

---

# Amazon Product Analysis

18<sup>th</sup> May 2023

## Introduction:

Introduction: In this project, I am investigating the application of text mining techniques on a dataset of laptops obtained through data scraping from the Amazon website. The project involves several steps, starting with data scraping, followed by analyzing patterns in the scraped data, and utilizing clustering techniques for further analysis. The experiments conducted in this project are interesting from a text-mining standpoint for the following reasons:

- **Data Scraping:** Data scraping involves extracting data from web pages, which in this case, includes laptop information from the Amazon website. By scraping data, we can gather a substantial amount of information, such as laptop titles, prices, ratings, availability, and other relevant attributes. This data becomes the foundation for subsequent text mining analysis.
- **Pattern Analysis:** After obtaining the scraped data, the next step involves analyzing patterns within the dataset. This analysis can include identifying common keywords, phrases, or characteristics within laptop titles or other textual attributes. By exploring patterns, we can gain insights into the features, specifications, or trends associated with different laptop brands or models.
- **Clustering Analysis:** Clustering techniques can be applied to the dataset to group similar laptops together based on their attributes. Clustering helps identify natural groupings or clusters within the data, allowing us to understand similarities and differences among laptops. This analysis can reveal patterns or segments within the dataset, aiding in market segmentation or targeted marketing strategies.
- **Insights for Business Decision-Making:** The text mining techniques used in this project provide valuable insights for business decision-making. By analyzing the scraped data, patterns, and clusters, we can uncover information about customer preferences, market trends, and competitive landscapes. These insights can inform marketing strategies, product development decisions, and help businesses stay competitive in the laptop market.
- **Scalability and Automation:** Data scraping and text mining techniques allow for scalability and automation. By automating the data scraping process, we can collect a large amount of data efficiently. Text mining techniques enable the analysis of this data in an automated manner, allowing for quick identification of patterns and clusters within the dataset.

---

In conclusion, the project involving data scraping, pattern analysis, and clustering using text mining techniques provides valuable insights into laptop data obtained from the Amazon website. The process allows for scalable data collection, automated analysis, and helps businesses make informed decisions based on patterns, clusters, and market trends.

## **Project Proposal:**

### **1. Introduction:**

In this project, we propose to perform a data mining analysis on a dataset of laptops obtained from the Amazon website. The aim is to extract valuable insights, identify patterns, and discover meaningful information from the dataset. The analysis will involve various techniques, including data preprocessing, exploratory data analysis, clustering, and association rule mining.

### **2. Objectives:**

- Gather a dataset of laptops by scraping information from the Amazon website, including attributes such as title, price, rating, reviews, availability, and brand.
- Clean and preprocess the dataset to handle missing values, remove duplicates, and ensure data quality.
- Perform exploratory data analysis to gain insights into the distribution, statistics, and correlations of the laptop attributes.
- Utilize clustering techniques to group similar laptops based on their attributes, such as price, rating, and brand.
- Analyze the clusters to identify patterns and common characteristics.
- Apply association rule mining to discover interesting associations and relationships between different attributes of the laptops, such as price and availability or brand and rating.
- Generate meaningful visualizations and reports to present the findings and insights from the data mining analysis.

### **3. Methodology:**

- **Data Collection:** Scrape the laptop data from the Amazon website, focusing on attributes like title, price, rating, reviews, availability, and brand.
- **Data Preprocessing:** Clean the dataset by handling missing values, removing duplicates, and performing necessary data transformations.
- **Exploratory Data Analysis:** Conduct exploratory data analysis to understand the distribution, statistics, and relationships between the laptop attributes. Visualize the data using plots and charts.

- 
- Clustering Analysis: Apply clustering algorithms (e.g., k-means, hierarchical clustering) to group similar laptops based on their attributes. Evaluate and interpret the clusters to gain insights.
  - Association Rule Mining: Utilize association rule mining algorithms (e.g., Apriori) to discover interesting relationships between attributes and generate association rules.
  - f. Visualization and Reporting: Create visualizations (e.g., bar charts, scatter plots, network graphs) to present the findings and insights. Prepare a comprehensive report summarizing the analysis and key findings.
4. Expected Outcome:
- Insights into laptop pricing trends, customer ratings, and availability patterns.
  - Identification of distinct groups or clusters of laptops based on attributes like price, rating, and brand.
  - Discovering interesting associations and relationships between different attributes of the laptops.
  - Visualization and reporting of the findings to facilitate understanding and decision-making.

## **Purpose of Experiment:**

The experiments conducted in this project aim to gain insights and extract meaningful information from a dataset of laptops obtained from the Amazon website. The experiments involve pattern analysis, price analysis, rating analysis, and availability analysis. The goal is to identify patterns, understand pricing trends, assess customer ratings, and analyze availability patterns. By testing various aspects of the dataset, the project aims to generate valuable insights for informed decision-making in areas such as marketing, product development, and inventory management within the laptop market.

## **Dataset Explanation:**

The dataset used for conducting the experiments in this project is obtained from the Amazon website and focuses on laptops. It contains several attributes that provide information about different aspects of the laptops.

