UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+
ACCREDITED UNIVERSITY

Case Study on

EDA on Online Retail Dataset

24CAT-612 : Data Mining

2024-2025

Submitted By:

Name: Sukhveer Singh
UID: 24MCI10029
Branch: MCA (AI/ML)
Section/Group: 1/A

Submitted To:

Dr. Disha Handa

Acknowledgement

I would like to express my heartfelt gratitude to Dr. Disha Handa for her invaluable guidance and support throughout this case study on data cleaning in e-commerce customer analytics. Her insights and expertise have greatly enriched my understanding of the subject matter, guiding the direction of this research. I appreciate her encouragement in exploring advanced data cleaning techniques and ensuring data integrity for reliable analytics. This study would not have been possible without her mentorship and unwavering support.

I am also grateful to my peers and colleagues for their collaboration and encouragement. Their willingness to share knowledge has enriched my learning experience, making this project possible through collective effort.

UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Table of Contents

Introduction

Problem Statement

Relevant Dataset

Data Cleaning Techniques

Exploratory Data Analysis (EDA)

Findings and Insights

Graphical Representation

Future Enhancements

Introduction

In the fast-paced world of e-commerce, companies are no longer just platforms for shopping—they are intricate networks of customer engagement, driven by vast oceans of data. Every action, from a simple click to a completed purchase or an abandoned cart, provides valuable glimpses into customer preferences, emerging trends, and potential market shifts. However, these insights often remain hidden, buried within an overwhelming volume of data that can be difficult to navigate without structured analysis.

Exploratory Data Analysis (EDA) is the gateway to unveiling these hidden patterns. Through EDA, businesses can transform raw data into meaningful insights, revealing behavioral trends, identifying anomalies, and discovering subtle relationships across their customer interactions. For an e-commerce business, these insights are invaluable—they enable the personalization of customer journeys, optimization of inventory management, and refinement of targeted marketing strategies. By revealing the underlying dynamics of shopping behavior, EDA empowers companies to make informed, data-driven decisions.

2. Problem Statement

An e-commerce company, XYZ, aims to enhance customer experience and boost sales through data-driven insights. As the volume of transaction, interaction, and feedback data grows, inconsistencies, duplicates, and errors emerge, leading to unreliable analytics. Our objective is to apply data cleaning and EDA techniques on a retail dataset to identify patterns and draw actionable insights.

3. Relevant Dataset

Dataset Source: [Customer Shopping Dataset - Retail Sales Data | Kaggle]

The dataset includes anonymized online retail transaction data, capturing customer demographics, order details, and product information. It covers 10 shopping malls across Istanbul from 2021 to 2023, providing a comprehensive view of shopping habits. This dataset can help analyze sales trends, customer behaviors, and order management processes.

Attribute Information:

- **invoice_no**: Unique invoice identifier (nominal)
- **customer_id**: Unique customer identifier (nominal)
- **gender**: Customer's gender
- **age**: Customer's age
- **category**: Product category purchased
- **quantity**: Product quantity in the transaction
- **price**: Price per unit in Turkish Liras (TL)
- **payment_method**: Payment method used (cash, credit card, or debit card)
- **invoice_date**: Date of transaction
- **shopping_mall**: Location of transaction

