BLACK FRIDAY SALES ANALYSIS

Team members:

Likhitha Chandanati

Murali Manobhiram Penumutchu

Praharsha Mutyala

Amarnadh Kari

Abstract:

This study helps to analyze the black Friday sales dataset and would give an insight into the shopping trends of customers using a dataset containing attributes such as User_ID, Product_ID, Gender, Age, Occupation, product_catergory_1, product_category_2, product_category_3 and Purchase. The analysis evaluates how buying behaviors can vary by category of products, physical locations, age, gender, occupation and depending on different features. The analysis aims to figure out key factors impacting sales numbers and potential correlations among variables, how one feature is dependent on the other feature and some other empirical analysis using the application of mathematical and statistical techniques and the models.

Goals and Objectives:

The main objective and goal are to understand consumer trends, tendencies, and purchasing patterns, to identify the primary driving factors and features influencing consumer expenditures and quantity of sales. For the creation of predictive models that, using characteristics and product categories as a foundation, to examine how purchases spread among multiple groups—such as age, gender, and occupation and understand how these different factors affect the choices that customers make, to find connections between attributes, product categories, and expenditures by using regression analysis and identifying correlations between components.

- **Motivation:**
  The motivation behind Black Friday sales analysis is to gain insight into customer purchasing patterns based on gender, age, and their occupation, analyze the effectiveness of marketing, to get to know which attributes contribute most of the analysis i.e. important features and which features effect sales the most.

- **Significance**:
  Black Friday sales analysis is highly significant because it helps us to know the shopping trend, learn about client behavior, to retrieve the crucial features to improve the model performance and accuracy, to analyze the purchasing trends based on different attributes age, gender, occupation etc.

- **Objectives**:
  The main objective of analyzing Black Friday sales is to know the purchase trends, which age group buys what products and which gender buys what products taking into the consideration the attributes such as occupation, city, age, gender e.t.c. which product sales drive most of the overall sales by using different techniques and analysis methods such as bivariate analysis, univariate analysis, exploratory data analysis, tree based models and classification metrics.

- **Features**:
  Attributes like User_ID, Product_ID, Gender, Age, Occupation, City_Category, Stay_In_Current_City_Years, Marital status, Product_Category_1, Product_Category_2, and Product_Category_3 is included in Black Friday sales analysis features. These resources simplify the analysis of product categories, purchase patterns, and buyer characteristics.

Related work (Background):

The practical tutorials and scholar papers available on platforms like Kaggle and blogs of Ali Ahmad delve into the implementation of Decision tree classifier, XG Boost models for sales analysis. The Kaggle tutorial on Decision tree classifier by Prashant presents a step-by-step guide for employing the model to provide insights into customer purchasing decisions.

In addition to theoretical aspects, Kaggle competitions and real-world applications serve as valuable resources for understanding the practical implementation of these models in sales analysis scenarios. Competitions like "Analysis of Store Sales" on Kaggle provide insights into the diverse approaches and methodologies applied by practitioners to effectively analyze sales trends.

In summary, these sources collectively offer a broad understanding of how models like Decision tree classifier, Decision tree regressor and XGBoost are employed, both theoretically and practically, to analyze and predict sales data, enabling businesses to make informed decisions and optimize their operations.

Dataset:

We have taken our dataset from Kaggle (https://www.kaggle.com/datasets/pranavuikey/black-friday-sales-eda/data). The dataset comprises of 12 columns and 550069 rows. Data plays a major role in defining the efficiency of our model. Quality data refers to better identification of trends and patterns, which is vital for predicting future sales based on the resemblances drawn in the data.

To improve the quality of the dataset we have handled the missing values by replacing them with mode and removing outlier data. We need to drop the duplicate rows as they lead to biasing in model. The attributes of the dataset are described below to understand and get more insights into the data.

- Attributes in the Dataset:
    a. User_ID: A unique number allocated to user.
    b. Product_ID: A unique number allotted to the products.
    c. Gender: Describes the user as male or female.
    d. Age: Provides the range of age of the user.
    e. Occupation: Represents occupation code of user.
    f. City_Category: Provides the city where user lives.
    g. Stay_In_Current_City_Years: Gives number of years the user has been living in current city.
    h. Marital Status: Describes whether the user is married or unmarried.
    i. Product_Category_1: The year when the outlet was established.
    j. Product_Category_2: Type of location where the outlet is situated.
    k. Product_Category_3: Unique identifier for the item.
    l. Purchase: target variable - the sales figure for the item at the outlet.

- Dataset Structure:

a. Predictor Variables: features like User_ID, Product_ID, Gender, Age, Occupation, City_Category, Stay_In_Current_City_Years, Marital Status, and Product_Category_1, Product_Category_2, and Product_Category_3 serve as predictor variables influencing the prediction of response variable.
b. Response Variable: purchase is the target variable that needs to be predicted based on the predictor variables.

These predictor variables provide a comprehensive insight into product details, purchasing patterns and customer characteristics. This dataset is used for training machine learning models such as Decision trees and XG Boost to analyze sales based on identified patterns.

Detail Design of Features:

Features are the input variables or attributes which are used in making predictions. Initially we explored the data by removing the irrelevant columns and converting categorical variables to numerical variables by label encoder as the algorithms cannot handle categorical data. Visualize a correlation matrix using a heat map to identify the correlation between features. We have split the dataset in 70-30 ratio to train, test datasets for evaluating the model performance on unseen data.

We have used 'SelectKBest' function imported from sklearn for feature selection. It will select the most relevant features related to target variable from the dataset. It identifies the important features based on the univariate analysis which is a statistical method that analyzes the distribution of single variable at a time. We have identified the top 7 most relevant features that are Age, occupation, City_Category, Marital_Status, Product_Category_1, Product_Category_2, Purchase.

Final submission:

We have Identifying Specific Gender Bought Product Category Using Heat Map, Exploring Seasonal Trends Using Time Series Analysis, Analyzing the Impact of Marketing Strategies Using Point Plot, One Way Anova, Resampling Technique-KFold Cross Validation, Model k – nearest neighbors and Random Forest classifiers.

Analysis:

Analysis was done of Black Friday Sales data using different techniques. Data preprocessing was the first step that was performed after the dataset was extracted from Kaggle. Data preprocessing has gone through many cleaning methods such as dropping the irrelevant columns that are not useful for analysis. Dropping duplicate rows, finding the missing values, and replacing those values with the numeric, replacing the valueless categorical values with some mapping values, removing the null valued rows was done as a part of data cleaning and making it ready for some training and analysis.

The cleaned data was then analyzed using various techniques and methods, such as Exploratory data analysis, univariate analysis, bivariate analysis using different display of plots, correlation analysis and feature selection to find the relations between the attributes and to retrieve the vital feature of the data which would help in drawing the analysis more accurately.

Training the model is done using the decision tree classifier on 70 percent of the data and the other 30 percent of the data went through the testing.

Then the evaluation of the model is done to make some predictions after the data is reshaped. The data that went through the testing after the training was then tested to find how accurate the results that were given are. We have used confusion matrix techniques, precision, recall, f1 score and support techniques to find the accuracy of the data.

Final submission:

To provide more insights into the data and derive more analyzed results leaving the basic analysis, we have analyzed the impact of marketing strategies using point plot. To perform this, we have considered all the attributes that are present in our dataset.

We have also conducted One-way Anova analysis on the data to analyze how the independent attributes like gender, age, occupation etc. have a measurable effect on the dependent variable purchase.

Exploratory data analysis:

EDA facilitates the determination of values that are unavailable, outliers, or gaps in the data as well as offers an early evaluation of links and structures within the data. It further helps in figuring out the structure and concentrations of many different attributes. It offers fundamental knowledge about correlations.

With in the Exploratory data analysis, we have dropped all the irrelevant columns from the dataset.

After dropping the irrelevant columns this is how the data looks:

	User_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Category_2	Product_Category_3
0	1000001	F	0-17	10	A	2	0	3	NaN	NaN
1	1000001	F	0-17	10	A	2	0	1	6.0	14.0
2	1000001	F	0-17	10	A	2	0	12	NaN	NaN
3	1000001	F	0-17	10	A	2	0	12	14.0	NaN
4	1000002	M	55+	16	C	4+	0	8	NaN	NaN

Dropped the duplicate rows (present a number of times in the data) and then found out the total number of rows present in the data.

Counting the total number of rows first

```
#Here i am counting the number of rows.
Sales.count()
```

```
User_ID                     550068
Gender                      550068
Age                         550068
Occupation                  550068
City_Category               550068
Stay_In_Current_City_Years  550068
Marital_Status              550068
Product_Category_1          550068
Product_Category_2          376430
Product_Category_3          166821
Purchase                    550068
dtype: int64
```

Getting the number of duplicate rows present in the data.

```
[ ]  # @title Dropping The Duplicate Rows
     # Here we are finding the duplicate rows using the duplicated() command.
     rows_duplicate = Sales[Sales.duplicated()]
     # Printing the duplicate rows.
     print("Number of duplicate rows: ", rows_duplicate.shape)
```

```
Number of duplicate rows:  (2543, 11)
```

Dropping these duplicate rows and counting the number of total rows present in the data

```
# After dropping the duplicate rows again i am counting the number of rows.
Sales.count()
```

```
User_ID                       547525
Gender                        547525
Age                           547525
Occupation                    547525
City_Category                 547525
Stay_In_Current_City_Years    547525
Marital_Status                547525
Product_Category_1            547525
Product_Category_2            375825
Product_Category_3            166734
Purchase                      547525
dtype: int64
```

Then as a part of exploratory data analysis, we have replaced the missing values with the numeric values

Where after that the data looks like:

```
# @title Replacing The Missing Data With Numeric Values
# In the 'Gender' column we are Replacing 'F' with 0 and 'no' with 1 using replace command.
Sales['Gender'] = Sales['Gender'].replace({'F': 0, 'M': 1})

# Here we are displaying the dataFrame with the replaced values of 'Gender' column.
print(Sales)
```

```
        User_ID  Gender    Age  Occupation City_Category  \
0       1000001       0   0-17          10             A
1       1000001       0   0-17          10             A
2       1000001       0   0-17          10             A
3       1000001       0   0-17          10             A
4       1000002       1    55+          16             C
...         ...     ...    ...         ...           ...
550063  1006033       1  51-55          13             B
550064  1006035       0  26-35           1             C
550065  1006036       0  26-35          15             B
550066  1006038       0    55+           1             C
550067  1006039       0  46-50           0             B

       Stay_In_Current_City_Years  Marital_Status  Product_Category_1  \
0                               2               0                   3
1                               2               0                   1
2                               2               0                  12
3                               2               0                  12
4                              4+               0                   8
...                           ...             ...                 ...
550063                          1               1                  20
550064                          3               0                  20
550065                         4+               1                  20
550066                          2               0                  20

        Product_Category_2  Product_Category_3  Purchase
0                      NaN                 NaN      8370
1                      6.0                14.0     15200
2                      NaN                 NaN      1422
3                     14.0                 NaN      1057
4                      NaN                 NaN      7969
...                    ...                 ...       ...
550063                 NaN                 NaN       368
550064                 NaN                 NaN       371
550065                 NaN                 NaN       137
550066                 NaN                 NaN       365
550067                 NaN                 NaN       490

[547525 rows x 11 columns]
```

For the analysis part, we had replaced the missing values with the numeric values

First finding the missing values from the dataset goes as:

```
# @title Filling The Missing Values
# Using the isnull() command i am checking the missing values in the dataset
print(Sales.isnull().sum())
```

For the analysis part, we had replaced the missing values in product category_2 with the numeric values

```
[ ] # We are filling the missing values in Product_Category_2 with the mode value of Product_Category_2.
    Sales["Product_Category_2"].fillna(Sales["Product_Category_2"].mode()[0],inplace=True)
    # Here we are displaying the dataFrame after filling the missing values of 'Product_Category_2' column.
    print(Sales)
```

