0	Lloyd 1.5 Ton 3 Star Inverter Split Ac (5 In 1 Convertible, Copper, Anti-Viral + Pm 2.5 Fi	appliances	Air C
1	LG 1.5 Ton 5 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
2	LG 1 Ton 4 Star Ai Dual Inverter Split Ac (Copper, Super Convertible 6-In-1 Cooling, Hd	appliances	Air C
3	LG 1.5 Ton 3 Star AI DUAL Inverter Split AC (Copper, Super Convertible 6-in-1 Cooling,	appliances	Air C
4	Carrier 1.5 Ton 3 Star Inverter Split AC (Copper,ESTER Dxi, 4-in-1 Flexicool Inverter, 20	appliances	Air C

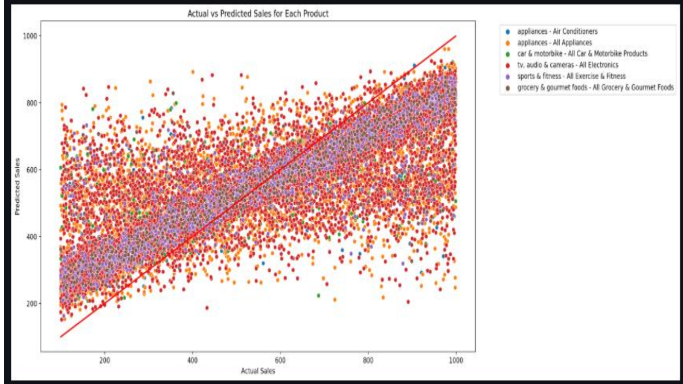
Remaining NaN values in the features:

	0
discount_price	0
actual_price	0
ratings	0
no_of_ratings	0

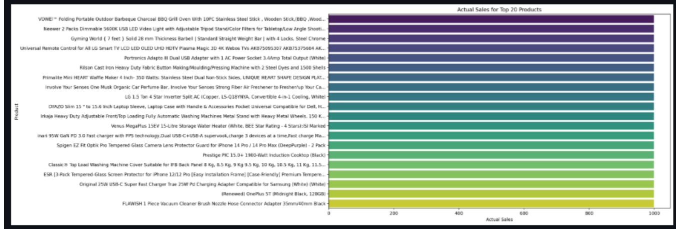
Mean Squared Error: 77539.57575764932

R^2 Score: -0.1612028265661023

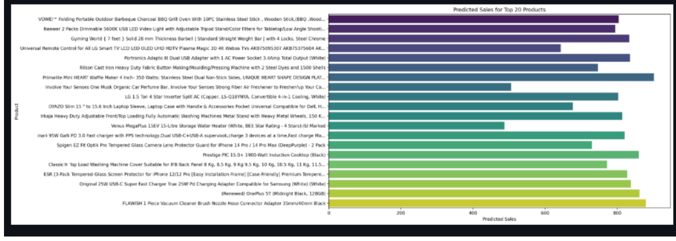
Actual vs Predicted Sales for Each Product



