Fundamentals of Data Science project.

Supervised by:

- Dr.Zaher Salah
-

Done by:

- Shahed Al Zu'bi (2137097)
- Raghad Jamal (2137599)
- Login Jamal (2144570)
- Mera Akram (2140136)

Introduction:

In the dynamic landscape of information, data science stands as a linchpin for extracting valuable insights from vast datasets. This assignment navigates the advanced facets of data science, emphasizing exploratory data analysis, statistical modeling, and machine learning. Leveraging tools like Python and R, we dive into real-world applications, from preprocessing to deploying intricate algorithms. Ethical considerations are woven throughout, highlighting the responsible use of data.

Data exploration and preparation:

Effective data exploration and preparation are essential for building accurate and robust models. It helps analysts and data scientists to better understand their data, address potential issues, and create a clean, structured dataset that is ready for analysis or machine learning applications.

Our dataset (Scoring-Dataset-9) contain 10 attributes from various data types:

A1.

1. User_id: Numeric (Ratio).
2. Gender: Binary (Symmetric).
3. Age: Numeric (Ratio).
4. Marital_status: Binary (Asymmetric) / Nominal.
5. Website_activity: Nominal.
6. Browsed_Electronics_12Mo: Binary(Asymmetric).
7. Bought_Electronics_12Mo: Binary(Asymmetric).
8. Bought_Digital_Media_18Mo: Binary(Asymmetric).
9. Bought_Digital_Books: Binary(Asymmetric).
10. Payment_Method: Nominal.

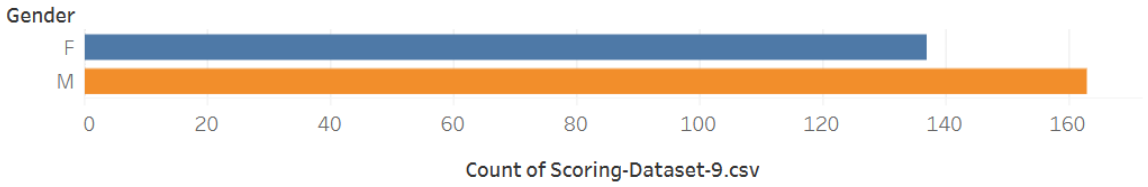
Our meticulous attribute analysis has unveiled the inherent characteristics of each variable, laying the groundwork for subsequent advanced analytics and modeling endeavors. The identification of attribute types provides a crucial foundation for shaping the trajectory of our data science journey, facilitating informed decisions and actionable insights. As we navigate the data landscape, our understanding deepens, paving the way for a comprehensive and impactful exploration of the dataset.

A2.

In this part we took each attribute to analyze and visualize it, tried to find patterns or relationships between the attributes using python codes and Tableau visualization tool.

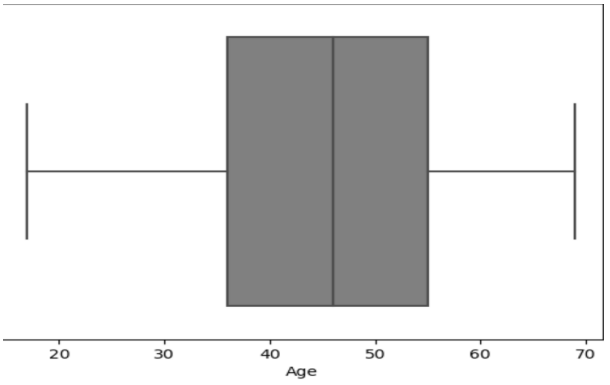
- Gender attribute:

Gender:

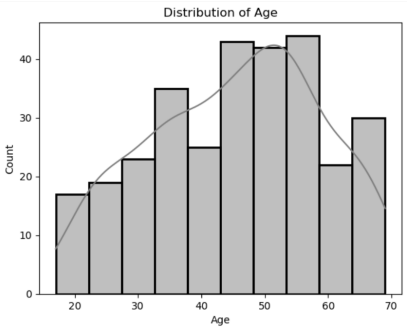


- Age attribute:
we used this blocks of code to see boxplot, distribution plot and bar plot results:

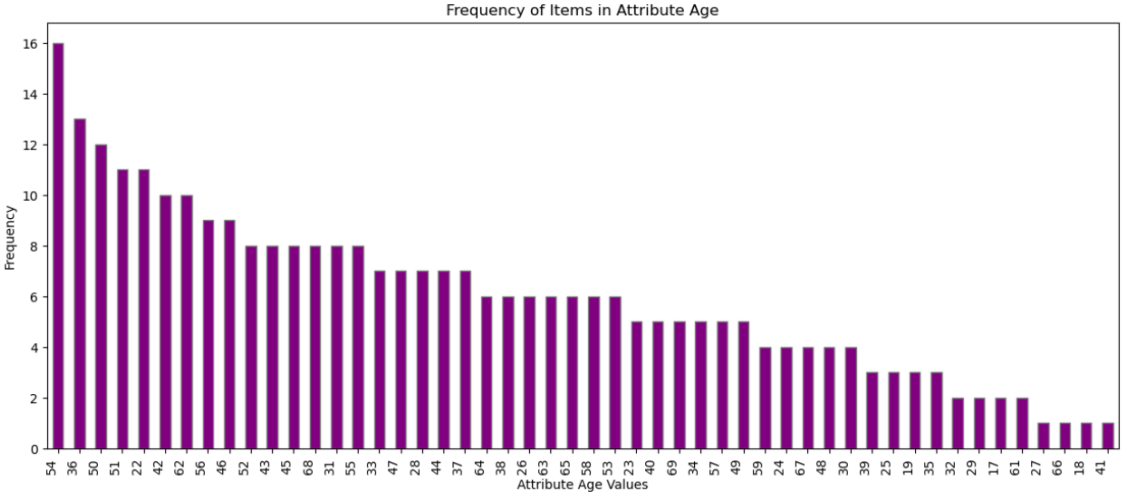
```
# Box plot
sns.boxplot(x=df['Age'], color='gray')
plt.title('Boxplot of Age')
plt.show()
```



```python
# Distribution plot
sns.histplot(df['Age'], kde=True, color='gray', linewidth=2)
plt.title('Distribution of Age')
plt.show()
```




```python
# Frequency of values:
value_counts = df['Age'].value_counts()
plt.figure(figsize=(15, 6))
value_counts.plot(kind='bar', color='purple', edgecolor='gray')
plt.title('Frequency of Items in Attribute Age')
plt.xlabel('Attribute Age Values')
plt.ylabel('Frequency')
plt.xticks(rotation=90, ha='right')
plt.show()
```

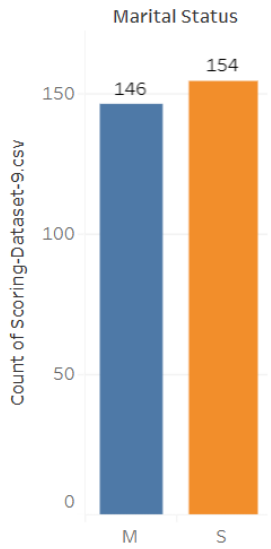



```python
# Print the results:
print("Max:", df['Age'].max())
print("Min:", df['Age'].min())
print("\nCentral tendency:")
print("Median:", df['Age'].median())
print("Mean:", df['Age'].mean())
print("Mode:", df['Age'].mode())
print("Variance:", df['Age'].var())
print("std:",math.sqrt( df['Age'].var()))
print("Percentiles [25th, 50th, 75th]:", percentiles)
```

Max: 69
Min: 17
Central tendency:
Median: 46.0
Mean: 45.42
Mode: 54
Name: Age, dtype: int64
Variance: 181
std: 13.45
Percentiles [25th, 50th, 75th]: [36. 46. 55.]

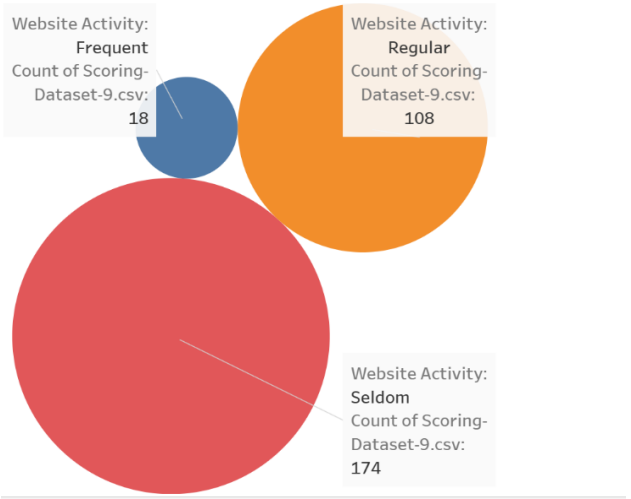
- Marital status attribute:

Tableau result:



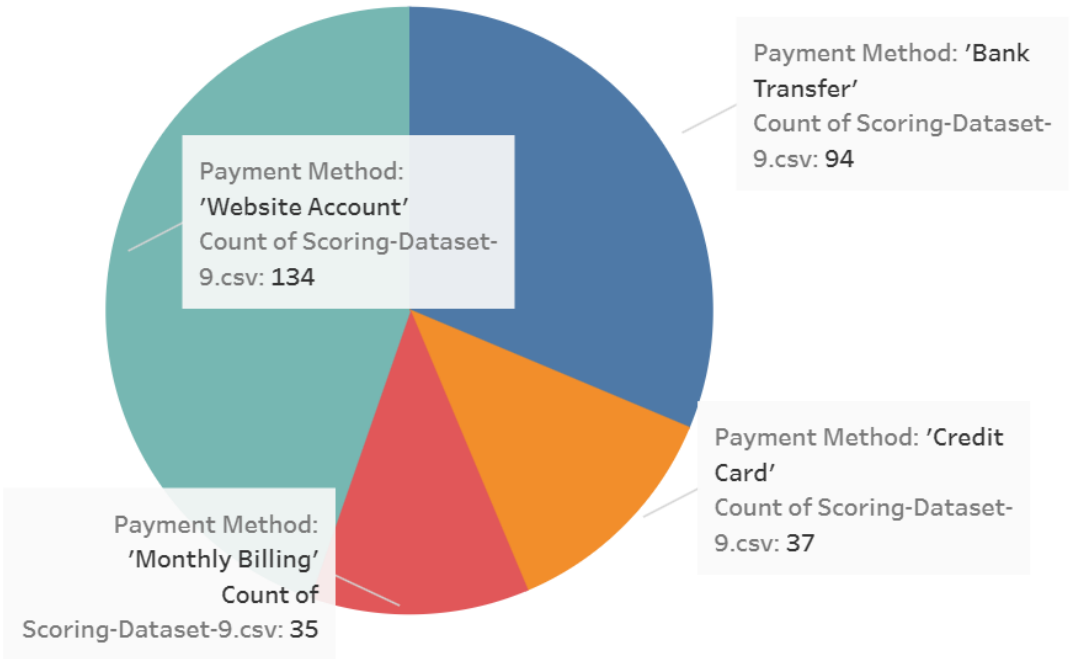
- Web site attribute:

Tableau result:

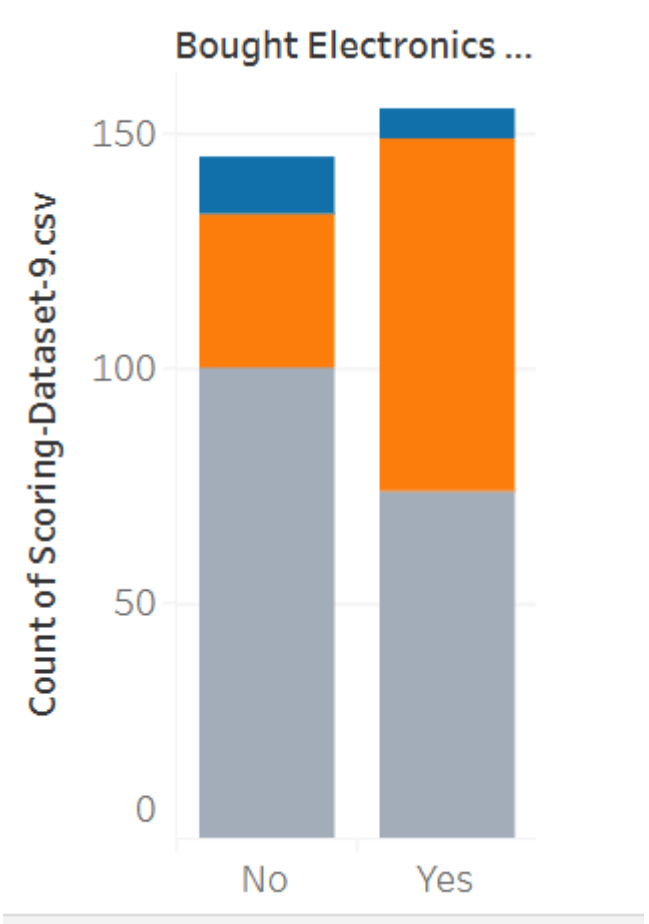


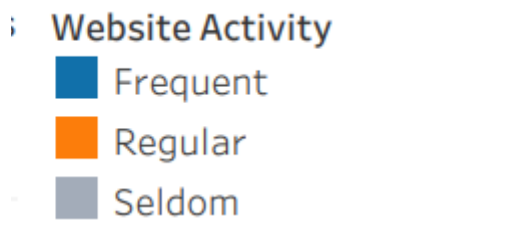
- Marital status attribute:

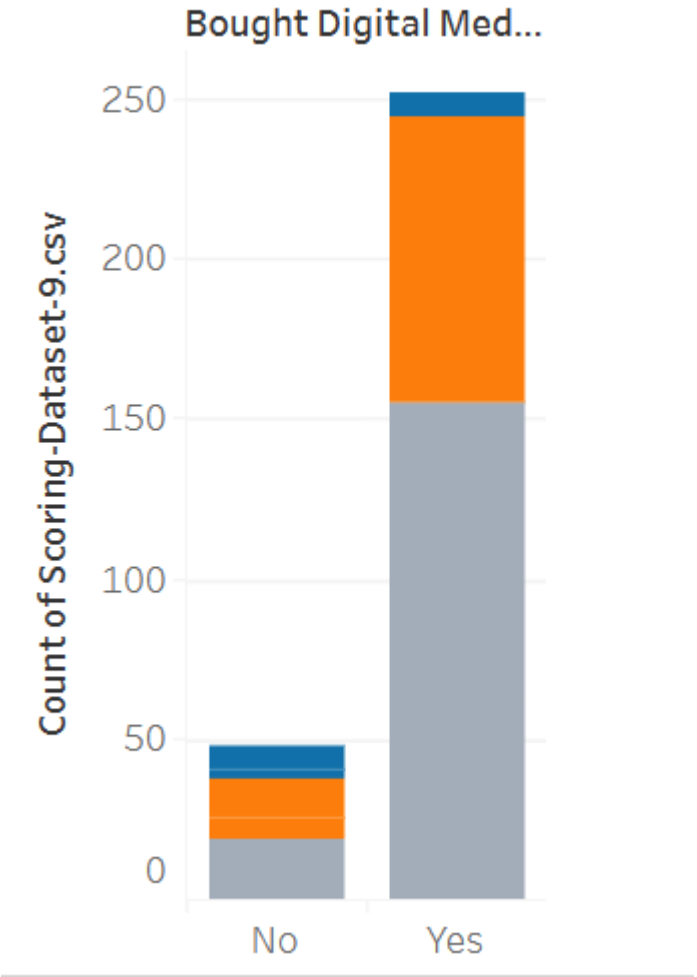
Tableau result:

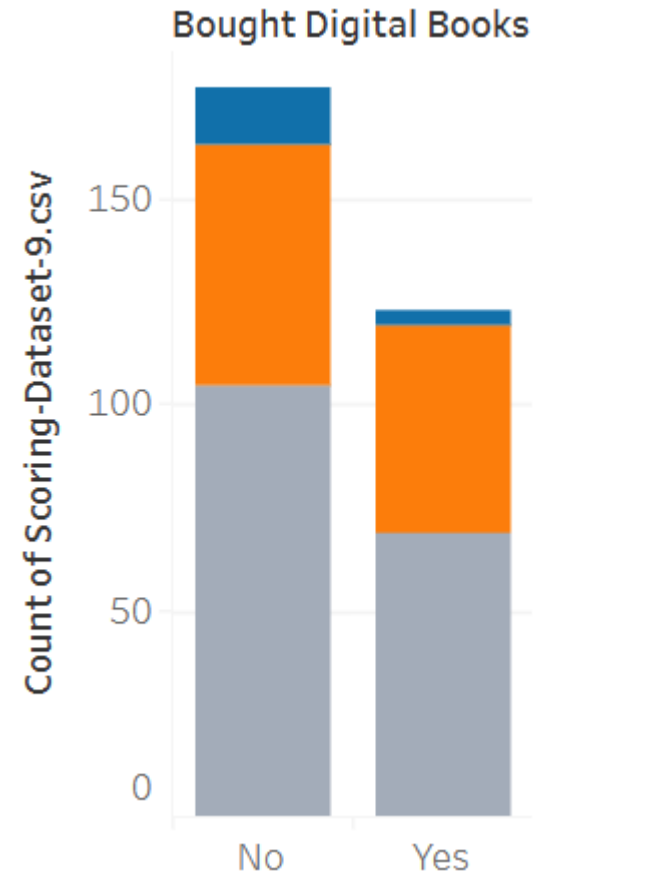


Comparison of website activity attribute with various attributes:



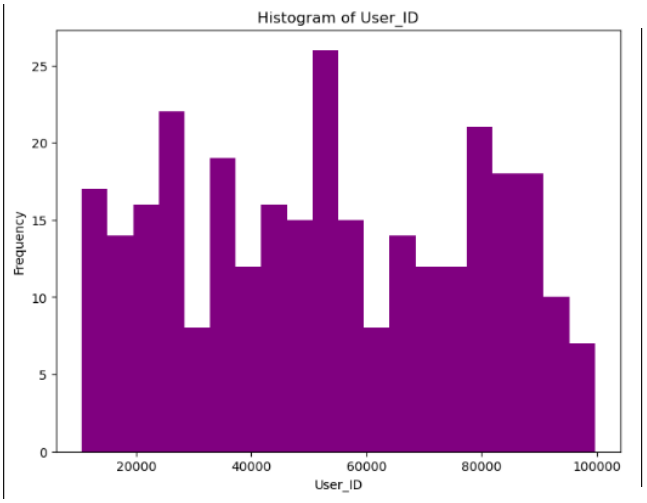


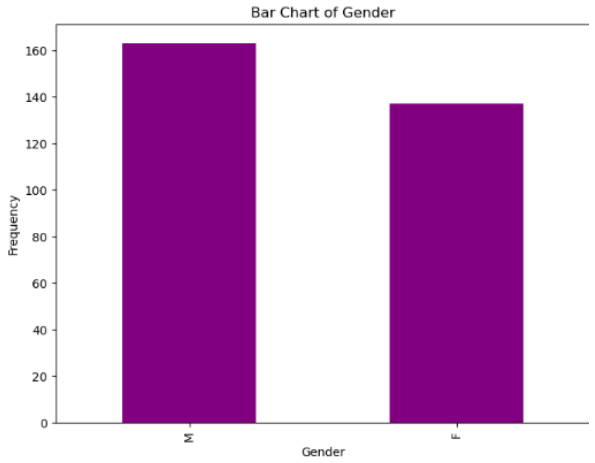


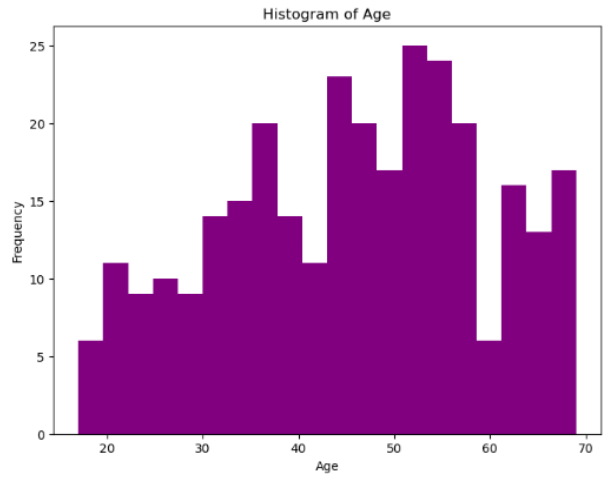


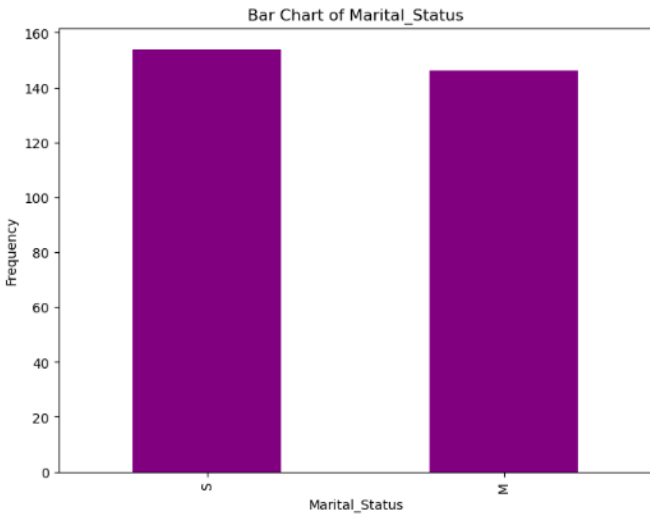
Finally, using this code bar chart for each attribute:

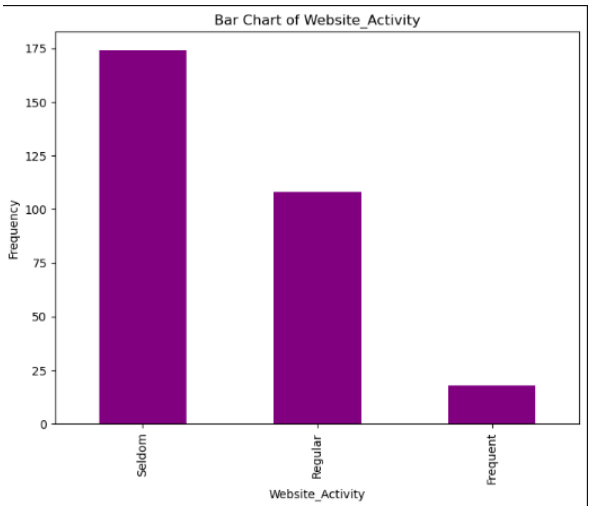
```python
def summarize_dataset(dataset_path):
    # Read the dataset into a pandas DataFrame
    df = pd.read_csv("Scoring-Dataset-9.csv")

    # Summary statistics for each attribute
    summary_statistics = df.describe()

    # Frequency of values for each attribute
    frequency_counts = {}
    for column in df.columns:
        frequency_counts[column] = df[column].value_counts()

    # Visualization for each attribute
    for column in df.columns:
        plt.figure(figsize=(8, 6))

        # Histogram for numeric attributes
        if pd.api.types.is_numeric_dtype(df[column]):
            df[column].plot(kind='hist', bins=20, color='purple', title=f'Histogram of {column}')
            plt.xlabel(column)
            plt.ylabel('Frequency')
            plt.show()
        else:
            # Bar chart for categorical attributes
            frequency_counts[column].plot(kind='bar', color='purple', title=f'Bar Chart of {column}')
            plt.xlabel(column)
            plt.ylabel('Frequency')
            plt.show()

    return summary_statistics, frequency_counts
```