Filling the missing values in the product category_3

```
# We are filling the missing values in Product_Category_3 with the mode value of Product_Category_3.
Sales["Product_Category_3"].fillna(Sales["Product_Category_3"].mode()[0],inplace=True)
# Here we are displaying the dataFrame after filling the missing values of 'Product_Category_3' column.
print(Sales)
```

Here is the display for the dataset statistics

```
[ ] <class 'pandas.core.frame.DataFrame'>
    Int64Index: 547525 entries, 0 to 550067
    Data columns (total 11 columns):
     #   Column                      Non-Null Count   Dtype
    ---  ------                      --------------   -----
     0   User_ID                     547525 non-null  int64
     1   Gender                      547525 non-null  int64
     2   Age                         547525 non-null  int64
     3   Occupation                  547525 non-null  int64
     4   City_Category               547525 non-null  object
     5   Stay_In_Current_City_Years  547525 non-null  object
     6   Marital_Status              547525 non-null  int64
     7   Product_Category_1          547525 non-null  int64
     8   Product_Category_2          547525 non-null  float64
     9   Product_Category_3          547525 non-null  float64
     10  Purchase                    547525 non-null  int64
    dtypes: float64(2), int64(7), object(2)
    memory usage: 50.1+ MB
```

	User_ID	Gender	Age	Occupation	Marital_Status	Product_Category_1	Product_Category_2	Product_C
count	5.475250e+05	547525.000000	547525.000000	547525.000000	547525.000000	547525.000000	547525.000000	547
mean	1.003029e+06	0.753186	3.496251	8.078466	0.409563	5.398765	9.263243	
std	1.727581e+03	0.431158	1.353850	6.521839	0.491754	3.940801	4.300364	
min	1.000001e+06	0.000000	1.000000	0.000000	0.000000	1.000000	2.000000	
25%	1.001516e+06	1.000000	3.000000	2.000000	0.000000	1.000000	8.000000	
50%	1.003077e+06	1.000000	3.000000	7.000000	0.000000	5.000000	8.000000	

As a part of the exploratory data analysis, we have detected the outliners for gender, age, occupation, purchase, product_category_1, product_category_2, product_category3 as follows:

Detecting Outliners for Gender attribute:

```
# Using the boxplot we are checking outliers for the Gender
# Creating a boxplot for gender
sns.boxplot(x=Sales['Gender'])
# For the plot we are adding a title
plt.title('Boxplot of Gender')
# Displaying the plot
plt.show()
```

Detecting Outliners for age:

```
# Using the boxplot we are checking outliers for the Age
# Creating a boxplot for Age
sns.boxplot(x=Sales['Age'])
# For the plot we are adding a title
plt.title('Boxplot of Age')
# Displaying the plot
plt.show()
```

Outliners for Occupation:

```
# Using the boxplot we are checking outliers for the Occupation
# Creating a boxplot for Occupation
sns.boxplot(x=Sales['Occupation'])
# For the plot we are adding a title
plt.title('Boxplot of Occupation')
# Displaying the plot
plt.show()
```

Outliners for Marital_status:

```
# Using the boxplot we are checking outliers for the Martial Status
# Creating a boxplot for Martial_Status
sns.boxplot(x=Sales['Marital_Status'])
# For the plot we are adding a title
plt.title('Boxplot of Marital_Status')
# Displaying the plot
plt.show()
```

Outliners for product_category_1:

```
# Using the boxplot we are checking outliers for the Product_Category_1
# Creating a boxplot for Product_Category_1
sns.boxplot(x=Sales['Product_Category_1'])
# For the plot we are adding a title
plt.title('Boxplot of Product_Category_1')
# Displaying the plot
plt.show()
```

Outliners for product_category_2:

```
# Using the boxplot we are checking outliers for the Product_Category_2
# Creating a boxplot for Product_Category_2
sns.boxplot(x=Sales['Product_Category_2'])
# For the plot we are adding a title
plt.title('Boxplot of Product_Category_2')
# Displaying the plot
plt.show()
```

Outliners for Product_category_3:

```
[ ] # Using the boxplot we are checking outliers for the Product_Category_3
# Creating a boxplot for Product_Category_3
sns.boxplot(x=Sales['Product_Category_3'])
# For the plot we are adding a title
plt.title('Boxplot of Product_Category_3')
# Displaying the plot
plt.show()
```

Detecting outliners for purchase using the boxplot:

```
# Using the boxplot we are checking outliers for the Purchase
# Creating a boxplot for Purchase
sns.boxplot(x=Sales['Purchase'])
# For the plot we are adding a title
plt.title('Boxplot of Purchase')
# Displaying the plot
plt.show()
```

Implementation:

After the data had been cleansed, it was analyzed using a variety of approaches and implements, including Exploratory data analysis, detecting the outliners for age, detecting outliners for gender, detecting outliners for occupation, checking outliners for marital status, checking outliners for product categories and purchase was done through a boxplot display to point out any disparities or abnormalities in the data.

We have performed univariate analysis to get the early insight into the characteristics of the data and the distribution of each variable independently, discovering patterns and unusual values, to support in recognizing the features and actions of specific elements using the histogram representation.

Performed bivariate analysis using the bar plot visualization and scatter plot visualization to understand the relation between two variables such as how age feature effects the purchase trends, how occupation adds for the purchase, how a particular gender or product is affecting the purchase.

Implemented the co relation analysis using heat maps for all the attributes of the data set to find the to select the model's strongest determinants can be assisted by knowing how alterations to one variable relate to modifications in another.

To select the vital features, label conder was imported to encode the categorical values which helps in converting the categorical values to the numerical values which would help us to analyze the data more accurately to share the crucial features.

Feature selection is done using the installation of selectKBest function which would help to find the most impacting element/feature in the data.

For Model selection and model training, we have imported all the libraries such as train_test_split, DecisionTreeClassifier, metrics, confusion_matrix, classification_report and accuracy_score. Next, the training the test cases were created by splitting the data into training_size which comprises of 70 percent of the data and testing size as 30 percent of the data.

Training the model is done using the decision tree classifier technique. But the training data set is in 1D array form, but to perform decision tree classifier the data must be in 2d array format. So, we applied the reshaping technique first and then fitted the decision tree model using the training data which would help us coordinate with the model selected by altering the parameters to reduce errors to promote performance. In the following phase, the model gathers up on patterns and connections seen in the initial training data.

To Evaluate the data that was trained, testing of the data is done on the 30 percent of the data. So, we applied the reshaping technique first and then fitted the decision tree model using the testing data.

To find the accuracy of the data which was tested we have evaluated the model creating a confusion matrix which helps to evaluate the effectiveness of the model and It improves evaluating a model's expected accuracy.

Next to that we have applied precision, f1 score, recall and support in the classification report to determine the efficiency of the classifying model. F1-score optimizes recall and precision by prioritizing precise predictions of positives over actual positive identifications. Support is the dispersion of groups.

The model training and testing goes as follows:

Model selection and training:

Creating the train class and test data

Creating The Train And Test Cases

```
# @title Creating The Train And Test Cases
# Here we are Ssplitting the varaibles as training_size as 70 and testing_size as 30
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7,
                                                    test_size = 0.3, random_state = 100)
```

Training the model decision Tree classifier

```
# @title Training The Model-Decision Tree Classifier
# Here X_train data is 1D array to perform the decision tree classifier data should be in 2D array. So we are reshaping the X_train data using values.re
X_train = X_train.values.reshape(-1, 1)
# Here we are creating Decision Tree model
dt_model = DecisionTreeClassifier()
# We are fitting the Decision Tree model Using training data
dt_model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

Model Evaluation:

```
# @title Evaluating The Model
# Here X_test data is 1D array to perform the decision tree classifier data should be in 2D array. So we are reshaping the x_test data using values.resh
X_test = X_test.values.reshape(-1, 1)
# Here we are using the decision tree model to make the predictions on the data which is reshaped
y_pred = dt_model.predict(X_test)
# Here we are printing the y_predicted values
print(y_pred)
```

```
[2 1 8 ... 8 8 1]
```

Finding the accuracy of the model:

```
# @title Finding The Accuracy
# Here we are comparing y_test,y_pred using accuracy_score()
accuracy = accuracy_score(y_test, y_pred)*100
# Here we are printing the accuracy of the testing data
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 86.4118331814961
```

Creating a confusion matrix:

```
[92] # @title Creating A Confusion Matrix
     # Here we are comparing y_test,y_pred using confusion_matrix()
     conf_matrix = confusion_matrix(y_test, y_pred)
     # Here we are displaying the confusion matrix
     print("Confusion Matrix:")
     print(conf_matrix)
```

Decision Tree classifier results:

```
[93] # @title Decision Tree Classifier Results
     # Here we are printing the classification report
     print("Classification Report:")
     # Here we are comparing y_test,y_pred using classification_report()
     print(classification_report(y_test, y_pred))
```

Creating other model and training and testing the data for that:

```
# @title Creating The Train And Test Cases
# Here we are Ssplitting the varaibles as training_size as 70 and testing_size as 30
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7,
                                                    test_size = 0.3, random_state = 100)
```

Training the data:

```
# @title Training Data
# To verify the split we are printing the shapes
print(X_train)
print(y_train)
```

Testing the data:

```
[98] # @title Testing Data
     # To verify the split we are printing the shapes
     print(X_test)
     print(y_test)
```

This is how training the model Gaussian NB happens:

▾ Training The Model-Guassian Model

```
[ ] # @title Training The Model-Guassian Model
    # Here X_train data is 1D array to perform the Gaussian Modle data should be in 2D array. So we are reshaping the X_train data using values.reshape() command
    X_train = X_train.values.reshape(-1, 1)
    # Here we are creating Gaussian model
    Gaussian_model = GaussianNB()
    # We are fitting the Guassian model Using training data
    Gaussian_model.fit(X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

Coming to the evaluation of the model we are transforming the testing data which is in 1D array to 2D array

▾ Evaluating The Model

```
[ ] # @title Evaluating The Model
    # Here X_test data is 1D array to perform the Gaussian Model data should be in 2D array. So we are reshaping the x_test data using values.reshape() command
    X_test = X_test.values.reshape(-1, 1)
    # Here we are using the Gaussian model to make the predictions on the data which is reshaped
    y_pred = Gaussian_model.predict(X_test)
    # Here we are printing the y_predicted values
    print(y_pred)
```

```
[1 1 5 ... 8 8 1]
```

Now we had found the accuracy of the testing data with the predicted data:

Finding The Accuracy

```
[ ] # @title Finding The Accuracy
    # Here we are comparing y_test,y_pred using accuracy_score()
    accuracy = accuracy_score(y_test, y_pred)*100
    # Here we are printing the accuracy of the testing data
    print(f"Accuracy: {accuracy}")

    Accuracy: 54.43870837129264
```

Then we have compared the testing data and predicted data using the confusion matrix:

```
# @title Creating A Confusion Matrix
# Here we are comparing y_test,y_pred using confusion_matrix()
conf_matrix = confusion_matrix(y_test, y_pred)
# Here we are displaying the confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

Final submission:

For the final draft we have implemented two more models such as K-Nearest neighbor model and random forest classifier.

Implementing K-Nearest Neighbours (KNN):

For implementing the K-Nearest Neighbour Model we have imported all the required libraries and evaluation metrics. Later created two variables with independent variable X as Purchase and dependent variable Y as Product Category 1.

Next splitted the dataset into train and test parts out of which 70% is used for training and the remaining 30% is used for testing. Later to ensure the split is done correctly we printed the shape of the dataset. Next converted the training feature (X_train) from 1-d array to 2-d array with the help of a reshaping technique and then created the K-nearest neighbor model and then fitted it with the training data. Like the train data now converted the testing feature (X_test) from 1-d array to 2-d array using a reshaping technique and made predictions on test data and printed them.

Next, we used a few evaluation metrics such as accuracy, confusion matrix and classification report to verify the performance of the K-nearest neighbor's model.

Training the model K-Nearest Neighbours (KNN):