4. Data Cleaning Techniques

To ensure data integrity and enhance analysis accuracy, the following data cleaning techniques are applied:

```
daa > data_mining.py > ...
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv("C:\\Users\\my\\Downloads\\ecommerce_sales_analysis.csv")
# Displaying Dataset Overview
```

```python
# 2. Show the column names and data types
print("Column Names and Data Types:")
print(df.dtypes)
print("\n")

# 3. Display basic statistics for numerical columns
print("Summary Statistics for Numerical Columns:")
print(df.describe())
print("\n")

# 4. Calculate and display the mean for specific columns
mean_price = df['price'].mean()
mean_review_score = df['review_score'].mean()
mean_review_count = df['review_count'].mean()

print("Mean Values:")
print(f"Mean Price: {mean_price:.2f}")
```

```
Dimensions of the dataset:
Rows: 1000, Columns: 18
```

```
Column Names and Data Types:
product_id          int64
product_name       object
category           object
price             float64
review_score      float64
review_count        int64
sales_month_1       int64
sales_month_2       int64
sales_month_3       int64
sales_month_4       int64
sales_month_5       int64
sales_month_6       int64
sales_month_7       int64
sales_month_8       int64
sales_month_9       int64
sales_month_10      int64
sales_month_11      int64
sales_month_12      int64
dtype: object
```

```
Summary Statistics for Numerical Columns:
         product_id        price  review_score  review_count  ...  sales_month_9  sales_month_10  sales_month_11  sales_month_12
count   1000.000000  1000.000000   1000.000000   1000.000000  ...    1000.000000     1000.000000      1000.00000     1000.000000
mean     500.500000   247.677130      3.027600    526.506000  ...     491.934000      514.798000       505.83800      500.386000
std      288.819436   144.607983      1.171243    282.269932  ...     287.514731      288.710119       288.82451      278.509459
min        1.000000     7.290000      1.000000      1.000000  ...       0.000000        1.000000         0.00000        4.000000
25%      250.750000   121.810000      2.000000    283.750000  ...     247.250000      267.000000       251.25000      259.000000
50%      500.500000   250.920000      3.100000    543.000000  ...     495.500000      532.000000       502.00000      500.500000
75%      750.250000   373.435000      4.000000    772.000000  ...     735.250000      770.250000       761.00000      730.000000
max     1000.000000   499.860000      5.000000    999.000000  ...    1000.000000     1000.000000      1000.00000     1000.000000

[8 rows x 16 columns]
```

```
Mean Values:
Mean Price: 247.68
Mean Review Score: 3.03
Mean Review Count: 526.51
```

1. Handling Missing Values

- **Explanation**: Missing values can bias results and reduce the dataset's reliability.
    - **Techniques Applied**:
    - Drop rows where essential attributes (e.g., customer_id) are missing.
    - Impute missing values in non-essential columns (e.g., product descriptions) using placeholders if they constitute a small portion of data.

```python
# 5. Checking for missing values in each column
print("Missing Values per Column:")
print(df.isnull().sum())
print("\n")
```

```
Missing Values per Column:
product_id         0
product_name       0
category           0
price              0
review_score       0
review_count       0
sales_month_1      0
sales_month_2      0
sales_month_3      0
sales_month_4      0
sales_month_5      0
sales_month_6      0
sales_month_7      0
sales_month_8      0
sales_month_9      0
sales_month_10     0
sales_month_11     0
sales_month_12     0
dtype: int64
```

```
# Data Cleaning and Transformation Steps

# 1. Handling Missing Values: Drop rows with missing 'product_id' (assumed essential) and fill missing 'product_n
df = df.dropna(subset=['product_id'])
df['product_name'].fillna('Unknown Product', inplace=True)
```

b. Removing Duplicates

- **Explanation**: Duplicate records inflate metrics and distort insights.
- **Technique Applied**: Use the drop_duplicates() function to eliminate duplicate rows.

```
# 2. Removing Duplicates: Drop any duplicate rows
df = df.drop_duplicates()
```

c. Data Type Conversion

- **Explanation**: Correct data types enable more accurate analyses, especially with dates and quantities.
- **Techniques Applied**:
  - Convert invoice_date to a datetime format for time-based analysis.
  - Ensure quantity and price are numeric to facilitate calculations.

```
# 5. Standardizing Categorical Data: Ensure 'category' values are consistent by standardizing text format
df['category'] = df['category'].str.strip().str.lower()
```

d. Outlier Detection

- **Explanation**: Outliers can skew analysis and lead to misleading conclusions.
- **Technique Applied**: Use the **interquartile range (IQR)** method to detect and handle outliers in quantity and price, either by removing or capping based on business rules.

```
# 4. Outlier Detection: Identify and handle outliers in 'price' using IQR
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1
df = df[(df['price'] >= Q1 - 1.5 * IQR) & (df['price'] <= Q3 + 1.5 * IQR)]

```

e. Standardizing Categorical Data

- **Explanation**: Consistent categorical data ensures reliable grouping and analysis.
- **Technique Applied**: Standardize values in fields like payment_method and gender to a consistent format.

