Onboarding Assignment

Big Data Scientist & Engineer

Hogeschool van Amsterdam

Name: *Bram Janssen (500748908)*
Minor: *Big Data*
Teacher: *Meij, D. van der*
Group: *BD2_4*
Date: *25 Sept*
Version: 1.0

Summary

The Big Data and Software Engineering (BDSE) minor program provides an intensive three-week immersion into data engineering and data science. This includes workshops and an individual onboarding assignment, focusing on data storage, cleaning, machine learning, and data visualisation, with a specific emphasis on sentiment analysis and interactive visualisation components.

The assignment centers on a Kaggle dataset of Amazon product reviews, employing Python for exploration, cleansing, and machine learning for sentiment analysis. The outcome is a comprehensive report elucidating data processing and machine learning algorithms.

Participants encompass 111,000 Amazon customers, with data quality relying on the Amazon review dataset. Potential sampling bias is acknowledged. Sentiment analysis, executed with Logistic Regression and SVM classifiers, demonstrates strong performance, with word cloud visualisations revealing significant terms in reviews.

Price distribution analysis utilises histograms and box plots. Sales rank displays a right-skewed distribution, while the price histogram exhibits a bimodal shape. Notably, sales rank outliers were retained due to their relevance.

In conclusion, this program equips participants with valuable skills in data engineering and data science, demonstrated through effective sentiment analysis models and insights into price and sales rank trends within the Amazon product review dataset.

Contents

1. Introduction

The commencement of the Big Data and Software Engineering (BDSE) minor program initiates an intensive three-week immersion into the domains of data engineering and data science. This period encompasses a series of high-intensity workshops and a consequential individual onboarding assignment. This assignment is structured to facilitate a multifaceted exploration of critical data-related subjects, namely data storage in databases, data cleaning and processing methodologies, machine learning techniques, and data visualisation. Of paramount importance is the practical application of sentiment analysis using various classifiers and the development of interactive data visualisation components (*Onboarding Assignment*, n.d.).

1.1 Assignment

The assignment revolves around the examination of one publicly available dataset from Kaggle, housing Amazon product reviews. Python programming is used for a comprehensive exploration of the data, including data cleansing and transformation to enhance its quality. The research delves into machine learning, entailing the training of at least two machine learning models for sentiment analysis on the reviews. Additionally, a comprehensive report adhering to English language standards is generated. This report describes the data processing design, rationale behind data transformations, and detailed explanations of the applied machine learning algorithms.

2. Methods

2.1 Participants

The participants in this study consist of 111,000 Amazon customers who have provided reviews for various products available on the Amazon platform. These participants were not directly involved in the research process, and their identities remain anonymous. The dataset used for this study is considered secondary data, as it was collected for a different purpose – namely, as part of routine Amazon operations. Therefore, there was no direct interaction with or manipulation of participants.

2.2 Materials

2.2.1 Primary and Secondary Measures

The primary data source for this study is a comprehensive dataset of Amazon reviews, encompassing variables such as reviewer information, product details, review text, ratings, and related information. Additionally, this study relies on secondary data sources, which include official records of product categories and genres provided by Amazon.

2.2.2 Quality of Measurements

The quality of measurements in this study is contingent on the reliability and accuracy of the Amazon review dataset. This secondary data source is considered to be relatively reliable, given its origin as part of Amazon's operational data collection. However, the quality of written reviews may vary, and outlier reviews, if present, could impact the analyses.

2.3 Procedure

2.3.1 Data Collection Methods

Data collection for this study involves obtaining access to the existing dataset of Amazon reviews, which is publicly available for research purposes. No new data collection procedures were employed in this study. As the dataset comprises secondary data, there was no direct involvement or interaction with participants.

2.3.2 Research Design

This study employs a correlational research design to investigate the associations between various variables within the Amazon review dataset. It does not involve experimental manipulations or interventions, as it primarily focuses on observed relationships.

2.3.3 Data Processing and Diagnostics

Prior to analysis, data processing steps include cleaning and preparing the dataset. This involves handling missing values, outlier detection, and data transformation to ensure the data's suitability for analysis. Outliers, if identified, will be addressed appropriately.

Database Storage

For this research, the choice of database storage is SQLite, and this decision is supported by several compelling reasons:

- **Efficient Reads and Writes:** SQLite allows for both reading and writing operations on the same computer. This is advantageous as it facilitates faster querying and analysis, crucial for handling a dataset of this size (Sqlitebrowser, n.d.).
- **Simplicity and Portability:** SQLite is a self-contained, serverless, and zero-configuration database engine. It's easy to set up and manage, making it ideal for research projects. Additionally, SQLite databases are portable and can be easily shared across different platforms (Sqlitebrowser, n.d.).
- **Data Integrity:** SQLite provides features like foreign key support, ensuring data integrity and consistency, especially when dealing with relationships between tables, as seen in the database schema (Sqlitebrowser, n.d.).