```
[ ]  # @title Training The Model-K-Nearest Neighbors (KNN) Model
     # Here X_train data is 1D array to perform the K-Nearest Neighbors (KNN) Model data should be in 2D array. So we are reshaping the X_train data using valu
     X_train = X_train.values.reshape(-1, 1)
     # Here we are creating K-Nearest Neighbors (KNN) Model
     Knn_model = KNeighborsClassifier()
     # We are fitting the K-Nearest Neighbors (KNN) Model Using training data
     Knn_model.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

Model Evaluation:

```
[ ]  # @title Evaluating The Model
     # Here X_test data is 1D array to perform the K-Nearest Neighbors (KNN) Model data should be in 2D array. So we are reshaping the x_test data using values
     X_test = X_test.values.reshape(-1, 1)
     # Here we are using the K-Nearest Neighbors (KNN) Model to make the predictions on the data which is reshaped
     y_pred = Knn_model.predict(X_test)
     # Here we are printing the y_predicted values
     print(y_pred)
```

```
[2 1 8 ... 8 8 1]
```

Finding the accuracy of the model:

```
[ ]  # @title Finding The Accuracy
     # Here we are comparing y_test,y_pred using accuracy_score()
     accuracy = accuracy_score(y_test, y_pred)*100
     # Here we are printing the accuracy of the testing data
     print(f"Accuracy: {accuracy}")
```

```
Accuracy: 85.32746229375444
```

Creating a confusion matrix:

```
# @title Creating A Confusion Matrix
# Here we are comparing y_test,y_pred using confusion_matrix()
conf_matrix = confusion_matrix(y_test, y_pred)
# Here we are displaying the confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[13162     0    24     0     0    58     0  1377     0    17   224     0
      0     0     1     5     0    72]
 [   21   762    41     0    97   211     7   266     0     2    11     0
      0     0    53    49     2     7]
 [  148    73  1101    17    34     9     1   183     0     0    31     0
      0     0     8     3     1     5]
 [    0     1     0   734    99     0     0   113     0     0    29    92
     36     0     0     0     0     0]
 [    0     6     0    14 25006     0     0    49     0     0    63    38
      0     0     0     0     0     8]
 [  233    97    29     0    49  2030    66    75     0     0    21     0
      0     0    58   354     0     0]
 [   24    37     4     0    17   267   114     6     0     1     9     0
      0     0   106   146     0     0]
 [  439    16     0   167   162     9     0 17819     0     0   667     0
      0     0     0     0     0    27]
 [    0     5     2     0     0     0     0     5     9    34     0     0
      0     2     7     0     0     0]
 [  151    11     1     0     5     1     0    17     2   359     2     0
      0    11     3     0     0     0]
 [  176     0     0    23    83     0     0  1125     0     0  3683     6
      0     0     0     0     0   105]
 [    0     0     0    61    99     0     0     0     0     0    28   310
     36     0     0     0     0     0]
 [    0     0     0     0     0     0     0     0     0     0     0     0
   1003     0     0     0     0     0]
 [   43     0    17     0     3     0     0    26     5    37    10     0
      0    91     0     0     0     4]
 [   46   170    28     0    76   114    80    12     0     1    36     0
      0     0   537   173     0     0]
 [   90   118    16     0    89   864   124    42     0     2    25     0
      0     0   157   442     0     0]
 [    0    23    18     1     4     0     0    19     0     0     0     0
      0     0     6     4    37     0]
 [   46     0     0     7    20     0     0    87     0     0   256     0
      4     0     0     0     0   237]]
```

K-Nearest Neighbours (KNN) results:

```
# @title K-Nearest Neighbors (KNN) Model Results
# Here we are printing the classification report
print("Classification Report:")
# Here we are comparing y_test,y_pred using classification_report()
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.90      0.88      0.89     14940
           2       0.58      0.50      0.54      1529
           3       0.86      0.68      0.76      1614
           4       0.72      0.66      0.69      1104
           5       0.97      0.99      0.98     25184
           6       0.57      0.67      0.62      3012
           7       0.29      0.16      0.20       731
           8       0.84      0.92      0.88     19306
           9       0.56      0.14      0.23        64
          10       0.79      0.64      0.71       563
          11       0.72      0.71      0.72      5201
          12       0.70      0.58      0.63       534
          13       0.93      1.00      0.96      1003
          14       0.88      0.39      0.54       236
          15       0.57      0.42      0.49      1273
          16       0.38      0.22      0.28      1969
          17       0.93      0.33      0.49       112
          18       0.51      0.36      0.42       657

    accuracy                           0.85     79032
   macro avg       0.70      0.57      0.61     79032
weighted avg       0.84      0.85      0.85     79032
```

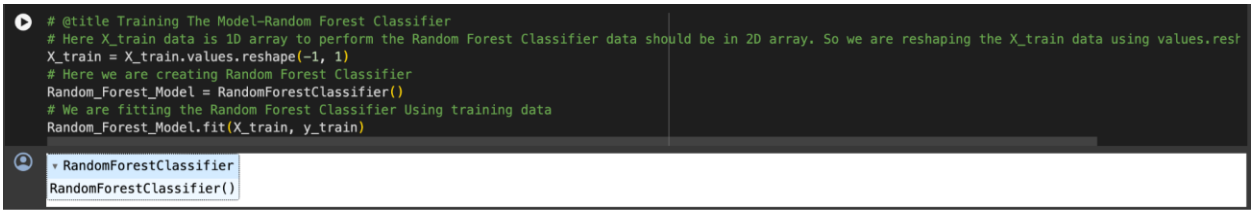
Implementing Random Tree Classifier:

For implementing the Random Forest Classifier, we have imported all the required libraries and evaluation metrics. Later created two variables with independent variable X as Purchase and dependent variable Y as Product Category 1.

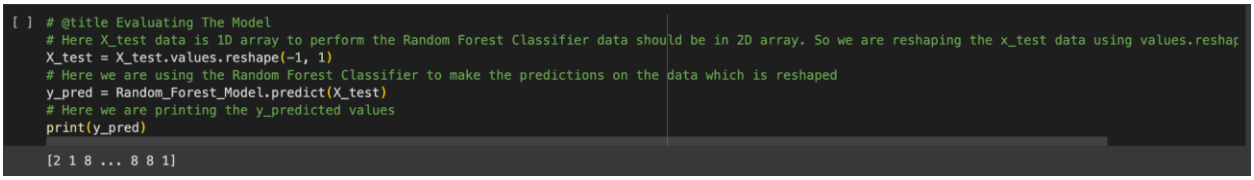
Next splitted the dataset into train and test parts out of which 70% is used for training and the remaining 30% is used for testing. Later to ensure the split is done correctly we printed the shape of the dataset. Next converted the training feature (X_train) from 1-d array to 2-d array with the help of a reshaping technique and then created the Random Forest Classifier model and then fitted it with the training data. Like the train data now converted the testing feature (X_test) from 1-d array to 2-d array using a reshaping technique and made predictions on test data and printed them.

Next, we used a few evaluation metrics such as accuracy, confusion matrix and classification report to verify the performance of the Random Forest Classifier model.

Training the model Random Tree Classifier:



Model Evaluation:



Finding the accuracy of the model:

Creating a confusion matrix:

```python
# @title Creating A Confusion Matrix
# Here we are comparing y_test,y_pred using confusion_matrix()
conf_matrix = confusion_matrix(y_test, y_pred)
# Here we are displaying the confusion matrix
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[13151     0    25     0     0     0     1  1448     0     5   222     0
      0     0     4     8     0    76]
 [   22   693    42     0    97   208    11   267     2     8    11     0
      0     0    80    69    12     7]
 [  146    69  1077    16    34     8     0   189     0     0    33     0
      0     5    20     9     5     3]
 [    0     0    11   711   104     0     0   107     0     0    28   104
     36     0     0     0     0     3]
 [    0     0     0     0 25106     0     0    17     0     0    37    24
      0     0     0     0     0     0]
 [  224    67    25     0    49  1906    86    84     0     0    23     0
      0     0    54   494     0     0]
 [   24    24     3     0    17   240   112     6     0     1     9     0
      0     0   105   190     0     0]
 [   60     0     0   173   186     0     0 18847     0     0    17     0
      0     0     0     0     0    23]
 [    0     4     2     0     0     0     0     5    11    34     0     0
      0     3     5     0     0     0]
 [  150    11     1     0     5     1     0    17    20   329     2     0
      0    25     1     1     0     0]
 [  119     0     0    19   106     0     0  1379     0     0  3507     9
      0     0     0     0     0    62]
 [    0     0     0    48   107     0     0     0     0     0    15   328
     36     0     0     0     0     0]
 [    0     0     0     0     0     0     0     0     0     0     0     3
   1000     0     0     0     0     0]
 [   45     1    17     0     3     0     0    26     6    31     8     0
      0    96     0     0     0     3]
 [   41   145    22     0    76    95    91    12     4     2    40     0
      0     0   536   205     4     0]
 [   85    86    15     0    88   786   103    46     1     1    25     0
      0     0   155   574     4     0]
 [    0    17    16     1     4     0     0    19     0     0     0     0
      0     0     9     6    40     0]
 [   32     5     0     7    18     0     0    98     0     0   278     0
      4     0     0     0     0   215]]
```

Random Tree classifier results:

```python
# @title Random Forest Classifier Results
# Here we are printing the classification report
print("Classification Report:")
# Here we are comparing y_test,y_pred using classification_report()
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.93      0.88      0.91     14940
           2       0.62      0.45      0.52      1529
           3       0.86      0.67      0.75      1614
           4       0.73      0.64      0.68      1104
           5       0.97      1.00      0.98     25184
           6       0.59      0.63      0.61      3012
           7       0.28      0.15      0.20       731
           8       0.84      0.98      0.90     19306
           9       0.25      0.17      0.20        64
          10       0.80      0.58      0.68       563
          11       0.82      0.67      0.74      5201
          12       0.70      0.61      0.65       534
          13       0.93      1.00      0.96      1003
          14       0.74      0.41      0.53       236
          15       0.55      0.42      0.48      1273
          16       0.37      0.29      0.33      1969
          17       0.62      0.36      0.45       112
          18       0.55      0.33      0.41       657

    accuracy                           0.86     79032
   macro avg       0.67      0.57      0.61     79032
weighted avg       0.86      0.86      0.86     79032
```

Comparison:

To check the most efficient model we compared several metrics such as accuracy and classification report. For the K-Nearest Neighbor Model We got an accuracy of 85.327 and for Random tree classifier we got an accuracy of 86.34.