Actual Sales for Top 20 Products

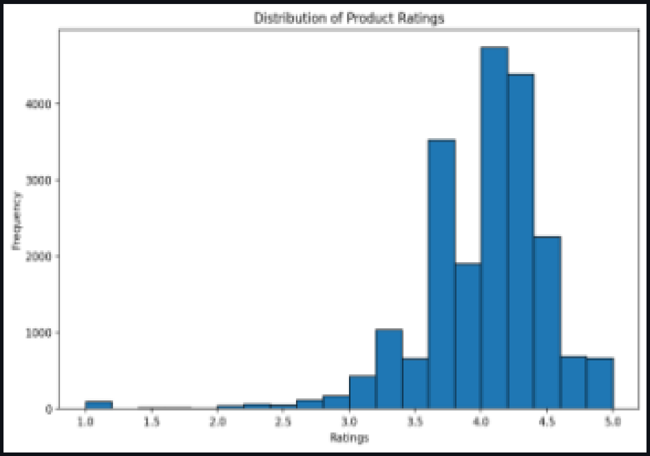


Predicted Sales for Top 20 Products

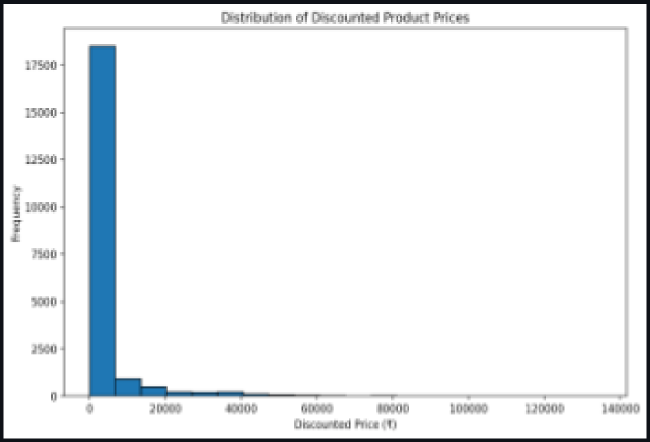


Price Distribution Analysis

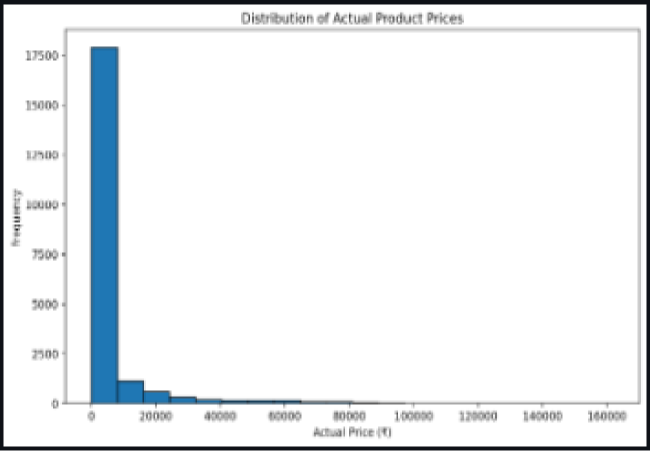
Distribution of Product Ratings



Distribution of Discounted Product Prices



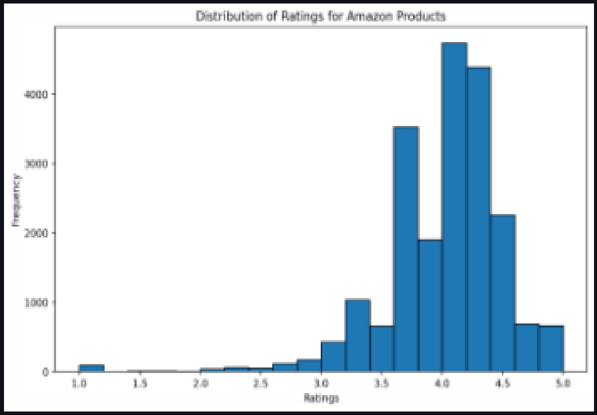
Distribution of Actual Product Prices