Regarding the structure of the database, the CSV columns were dissected into separate tables to enhance data organisation and maintain data integrity. This design choice follows principles of database normalisation, allowing for efficient querying and data management. The tables were created as follows:

- **Reviews Table:** Contains information about individual reviews, including reviewer details, review text, and related data. The "amazon_id" field establishes a foreign key relationship with the "products" table.
- Products Table: Stores information about the products being reviewed, such as title, artist, price, and sales rank. The "root_genre" field establishes a foreign key relationship with the "categories" table.
- **Categories Table:** Contains unique categories extracted from the dataset, ensuring data consistency and making it easier to manage and analyse categorical information.
- **Product_Category Table:** Establishes a many-to-many relationship between products and categories, allowing each product to belong to multiple categories. This structure accommodates the multifaceted nature of products in an online marketplace.

The use of SQLite and this structured table design enables efficient storage, retrieval, and analysis of the dataset, aligning with the research's objectives.

The code snippets for database schema creation can be found in the Appendix, accessible through the following link: 6.1 Sqlite Database Setup.

Dataframe Cleaning and Seeding Database

This section outlines the steps taken to import and clean the dataset obtained from Amazon reviews and describes the process of seeding the data into the SQLite database. The complete code for these procedures can be found in 6.2 Import and clean dataframe, seed database.

The dataset was initially imported using the Pandas library from the provided CSV file. To ensure consistency and alignment with the database schema, several data cleaning operations were applied. Specifically, columns were renamed for uniformity, the "first_release_year" column was converted to an integer data type while addressing missing values, and the "review_time" column was dropped as it was deemed unnecessary for the analysis.

For the database portion, SQLite was chosen as the database management system due to its advantages, including the ability to perform reads and writes on the same computer, facilitating faster querying (Sqlitebrowser, n.d.). The dataset's columns were disassembled and stored in separate tables to create a structured relational database. The cleaning and preparation of the data were crucial to ensure the database's reliability and suitability for subsequent analysis.

2.3.4 Data Analysis Strategy

The data analysis strategy for this research encompasses a variety of quantitative approaches. Specifically, this study employs sentiment analysis to assess the emotional tone of customer reviews, logistic regression, and Support Vector Machine (SVM) classifiers to predict sentiment labels accurately. Additionally, word cloud visualisations are used to provide intuitive representations of the most prevalent terms in both positive and negative reviews (Korab, 2023). Furthermore, a price distribution analysis will be conducted to examine the distribution of product prices within specific categories or genres.

Sentiment Analysis

In this section, we delve into the results of sentiment analysis performed on the review text using machine learning techniques, including logistic regression and SVM classifiers. The primary aim is to offer a comprehensive understanding of the sentiment's nature within the dataset. This encompasses the presentation of essential statistics related to sentiment polarity, sentiment scores, or sentiment categories, including positive, negative, or neutral classifications.

The sentiment analysis begins with preprocessing the text data, which involves tasks such as lowercasing, punctuation removal, tokenization, stopword elimination, and lemmatization. Subsequently, machine learning models, namely logistic regression and SVM, are utilised to predict sentiment labels based on the processed text. These models have been trained on labelled data and are capable of classifying reviews into positive and negative sentiment categories.

Additionally, to provide visual insights into the sentiment distribution, word cloud visualisations are employed to showcase the most frequently occurring words in both positive and negative reviews. These word clouds offer an intuitive representation of the significant terms that contribute to the overall sentiment of customer reviews.

Overall, this sentiment analysis approach combines both quantitative and visual methods to provide a comprehensive understanding of sentiment trends within the dataset, facilitating deeper insights into customer sentiments and preferences.

Price Distribution Analysis

In this section, an analysis has been made into the distribution of product prices and sales ranks within the dataset. The objective is to gain a deeper understanding of these variables, including their price ranges, averages, medians, and spreads.

To visualise the distribution clearly, two types of graphs are utilised: histograms and box plots. Histograms display the frequency of different price or sales rank values, aiding in the identification of common patterns and outliers. Conversely, box plots provide a concise summary of the data, illustrating the central tendency, variability, and any unusual data points (Mcleod, 2023). Outliers are removed from the dataset based on the well-established statistical criterion of the Interquartile Range (IQR) method (Bhandari, 2023). This criterion ensures that extreme data points, which could potentially skew the analysis, are excluded, allowing for a more accurate examination of the core distribution patterns.