6. Findings and Insights

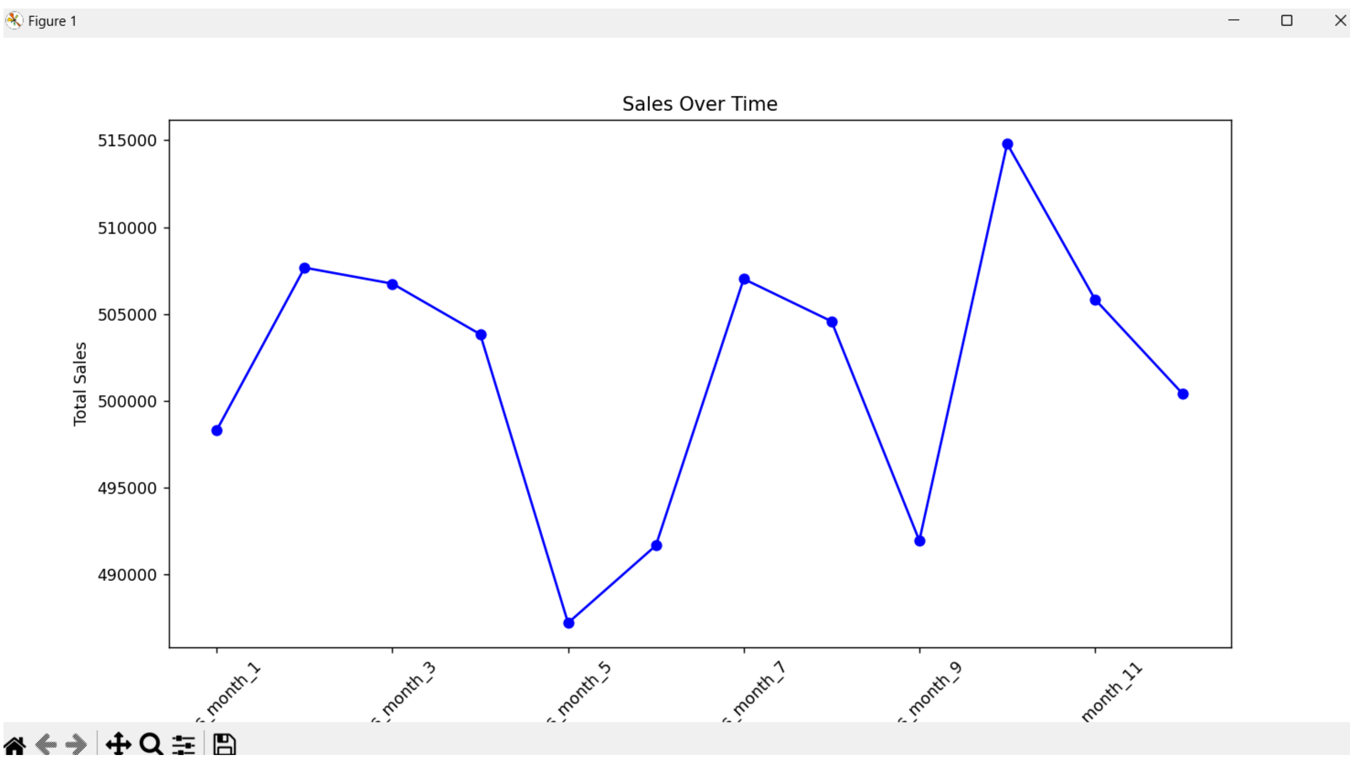
- **Customer Demographics**: Younger age groups may show higher purchasing frequency.
- **Product Preferences**: Certain categories are more popular, suggesting high-demand items.
- **Payment Preferences**: Majority of transactions through credit card payments, indicating customer preference for credit purchases.
- **Seasonal Trends**: Peak sales periods align with holidays and sales events, indicating optimal times for promotions.

