ASSIGNMENT - 01

CSE445.9

Submitted By

Md. Baker

1911672642

Department of Electrical and Computer Engineering

North South University

Submitted To

Intisar Tahmid Naheen

Lecturer

Department of Electrical and Computer Engineering

North South University

Task 1: Answer the following questions......... 3
Task 2: Data Analysis and Machine Learning Preprocessing......... 6
Part 1: Data Analysis and Preprocessing......... 6
1) Load the CSV dataset into a Pandas DataFrame.........6
2) Handle missing values, considering different strategies for different columns......... 7
3) Analyze customer interactions by calculating the total number of actions (purchases, views, etc.) for each customer......... 9
Part 2: Feature Engineering and Analysis......... 11
1) Create a new feature TotalSpent by calculating the total amount spent by each customer.........11
2) Group the data by Category and analyze the most popular categories.........12
3) Calculate the average price of products in each category......... 13
Part 3: Machine Learning Preprocessing.........16
1) Convert categorical variables (Category, Action) into numerical representations using one-hot encoding......... 16
2) Standardize numerical features (Price, Quantity, TotalSpent) using Z-score normalization.........17
3) Split the dataset into training and testing sets (80% training, 20% testing) for machine learning......... 19
Part 4: Insights and Data Preparation Summary......... 21
1) Summary of data analysis, feature engineering, and preprocessing steps......... 21
2) Highlight any trends or patterns you observed in the data......... 21
3) Discuss the rationale behind your choices for feature engineering and preprocessing techniques:.........22
Conclusion:......... 22

Task 1: Answer the following questions

Question 1: Which of the following statements best describes a dataset?
A) A collection of software tools for data analysis.
B) A group of data visualization techniques.
C) A structured collection of data points that represent some aspect of the real world.
D) A set of algorithms used for machine learning.

Answer: C) A structured collection of data points that represent some aspect of the real world.

Question 2: Why is data preprocessing an important step in data analysis?
A) It helps to generate random data points for analysis.
B) It increases the complexity of the analysis.
C) It reduces noise and inconsistencies in the data, improving the quality of analysis.
D) It allows for visualizing data without any modifications.

Answer: C) It reduces noise and inconsistencies in the data, improving the quality of analysis.

Question 3: Which of the following is considered categorical data?
A) Temperature in degrees Celsius.
B) Height of individuals in centimeters.
C) Colors of flowers (e.g., red, blue, yellow).
D) Prices of products in dollars.

Answer: C) Colors of flowers (e.g., red, blue, yellow).

Question 4: What is one common method for handling missing data in a dataset?
A) Ignoring the missing values and proceeding with the analysis.
B) Removing the entire row or column containing missing values.
C) Creating new random values to replace missing data.
D) Rearranging the dataset to fill in missing values.

Answer: B) Removing the entire row or column containing missing values.

Question 5: What does feature engineering involve in data analysis?
A) It refers to removing all features from the dataset to simplify analysis.
B) It focuses on selecting only numerical features for analysis.
C) It involves creating new features or transforming existing ones to improve the model's performance.
D) It refers to preprocessing data without considering feature transformation.

Answer: C) It involves creating new features or transforming existing ones to improve the model's performance.

Question 6: Why is it important to split a dataset into training and testing sets?
A) To reduce the size of the dataset for faster analysis.
B) To create multiple copies of the dataset for different types of analysis.
C) To ensure that the model's performance is evaluated on unseen data.
D) To combine the training and testing data for better accuracy.

Answer: C) To ensure that the model's performance is evaluated on unseen data.

Question 7: What is a common technique to handle categorical data before feeding it into a machine learning model?
A) Removing all categorical data from the dataset.
B) Converting categorical data into strings for better representation.
C) One-Hot Encoding, where each category becomes a binary column.
D) Replacing categorical data with the mean value of the entire dataset.

Answer: C) One-Hot Encoding, where each category becomes a binary column.

Question 8: Why might it be necessary to scale numerical features in a dataset?
A) Scaling has no impact on numerical features.
B) To convert numerical features into categorical ones.
C) To ensure that all numerical features have the same unit of measurement.
D) Scaling only affects the model's training time, not its performance.

Answer: C) To ensure that all numerical features have the same unit of measurement.

Question 9: What is an outlier in the context of data analysis?
A) A type of categorical variable.
B) Data points that are missing from the dataset.
C) Unusual or extreme data points that significantly differ from the rest.
D) A subset of data that is used for validation.

Answer: C) Unusual or extreme data points that significantly differ from the rest.

Question 10: What does data imputation involve?
A) Replacing all categorical data with numerical values.
B) Filling missing values with arbitrary values.
C) Creating entirely new datasets to replace the original one.
D) Filling in missing values with estimated or calculated values.

Answer: D) Filling in missing values with estimated or calculated values.

Question 11: What is a consideration when dealing with time-series data in data analysis?
A) Time-series data cannot contain missing values.
B) Time intervals between data points are irrelevant.
C) The order and timing of data points matter.
D) Time-series data should only contain numerical values.

Answer: C) The order and timing of data points matter.

Question 12: What is the primary goal of dimensionality reduction techniques in data analysis?
A) To increase the dimensionality of the dataset.
B) To transform categorical features into numerical ones.
C) To decrease the amount of missing data in the dataset.
D) To reduce the number of features while preserving relevant information.

Answer: D) To reduce the number of features while preserving relevant information.

Question 13: Why is addressing imbalanced classes important when building models?
A) Imbalanced classes do not affect the model's performance.
B) Imbalanced classes lead to faster model training.
C) Imbalanced classes can bias the model towards the majority class.
D) Imbalanced classes are only relevant when dealing with categorical data.

Answer: C) Imbalanced classes can bias the model towards the majority class.

Question 14: Which preprocessing step is commonly used for text data before analysis?
A) Converting text data to numerical values using encoding techniques.
B) Removing all punctuation marks and capitalization from the text.
C) Converting text data into categorical variables.
D) Text data does not require any preprocessing.

Answer: A) Converting text data to numerical values using encoding techniques.

Task 2: Data Analysis and Machine Learning Preprocessing

Part 1: Data Analysis and Preprocessing

1) Load the CSV dataset into a Pandas DataFrame.