The dataset includes the following columns:

- Title: This column represents the title or name of the laptop as displayed on the Amazon website. It provides a brief description of the laptop and may include information about the brand, model, and specific features.


- 
- **Price:** The price column indicates the cost of the laptop. It is a numerical attribute represented as a float or numeric data type. The prices may vary across different laptop models and brands.
  - **Rating:** The rating column represents the customer rating or satisfaction level for each laptop. It is a numerical attribute represented as a float or numeric data type. The rating is typically based on a scale, such as 1 to 5 stars, where higher values indicate better ratings.
  - **Reviews:** The reviews column indicates the total number of customer reviews received for each laptop. It is represented as an integer data type. The number of reviews provides an indication of the popularity and engagement of customers with a particular laptop.
  - **Availability:** The availability column represents the availability status of the laptop. It provides information about whether the laptop is currently in stock or out of stock. This column typically contains categorical values, such as "In Stock," "Out of Stock," or "Currently Unavailable."

These attributes provide essential information about the laptops, including their names, prices, customer ratings, number of reviews, and availability status. By analyzing and mining this dataset, we can gain insights into various aspects of the laptops, such as pricing trends, customer satisfaction levels, popularity, and availability patterns. The dataset allows us to explore relationships between these attributes, identify patterns, and make informed decisions based on the insights obtained from the analysis.

## Methodology:

- **Data Scraping:**

This function uses BeautifulSoup to locate and extract the product title from a web page. It handles any potential errors and returns the title as a string value.

```
 # Function to extract Product Title
def get_title(soup):

    try:
        # Outer Tag Object
        title = soup.find("span", attrs={"id": 'productTitle'})

        # Inner NavigableString Object
        title_value = title.text

        # Title as a string value
        title_string = title_value.strip()

    except AttributeError:
        title_string = ""

    return title_string
```

---

This function extracts the number of user reviews from the web page. It looks for a <span> tag with the id attribute 'acrCustomerReviewText' and extracts the review count as a string value. If the tag is not found, an empty string is returned.

```
# Function to extract Product Price

def get_price(soup):
    try:
        price_element = soup.find("span", attrs={'class': 'a-offscreen'})
        price = price_element.get_text().strip()

    except AttributeError:
        price = ""

    return price
```

This function extracts the product rating from the web page. It first tries to find an <i> tag with the class attribute 'a-icon a-icon-star a-star-4-5'. If found, the rating is extracted as a string value. If the first attempt fails, it tries to find a <span> tag with the class attribute 'a-icon-alt'. If the rating is found, it is returned as a string. If both attempts fail, an empty string is returned.

```
# Function to extract Product Rating
def get_rating(soup):

    try:
        rating = soup.find("i", attrs={'class': 'a-icon a-icon-star a-star-4-5'}).string.strip()

    except AttributeError:
        try:
            rating = soup.find("span", attrs={'class': 'a-icon-alt'}).string.strip()

        except:
            rating = ""

    return rating
```

This function extracts the number of user reviews from the web page. It looks for a <span> tag with the id attribute 'acrCustomerReviewText' and extracts the review count as a string value. If the tag is not found, an empty string is returned.

```
# Function to extract Number of User Reviews
def get_review_count(soup):
    try:
        review_count = soup.find("span", attrs={'id': 'acrCustomerReviewText'}).string.strip()

    except AttributeError:
        review_count = ""

    return review_count
```

This function extracts the availability status of the product from the web page. It first locates a <div> tag with the id attribute 'availability'. From this tag, it extracts the availability status as a string value by finding the <span> tag within it. If the availability information is not found, it sets the availability status to "Not Available".

```
# Function to extract Number of User Reviews
def get_review_count(soup):
    try:
        review_count = soup.find("span", attrs={'id': 'acrCustomerReviewText'}).string.strip()

    except AttributeError:
        review_count = ""

    return review_count
```

This function performs web scraping on a list of Amazon URLs, extracts product details using helper functions, and stores the scraped data in a DataFrame. The function also handles data cleaning and saves the resulting DataFrame as a CSV file.

```
def scrape_amazon_data(user_agent, url_list):
    # Store the scraped data
    data = {"title": [], "price": [], "rating": [], "reviews": [], "availability": []}

    # Headers for the HTTP requests
    headers = {'User-Agent': user_agent, 'Accept-Language': 'en-US, en;q=0.5'}

    for url in url_list:
        # HTTP Request
        webpage = requests.get(url, headers=headers)
        soup = BeautifulSoup(webpage.content, "html.parser")

        # Fetch links as List of Tag Objects
        links = soup.find_all("a", attrs={'class': 'a-link-normal s-no-outline'})

        # Store the links
        links_list = [link.get('href') for link in links]

        # Loop for extracting product details from each link
        for link in links_list:
            new_webpage = requests.get("https://www.amazon.com" + link, headers=headers)
            new_soup = BeautifulSoup(new_webpage.content, "html.parser")

            # Function calls to extract necessary product information
            data['title'].append(get_title(new_soup))
            data['price'].append(get_price(new_soup))
            data['rating'].append(get_rating(new_soup))
```

- Data Cleaning and EDA:

Data head

```
✓ [104] data.head()  
1s
```

	title	price	rating	reviews	availability
0	Dell 2022 Newest Inspiron 15 Laptop, 15.6" HD ...	\$390.00	4.1 out of 5 stars	204 ratings	In Stock
1	Dell 2022 Newest Inspiron 3000 Laptop, 15.6 HD...	\$296.02	3.9 out of 5 stars	484 ratings	Only 13 left in stock - order soon
2	Dell Latitude 7490 14' FHD Laptop PC - Intel C...	\$226.62	3.8 out of 5 stars	61 ratings	Only 12 left in stock - order soon.
3	Fast Dell Latitude E5470 HD Business Laptop No...	\$173.00	4.1 out of 5 stars	1,493 ratings	In Stock.
4	Dell Latitude E7470 14in Laptop, Core i5-6300U...	\$169.90	4.2 out of 5 stars	1,093 ratings	In Stock.

Data types

```
▶ data.dtypes
```

```
☞ title           object  
   price          object  
   rating         object  
   reviews        object  
   availability    object  
   dtype: object
```

Extracting Rating values

```
✓ [107] data['rating'] = data['rating'].str[:3]  
0s
```

```
✓ ▶ data['rating']  
0s
```

```
☞ 0      4.1  
   1      3.9  
   2      3.8  
   3      4.1  
   4      4.2  
   ...
```

---

This code replaces the dollar sign ('\$') and commas (',') in the 'price' column, and then converts the values to float using the `astype(float)` method. The resulting values will be numeric, allowing you to perform numerical operations and analysis on the 'price' column.

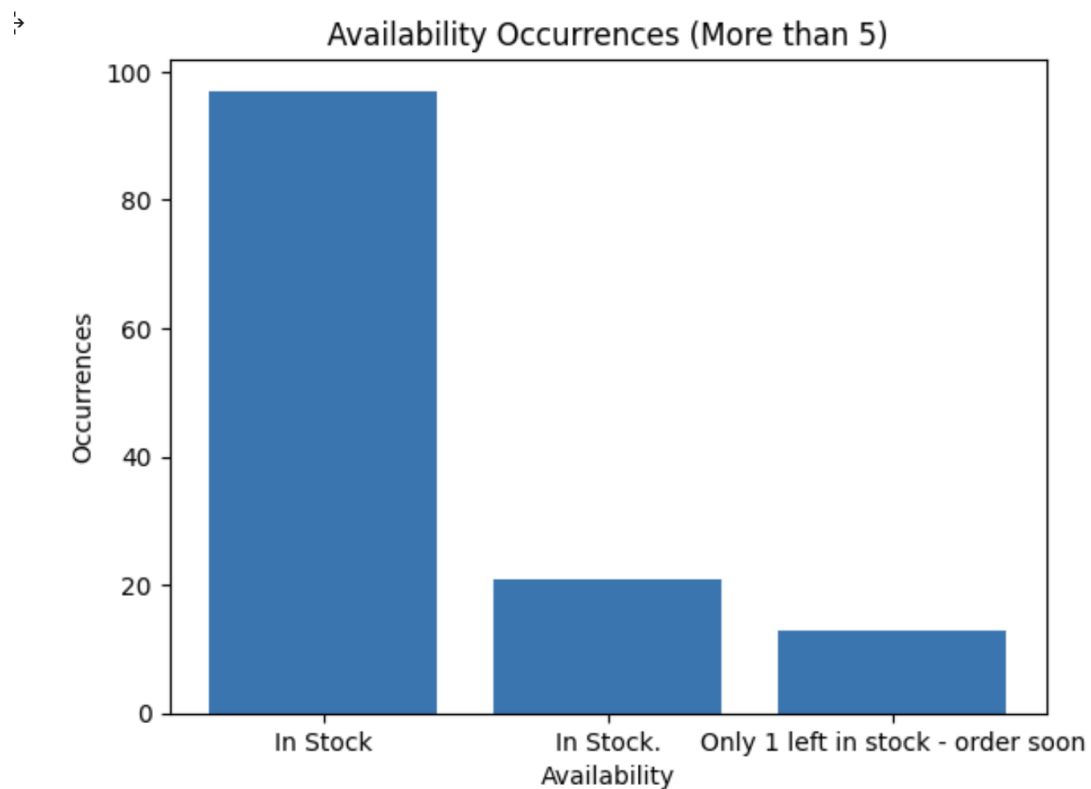
```
[112] data['price'] = pd.to_numeric(data['price'].str.replace('$', '').str.replace(',', ''), errors='coerce')
```

Selecting only Digits in reviews column

```
[116] data['reviews'] = pd.to_numeric(data['reviews'].str.extract('(\d+)')[0], errors='coerce')
```

```
[117] data['reviews'] = data['reviews'].astype('Int64')
```