The utilisation of both histograms and box plots facilitates a comprehensive exploration of how product prices and sales ranks are distributed in the dataset, enabling meaningful insights to be drawn from the analysis.

4. Results

4.1 Descriptive Statistics

The dataset used in this analysis comprises a total of 111,098 reviews, contributed by 90,040 unique reviewers, and encompasses information on 10,543 distinct products. On average, the products within the dataset are priced at approximately $19.49, and the average review rating stands at 4.47.

The code snippets to generate the descriptive statistics can be found in the Appendix, accessible through the following link: 6.4 Descriptive Statistics

4.2 Sentiment Analysis

4.2.1 Support Vector Machine

The SVM model exhibited strong performance in accurately categorising customer reviews into positive and negative sentiment classes. The classification report reveals the following statistics

Precision: The SVM model achieved a precision of 0.76 for negative sentiment and 0.92 for positive sentiment. This indicates the model's ability to correctly classify reviews, minimising false positives and false negatives.
Recall: The SVM model exhibited a recall of 0.41 for negative sentiment and 0.98 for positive sentiment. High recall values signify the model's capability to capture a significant portion of true positive cases.
F1-Score: The F1-score, a harmonic mean of precision and recall, reached 0.53 for negative sentiment and 0.95 for positive sentiment. These scores reflect a balanced trade-off between precision and recall.

The SVM model achieved an overall accuracy of 0.91 on a dataset of 20,000 reviews, highlighting its effectiveness in sentiment classification. The macro and weighted averages for precision, recall, and F1-score further emphasise the model's robust performance.

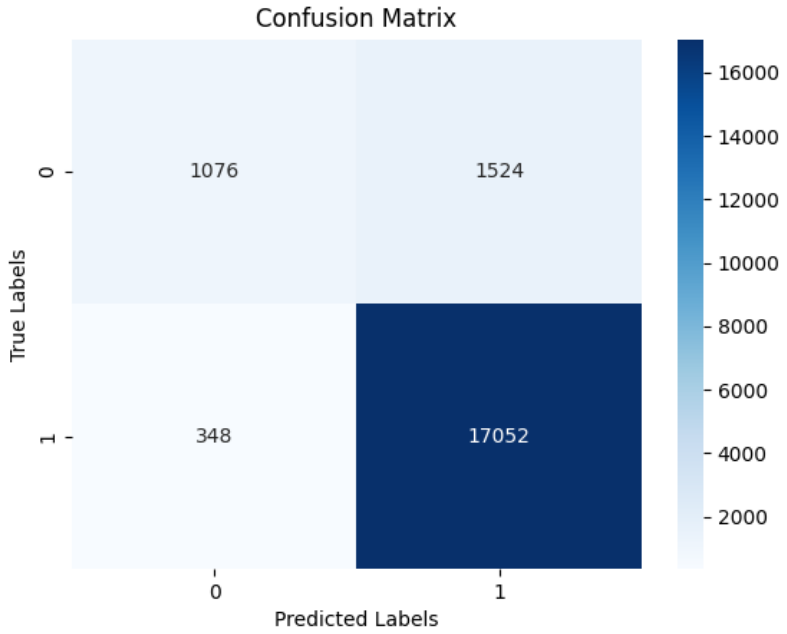

Figure 1: SVM Confusion Matrix

The code snippets to generate the descriptive statistics can be found in the Appendix, accessible through the following link: 6.5 SVM Sentiment Analysis and Confusion Matrix

4.2.2 Logistic Regression

In the sentiment analysis utilising Logistic Regression, the model also demonstrated commendable results. The classification report presents the following key metrics:

Precision: The Logistic Regression model achieved a precision of 0.75 for negative sentiment and an impressive 0.92 for positive sentiment. This indicates the model's effectiveness in minimising classification errors.
Recall: The model exhibited a recall of 0.40 for negative sentiment and 0.98 for positive sentiment. The high recall values signify the model's ability to accurately identify a substantial proportion of positive reviews.
F1-Score: The F1-scores for the Logistic Regression model reached 0.53 for negative sentiment and 0.95 for positive sentiment. These scores reflect a harmonious balance between precision and recall.

With an overall accuracy of 0.91 on a dataset comprising 22,220 reviews, the Logistic Regression model showcases its proficiency in sentiment classification. The macro and weighted averages further validate the model's strong performance.