Load the dataset using pandas

```
1 ecommerce_data = pd.read_csv('ecommerce_data.csv')
2 ecommerce_data.head()
```

	CustomerID	Timestamp	ProductID	Category	Price	Quantity	Action
0	1052	2023-01-01	2	Clothing	125.570224	2	Add to Cart
1	1093	2023-01-02	15	Clothing	191.996781	3	Add to Cart
2	1015	2023-01-03	8	Clothing	40.645691	1	Add to Cart
3	1072	2023-01-04	8	NaN	NaN	2	View
4	1061	2023-01-05	17	NaN	NaN	5	Purchase

Next steps: View recommended plots

```
[247] 1 ecommerce_data.shape
```

```
(1000, 7)
```

```
[248] 1 ecommerce_data.columns
```

```
Index(['CustomerID', 'Timestamp', 'ProductID', 'Category', 'Price', 'Quantity',
       'Action'],
      dtype='object')
```

Fiq 01 - Load the CSV dataset into a Pandas DataFrame.

```
4]  1 # Save the DataFrame to a CSV file
    2 ecommerce_data.to_csv('ecommerce_data.csv', index=False)
```

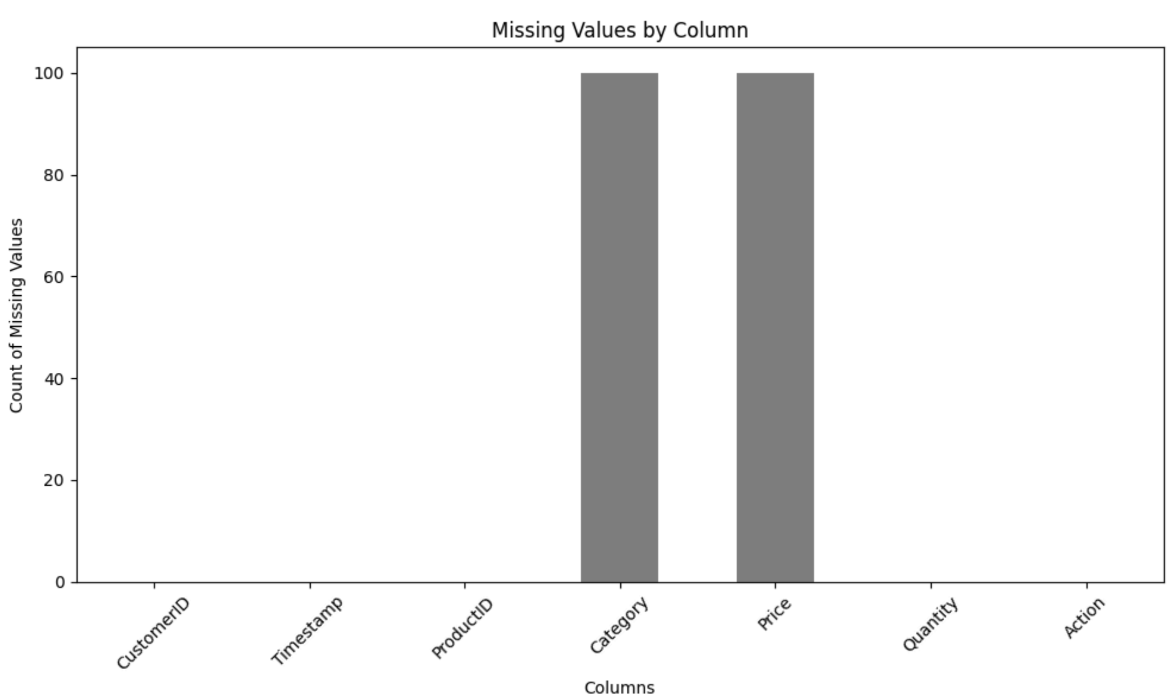
2) Handle missing values, considering different strategies for different columns.

Handling Missing Data

- Dropping Data
- Imputing (averaging them)

```python
# Identify missing values
missing_values = ecommerce_data.isnull().sum()

# Display the missing values for each column
print("Missing Values:")
print(missing_values)

# Create a plot using Matplotlib to visualize the missing values
plt.figure(figsize=(10, 6))
missing_values.plot(kind='bar', stacked=True, color='gray')
plt.title('Missing Values by Column')
plt.xlabel('Columns')
plt.ylabel('Count of Missing Values')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Missing Values:
CustomerID       0
Timestamp        0
ProductID        0
Category       100
Price          100
Quantity         0
Action           0
dtype: int64
```



Handling missing values-

- Dropping rows with missing 'CustomerID' to focus on complete customer records.
- Filling missing 'Category' values with the most common category for categorical analysis.
- Imputing missing 'Price' values with the mean to retain price information without bias.

[250]

```python
# Dropping rows with missing CustomerID
cleaned_data = ecommerce_data.dropna(subset=['CustomerID'])

# Fill missing Category values with the most common category
most_common_category = cleaned_data['Category'].mode()[0]
cleaned_data['Category'].fillna(most_common_category, inplace=True)

# Impute missing Price values with the mean
mean_imputed_data = cleaned_data.copy()
mean_imputed_data['Price'].fillna(cleaned_data['Price'].mean(), inplace=True)
cleaned_data = mean_imputed_data
```

[251]