Different Type of availabilities





---

### Updated data types:

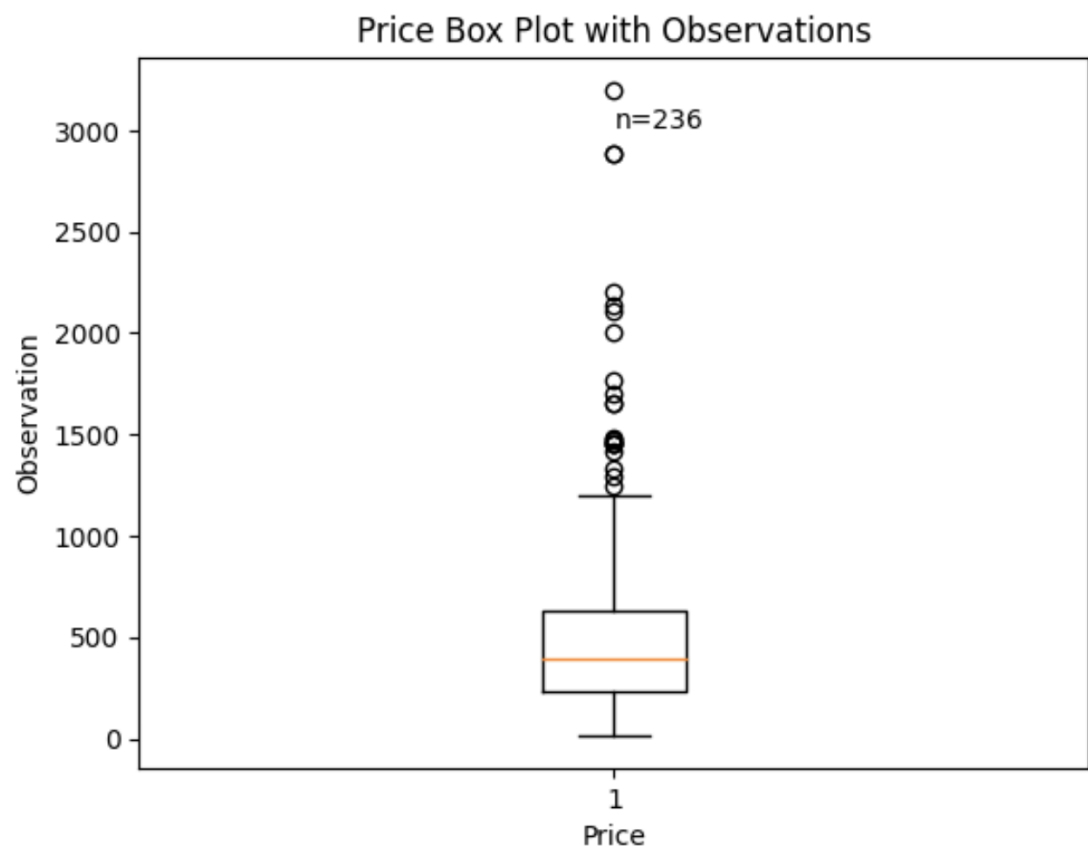
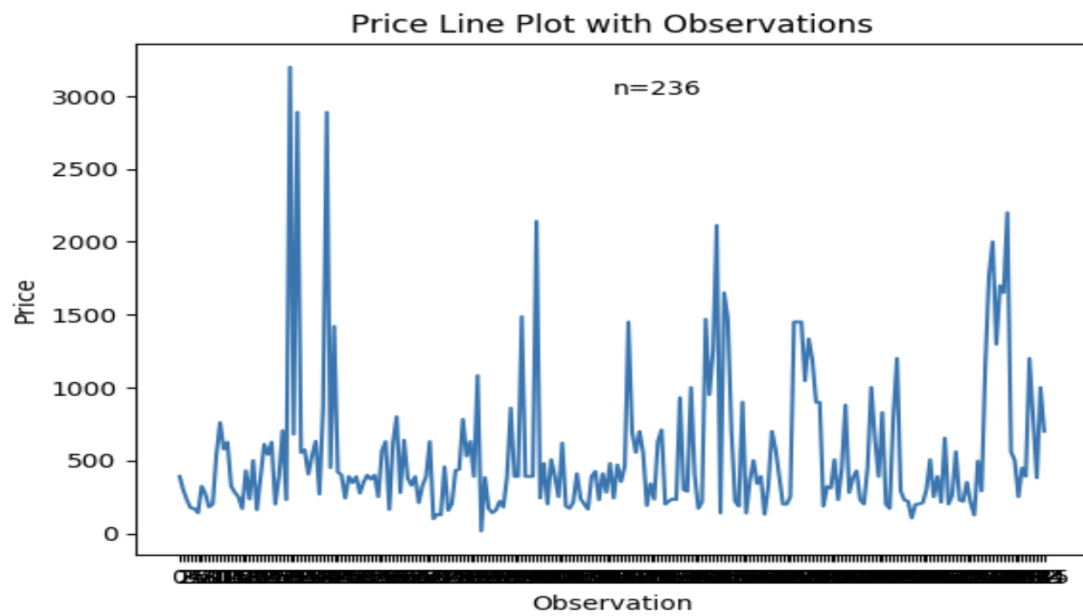
```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 236 entries, 0 to 235
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   title           236 non-null   object
1   price           222 non-null   float64
2   rating          228 non-null   float64
3   reviews         227 non-null   Int64
4   availability     219 non-null   object
dtypes: Int64(1), float64(2), object(2)
memory usage: 9.6+ KB
```