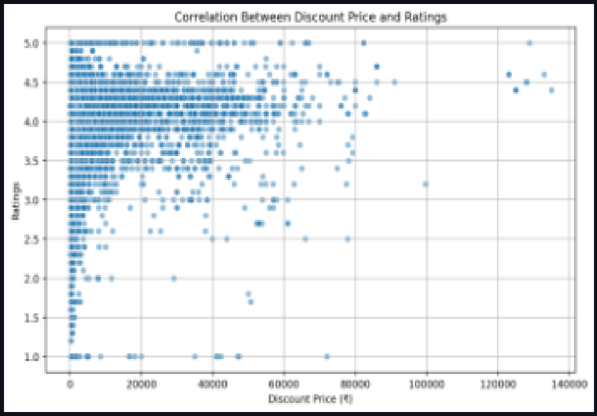
Ratings and Reviews Analysis

Average Rating of Amazon Products: 3.99

Distribution of Ratings for Amazon Products

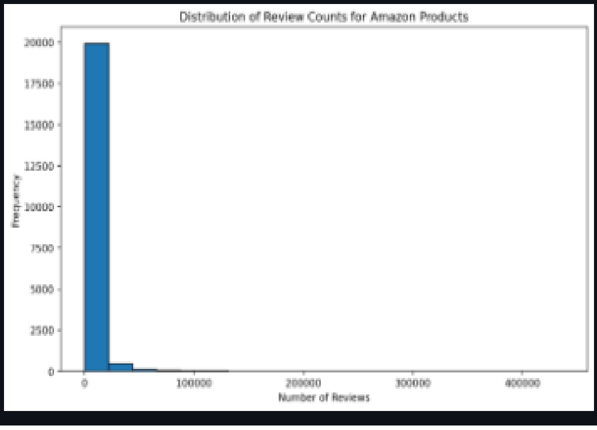


Correlation Between Discount Price and Ratings

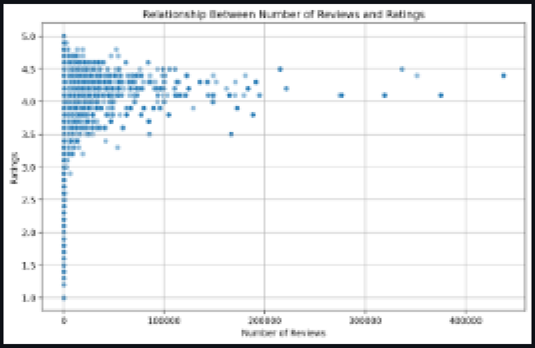


Review Count Analysis

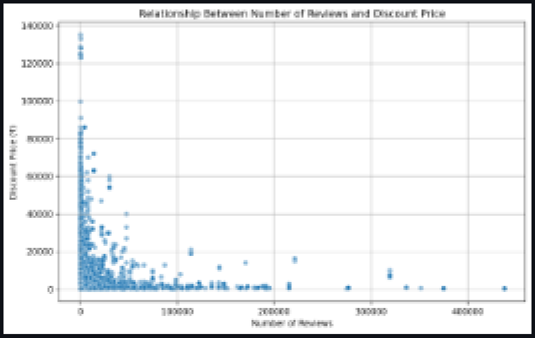
Distribution of Review Counts for Amazon Products