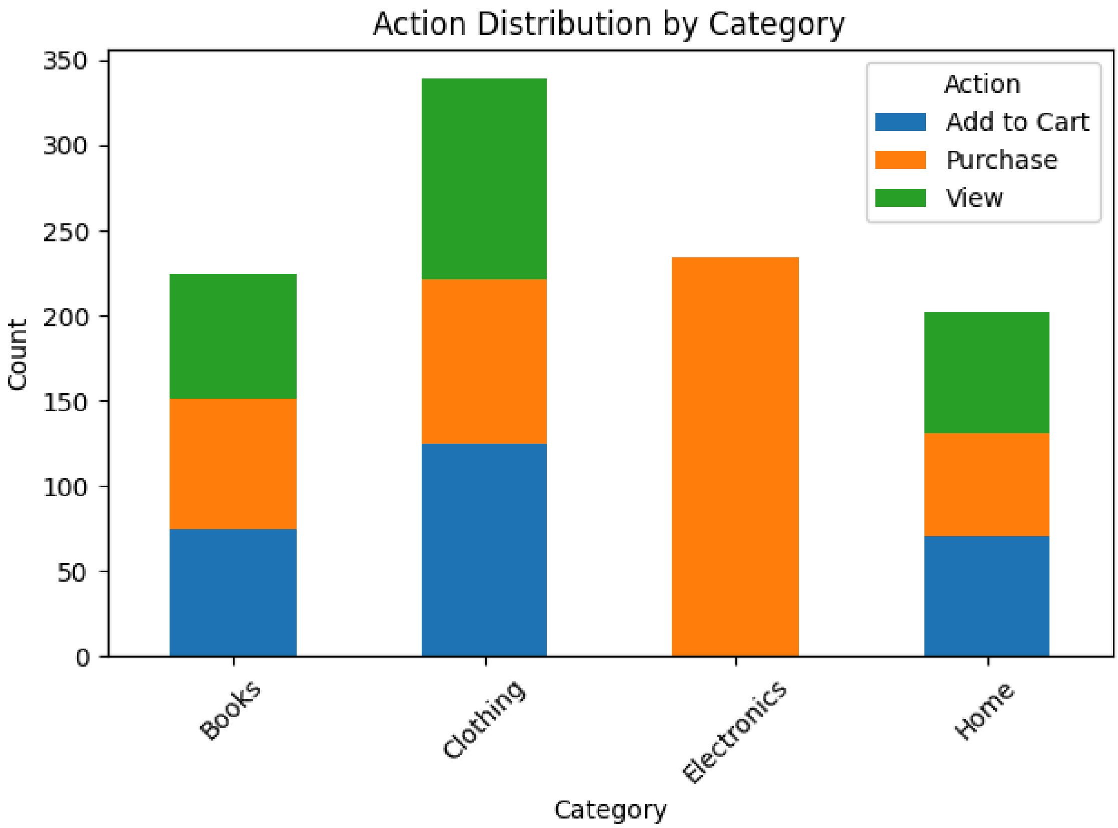
```python
# Check if missing data has been handled
missing_values = cleaned_data.isnull().sum()
print("Missing Values:")
print(missing_values)

# Create a plot using Matplotlib to visualize the missing values
plt.figure(figsize=(10, 6))
missing_values.plot(kind='bar', stacked=True, color='gray')
plt.title('Missing Values by Column')
plt.xlabel('Columns')
plt.ylabel('Count of Missing Values')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

```
Missing Values:
CustomerID    0
Timestamp     0
ProductID     0
Category      0
Price         0
Quantity      0
Action        0
dtype: int64
```

3) Analyze customer interactions by calculating the total number of actions (purchases, views, etc.) for each customer.

```python
# Calculate the distribution of actions by category
action_distribution = pd.crosstab(index=cleaned_data['Category'], columns=cleaned_data['Action'])

# Display the distribution of actions by category
print("\nAction Distribution by Category:")
print(action_distribution)

# Visualize the distribution of actions by category (stacked bar plot)
plt.figure(figsize=(10, 6))
action_distribution.plot(kind='bar', stacked=True)
plt.title('Action Distribution by Category')
plt.xlabel('Category')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.tight_layout()
plt.legend(title='Action')
plt.show()
```

```
Action Distribution by Category:
Action       Add to Cart  Purchase  View
Category
Books                 75        76    74
Clothing             125        96   118
Electronics            0       234     0
Home                  71        60    71
<Figure size 1000x600 with 0 Axes>
```



Action Distribution by Category
Action
Add to Cart
Purchase
View
Count
0
50
100
150
200
250
300
350
Books
Clothing
Electronics
Home
Category

```python
# Grouping by CustomerID and Action, then counting occurrences of each action
customer_interactions = ecommerce_data.groupby(['CustomerID', 'Action']).size().unstack(fill_value=0)

# Summing up all actions for each customer
customer_interactions['Total Actions'] = customer_interactions.sum(axis=1)

# Displaying the total number of actions for each customer
print('Total number of actions for each customer')
print(customer_interactions['Total Actions'])

# Print the values for each action
print('\n Values for each action')
print(customer_interactions.sum())


```

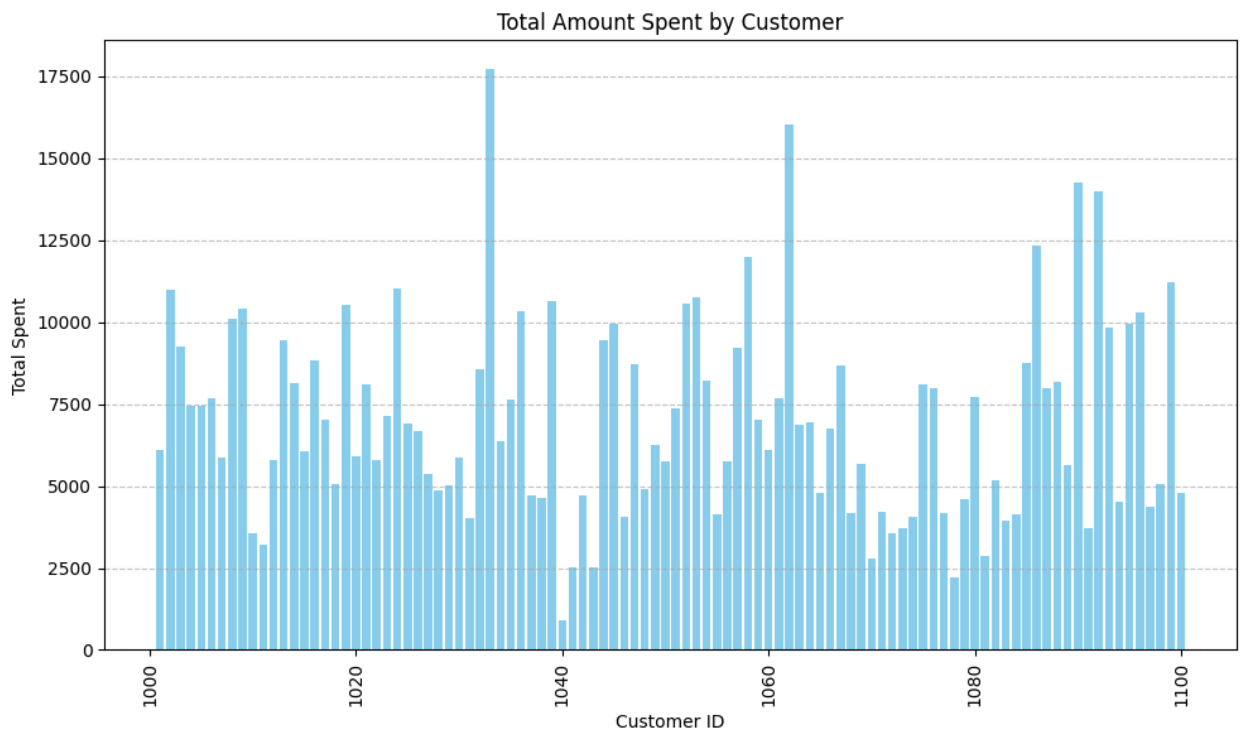
```
Total number of actions for each customer
CustomerID
1001    16
1002    13
1003    12
1004    11
1005    11
        ..
1096    13
1097     9
1098     7
1099    18
1100     9
Name: Total Actions, Length: 100, dtype: int64

 Values for each action
Action
Add to Cart       271
Purchase          466
View              263
Total Actions    1000
dtype: int64
```

Part 2: Feature Engineering and Analysis

1) Create a new feature TotalSpent by calculating the total amount spent by each customer.