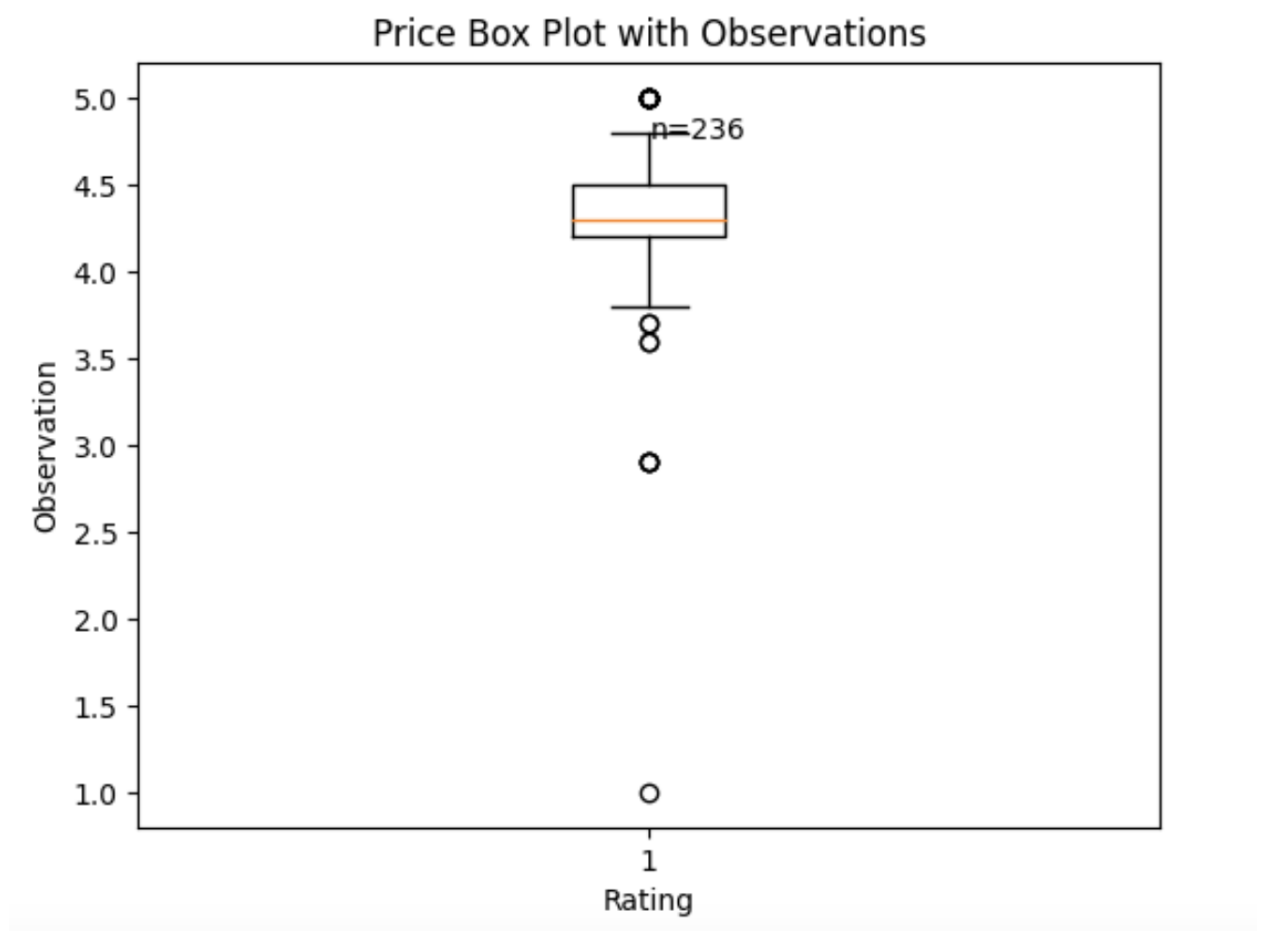
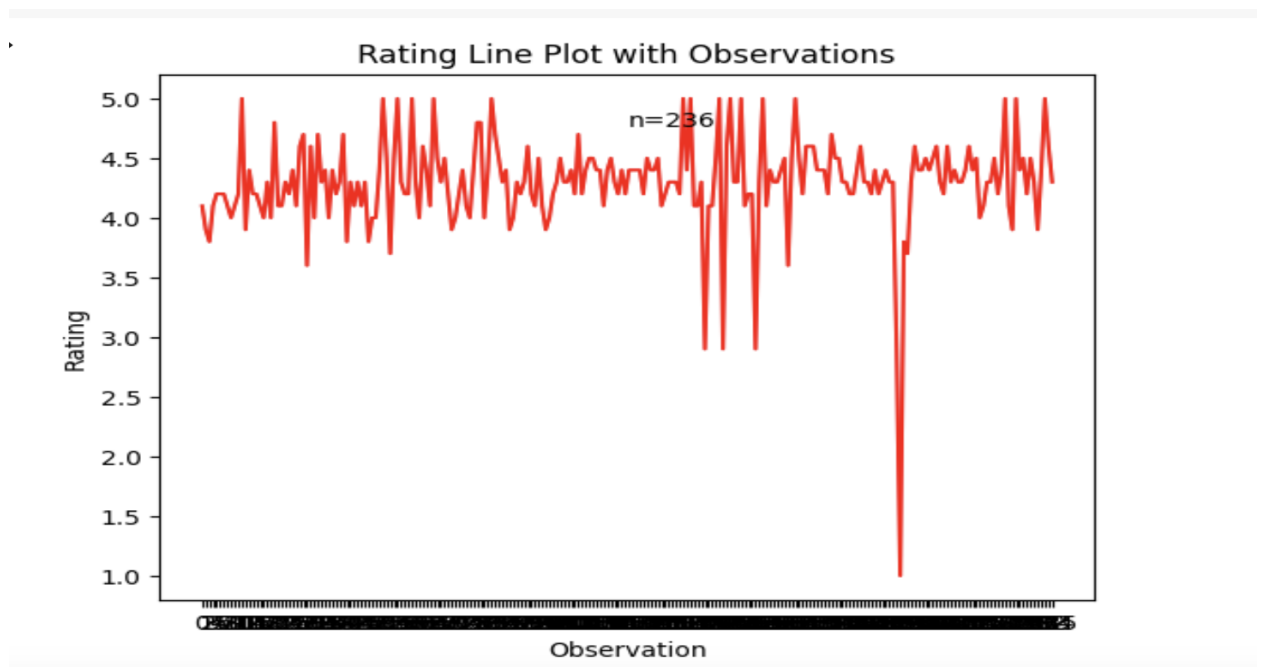
```
[ ] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 236 entries, 0 to 235
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   title           236 non-null   object
1   price           222 non-null   float64
2   rating          228 non-null   float64
3   reviews         227 non-null   Int64
4   availability     219 non-null   object
dtypes: Int64(1), float64(2), object(2)
memory usage: 9.6+ KB
```