7. Graphical Representation

To visualize insights effectively, several plots are used:

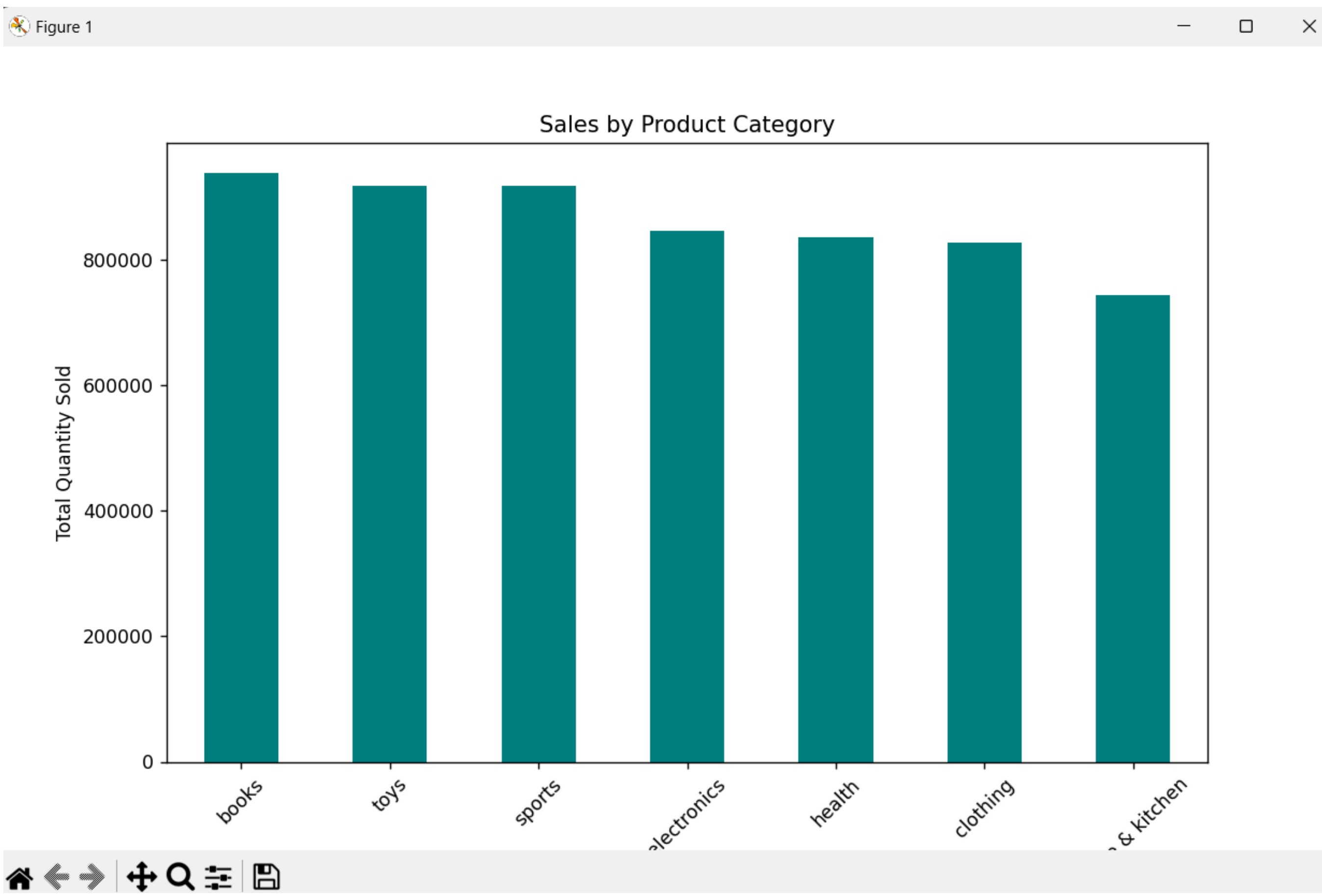
1. **Sales Over Time**: A line plot illustrating monthly or weekly sales trends to identify seasonality.The line plot of monthly sales shows how sales trends fluctuate over the year, helping identify any seasonality or peaks in demand. Periods of higher sales may indicate popular shopping seasons, promotional events, or holiday effects.

```python
# 1. Sales Over Time: Line plot of monthly sales
monthly_sales = df[sales_columns].sum()

plt.figure(figsize=(12, 6))
monthly_sales.plot(kind='line', marker='o', color='b')
plt.title('Sales Over Time')
plt.xlabel('Months')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.show()
```