Classification report for K-Nearest Neighbor:

```
# @title K-Nearest Neighbors (KNN) Model Results
# Here we are printing the classification report
print("Classification Report:")
# Here we are comparing y_test,y_pred using classification_report()
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.90      0.88      0.89     14940
           2       0.58      0.50      0.54      1529
           3       0.86      0.68      0.76      1614
           4       0.72      0.66      0.69      1104
           5       0.97      0.99      0.98     25184
           6       0.57      0.67      0.62      3012
           7       0.29      0.16      0.20       731
           8       0.84      0.92      0.88     19306
           9       0.56      0.14      0.23        64
          10       0.79      0.64      0.71       563
          11       0.72      0.71      0.72      5201
          12       0.70      0.58      0.63       534
          13       0.93      1.00      0.96      1003
          14       0.88      0.39      0.54       236
          15       0.57      0.42      0.49      1273
          16       0.38      0.22      0.28      1969
          17       0.93      0.33      0.49       112
          18       0.51      0.36      0.42       657

    accuracy                           0.85     79032
   macro avg       0.70      0.57      0.61     79032
weighted avg       0.84      0.85      0.85     79032
```

Classification report for Random Forest Classifier:

```
# @title Random Forest Classifier Results
# Here we are printing the classification report
print("Classification Report:")
# Here we are comparing y_test,y_pred using classification_report()
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           1       0.93      0.88      0.91     14940
           2       0.62      0.45      0.52      1529
           3       0.86      0.67      0.75      1614
           4       0.73      0.64      0.68      1104
           5       0.97      1.00      0.98     25184
           6       0.59      0.63      0.61      3012
           7       0.28      0.15      0.20       731
           8       0.84      0.98      0.90     19306
           9       0.25      0.17      0.20        64
          10       0.80      0.58      0.68       563
          11       0.82      0.67      0.74      5201
          12       0.70      0.61      0.65       534
          13       0.93      1.00      0.96      1003
          14       0.74      0.41      0.53       236
          15       0.55      0.42      0.48      1273
          16       0.37      0.29      0.33      1969
          17       0.62      0.36      0.45       112
          18       0.55      0.33      0.41       657

    accuracy                           0.86     79032
   macro avg       0.67      0.57      0.61     79032
weighted avg       0.86      0.86      0.86     79032
```

By comparing both reports we can conclude that random forest classifier is a bit better when compared with k-nearest neighbors.

Final submission:

To analyze the impact of the marketing strategies using point plot, we have considered Gender, Age, City category, Occupation, Stay_In_Current_City_Years, Marital_Status, product category1, product category2, product category3.

This is how we have implemented the impact of marketing strategy using point plot using gender, age, city category and all other attributes.

```
# @title # Analyzing the Impact of Marketing Strategies on Purchase Amount for Gender
#Importing the required libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Here i am loading the black friday sales dataset
Sales = pd.read_csv('Sales 3.csv')

# Here for Gender i am analyzing the impact of marketing strategies
Gender = Sales.groupby('Gender')['Purchase'].sum().reset_index()

# Here we setting the figure size
plt.figure(figsize=(10, 6))
# Here we are creating the point plot for Gender as x-axis and Purchase as y-axis using point plot() command
sns.pointplot(x='Gender', y='Purchase', data=Gender)
# We are adding a title to the plot
plt.title('Impact of Marketing Strategies on Purchase Amount For Gender')
# We are adding x-label to the plot
plt.xlabel('Gender')
# We are adding y-label to the plot
plt.ylabel('Purchase Amount')
# Displaying the plot using show() command
plt.show()
```

Age:

```
    # Here i am loading the black friday sales dataset
    Sales = pd.read_csv('Sales 3.csv')

    # Here for Age i am analyzing the impact of marketing strategies
    Age = Sales.groupby('Age')['Purchase'].mean().reset_index()

    # Here we setting the figure size
    plt.figure(figsize=(10, 6))
    # Here we are creating the point plot for Age as x-axis and Purchase as y-axis using point plot() command
    sns.pointplot(x='Age', y='Purchase', data=Age)
    # We are adding a title to the plot
    plt.title('Impact of Marketing Strategies on Purchase Amount For Age')
    # We are adding x-label to the plot
    plt.xlabel('Age')
    # We are adding y-label to the plot
    plt.ylabel('Purchase Amount')
    # Displaying the plot using show() command
    plt.show()
```

City category:

```
# Here for City_Category i am analyzing the impact of marketing strategies
City_Category = Sales.groupby('City_Category')['Purchase'].mean().reset_index()

# Here we setting the figure size
plt.figure(figsize=(10, 6))
# Here we are creating the point plot for City_Category as x-axis and Purchase as y-axis using point plot() command
sns.pointplot(x='City_Category', y='Purchase', data=City_Category)
# We are adding a title to the plot
plt.title('Impact of Marketing Strategies on Purchase Amount For City_Category')
# We are adding x-label to the plot
plt.xlabel('Age')
# We are adding y-label to the plot
plt.ylabel('Purchase Amount')
# Displaying the plot using show() command
plt.show()
```

Occupation:

```
# Here for Occupation i am analyzing the impact of marketing strategies
Occupation = Sales.groupby('Occupation')['Purchase'].mean().reset_index()

# Here we setting the figure size
plt.figure(figsize=(10, 6))
# Here we are creating the point plot for Occupation as x-axis and Purchase as y-axis using point plot() command
sns.pointplot(x='Occupation', y='Purchase', data=Occupation)
# We are adding a title to the plot
plt.title('Impact of Marketing Strategies on Purchase Amount For Occupation')
# We are adding x-label to the plot
plt.xlabel('Occupation')
# We are adding y-label to the plot
plt.ylabel('Purchase Amount')
# Displaying the plot using show() command
plt.show()
```

Marital Status:

```
# Here for Marital_Status i am analyzing the impact of marketing strategies
Marital_Status = Sales.groupby('Marital_Status')['Purchase'].mean().reset_index()

# Here we setting the figure size
plt.figure(figsize=(10, 6))
# Here we are creating the point plot for Marital_Status as x-axis and Purchase as y-axis using point plot() command
sns.pointplot(x='Marital_Status', y='Purchase', data=Marital_Status)
# We are adding a title to the plot
plt.title('Impact of Marketing Strategies on Purchase Amount For Marital_Status')
# We are adding x-label to the plot
plt.xlabel('Marital_Status')
# We are adding y-label to the plot
plt.ylabel('Purchase Amount')
# Displaying the plot using show() command
plt.show()
```

We have conducted the Anova test to determine how the independent factors product category1, product category2, product category3 are affecting the dependent variable purchase.

Performing One-Way ANOVA for Product Categories in Black Friday Sales Dataset

```
# @title Performing One-Way ANOVA for Product Categories in Black Friday Sales Dataset
# Importing the required libraries
from scipy.stats import f_oneway
import pandas as pd

# Assuming 'Sales' DataFrame has columns 'Product_Category_1', 'Product_Category_2', 'Product_Category_3'
Sales = pd.read_csv('Sales 3.csv')

# Here from the datset i am extracting the product_category1,product_category_2,product_category_3 columns as arrays
Product_Category_1 = Sales['Product_Category_1']
Product_Category_2 = Sales['Product_Category_2']
Product_Category_3 = Sales['Product_Category_3']

# Here i am calculating the f_statistic,p-values Using the f_oneway function
F_Statistic, P_Value = f_oneway(Product_Category_1, Product_Category_2, Product_Category_3)

# Displaying the one way anova results of f-statistic and p-value
print("One Way ANOVA Results:")
# Printing the F-Statistic
print("F-Statistic:", F_Statistic)
# Printing the P-Value
print("P-Value:", P_Value)

# Here we are setting the alpha value as 0.05
alpha = 0.05
# Here i am checking the p_value < alpha and printing the results
if P_Value < alpha:
    print("Reject the null hypothesis. There is a significant difference between means of the groups.")
else:
    print("Fail to reject the null hypothesis. There is no significant difference between means of the groups.")
```

Preliminary Results:

This section contains all the results of the analysis on Black Friday sales which goes from data preprocessing to the evaluation of the model.

Starting with the exploratory data analysis, we first read the dataset.

Reading The Black Friday Sales Dataset.

```
# @title Reading The Black Friday Sales Dataset.
# Reading the csv file
Sales = pd.read_csv("Black Friday Sales.csv")
# Here we are displaying the top 5 rows.
Sales.head(5)
```

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Prc
0	1000001	P00069042	F	0-17	10	A	2	0	3	
1	1000001	P00248942	F	0-17	10	A	2	0	1	
2	1000001	P00087842	F	0-17	10	A	2	0	12	
3	1000001	P00085442	F	0-17	10	A	2	0	12	
4	1000002	P00285442	M	55+	16	C	4+	0	8	

Dropping of the duplicate rows and irrelevant columns is done.

Dropping The Irrelevant Columns

```
# @title Dropping The Irrelevant Columns
#Here we are dropping the irrelevant columns, In my case i am dropping the Product_ID
Sales = Sales.drop(['Product_ID'], axis=1)
# Here we are displaying the top 5 rows
Sales.head(5)
```

	User_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Product_Category_1	Product_Categor
0	1000001	F	0-17	10	A	2	0	3	
1	1000001	F	0-17	10	A	2	0	1	
2	1000001	F	0-17	10	A	2	0	12	
3	1000001	F	0-17	10	A	2	0	12	
4	1000002	M	55+	16	C	4+	0	8	

Dropping The Duplicate Rows

```
[ ] # @title Dropping The Duplicate Rows
    # Here we are finding the duplicate rows using the duplicated() command.
    rows_duplicate = Sales[Sales.duplicated()]
    # Printing the duplicate rows.
    print("Number of duplicate rows: ", rows_duplicate.shape)
```

```
Number of duplicate rows:  (2543, 11)
```

```
[ ] #Here i am counting the number of rows.
    Sales.count()
```

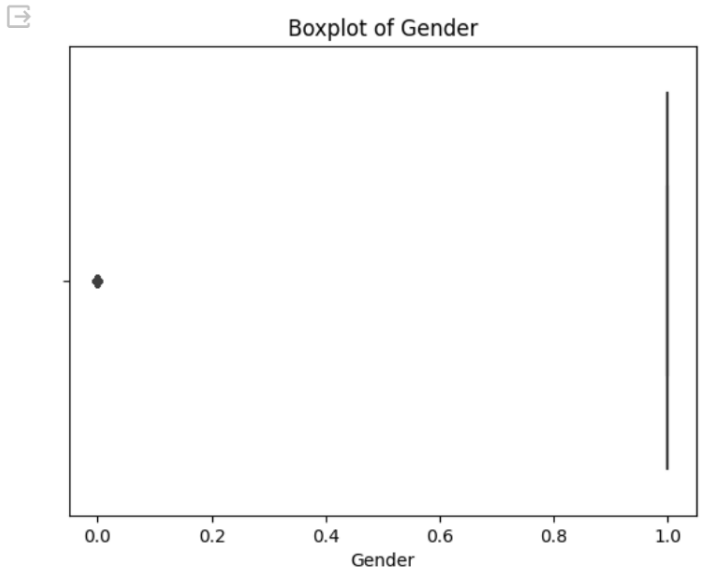
```
User_ID                     550068
Gender                      550068
Age                         550068
Occupation                  550068
City_Category               550068
Stay_In_Current_City_Years  550068
Marital_Status              550068
Product_Category_1          550068
Product_Category_2          376430
Product_Category_3          166821
Purchase                    550068
dtype: int64
```

After dropping the duplicate rows and irrelevant columns we got the data count.