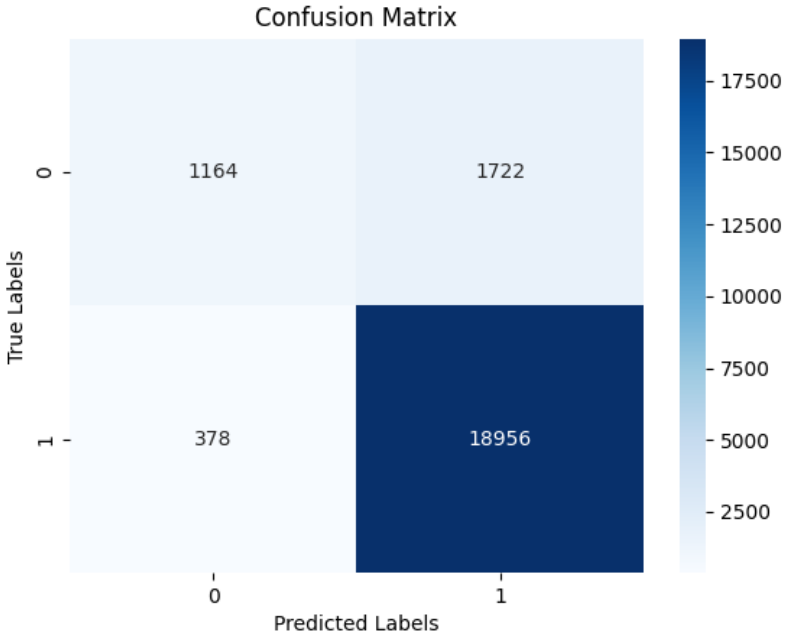

Figure 2: Logistic Regression Confusion Matrix

The code snippets to generate the descriptive statistics can be found in the Appendix, accessible through the following link: 6.6 Logistic Regression Sentiment Analysis and confusion matrix

4.2.3 Word Cloud Analysis

To gain insights into the most prominent terms in positive and negative reviews, word cloud visualisations were generated. These visualisations provide an intuitive representation of the prevalent words associated with each sentiment category.

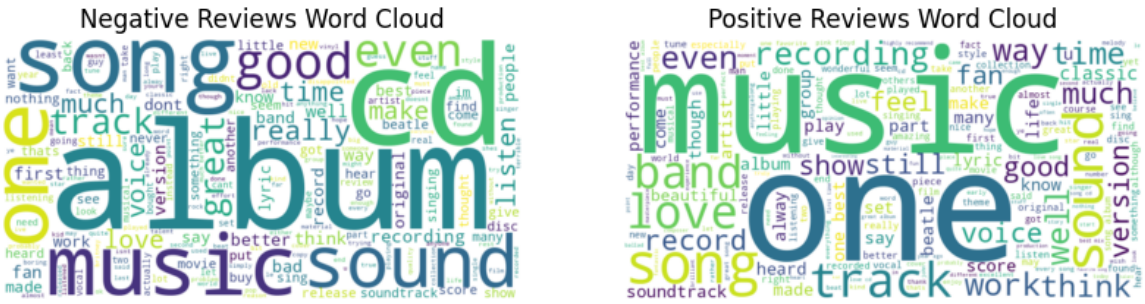

Figure 3: Review Word Clouds

Negative Words: In negative reviews, words such as "Album," "Cd," "Music," "sound," "song," and "one" appear frequently, indicating topics or aspects that might contribute to negative sentiments.

Positive Words: Conversely, in positive reviews, terms like "Music," "One," "Love," "band," "song," and "recording" dominate the word cloud, highlighting aspects that customers found favourable.

4.3 Price Distribution

In the examination of the price distribution within the dataset, boxplots and histograms were used.

The code snippets for the price distribution can be found in the Appendix, accessible through the following link: 6.3 Price Distribution plots.

Sales Rank

The sales rank histogram had a right-skewed distribution, suggesting that a greater number of products had higher sales ranks. The reason outliers were not removed in the sales rank histogram is primarily due to the nature of the sales rank variable and the context of the analysis. Sales rank, as a metric on platforms like Amazon, is typically designed to reflect the relative performance of a product compared to others in the same category. It's a ranking system where a higher number indicates a better rank.