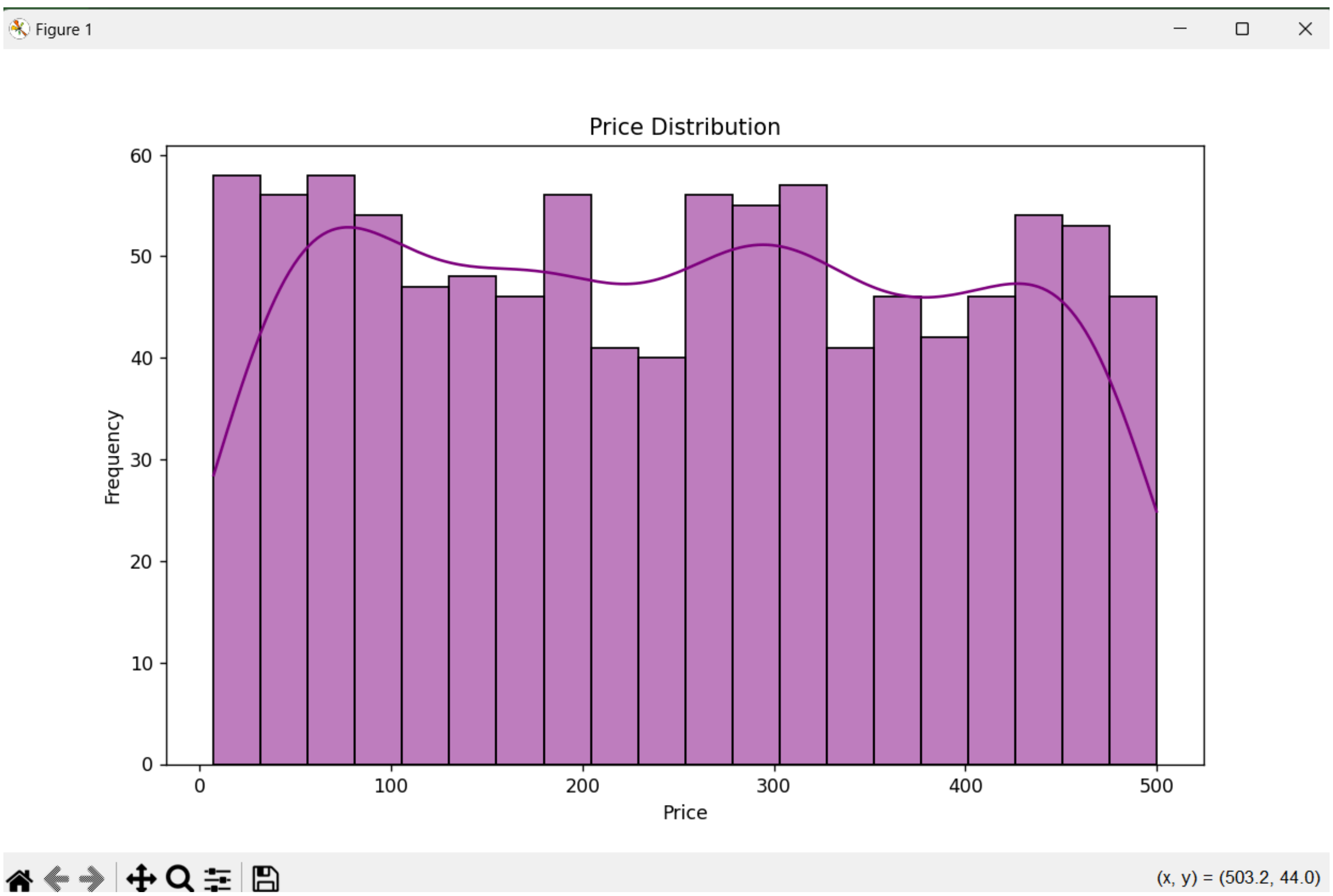
2. **Category Popularity**: A bar chart to show sales across product categories.The bar chart displays total sales across different product categories, showing which categories are most popular among customers. Categories with higher sales can guide inventory decisions and highlight areas to focus marketing efforts.

```python
# 2. Category Popularity: Bar chart for sales across categories
category_sales = df.groupby('category')[sales_columns].sum().sum(axis=1).sort_values(ascending=False)

plt.figure(figsize=(10, 6))
category_sales.plot(kind='bar', color='teal')
plt.title('Sales by Product Category')
plt.xlabel('Product Category')
plt.ylabel('Total Quantity Sold')
plt.xticks(rotation=45)
plt.show()
```