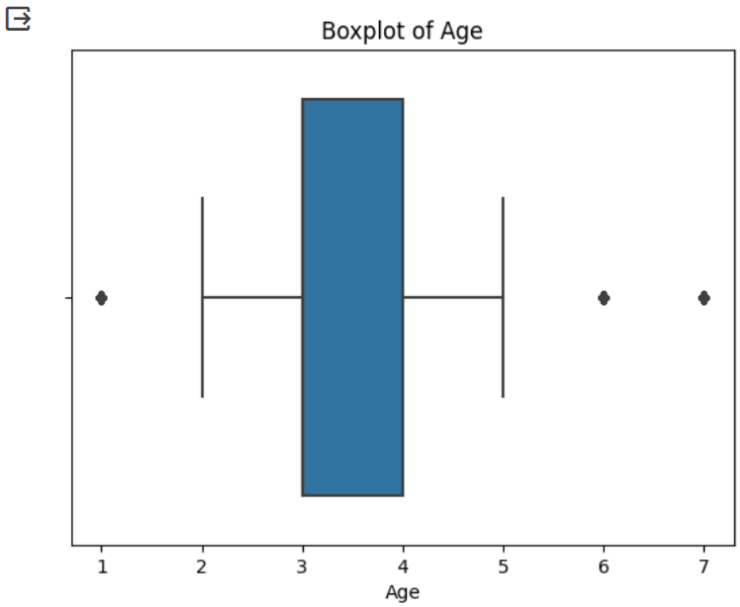
```
# After dropping the duplicate rows again i am counting the number of rows.
Sales.count()
```

```
User_ID                     547525
Gender                      547525
Age                         547525
Occupation                  547525
City_Category               547525
Stay_In_Current_City_Years  547525
Marital_Status              547525
Product_Category_1          547525
Product_Category_2          375825
Product_Category_3          166734
Purchase                    547525
dtype: int64
```

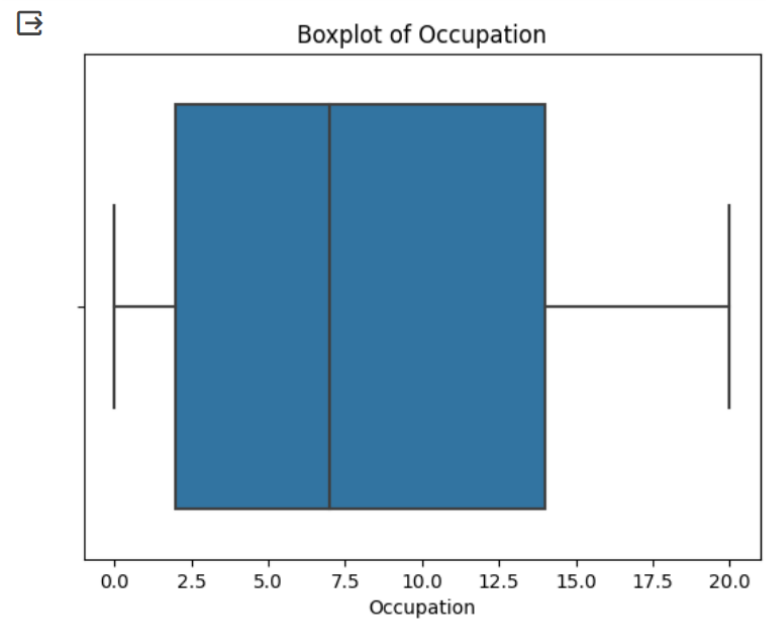
Outliners detection for Gender.



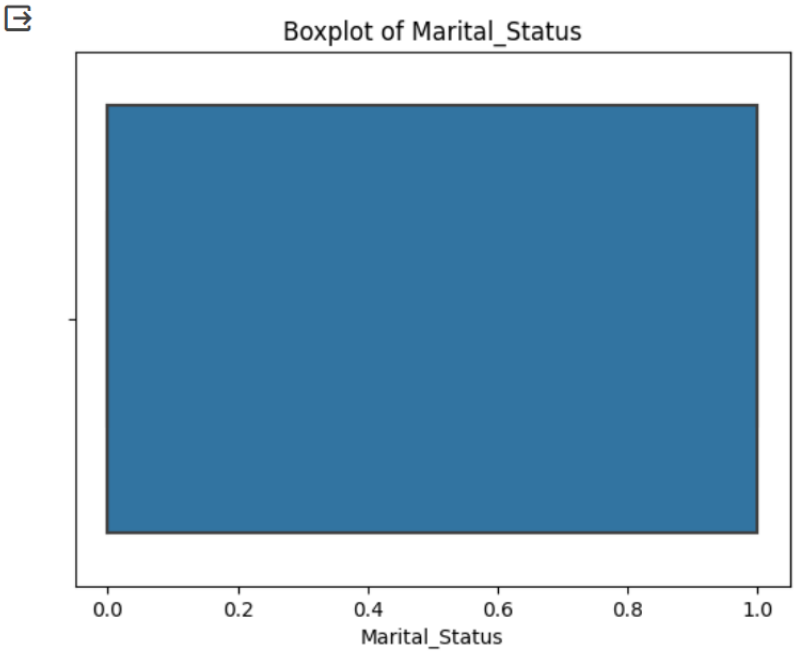
Outliners detection for Age using the boxplot.



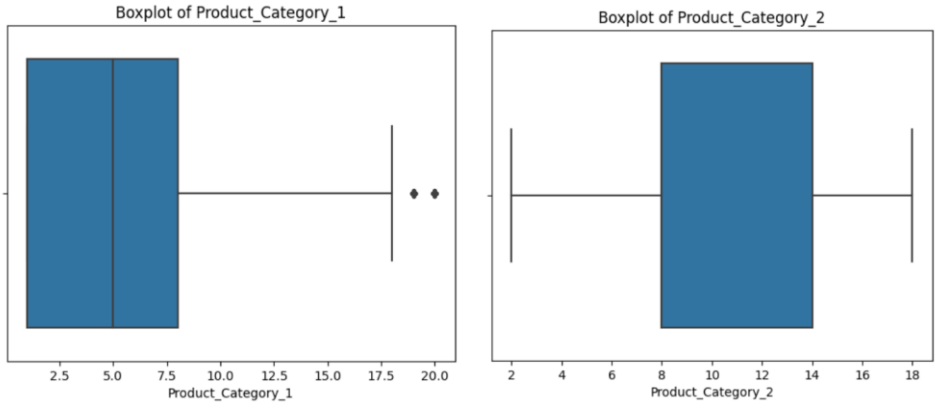
Outliners detection for Occupation

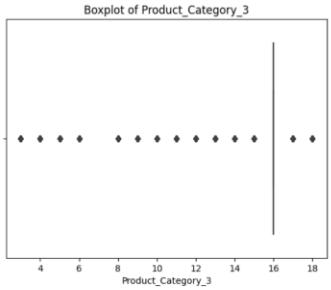


Outliners detection for Marital_status attribute using the box plot.

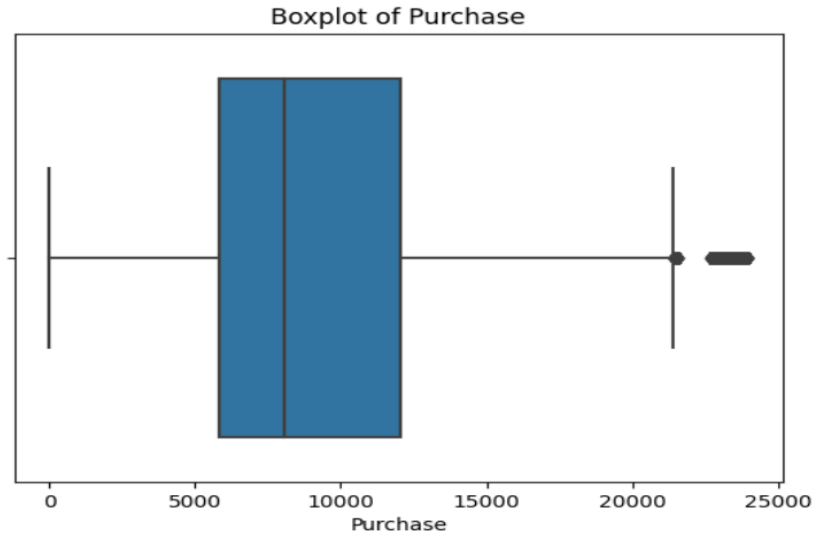


Outliners for product_category1, product_category2, product_category3.

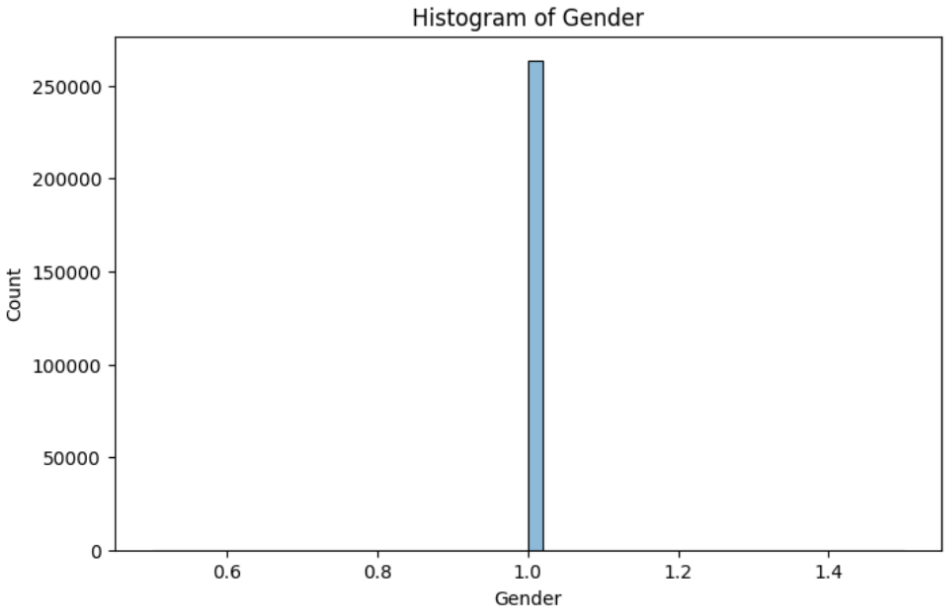




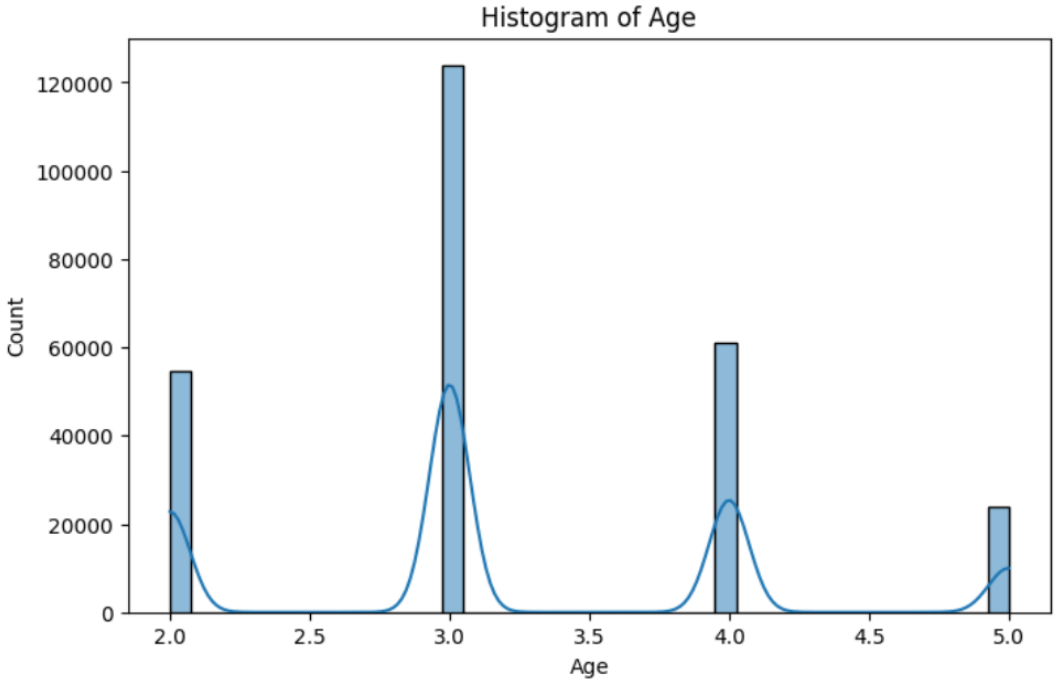
Visualization of the outliners for purchase attribute.