Relationship Between Number of Reviews and Ratings

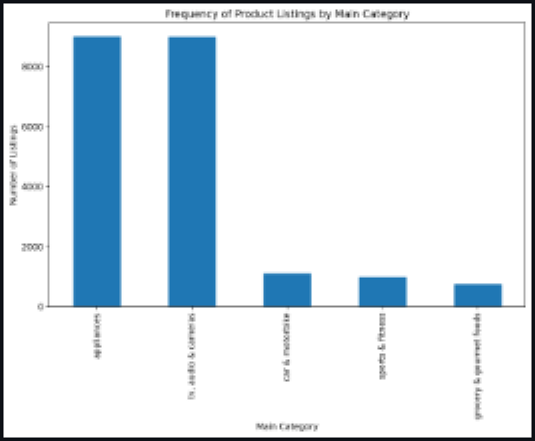


Relationship Between Number of Reviews and Discount Price

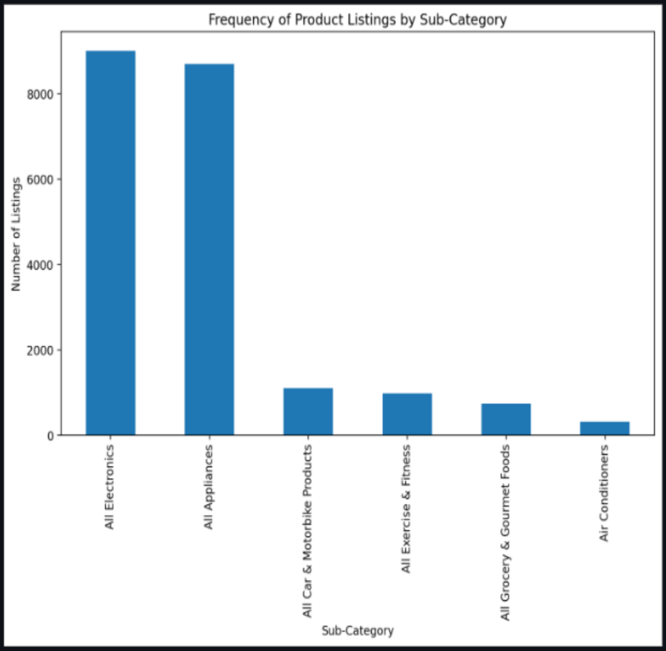


Product Popularity and Visibility

Frequency of Product Listings by Main Category

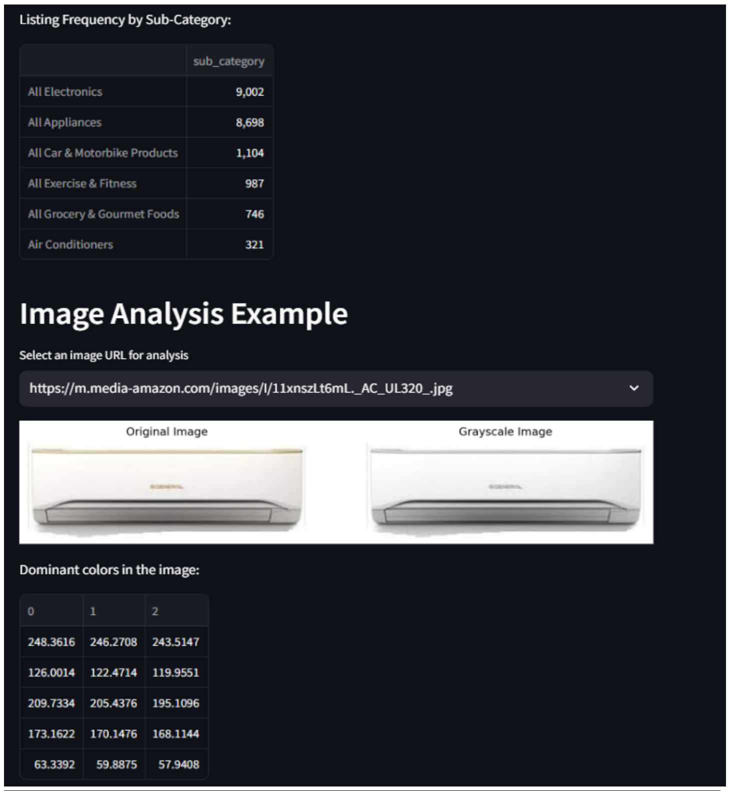


Frequency of Product Listings by Sub-Category



Listing Frequency by Main Category:

	main_cat
appliances	
tv, audio & cameras	
car & motorbike	
sports & fitness	



Listing Frequency by Sub-Category:

	sub_category
All Electronics	9,002
All Appliances	8,698
All Car & Motorbike Products	1,104
All Exercise & Fitness	987
All Grocery & Gourmet Foods	746
Air Conditioners	321