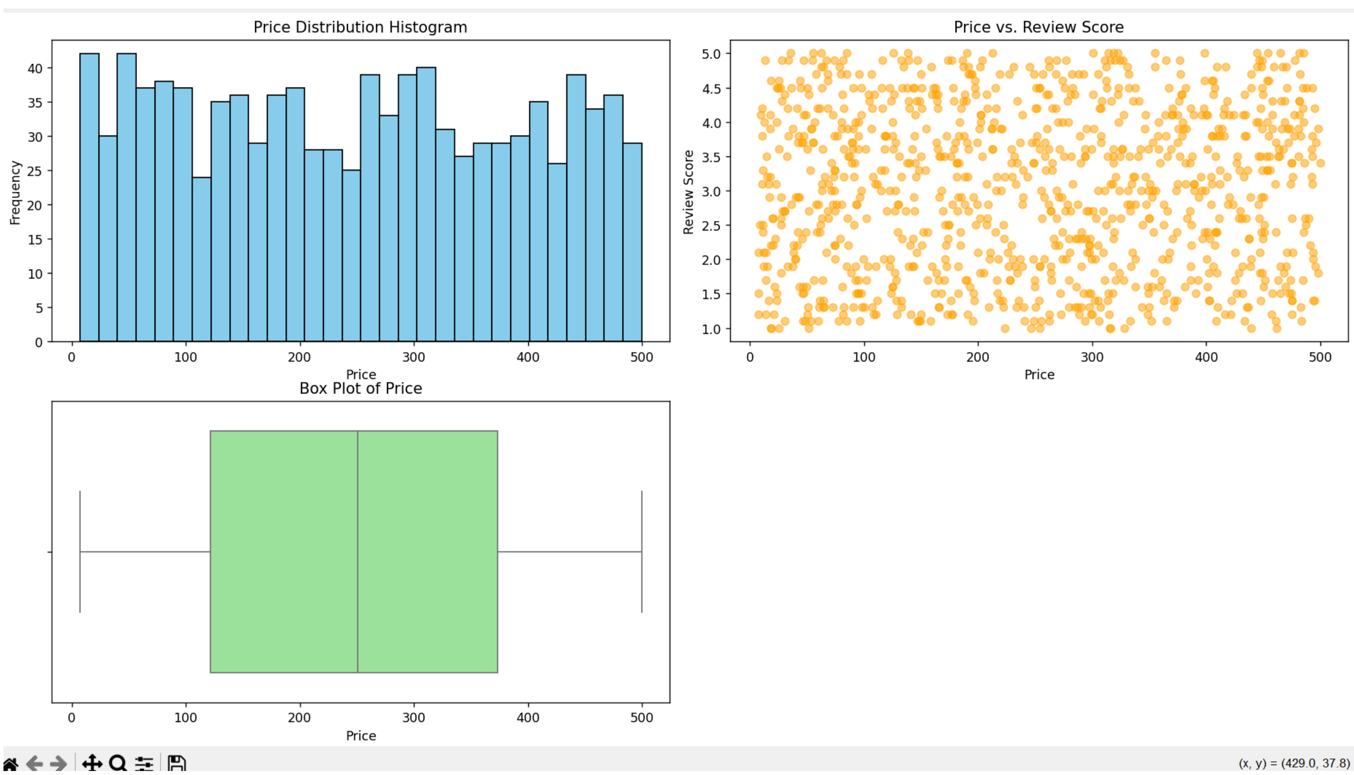


3. **Customer Age Distribution**: Histogram to display age demographics.The histogram of prices gives a sense of how products are priced overall. Peaks in this distribution might indicate common price ranges, while any long tail could show high-end products or potential outliers. The accompanying box plot provides further insight into the spread of prices and helps identify any outliers, ensuring the pricing strategy is well-calibrated.

```
# 3. Price Distribution: Histogram and box plot for 'price'
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], bins=20, kde=True, color='purple')
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()
```