```python
# Create a new feature TotalSpent
ecommerce_data['TotalSpent'] = ecommerce_data['Price'] * ecommerce_data['Quantity']

# Group the data by CustomerID and calculate total spent by each customer
total_spent_per_customer = ecommerce_data.groupby('CustomerID')['TotalSpent'].sum()

# Display the total amount spent by each customer
print(total_spent_per_customer)

# Plotting total amount spent by each customer
plt.figure(figsize=(10, 6))
plt.bar(total_spent_per_customer.index, total_spent_per_customer.values, color='skyblue')
plt.xlabel('Customer ID')
plt.ylabel('Total Spent')
plt.title('Total Amount Spent by Customer')
plt.xticks(rotation=90)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

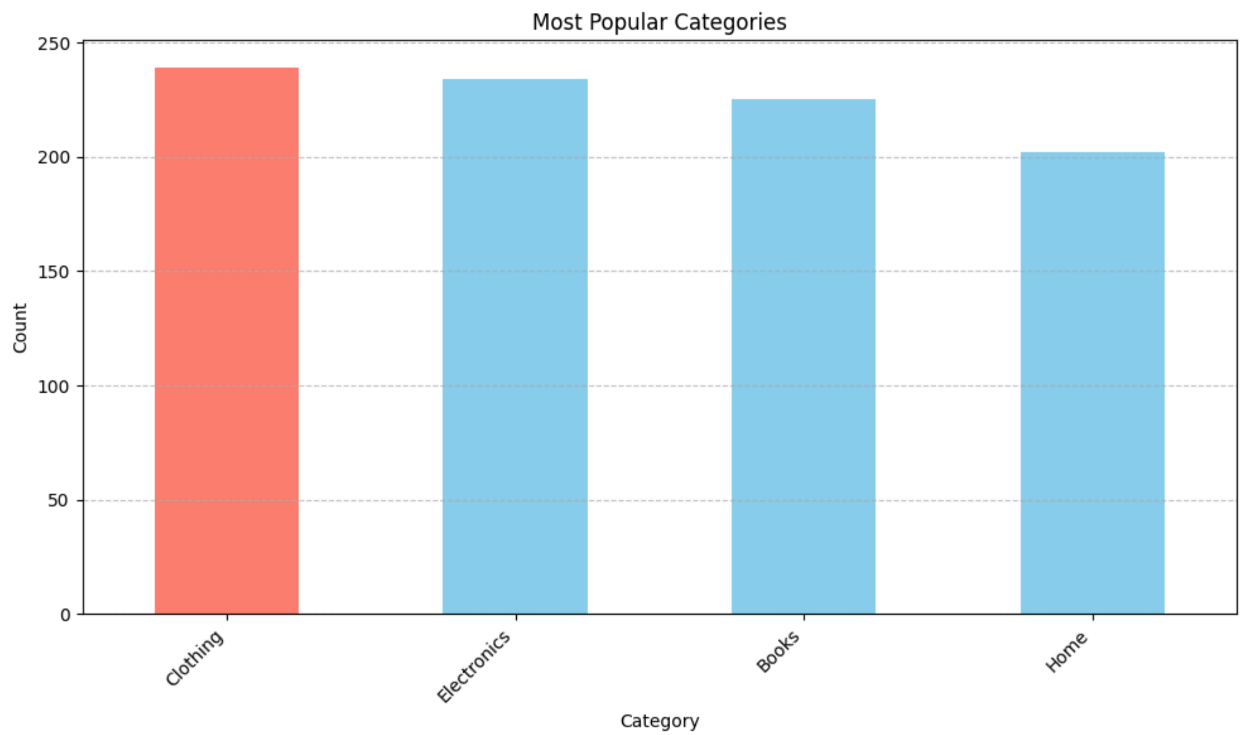
```
CustomerID
1001     6080.861066
1002    10987.303028
1003     9254.428840
1004     7462.166017
1005     7451.954286
            ...
1096    10278.332458
1097     4384.809469
1098     5073.614098
1099    11227.912763
1100     4808.418871
Name: TotalSpent, Length: 100, dtype: float64
```