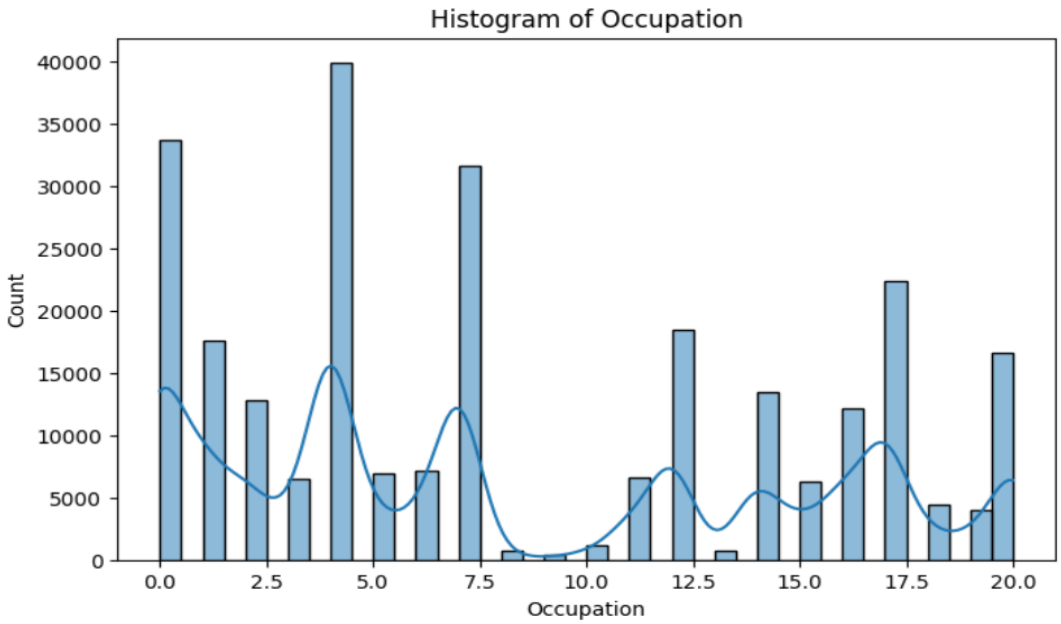
Pictorial result of the univariate analysis for Gender attribute using histogram.



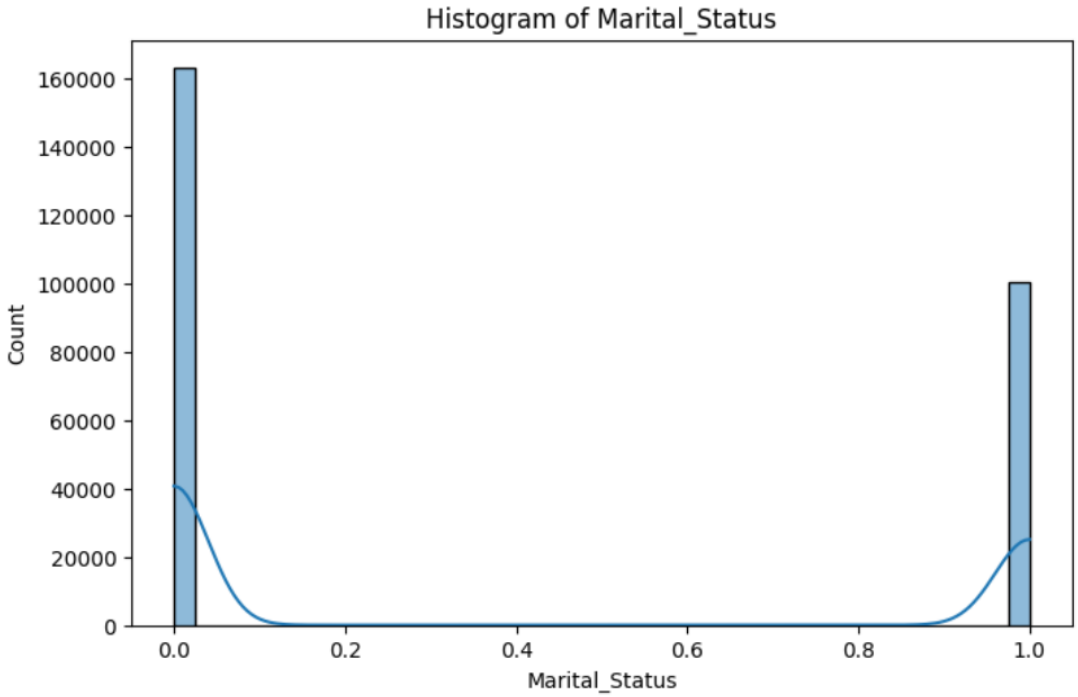
Pictorial result of the univariate analysis for Age attribute using histogram.



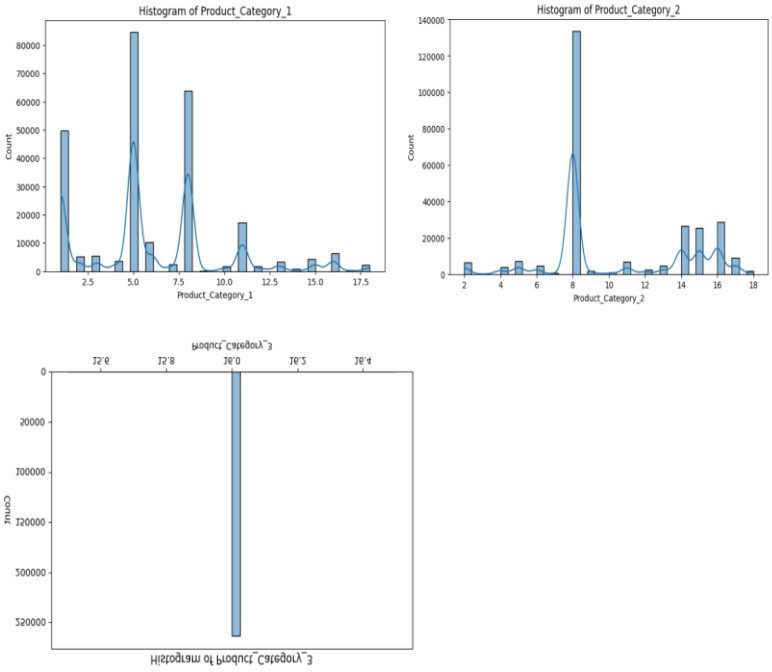
Pictorial result of the univariate analysis for Occupation attribute using histogram.



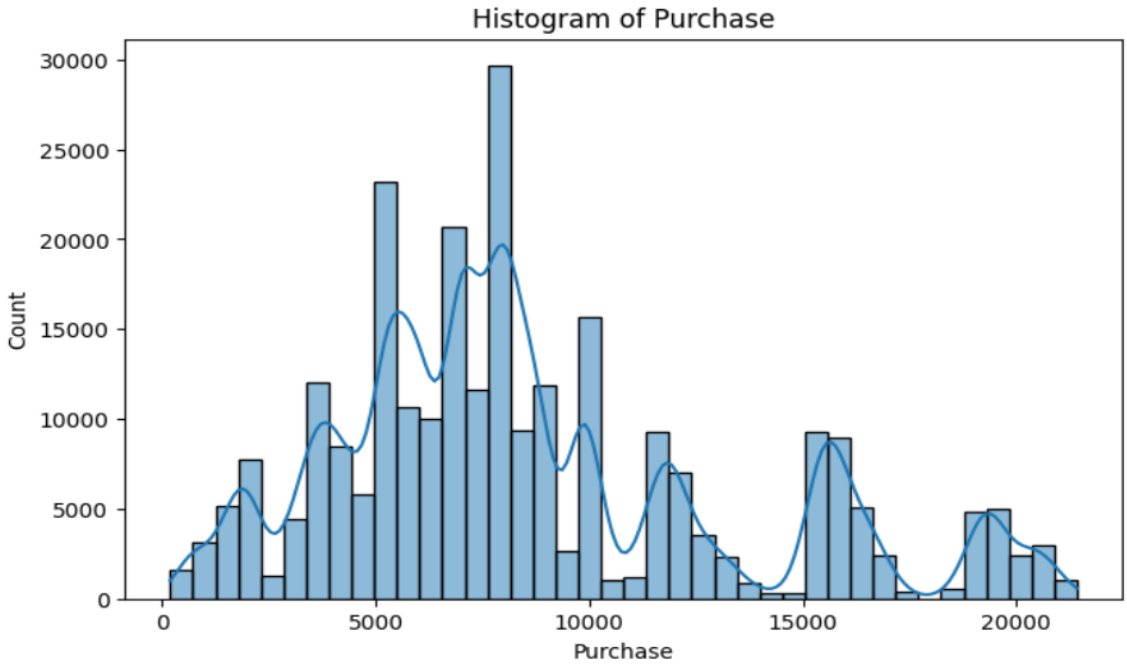
Pictorial result of the univariate analysis for Marital_status attribute using histogram.



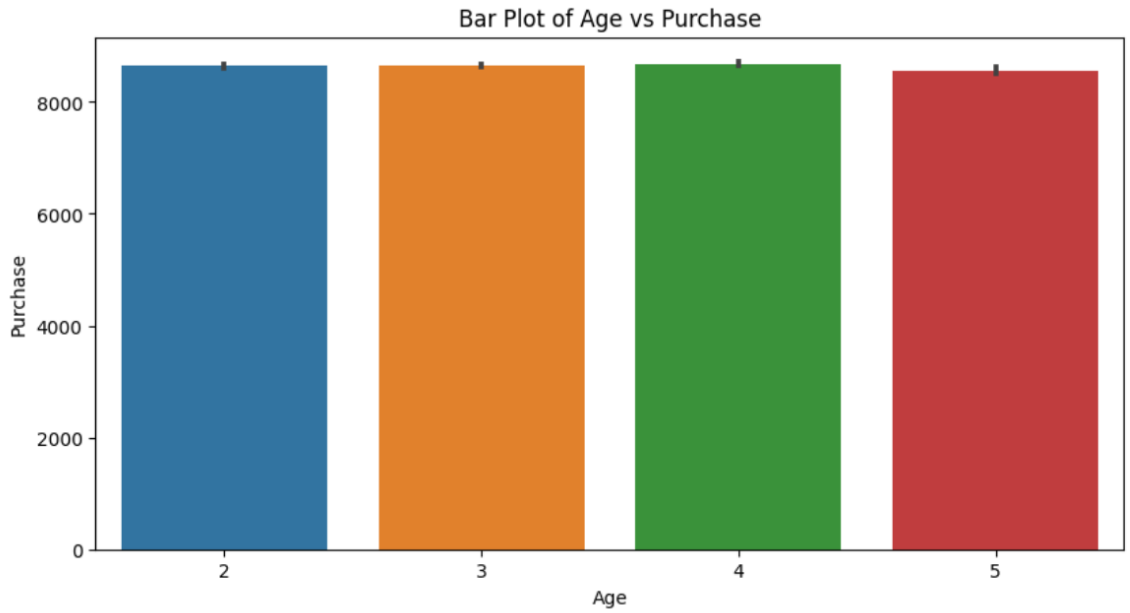
Pictorial result of the univariate analysis for Product_category_1, product_category_2, product_category_3 attribute using histogram.