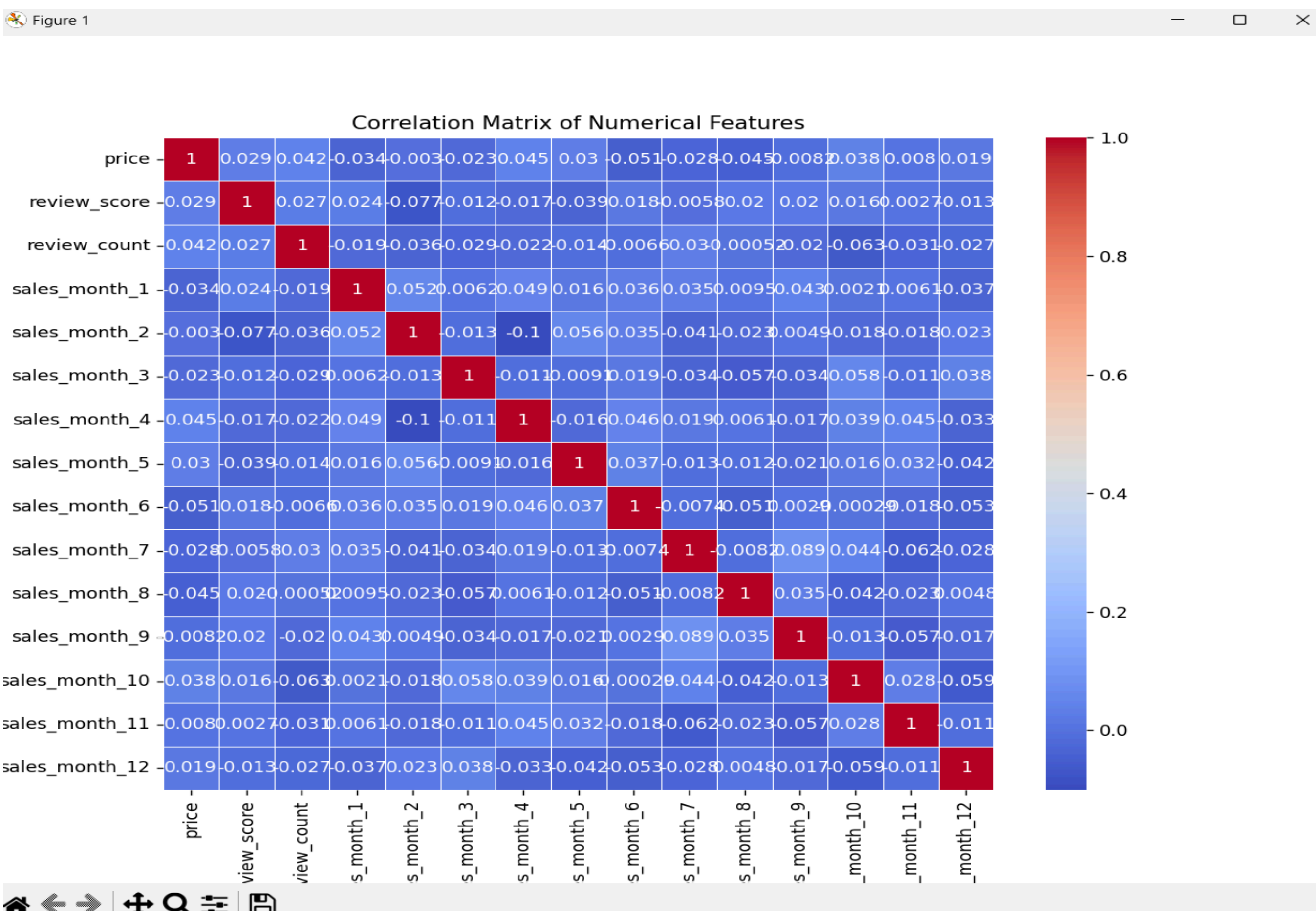


4. **Price Metrics**: Scatter plots and box plots for price distribution, helping detect pricing outliers.The histogram of prices gives a sense of how products are priced overall. Peaks in this distribution might indicate common price ranges, while any long tail could show high-end products or potential outliers. The accompanying box plot provides further insight into the spread of prices and helps identify any outliers, ensuring the pricing strategy is well-calibrated.



6. Correlation Matrix: A heatmap to show correlations between numerical variables, revealing relationships that inform model building.The heatmap shows correlations between numerical variables, such as price, review scores, review count, and monthly sales. Strong correlations (positive or negative) can reveal relationships; for instance, a correlation between review scores and sales could indicate the impact of product quality on customer purchases.

```python

# 4. Correlation Matrix: Heatmap for numerical features
# Select numerical columns for correlation matrix (excluding sales_month columns if needed)
numerical_cols = ['price', 'review_score', 'review_count'] + sales_columns
correlation_matrix = df[numerical_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of Numerical Features')
plt.show()
```



8. Future Enhancements

1. **Advanced Analytics**:
    - **Predictive Modeling**: Use machine learning to predict next purchases or churn rates.
    - **Sentiment Analysis**: Apply NLP to customer feedback to gauge satisfaction levels.
2. **Real-Time Processing**:
    - **Data Pipelines**: Create real-time data ingestion for dynamic insights.
    - **Interactive Dashboards**: Build dashboards for KPIs, offering real-time monitoring.
3. **Enhanced Segmentation**:
    - **Behavioral Segmentation**: Use clustering to segment customers by behavior and preferences, enabling targeted marketing.
    - **Personalization**: Implement personalized recommendations to boost engagement.
4. **Expanded Data Sources**:
    - **Cross-Platform Data**: Integrate social media and support data for a comprehensive view of customer interactions.
    - **API Integration**: Leverage logistics and payment processor APIs to track delivery times and payment methods.
5. **Improved Reporting and Inventory Management**:
    - **Automated Reports**: Set up automated reporting for routine insights on sales and operations.
    - **Demand Forecasting**: Use historical data to forecast demand, reducing overstock and optimizing supply chains.
6. **Customer Loyalty Programs**:
    - **Loyalty Analytics**: Track the impact of loyalty programs on engagement and retention.
    - **Gamification**: Introduce gamified shopping elements to encourage repeat purchases.

---

9. Github link:

https://github.com/suhkv/Data-Mining-case-study