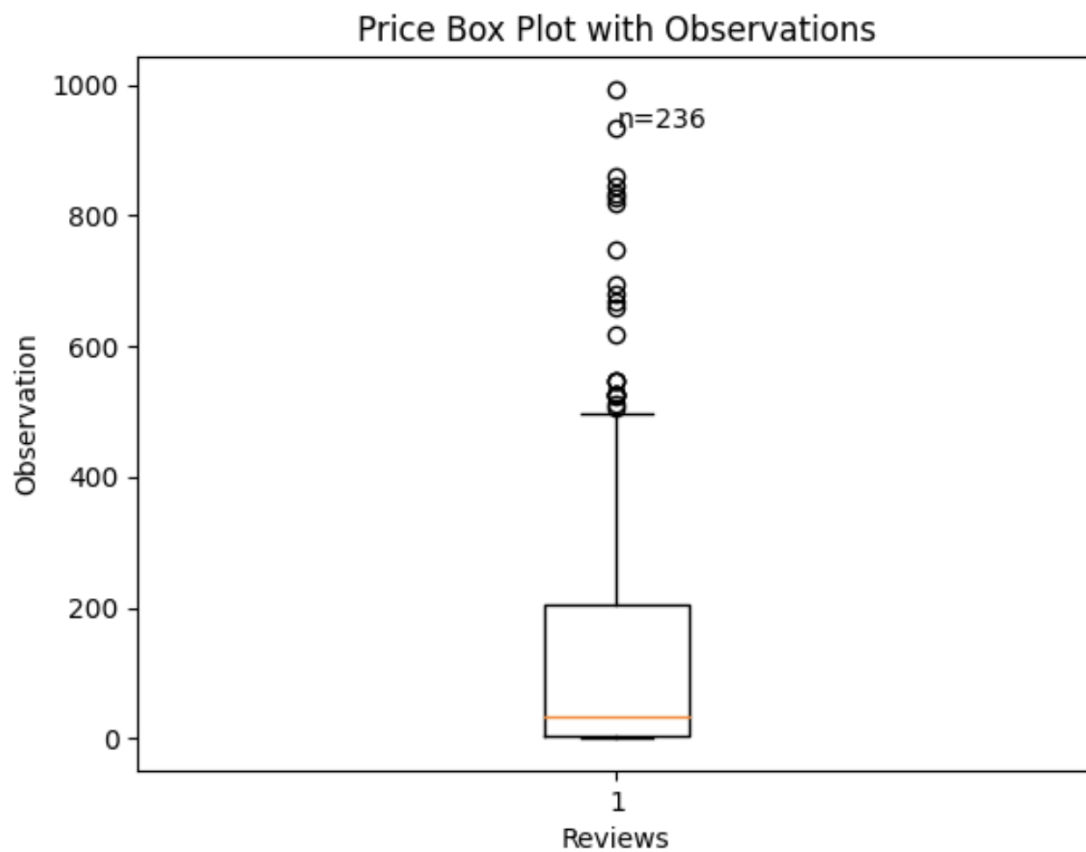
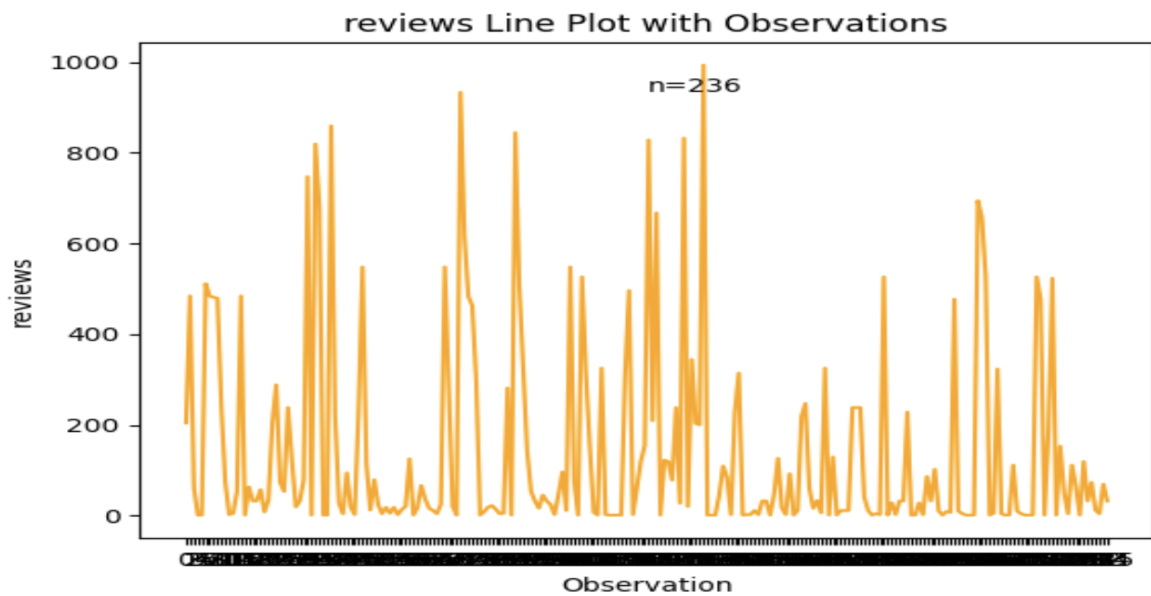
Plotting prices of different observations:



## Plotting Product Ratings:



## Plotting Reviews of products:



The code matches brand names from the given list with the lower-cased 'title' column and assigns the corresponding brand to a new 'brand' column in the DataFrame data. The code then removes the temporary 'lower\_title' column.

```
brands = ['Macbook', 'ASUS', 'DELL', 'SAMSUNG', 'Acer', 'HP', 'Toshiba', 'Lenovo', 'Lifebook', 'ApoloSign']

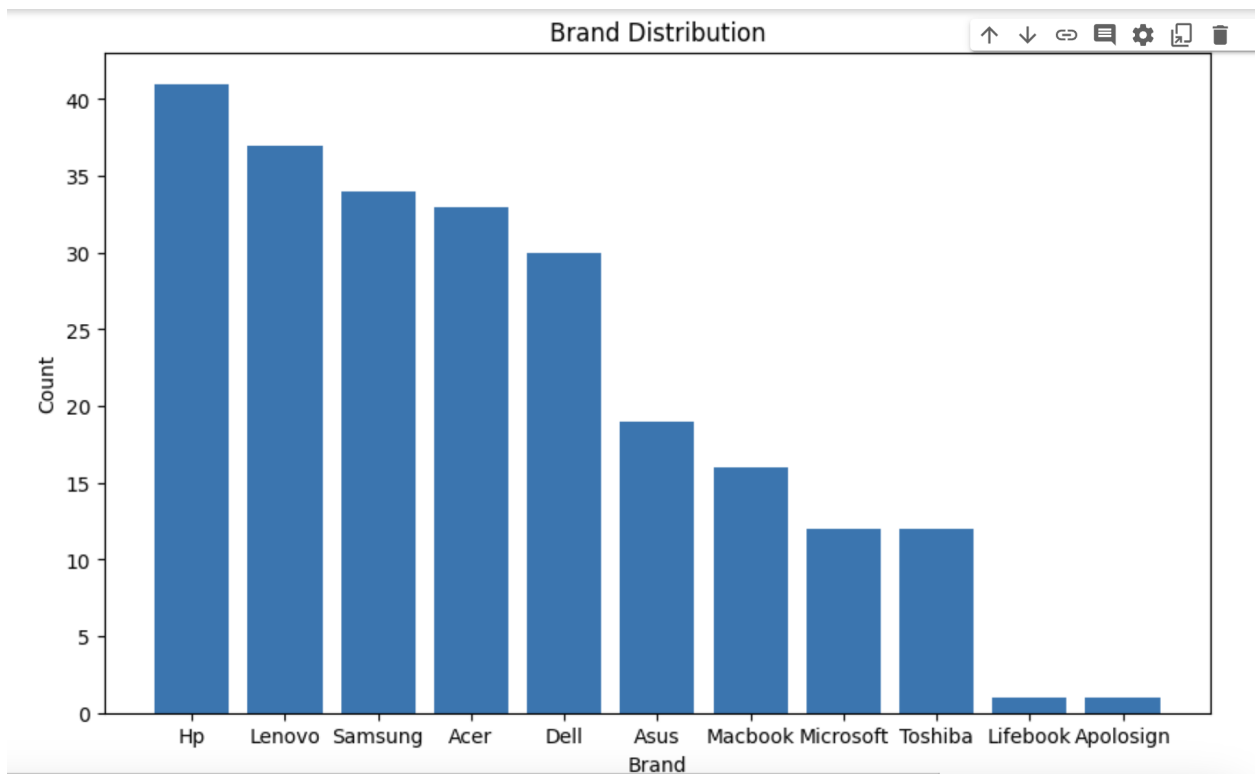
# Create brand column
data['brand'] = ''

# Convert title and brands to lower case
data['lower_title'] = data['title'].str.lower()
lower_brands = [brand.lower() for brand in brands]

# Iterate over each brand and check for a match in the lower_title
for brand in lower_brands:
    data.loc[data['lower_title'].str.contains(brand, case=False), 'brand'] = brand.capitalize()

# Remove the temporary lower_title column
data = data.drop(columns=['lower_title'])
```