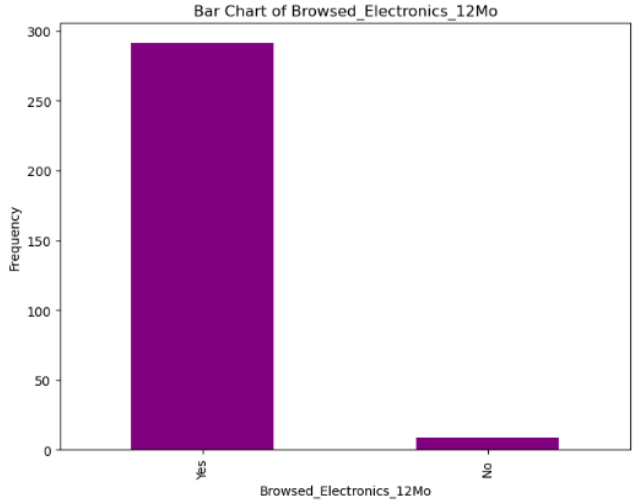


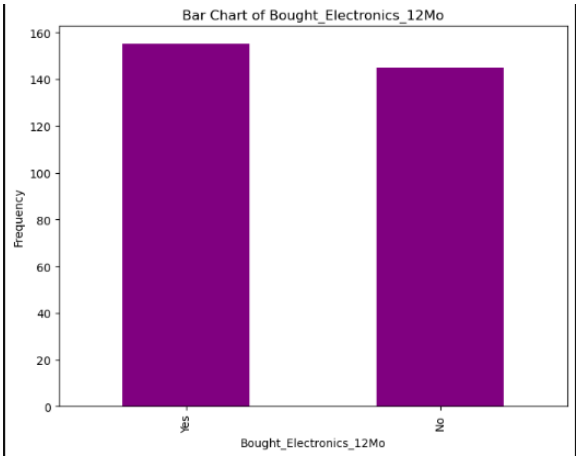


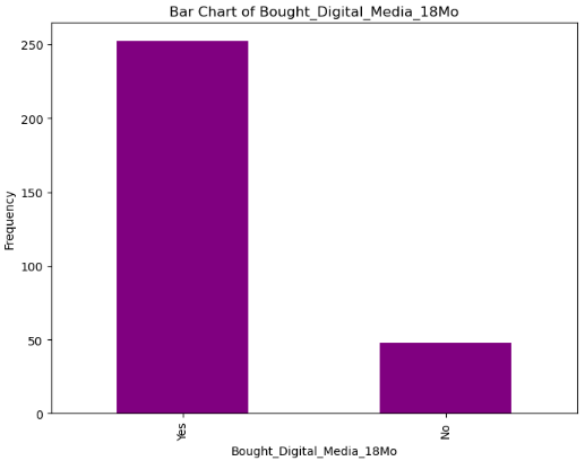


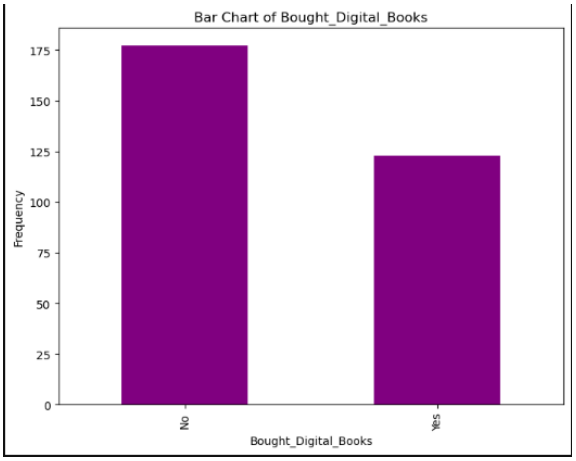


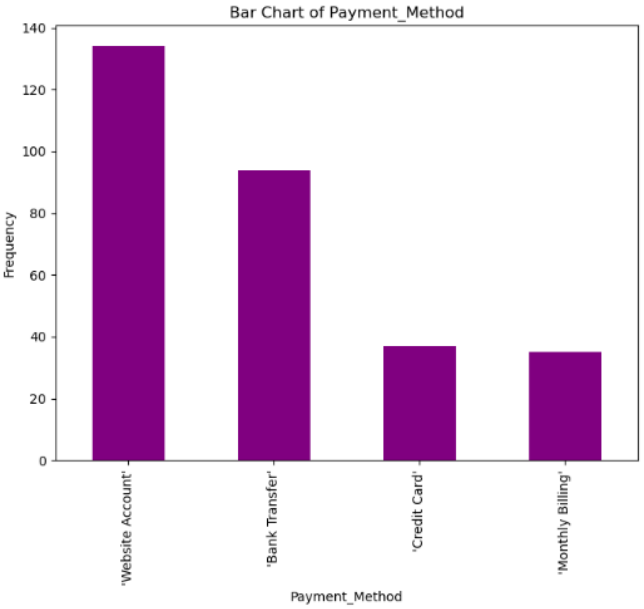

Bar Chart of Website_Activity
Frequency
0
25
50
75
100
125
150
175
Seldom
Regular
Frequent
Website_Activity