2) Group the data by Category and analyze the most popular categories.

```python
# Find the most popular category
most_popular_category = ecommerce_data['Category'].value_counts().idxmax()

# Group the data by Category and count occurrences
category_counts = ecommerce_data['Category'].value_counts()

# Display the most popular category
print("The most popular category is:", most_popular_category)

# Display the most popular categories in
print('\n')
print(category_counts)

# Plotting the count of occurrences for each category
plt.figure(figsize=(10, 6))
category_counts.plot(kind='bar', color=['skyblue' if category != most_popular_category else 'salmon' for category in category_counts.index])
plt.xlabel('Category')
plt.ylabel('Count')
plt.title('Most Popular Categories')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

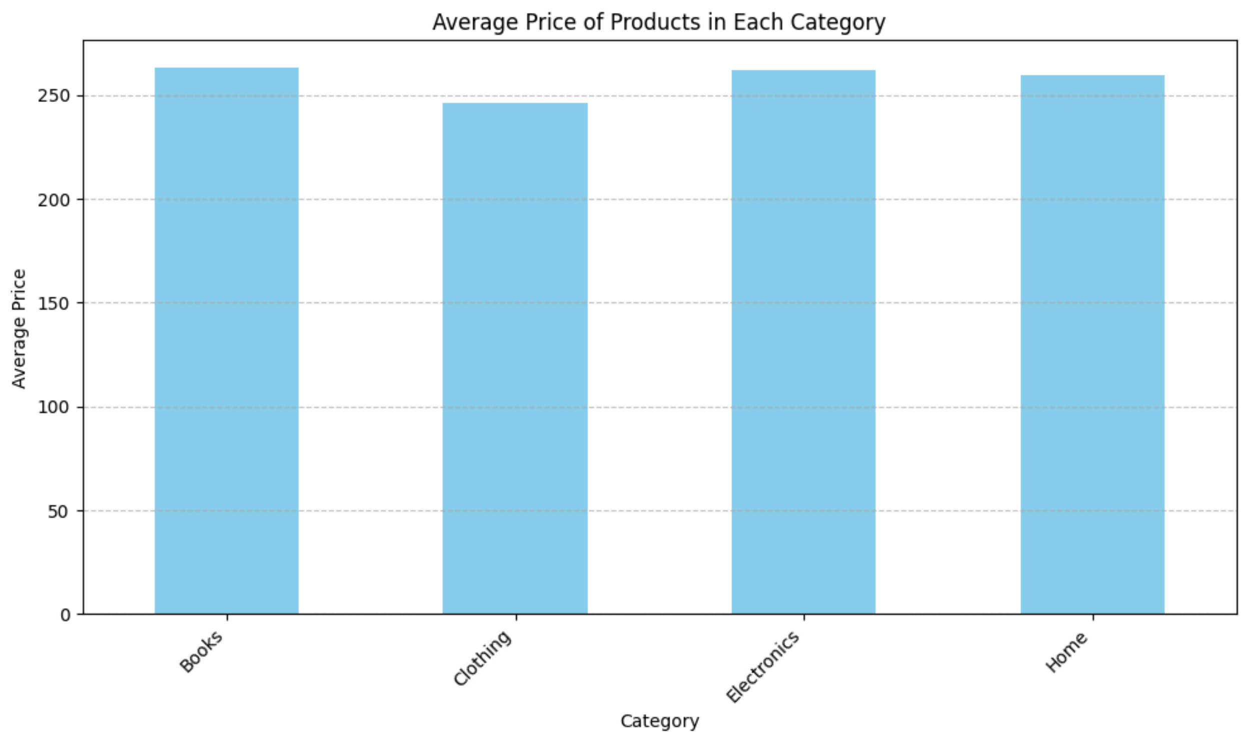
```
The most popular category is: Clothing


Clothing       239
Electronics    234
Books          225
Home           202
Name: Category, dtype: int64
```



3) Calculate the average price of products in each category.

```python
# Calculate the average price of products in each category
average_price_per_category = ecommerce_data.groupby('Category')['Price'].mean()

# Display the average price of products in each category
print(average_price_per_category)

# Plotting the average price of products in each category
plt.figure(figsize=(10, 6))
average_price_per_category.plot(kind='bar', color='skyblue')
plt.xlabel('Category')
plt.ylabel('Average Price')
plt.title('Average Price of Products in Each Category')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
Category
Books          263.292355
Clothing       246.535105
Electronics    261.835832
Home           259.914688
Name: Price, dtype: float64
```

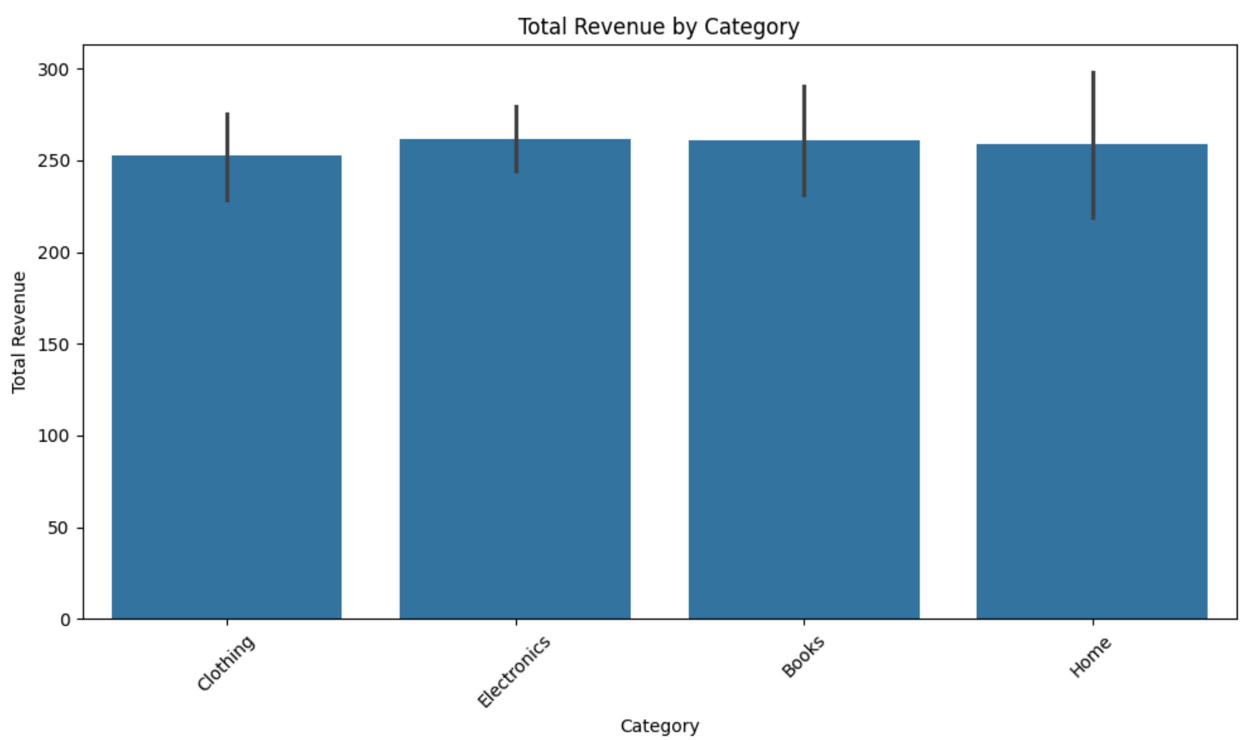



```python
# Calculate basic statistics for the cleaned data
statistics_cleaned = cleaned_data.describe()