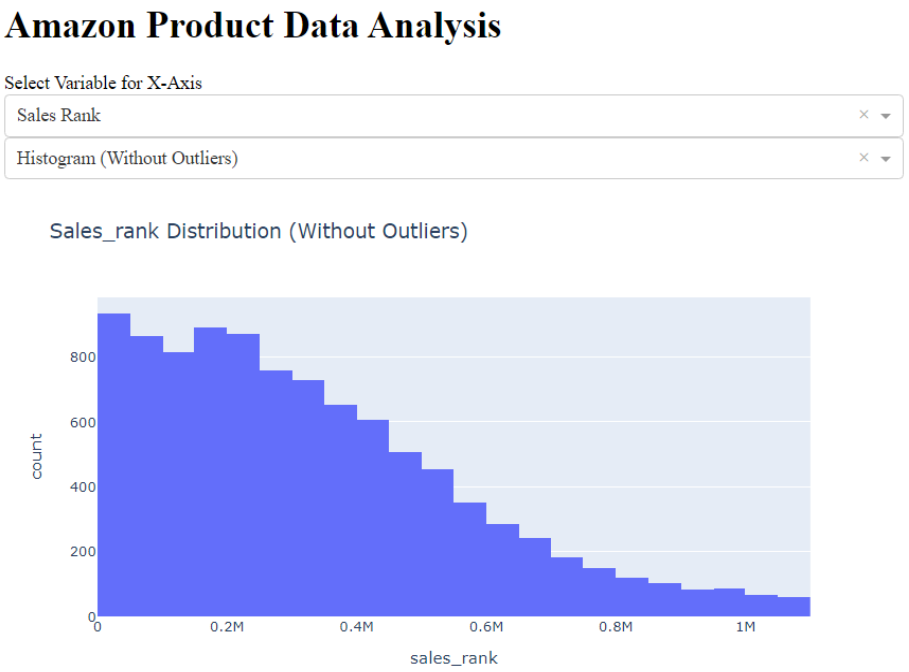

Figure 1: Sales Rank Distribution

Price

The price histogram exhibited a bimodal shape, indicating the presence of two prominent peaks in the distribution. The highest peak in this histogram corresponded to products priced between $8 and $8.99, with a count of 896 products falling within this price range. However, it's important to note that outliers were removed from the price histogram to focus on the central distribution of prices and reduce the impact of extreme values.