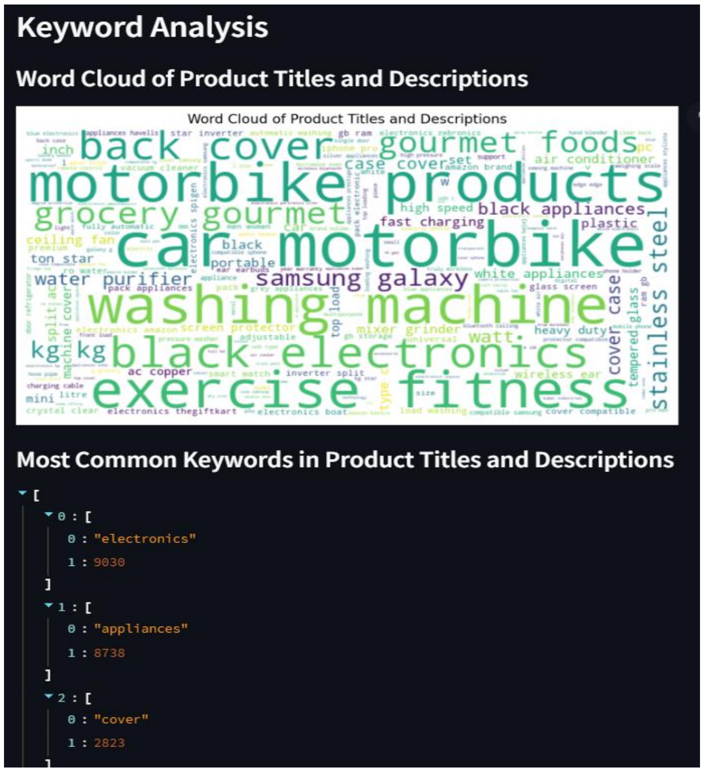
Image Analysis Example

Select an image URL for analysis

https://m.media-amazon.com/images/I/11xnszLt6mL._AC_UL320_.jpg

Dominant colors in the image:

0	1	2
248.3616	246.2708	243.5147
126.0014	122.4714	119.9551
209.7334	205.4376	195.1096
173.1622	170.1476	168.1144
63.3392	59.8875	57.9408

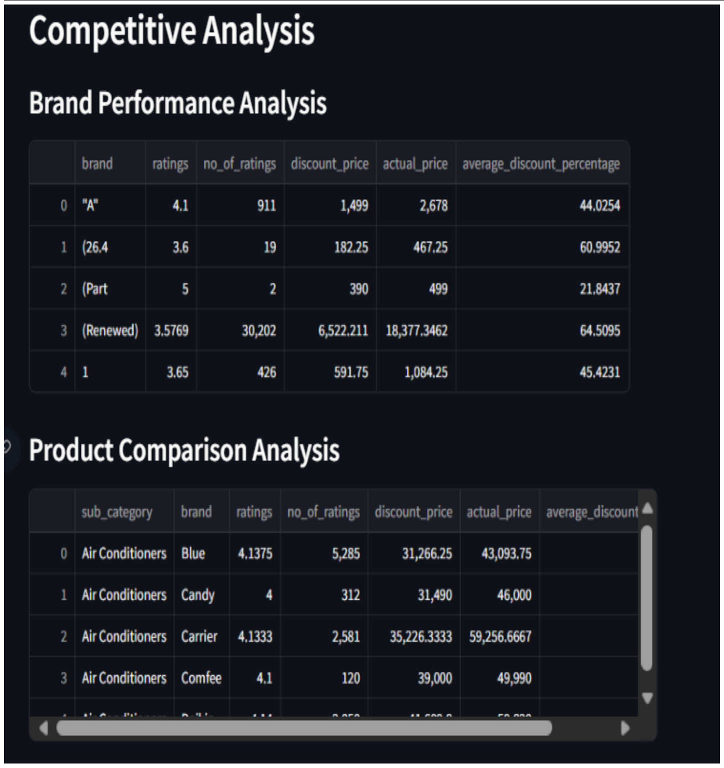


Keyword Analysis

Word Cloud of Product Titles and Descriptions

Most Common Keywords in Product Titles and Descriptions

```
[
  0 : [
    0 : "electronics"
    1 : 9030
  ]
  1 : [
    0 : "appliances"
    1 : 8738
  ]
  2 : [
    0 : "cover"
    1 : 2823
  ]
```