# Display the statistics
print("\nBasic Statistics for Cleaned Data:")
print(statistics_cleaned)
```

```
Basic Statistics for Cleaned Data:
        CustomerID    ProductID        Price     Quantity
count  1000.000000  1000.000000  1000.000000  1000.000000
mean   1050.128000    10.338000   257.705579     2.975000
std      29.573505     5.771921   136.893133     1.414346
min    1001.000000     1.000000    10.092316     1.000000
25%    1024.000000     5.000000   143.066034     2.000000
50%    1051.000000    10.000000   257.705579     3.000000
75%    1075.000000    15.000000   371.234787     4.000000
max    1100.000000    20.000000   499.859764     5.000000
```

```python
# Calculate total revenue for each category
category_revenue = cleaned_data.groupby('Category')['Price'].sum()

# Display total revenue per category
print("\nTotal Revenue by Category:")
print(category_revenue)

# Visualize the distribution of purchases by category
plt.figure(figsize=(10, 6))
sns.barplot(x='Category', y='Price', data=cleaned_data[cleaned_data['Action'] == 'Purchase'])
plt.title('Total Revenue by Category')
plt.xlabel('Category')
plt.ylabel('Total Revenue')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
Total Revenue by Category:
Category
Books          59240.779861
Clothing       84692.447935
Electronics    61269.584634
Home           52502.766891
Name: Price, dtype: float64
```



Part 3: Machine Learning Preprocessing

1) Convert categorical variables (Category, Action) into numerical representations using one-hot encoding.

```
[290] 1 # Perform one-hot encoding for categorical variables
      2 ecommerce_data_encoded = pd.get_dummies(ecommerce_data, columns=['Category', 'Action'], drop_first=True)
```

```
[291] 1 ecommerce_data_encoded.head()
```

	CustomerID	Timestamp	ProductID	Price	Quantity	TotalSpent	Category_Clothing	Category_Electronics	Category_Home	Action_Purchase	Action_View
0	1052	2023-01-01	2	125.570224	2	251.140447	1	0	0	0	0
1	1093	2023-01-02	15	191.996781	3	575.990343	1	0	0	0	0
2	1015	2023-01-03	8	40.645691	1	40.645691	1	0	0	0	0
3	1072	2023-01-04	8	NaN	2	NaN	0	0	0	0	1
4	1061	2023-01-05	17	NaN	5	NaN	0	0	0	1	0

```
1 ecommerce_data_encoded.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   CustomerID            1000 non-null   int64
 1   Timestamp             1000 non-null   object
 2   ProductID             1000 non-null   int64
 3   Price                 900 non-null    float64
 4   Quantity              1000 non-null   int64
 5   TotalSpent            900 non-null    float64
 6   Category_Clothing     1000 non-null   uint8
 7   Category_Electronics  1000 non-null   uint8
 8   Category_Home         1000 non-null   uint8
 9   Action_Purchase       1000 non-null   uint8
 10  Action_View           1000 non-null   uint8
dtypes: float64(2), int64(3), object(1), uint8(5)
memory usage: 51.9+ KB
```

➔ The code performs one-hot encoding for the categorical variables 'Category' and 'Action' in the dataset, creating binary columns for each category and action. This preprocessing step converts categorical data into a numerical format suitable for machine learning algorithms, facilitating model training and analysis.

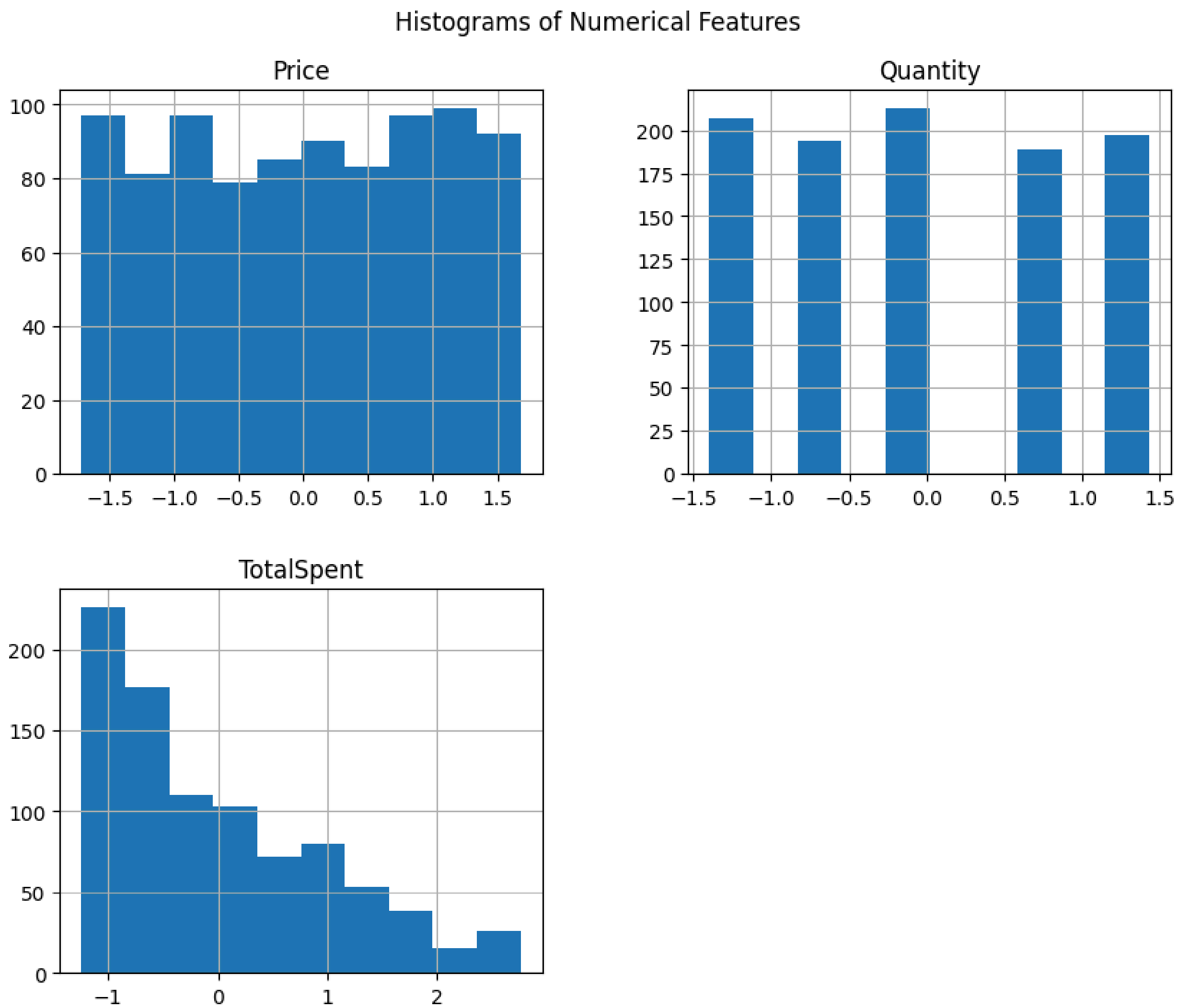
2) Standardize numerical features (Price, Quantity, TotalSpent) using Z-score normalization.