## Brand Distributions:



- **K-Means Clustering:**

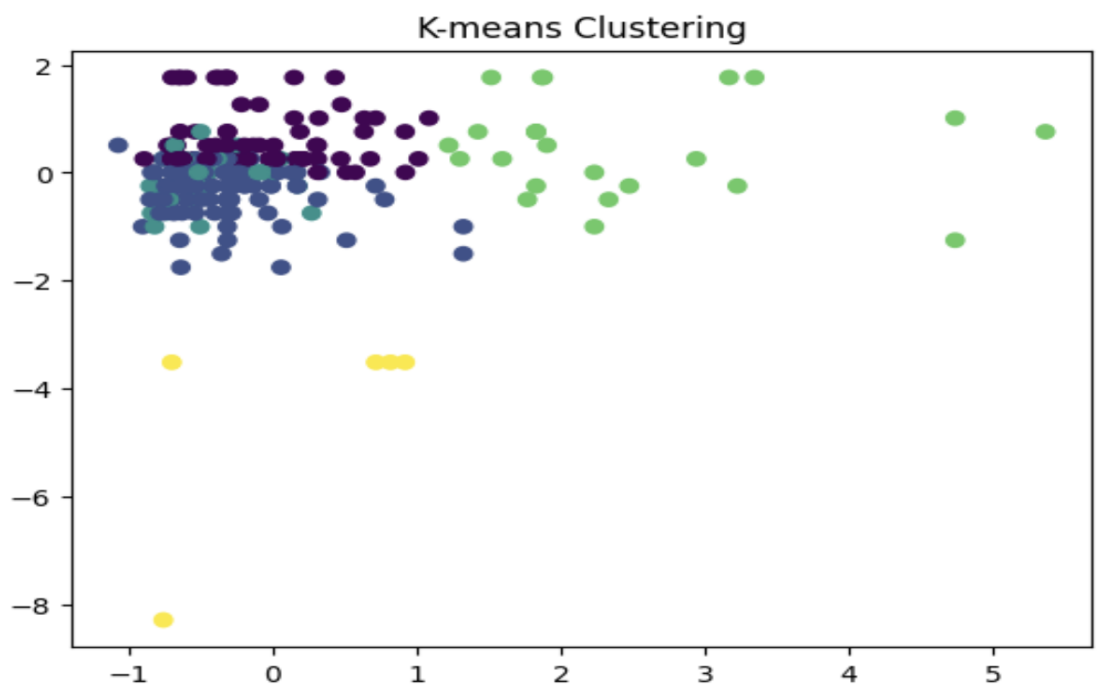
K-means clustering is an unsupervised machine learning algorithm used for grouping data points into K distinct clusters based on their similarities. The algorithm starts by randomly initializing K cluster centers and iteratively assigns data points to the nearest cluster based on distance calculations. The cluster centers are then updated by calculating the mean of the assigned data points. This process continues until convergence is reached. K-means clustering is commonly used in various applications such as customer segmentation, image segmentation, anomaly detection, and recommendation systems. However, it is sensitive to the initial choice of cluster centers and may converge to suboptimal solutions. Techniques such as multiple runs with different initializations or using variations of K-means can help mitigate this issue. By applying K-means clustering, patterns and structures within the data can be identified, providing valuable insights for analysis and decision-making.

✓  
0s



# Apply K-means clustering

```
kmeans = KMeans(n_clusters=num_clusters)
kmeans.fit(scaled_data)
cluster_labels = kmeans.labels_
```



- **AgglomerativeClustering:**

Agglomerative clustering is a hierarchical clustering algorithm that groups similar data points into clusters by iteratively merging the closest clusters based on a proximity measure. It starts with each data point as a separate cluster and progressively merges them until the desired number of clusters is obtained. The algorithm forms a dendrogram, illustrating the hierarchy of cluster merges.

Agglomerative clustering is useful for exploring complex data structures, identifying clusters at different granularity levels, and understanding inter-cluster relationships. The choice of proximity measure and linkage criterion impacts the merging process. Agglomerative clustering provides a flexible and interpretable approach to clustering, enabling the discovery of meaningful groups based on similarity.

```
▶ agg_cluster = AgglomerativeClustering(n_clusters=num_clusters)

# Fit the Agglomerative Clustering model to the selected data
cluster_labels = agg_cluster.fit_predict(selected_data)
```