Bar Chart of Browsed_Electronics_12Mo
Frequency
0
50
100
150
200
250
300
Yes
No
Browsed_Electronics_12Mo


Bar Chart of Bought_Electronics_12Mo
Frequency
0
20
40
60
80
100
120
140
160
Yes
No
Bought_Electronics_12Mo


Bar Chart of Bought_Digital_Media_18Mo
Frequency
0
50
100
150
200
250
Yes
No
Bought_Digital_Media_18Mo


Bar Chart of Bought_Digital_Books
Frequency
0
25
50
75
100
125
150
175
No
Yes
Bought_Digital_Books

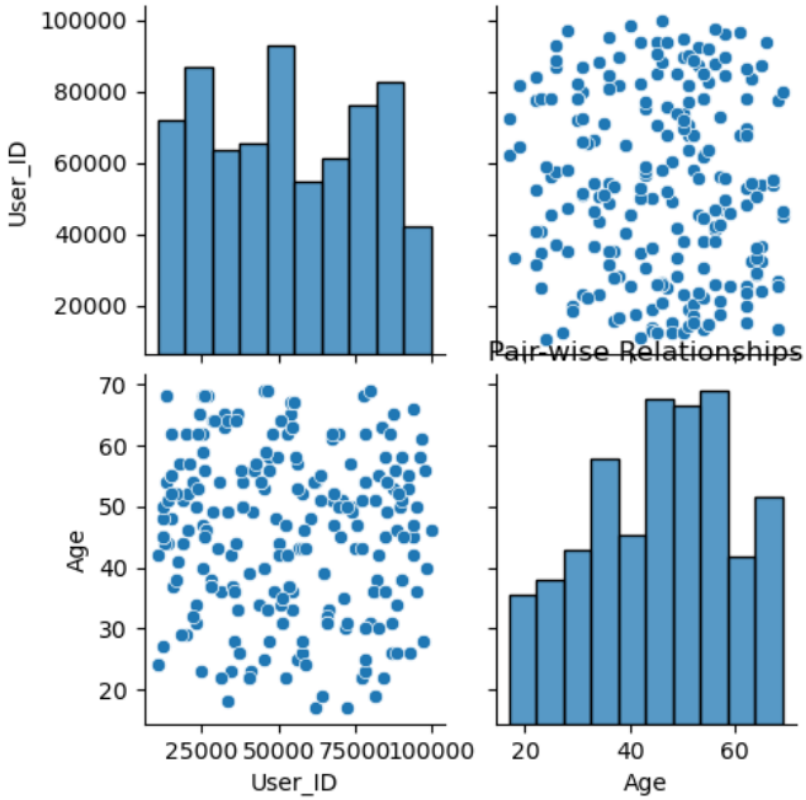

Bar Chart of Payment_Method
Frequency
0
20
40
60
80
100
120
140
'Website Account'
'Bank Transfer'
'Credit Card'
'Monthly Billing'
Payment_Method

A3. To explore our data set and identify any outliers, clusters of similar instances, "interesting" attributes and specific values of those attributes we wrote this block of code:

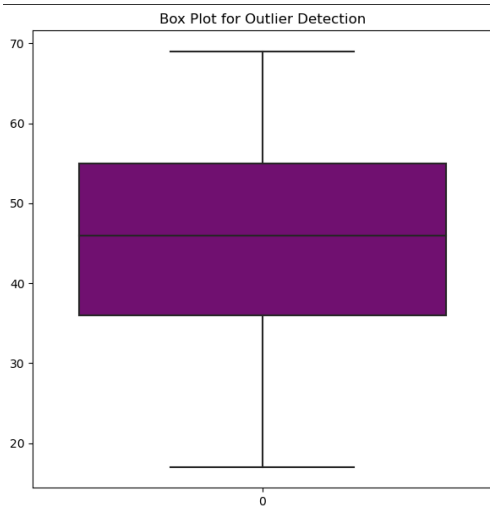
```python
# Read the dataset into a pandas DataFrame
df = pd.read_csv('Scoring-Dataset-9.csv')

# Display basic information about the dataset
print("Dataset Info:")
print(df.info())

# Display summary statistics
print("\nSummary Statistics:")
print(df.describe())
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300 entries, 0 to 299
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   User_ID                   300 non-null    int64
 1   Gender                    300 non-null    object
 2   Age                       300 non-null    int64
 3   Marital_Status            300 non-null    object
 4   Website_Activity          300 non-null    object
 5   Browsed_Electronics_12Mo  300 non-null    object
 6   Bought_Electronics_12Mo   300 non-null    object
 7   Bought_Digital_Media_18Mo 300 non-null    object
 8   Bought_Digital_Books      300 non-null    object
 9   Payment_Method            300 non-null    object
dtypes: int64(2), object(8)
memory usage: 23.6+ KB
None

Summary Statistics:
            User_ID         Age
count    300.000000  300.000000
mean   53462.776667   45.420000
std    25194.666343   13.453883
min    10591.000000   17.000000
25%    32011.500000   36.000000
50%    52945.000000   46.000000
75%    77370.000000   55.000000
max    99694.000000   69.000000
```

```python
# Visualize pair-wise relationships and scatter plots
sns.pairplot(df)
plt.title("Pair-wise Relationships")
plt.show()
```




```python
# Identify outliers using box plots
plt.figure(figsize=(7, 7))
sns.boxplot(data=df['Age'], color='purple')
plt.title("Box Plot for Outlier Detection")
plt.show()
```