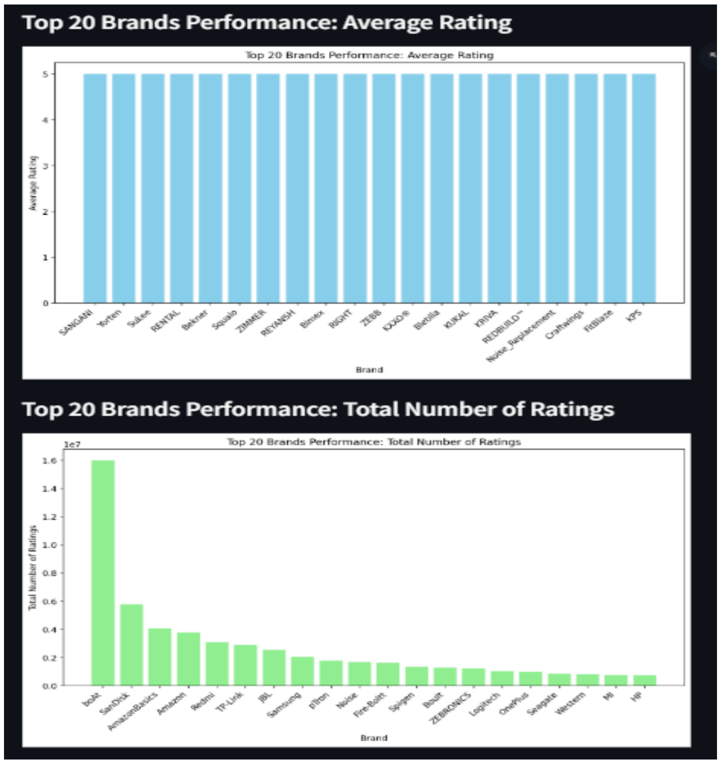
Competitive Analysis

Brand Performance Analysis

	brand	ratings	no_of_ratings	discount_price	actual_price	average_discount_percentage
0	"A"	4.1	911	1,499	2,678	44.0254
1	(26.4	3.6	19	182.25	467.25	60.9952
2	(Part	5	2	390	499	21.8437
3	(Renewed)	3.5769	30,202	6,522.211	18,377.3462	64.5095
4	1	3.65	426	591.75	1,084.25	45.4231

Product Comparison Analysis

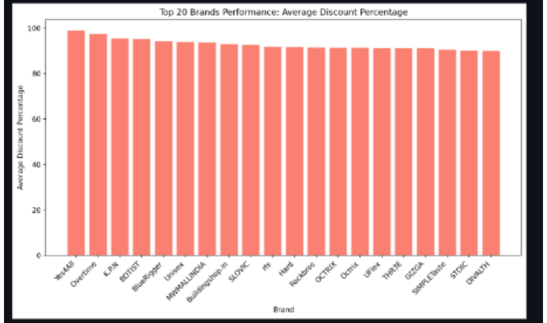
	sub_category	brand	ratings	no_of_ratings	discount_price	actual_price	average_discount
0	Air Conditioners	Blue	4.1375	5,285	31,266.25	43,093.75	
1	Air Conditioners	Candy	4	312	31,490	46,000	
2	Air Conditioners	Carrier	4.1333	2,581	35,226.3333	59,256.6667	
3	Air Conditioners	Comfee	4.1	120	39,000	49,990	



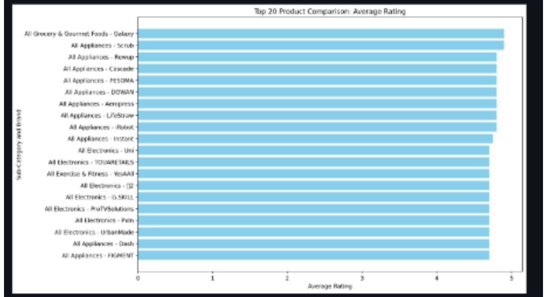
Top 20 Brands Performance: Average Rating

Top 20 Brands Performance: Total Number of Ratings

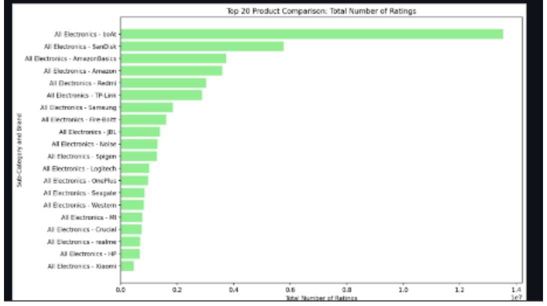
Top 20 Brands Performance: Average Discount Percentage