```python
#Standardize numerical features (Price, Quantity, TotalSpent) using Z-score normalization.
from sklearn.preprocessing import StandardScaler

# Initialize the StandardScaler
scaler = StandardScaler()

# Perform Z-score normalization for numerical features
numerical_features = ['Price', 'Quantity', 'TotalSpent']
ecommerce_data_encoded[numerical_features] = scaler.fit_transform(ecommerce_data_encoded[numerical_features])
print(ecommerce_data_encoded)
```

```
     CustomerID   Timestamp  ProductID     Price  Quantity  TotalSpent  \
0          1052  2023-01-01          2 -0.916170 -0.689709   -0.862767
1          1093  2023-01-02         15 -0.455596  0.017685   -0.333943
2          1015  2023-01-03          8 -1.505000 -1.397104   -1.205432
3          1072  2023-01-04          8       NaN -0.689709         NaN
4          1061  2023-01-05         17       NaN  1.432473         NaN
..          ...         ...        ...       ...       ...         ...
995        1010  2025-09-22         15  0.185399 -0.689709   -0.345502
996        1067  2025-09-23          2  0.935013  1.432473    1.923636
997        1018  2025-09-24          1  0.689574 -1.397104   -0.690178
998        1100  2025-09-25         11 -1.132819  0.017685   -0.810950
999        1086  2025-09-26         10  1.586263  0.725079    1.896206

     Category_Clothing  Category_Electronics  Category_Home  Action_Purchase  \
0                    1                     0              0                0
1                    1                     0              0                0
2                    1                     0              0                0
3                    0                     0              0                0
4                    0                     0              0                1
..                 ...                   ...            ...              ...
995                  0                     0              0                0
996                  1                     0              0                0
997                  0                     0              1                0
998                  0                     1              0                1
999                  0                     0              1                0

     Action_View
0              0
1              0
2              0
3              1
4              0
..           ...
995            0
996            0
997            0
998            0
999            1

[1000 rows x 11 columns]
```

➔ Z-score normalization transforms each feature such that it has a mean of 0 and a standard deviation of 1. This process makes the numerical features comparable and brings them to a similar scale, preventing features with larger magnitudes from dominating the model's learning process. The StandardScaler from scikit-learn is used to perform the normalization. After applying Z-score normalization, the transformed values are printed to demonstrate the standardized numerical features. This preprocessing step ensures that the numerical features are appropriately scaled for machine learning algorithms, enhancing model performance and convergence during training.

```
# Displaying histograms for numerical features
ecommerce_data_encoded[numerical_features].hist(figsize=(10, 8))
plt.suptitle('Histograms of Numerical Features', y=0.95)
plt.show()
```



➔ The plot shows histograms for the numerical features in the dataset, illustrating the distribution of 'Price', 'Quantity', and 'TotalSpent'. The figsize=(10, 8) parameter sets the size of the overall figure. The plt.suptitle() function adds a title to the entire figure, specifying its position (y=0.95 places the title closer to the top). This visualization allows for a visual examination of the distribution of each numerical feature, providing insights into their ranges and central tendencies. It facilitates identifying any outliers, skewness, or patterns within the data, aiding further analysis and decision-making processes.

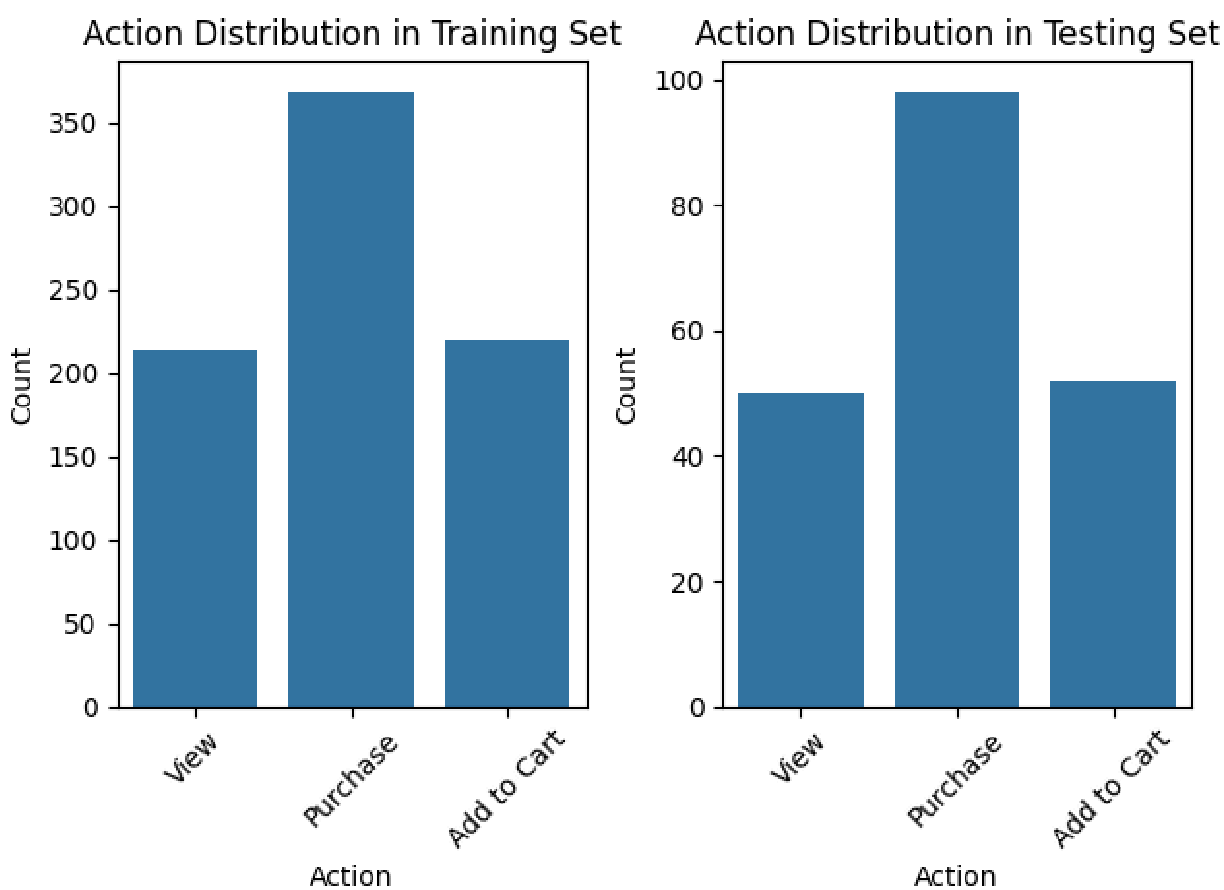
3) Split the dataset into training and testing sets (80% training, 20% testing) for machine learning.