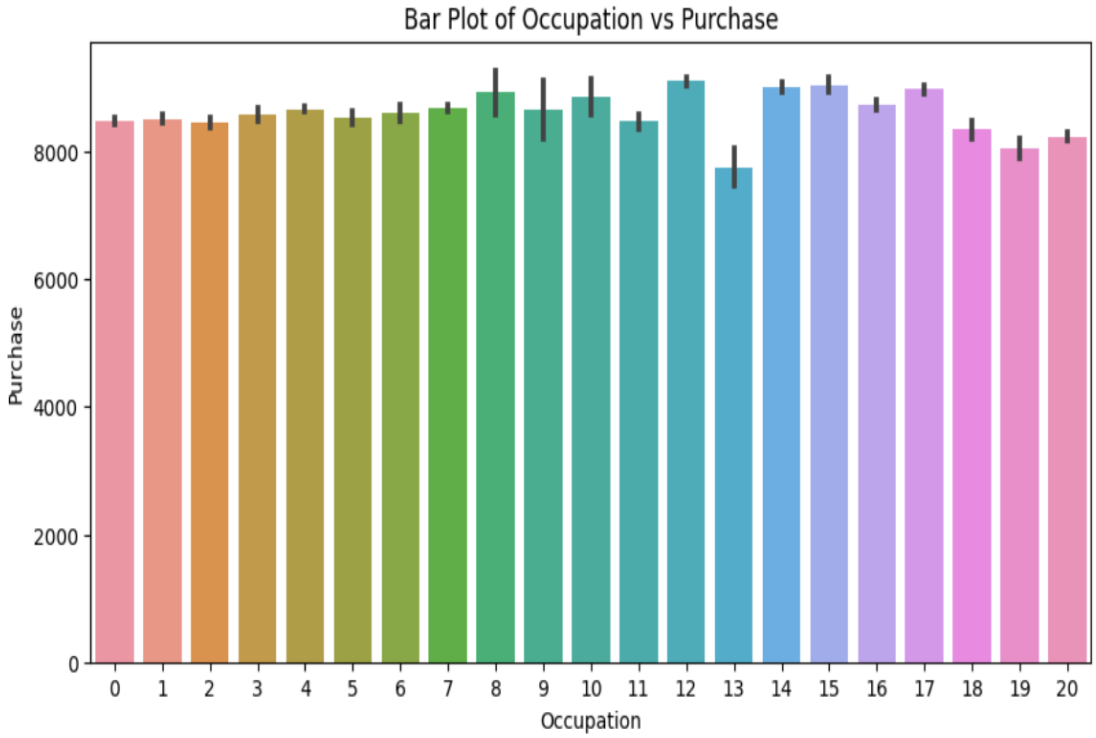
Pictorial result of the univariate analysis for purchase attribute using histogram.



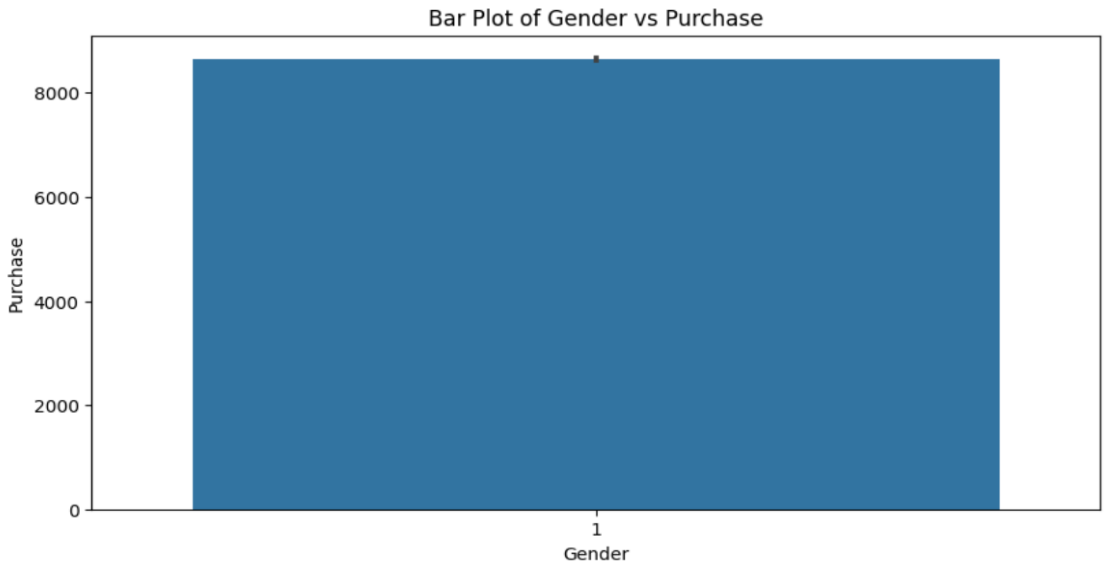
Pictorial representation for bivariate analysis of age and purchase features.



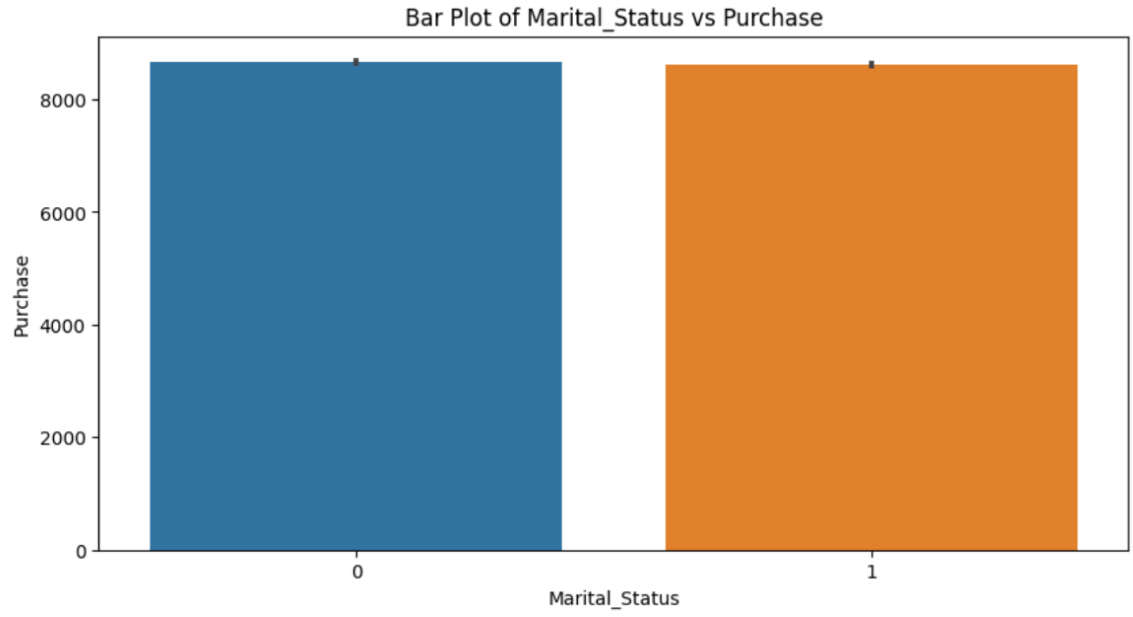
Pictorial representation for bivariate analysis of occupation and purchase features.



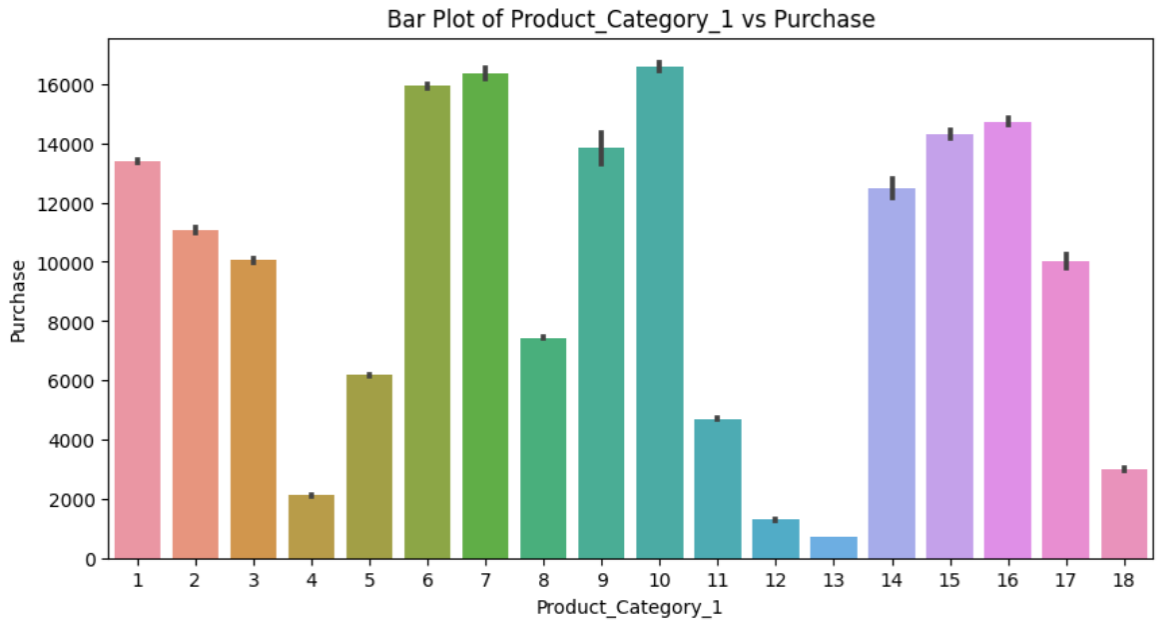
Pictorial representation for bivariate analysis of gender and purchase features using Bar plot.



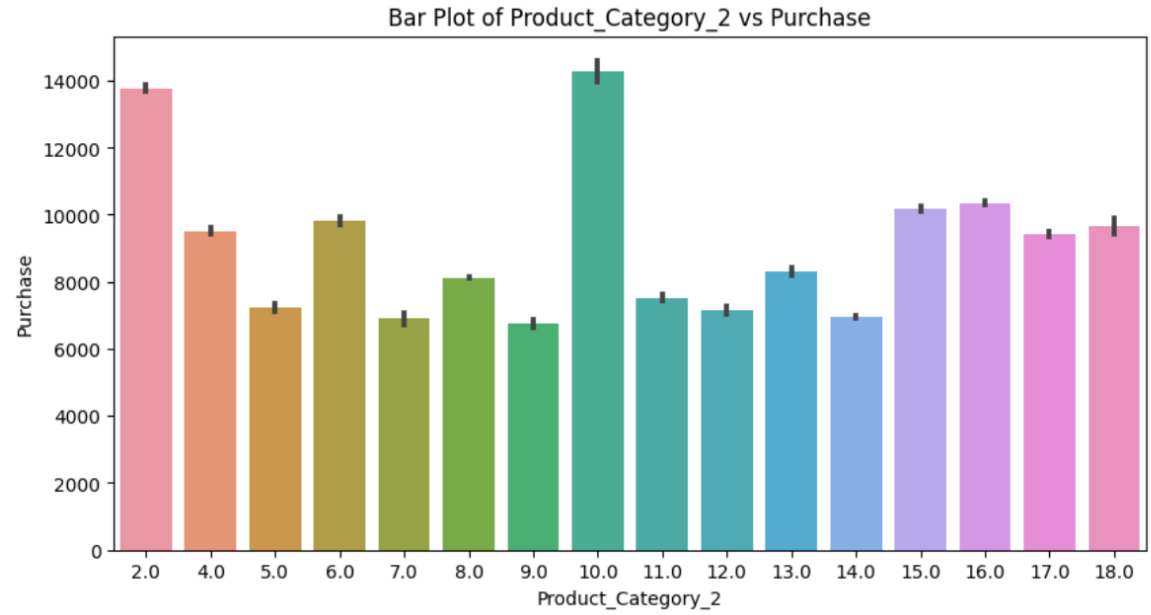
Pictorial representation for bivariate analysis of Marital_status and purchase features using Bar plot.