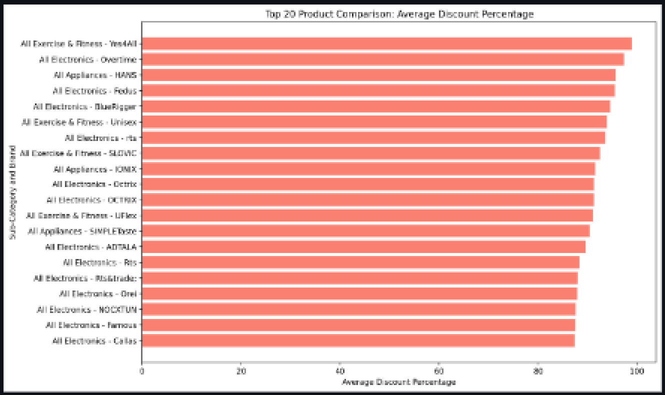
Top 20 Product Comparison: Average Rating



Top 20 Product Comparison: Total Number of Ratings



Top 20 Product Comparison: Average Discount Percentage



Product Reviews Sentiment Analysis and Recommendation System

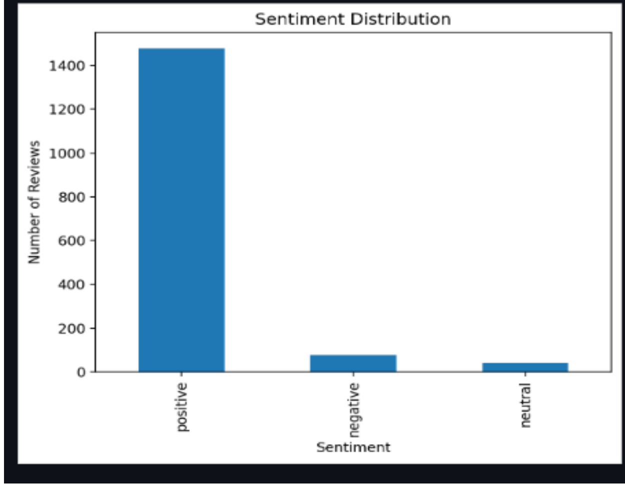
Choose a CSV file

Drag and drop file here
Limit 200MB per file • CSV

Browse files

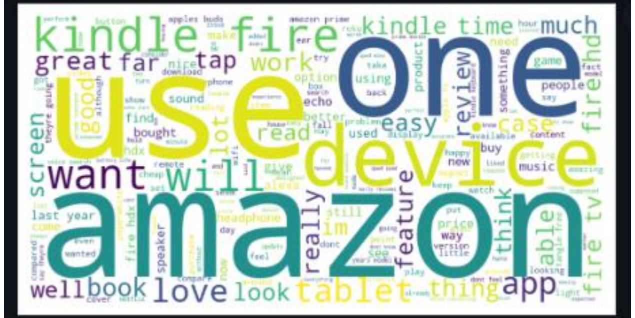
7817_1.csv 17.5MB

Sentiment Analysis

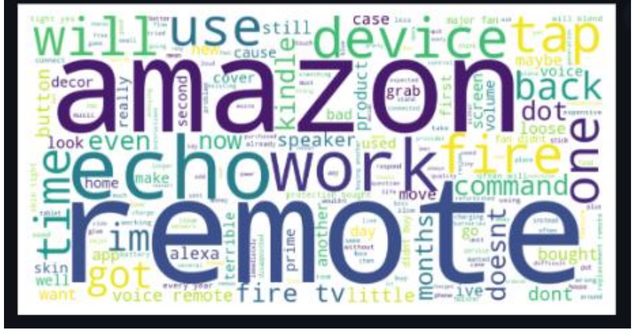


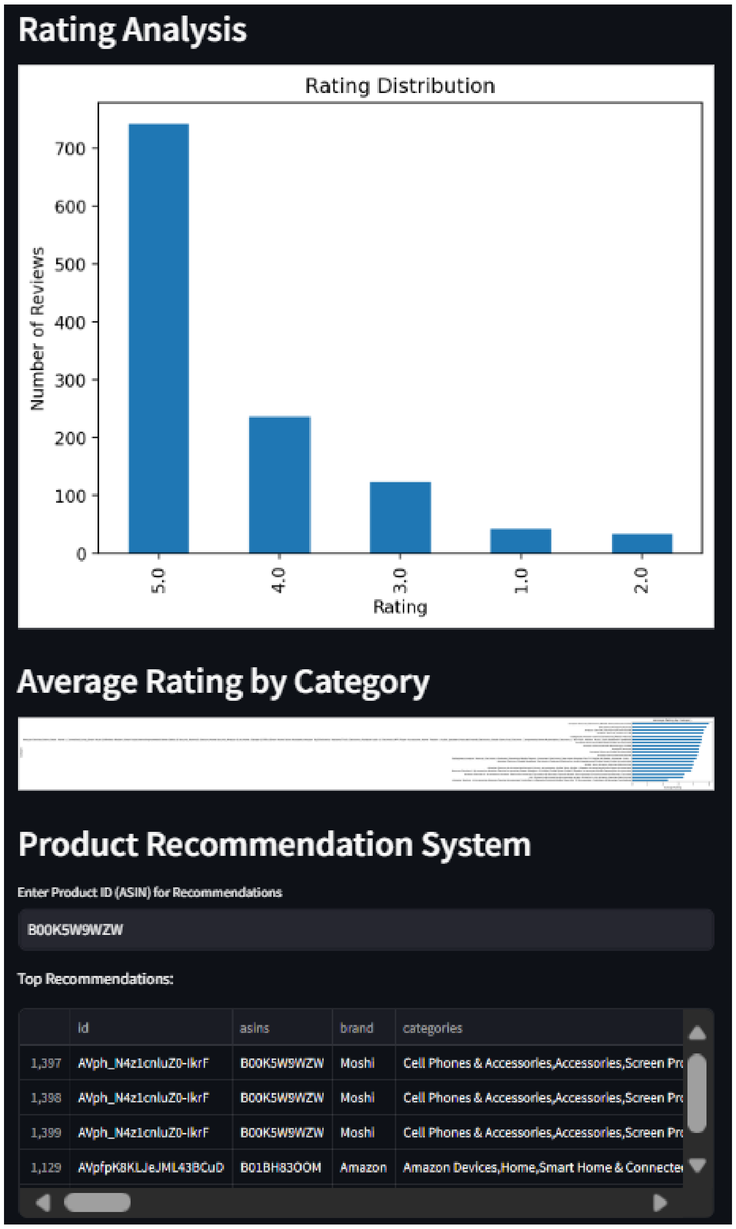
Word Clouds

Positive Reviews



Negative Reviews





Product Recommendation System

Enter Product ID (ASIN) for Recommendations

B00K5W9WZW

Top Recommendations:

	id	asins	brand	categories
1,397	AVph_N4z1cnluZ0-lkrF	B00K5W9WZW	Moshi	Cell Phones & Accessories,Accessories,Screen Pr
1,398	AVph_N4z1cnluZ0-lkrF	B00K5W9WZW	Moshi	Cell Phones & Accessories,Accessories,Screen Pr
1,399	AVph_N4z1cnluZ0-lkrF	B00K5W9WZW	Moshi	Cell Phones & Accessories,Accessories,Screen Pr
1,129	AVpfpK8KLJeJML43BCuD	B01BH83OOM	Amazon	Amazon Devices,Home,Smart Home & Connecte

CONCLUSION

By following this methodology, the analysis will help in understanding customer satisfaction and preferences, identifying key areas for product improvement, and enhancing the overall customer experience with targeted product recommendations.