Data pre-processing:

To change the raw data into a clean data set. The dataset is preprocessed in order to check missing values, noisy data, and other inconsistencies before executing it to the algorithm. Data must be in a format appropriate for ML.

B1.

```python
#B1:
# Min-Max Normalization
min_max_scaler = MinMaxScaler()
df['Age_MinMax'] = min_max_scaler.fit_transform(df[['Age']])
print("Min-Max Normalization sample:\n",df['Age_MinMax'].head(10))

# Z-Score Normalization
z_score_scaler = StandardScaler()
df['Age_ZScore'] = z_score_scaler.fit_transform(df[['Age']])
print("\nZ- score Normalization sample:\n",df['Age_ZScore'] .head(10))


# Save the DataFrame with normalized values to a CSV file
df.to_csv("C:/Users/User/Downloads/Scoring-Dataset-9_normalized.csv", index=False)
```

```
Min-Max Normalization sample:
 0    0.634615
1    0.576923
2    0.653846
3    0.711538
4    0.923077
5    0.711538
6    0.096154
7    0.692308
8    0.038462
9    0.923077
Name: Age_MinMax, dtype: float64
```

```
Z- score Normalization sample:
 0    0.340991
1    0.117634
2    0.415443
3    0.638800
4    1.457774
5    0.638800
6   -1.743670
7    0.564348
8   -1.967027
9    1.457774
Name: Age_ZScore, dtype: float64
```

B2. Discretise the Age attribute:

```python
# Define the bins and labels for discretization
age_bins = [0, 16, 35, 55, 70, float('inf')]
age_labels = ['Teenager', 'Young', 'Mid_Age', 'Mature', 'Old']

# Discretize the 'Age' column
df['Age_Category'] = pd.cut(df['Age'], bins=bins, labels=labels, right=False)

# Display the result:
print("\nAge categories sample:")
print(df['Age_Category'].head(10))

print("\nFrequency of Each Age Category:")
print(df['Age_Category'].value_counts())
```

```
Age categories sample:
0    Mid_Age
1    Mid_Age
2    Mid_Age
3    Mid_Age
4     Mature
5    Mid_Age
6      Young
7    Mid_Age
8      Young
9     Mature
Name: Age_Category, dtype: category
Categories (5, object): ['Teenager' < 'Young' < 'Mid_Age' < 'Mature' < 'Old']
```

```
Frequency of Each Age Category:
Mid_Age     149
Mature       80
Young        71
Teenager      0
Old           0
Name: Age_Category, dtype: int64
```

B3. Convert the "Gender" variable into binary variables [with values "0" or "1"]:

```
label_encoder = LabelEncoder()
df['Gender'] = label_encoder.fit_transform(df['Gender'])

# Get the mapping of original categories to encoded labels
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.classes_)))
print("Gender:")
print(label_mapping)
print(df['Gender'].head(10))


with pd.ExcelWriter('C:/Users/User/Downloads/Scoring-Dataset-9_normalized.xlsx', engine='openpyxl') as writer:
    df.to_excel(writer, sheet_name='Sheet1', index=False)
    encoding_results.to_excel(writer, sheet_name='EncodingResults', index=False)
```

```
Gender:
{'F': 0, 'M': 1}
0    0
1    0
2    1
3    1
4    1
5    0
6    1
7    1
8    1
9    0
Name: Gender, dtype: int32
```

```
# Create a pie chart to visualize the distribution of genders
gender_counts = df['Gender'].value_counts()
labels = gender_counts.index
sizes = gender_counts.values

colors = ['lightblue', 'purple']  # Light Blue for Male, Purple for Female

plt.figure(figsize=(5, 5))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90, colors=colors)
plt.title('Distribution of Genders')
plt.show()
```