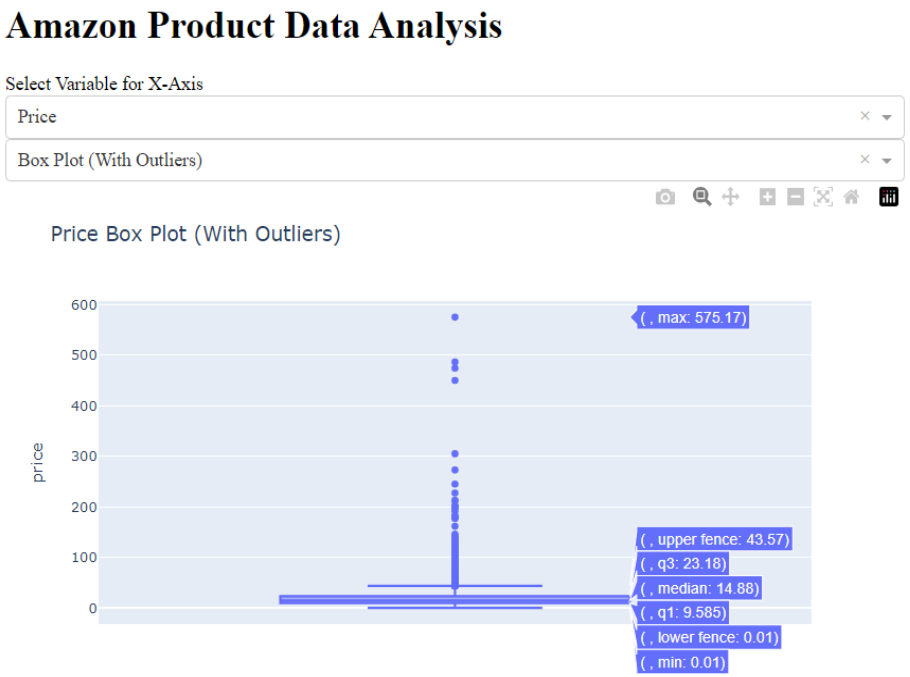

Figure 2: Price Box Plot

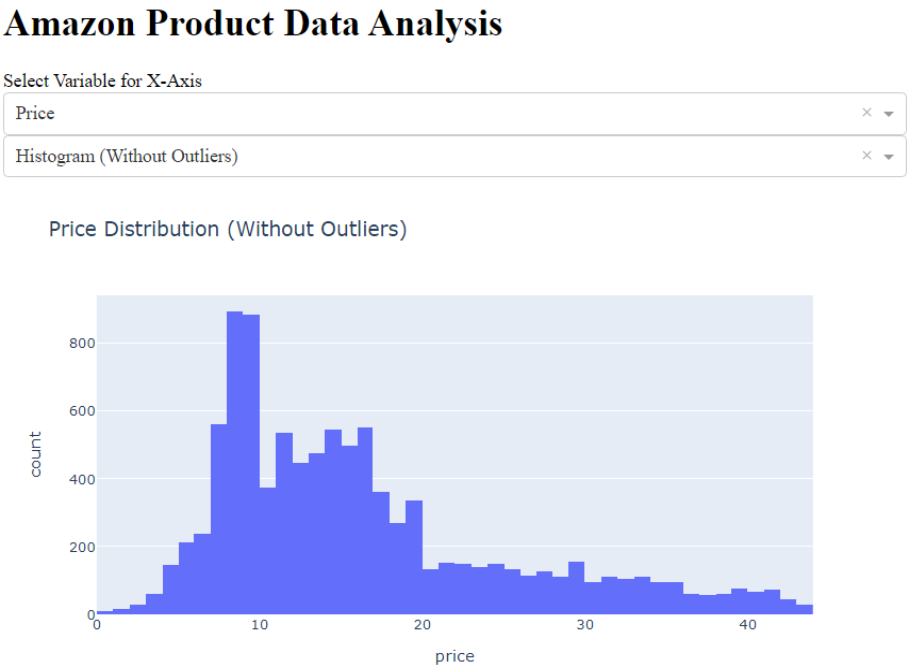

Figure 3: Price Histogram

5. Conclusion

The discussion of findings involves the assessment of the analyses conducted. The dataset comprises 111,098 reviews from 90,040 unique reviewers, covering details about 10,543 different products, with an average product cost of approximately $19.49 and an average review rating of 4.47.

In terms of sentiment analysis, the Support Vector Machine (SVM) model demonstrates strong performance, achieving a precision of 0.76 for negative sentiment and 0.92 for positive sentiment. It also exhibits a recall of 0.41 for negative sentiment and 0.98 for positive sentiment, along with F1-scores of 0.53 for negative sentiment and 0.95 for positive sentiment. The SVM model attains an overall accuracy of 0.91 on a dataset of 20,000 reviews.

Similarly, the Logistic Regression model performs effectively in sentiment analysis, achieving a precision of 0.75 for negative sentiment and 0.92 for positive sentiment. It also exhibits recall values of 0.40 for negative sentiment and 0.98 for positive sentiment, with corresponding F1-scores of 0.53 for negative sentiment and 0.95 for positive sentiment. The Logistic Regression model demonstrates an overall accuracy of 0.91 on a dataset comprising 22,220 reviews.

Word cloud visualisations reveal prevalent terms in both positive and negative reviews.

In the analysis of price distribution, sales rank exhibits a right-skewed distribution, while the price histogram displays a bimodal shape, with a prominent peak for products priced between $8 and $8.99. It's important to note that sales rank outliers were retained due to their relevance, whereas price outliers were removed to focus on the central price distribution.

6. Appendixes

6.1 Sqlite Database Setup

```
import sqlite3
import pandas as pd

conn = sqlite3.connect('amazon_reviews.sqlite3')
conn.execute("PRAGMA foreign_keys = 1")  # Enable foreign key support

create_review_table = """
CREATE TABLE IF NOT EXISTS reviews (
    review_id INTEGER PRIMARY KEY AUTOINCREMENT,
    reviewer_id INTEGER,
    amazon_id INTEGER,
    unix_review_time INTEGER,
    review_text TEXT,
    overall INTEGER,
    helpful TEXT,
    summary TEXT,
    related TEXT,
    FOREIGN KEY (amazon_id) REFERENCES products (amazon_id)
);
"""
create_product_table = """
CREATE TABLE IF NOT EXISTS products (
    amazon_id INTEGER PRIMARY KEY,
```

```
    title INTEGER,
    artist INTEGER,
    label TEXT,
    first_release_year REAL,
    root_genre INTEGER,
    price REAL,
    sales_rank INTEGER,
    songs TEXT,
    FOREIGN KEY (root_genre) REFERENCES categories (category_id)
);
"""
create_categories_table = """
CREATE TABLE IF NOT EXISTS categories (
    category_id INTEGER PRIMARY KEY AUTOINCREMENT,
    category_name TEXT
);
"""
create_product_category_table = """
CREATE TABLE IF NOT EXISTS product_category (
    product_id INTEGER,
    category_id INTEGER,
    PRIMARY KEY (product_id, category_id),
    FOREIGN KEY (product_id) REFERENCES products (amazon_id),
    FOREIGN KEY (category_id) REFERENCES categories (category_id)
);
"""

# Execute the SQL CREATE TABLE statements
conn.execute(create_review_table)
conn.execute(create_product_table)
conn.execute(create_categories_table)
conn.execute(create_product_category_table)
```