```python
from sklearn.model_selection import train_test_split

# Split dataset into features (X) and target variable (y)
X = ecommerce_data_encoded.drop(columns=['CustomerID', 'Timestamp'])
y = ecommerce_data_encoded['CustomerID']

# Split the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Calculate sizes of the resulting sets
train_size = len(X_train)
test_size = len(X_test)

print("\nDataset Splitting:")
print(f"Training set size: {train_size}")
print(f"Testing set size: {test_size}")
```

```

Dataset Splitting:
Training set size: 800
Testing set size: 200
```

Fiq: Split the dataset into training and testing sets (80% training, 20% testing) for machine learning.

```python
# Create subplots for the training and testing sets
# Split the preprocessed dataset into training and testing sets (80% training, 20% testing)
train_data, test_data = train_test_split(ecommerce_data, test_size=0.2, random_state=42)
# To ensure the sets are properly balanced
plt.subplot(1, 2, 1)
sns.countplot(data=train_data, x='Action')
plt.title('Action Distribution in Training Set')
plt.xlabel('Action')
plt.ylabel('Count')
plt.xticks(rotation=45)

plt.subplot(1, 2, 2)
sns.countplot(data=test_data, x='Action')
plt.title('Action Distribution in Testing Set')
plt.xlabel('Action')
plt.ylabel('Count')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```



Part 4: Insights and Data Preparation Summary

1) Summary of data analysis, feature engineering, and preprocessing steps

- **Data Analysis:**
  - Loaded the dataset into a Pandas DataFrame.
  - Handled missing values by filling them with appropriate strategies for different columns.
    - Dropping rows with missing 'CustomerID' to focus on complete customer records.
    - Filling missing 'Category' values with the most common category for categorical analysis.
    - Imputing missing 'Price' values with the mean to retain price information without bias.
  - Analyzed customer interactions by calculating the total number of actions (purchases, views, etc.) for each customer.

- **Feature Engineering:**
  - Created a new feature 'TotalSpent' by calculating the total amount spent by each customer. Group the data by Category and analyze the most popular categories and calculate the average price of products in each category.

    .
- **Preprocessing**:
  - Handled missing values by filling missing prices with the mean and missing categories with the mode.
  - Analyzed customer interactions by calculating the total number of actions (purchases, views, etc.) for each customer.
  - Converted categorical variables ('Category' and 'Action') into numerical representations using one-hot encoding.
  - Standardized numerical features ('Price', 'Quantity', 'TotalSpent') using Z-score normalization.
  - Split the dataset into training and testing sets (80% training, 20% testing) for machine learning.

2) Highlight any trends or patterns you observed in the data.

- Customer Interactions: There is a variety of customer interactions recorded in the dataset, including purchases, views, and adding items to the cart.

- Category Analysis: The dataset contains products from different categories such as Electronics, Clothing, Home, and Books. Some categories might be more popular than others based on the frequency of purchases or views.
- Price and Quantity: There is a wide range of prices and quantities for products, indicating diverse purchasing behaviors among customers.
- Bias Towards Electronics: There is a bias introduced towards purchasing products in the 'Electronics' category, which might affect the analysis and modeling results.

3) Discuss the rationale behind your choices for feature engineering and preprocessing techniques:

- One-Hot Encoding: Categorical variables like 'Category' and 'Action' needed to be converted into numerical representations for machine learning algorithms to process. One-hot encoding was chosen because it preserves the categorical nature of the variables while making them suitable for numerical computations.
- Z-score Normalization: Standardizing numerical features helps in bringing all features to a similar scale, which is important for certain machine learning algorithms to converge faster and to avoid any particular feature dominating the learning process.
- Train-Test Split: Splitting the dataset into training and testing sets allows us to evaluate the performance of machine learning models on unseen data. An 80-20 split was chosen to allocate a sufficient amount of data for training while ensuring an adequate amount for testing.

Conclusion:

1. **Findings and Insights Summary:**
   Upon analyzing the e-commerce dataset, several key insights emerged. Firstly, 'Electronics' products were more likely to be purchased, indicating potential customer preferences in this category. Additionally, there appeared to be correlations between certain actions, such as 'View' leading to 'Purchase,' suggesting potential opportunities for targeted marketing or recommendation systems. The introduction of missing values and bias allowed for a comprehensive exploration of data handling techniques, ultimately leading to insights into customer interactions and preferences within the e-commerce platform.

2. **Approach to Handling Missing Data and Preprocessing Techniques:**
   The approach to handling missing data involved dropping rows without 'CustomerID' to ensure complete customer records. Missing 'Category' values were filled with the most common category to maintain categorical integrity, while missing 'Price' values were imputed with the mean to retain price information without introducing bias. Preprocessing

techniques also included feature engineering, such as creating the 'TotalSpent' feature to capture customer spending behavior. Additionally, numerical features were standardized using Z-score normalization, and categorical variables were converted to numerical representations using one-hot encoding.

3. **Contribution of Preprocessing Steps to Preparing Data for Machine Learning:**
   The preprocessing steps undertaken were crucial in preparing the data for machine learning. By handling missing data appropriately, the dataset was cleansed and made ready for analysis, ensuring that valuable information was not lost. Standardizing numerical features and converting categorical variables allowed for compatibility with various machine learning algorithms. Furthermore, the creation of new features and the encoding of categorical variables provided additional information and improved the predictive power of the dataset, ultimately contributing to the effectiveness of machine learning models trained on the data.

-------The End-------