6.2 Import and Clean Dataframe, Seed Database

```
import pandas as pd
import sqlite3

# import and clean csv
df = pd.read_csv("../data/amazon_reviews_train.csv")
df = df.rename(columns={'reviewerID': 'reviewer_id', 'amazon-id': 'amazon_id',
'helpful': 'helpful', 'unixReviewTime': 'unix_review_time', 'reviewText':
'review_text','overall': 'overall', 'reviewTime': 'review_time', 'summary':
'summary', 'price': 'price', 'categories': 'categories', 'root-genre':
'root_genre','title': 'title', 'artist': 'artist', 'label':
'label','first-release-year': 'first_release_year', 'songs':
'songs','salesRank': 'sales_rank', 'related': 'related'})

df.drop(columns=['review_time'], inplace=True)
```

```
df['first_release_year'] = pd.to_numeric(df['first_release_year'],
errors='coerce').fillna(9999).astype('int64')

# connect to db
conn = sqlite3.connect('amazon_reviews.sqlite3')
cursor = conn.cursor()

#  Seed categories
categories_lists = df['categories'].apply(lambda x: x.strip("[]").replace("'",
"").split(", "))
unique_categories = set()

for category_list in categories_lists:
    unique_categories.update(category_list)

for category_name in unique_categories:
    cursor.execute("INSERT INTO categories (category_name) VALUES (?)",
(category_name,))

conn.commit()

# Cet dict of categories
cursor.execute("SELECT category_id, category_name FROM categories")
categories_data = cursor.fetchall()
category_id_dict = {category_name: category_id for category_id, category_name in
categories_data}


# Seed products and product_category
unique_amazon_ids_list = df['amazon_id'].unique().tolist()

for amazon_id in unique_amazon_ids_list:
     # Get the first row with the matching amazon_id
    product_info = df[df['amazon_id'] == amazon_id].iloc[0]

    title = int(product_info['title'])
    artist = int(product_info['artist'])
    label = product_info['label']
    first_release_year = int(product_info['first_release_year'])
    price = float(product_info['price'])
    sales_rank = int(product_info['sales_rank'])
    songs = product_info['songs']

    category_name = product_info['root_genre']
    root_genre = int(category_id_dict.get(category_name, None))

    cursor.execute("INSERT INTO products (amazon_id, title, artist, label,
first_release_year, root_genre, price, sales_rank, songs) VALUES (?, ?, ?, ?, ?,
?, ?, ?, ?)",
                   (amazon_id, title, artist, label, first_release_year,
root_genre, price, sales_rank, songs))
```

```
    categories = product_info['categories']
    category_list = [category.strip("[]").replace("'", "") for category in
categories.split(", ")]

    for category_name_in_list in category_list:
        category_id = category_id_dict.get(category_name_in_list, None)
        if category_id is not None:
            cursor.execute("INSERT INTO product_category (product_id,
category_id) VALUES (?, ?)",
                           (amazon_id, category_id))

conn.commit()

# Seed reviews

for index, row in df.iterrows():
    reviewer_id = row['reviewer_id']
    amazon_id = row['amazon_id']
    unix_review_time = row['unix_review_time']
    review_text = row['review_text']
    overall = row['overall']
    helpful = row['helpful']
    summary = row['summary']
    related = row['related']

    cursor.execute("INSERT INTO reviews (reviewer_id, amazon_id,
unix_review_time, review_text, overall, helpful, summary, related) VALUES (?, ?,
?, ?, ?, ?, ?, ?)",
                   (reviewer_id, amazon_id, unix_review_time, review_text,
overall, helpful, summary, related))

# Commit the changes and close the database connection
conn.commit()
conn.close()
```

6.3 Price Distribution Plots

```
import sqlite3
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import sqlalchemy
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output

engine = sqlalchemy.create_engine('sqlite:///amazon_reviews.sqlite3')
```

```
products_df = pd.read_sql_table('products', engine)

def remove_outliers(df, variable):
    Q1 = df[variable].quantile(0.25)
    Q3 = df[variable].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[variable] >= lower_bound) & (df[variable] <= upper_bound)]

app = dash.Dash(__name__)

app.layout = html.Div([
    html.H1("Amazon Product Data Analysis"),
    html.Label("Select Variable for X-Axis"),
    dcc.Dropdown(
        id='x-axis-variable',
        options=[
            {'label': 'Price', 'value': 'price'},
            {'label': 'Sales Rank', 'value': 'sales_rank'}
        ],
        value='price'
    ),
    dcc.Dropdown(
        id='plot-type',
        options=[
            {'label': 'Histogram (Without Outliers)', 'value': 'histogram_without_outliers'},
            {'label': 'Box Plot (With Outliers)', 'value': 'boxplot_with_outliers'}
        ],
        value='histogram_without_outliers'
    ),
    dcc.Graph(id='plot')
])

@app.callback(
    Output('plot', 'figure'),
    Input('x-axis-variable', 'value'),
    Input('plot-type', 'value')
)
def update_plot(selected_variable, plot_type):
    if plot_type == 'histogram_without_outliers':
        products_df_filtered = remove_outliers(products_df, selected_variable)
        fig = px.histogram(
            products_df_filtered,
            x=selected_variable,
            title=f'{selected_variable.capitalize()} Distribution (Without Outliers)',
            labels={'value': selected_variable, 'count': 'Frequency'},
            nbins=50
        )
    elif plot_type == 'boxplot_with_outliers':
        fig = px.box(
```

```
            products_df,
            y=selected_variable,
            title=f'{selected_variable.capitalize()} Box Plot (With Outliers)',
            labels={'value': selected_variable}
        )

    return fig

if __name__ == '__main__':
    app.run_server(debug=True)
```

6.4 Descriptive Statistics

```
import sqlite3
import pandas as pd

conn = sqlite3.connect('amazon_reviews.sqlite3')
reviews_df = pd.read_sql_query("SELECT * FROM reviews", conn)
products_df = pd.read_sql_query("SELECT * FROM products", conn)

num_reviews = len(reviews_df)
num_reviewers = reviews_df['reviewer_id'].nunique()  # Count of unique reviewer_id values
num_products = len(products_df)

average_price = products_df['price'].mean()
average_review_rating = reviews_df['overall'].mean()

print("Overview of the Dataset:")
print(f"Number of Reviews: {num_reviews}")
print(f"Number of Unique Reviewers: {num_reviewers}")
print(f"Number of Products: {num_products}")
print(f"Average Price: ${average_price:.2f}")
print(f"Average Review Rating: {average_review_rating:.2f}")
```

6.5 SVM Sentiment Analysis and Confusion Matrix

```
import sqlite3
import pandas as pd
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score,
```

```
confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

conn = sqlite3.connect('amazon_reviews.sqlite3')
df = pd.read_sql_query("SELECT * FROM reviews ORDER BY RANDOM() LIMIT 100000",
conn)

def preprocess_text(text):
    if text is None:
        return ""
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    words = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]
    words = [word for word in words if word.isalpha()]
    cleaned_text = ' '.join(words)
    return cleaned_text

df['cleaned_text'] = df['review_text'].apply(preprocess_text)
df['sentiment'] = df['overall'].apply(lambda x: 1 if x in [4, 5] else 0)

X_train, X_test, y_train, y_test = train_test_split(df['cleaned_text'],
df['sentiment'], test_size=0.2, random_state=42)

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train_tfidf, y_train)

y_pred = svm_classifier.predict(X_test_tfidf)

accuracy = accuracy_score(y_test, y_pred)
classification_report_str = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}\n")
print("Classification Report:\n")
print(classification_report_str)

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

6.6 Logistic Regression Sentiment Analysis and Confusion Matrix

```
import sqlite3
import pandas as pd
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
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 classification_report, accuracy_score
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import seaborn as sns


conn = sqlite3.connect('amazon_reviews.sqlite3')
df = pd.read_sql_query("SELECT * FROM reviews", conn)

def preprocess_text(text):
    if text is None:
        return ""

    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    words = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]
    lemmatizer = WordNetLemmatizer()
    words = [lemmatizer.lemmatize(word) for word in words]
    words = [word for word in words if word.isalpha()]
    cleaned_text = ' '.join(words)
    return cleaned_text

df['cleaned_text'] = df['review_text'].apply(preprocess_text)
df['sentiment'] = df['overall'].apply(lambda x: 1 if x in [4, 5] else 0)

X_train, X_test, y_train, y_test = train_test_split(df['cleaned_text'],
df['sentiment'], test_size=0.2, random_state=42)

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

logistic_regression = LogisticRegression()
logistic_regression.fit(X_train_tfidf, y_train)
```

```
y_pred = logistic_regression.predict(X_test_tfidf)

accuracy = accuracy_score(y_test, y_pred)
classification_report_str = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy:.2f}\n")
print("Classification Report:\n")
print(classification_report_str)

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

7. References

Bhandari, P. (2023). How to find interquartile range (IQR) | Calculator & examples. *Scribbr*. https://www.scribbr.co.uk/stats/interquartile-range-meaning/

Korab, P. (2023, September 12). Guide to Using Word Clouds for Applied Research Design. *Medium*. https://towardsdatascience.com/guide-to-using-word-clouds-for-applied-research-design-2e07a6a1a513

Mcleod, S., PhD. (2023). Box Plot Explained: Interpretation, Examples, & Comparison. *Simply Psychology*. https://www.simplypsychology.org/boxplots.html

Onboarding Assignment. (n.d.). https://dlo.mijnhva.nl/d2l/le/content/537119/Home

Sqlitebrowser. (n.d.). *Home*. GitHub. https://github.com/sqlitebrowser/sqlitebrowser/wiki