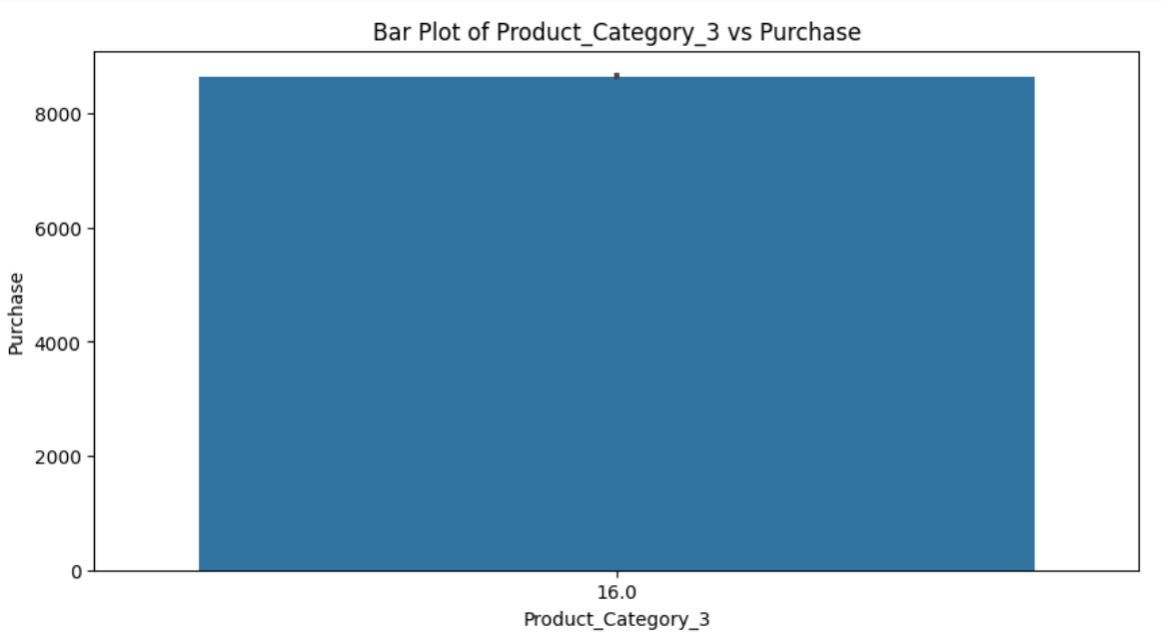
Pictorial representation for bivariate analysis of product_category_1 and purchase features using Bar plot.



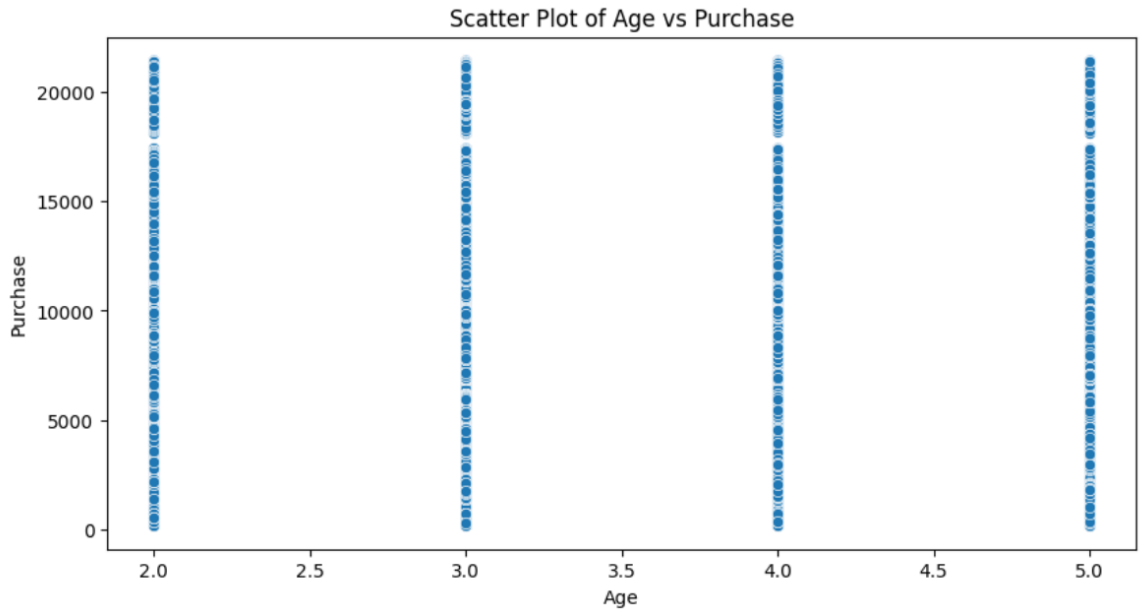
Pictorial representation for bivariate analysis of product_category_2 and purchase features using Bar plot.



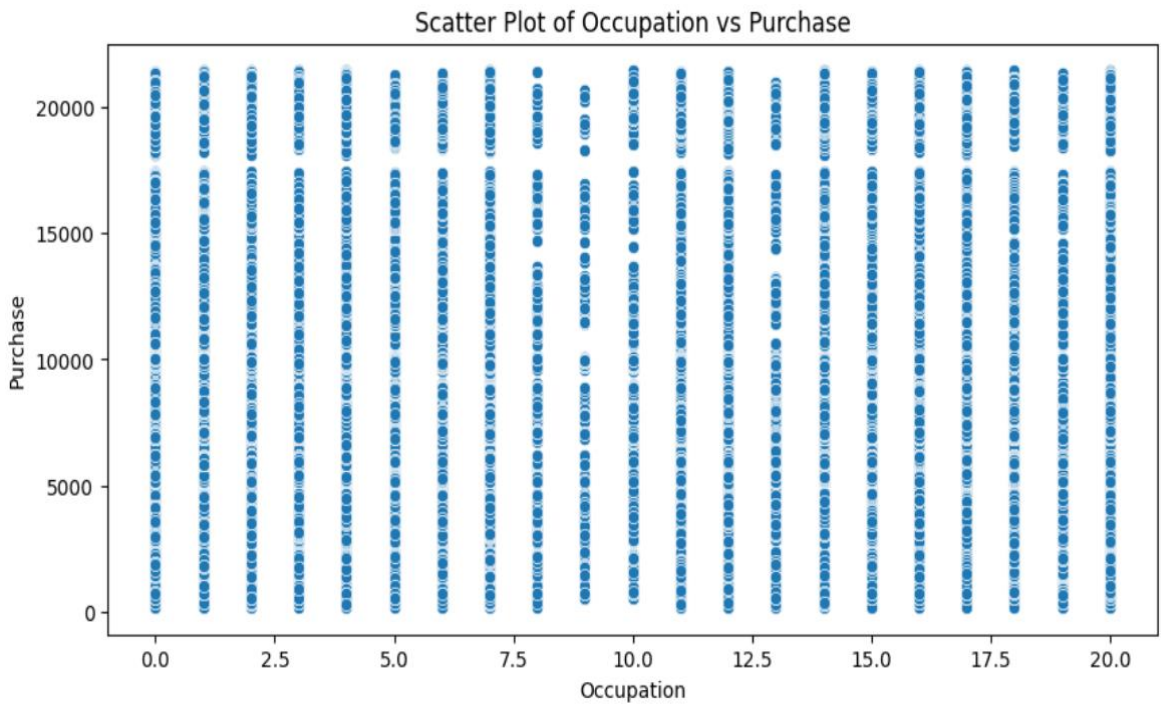
representation for bivariate analysis of product_category_3 and purchase features using Bar plot.



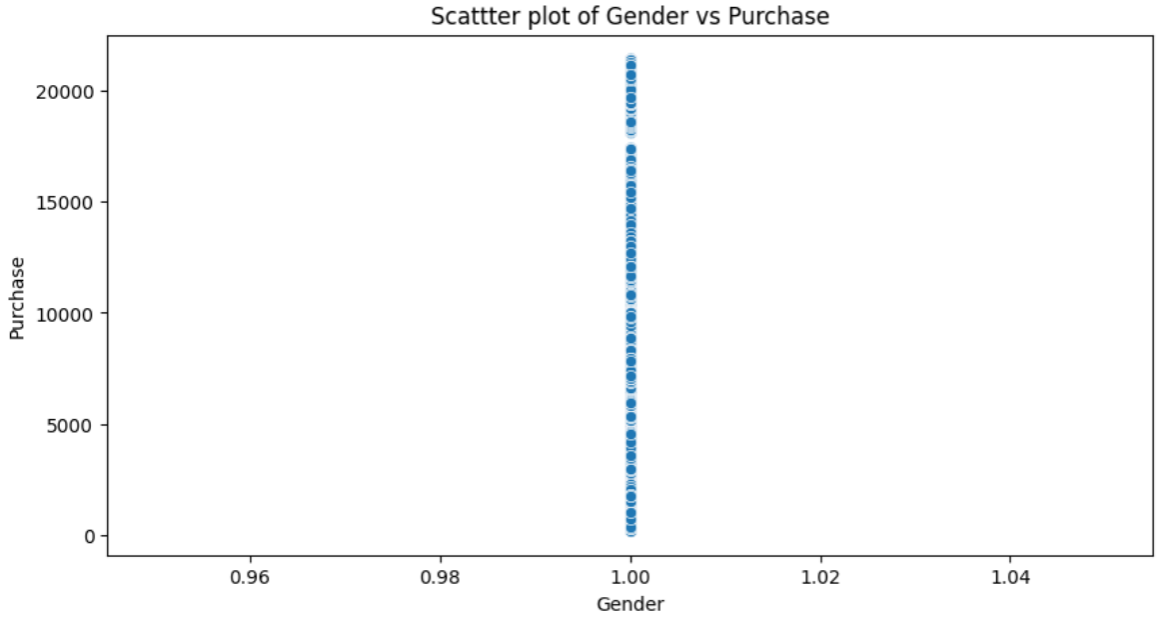
Pictorial representation for bivariate analysis of age and purchase features using scatter plot.



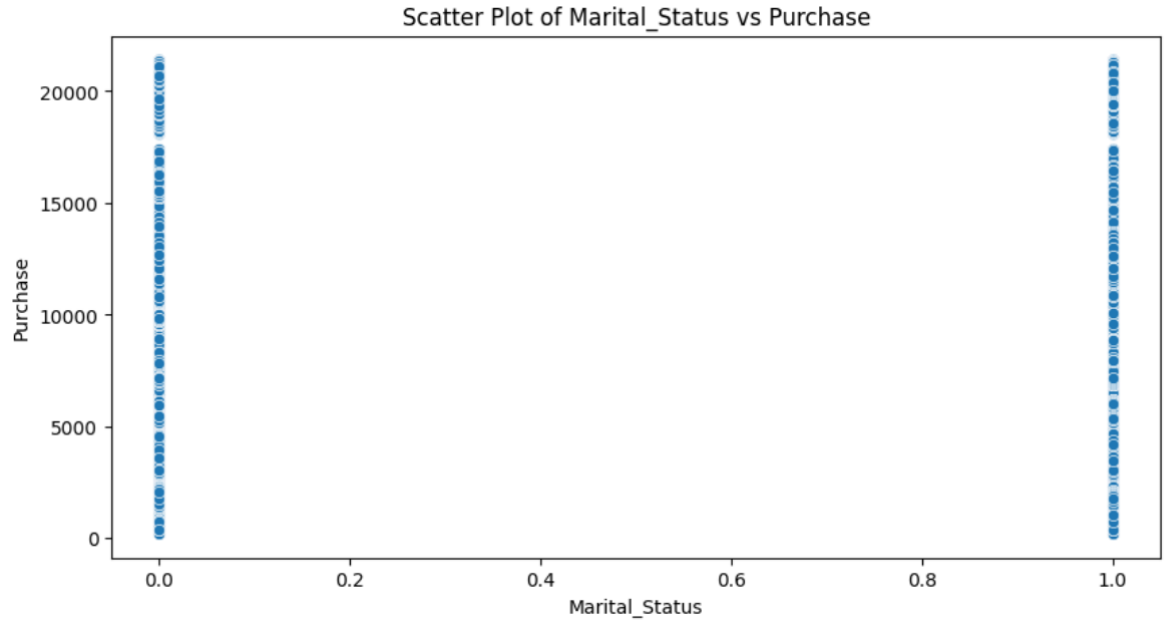
Pictorial representation for bivariate analysis of occupation and purchase features using scatter plot.



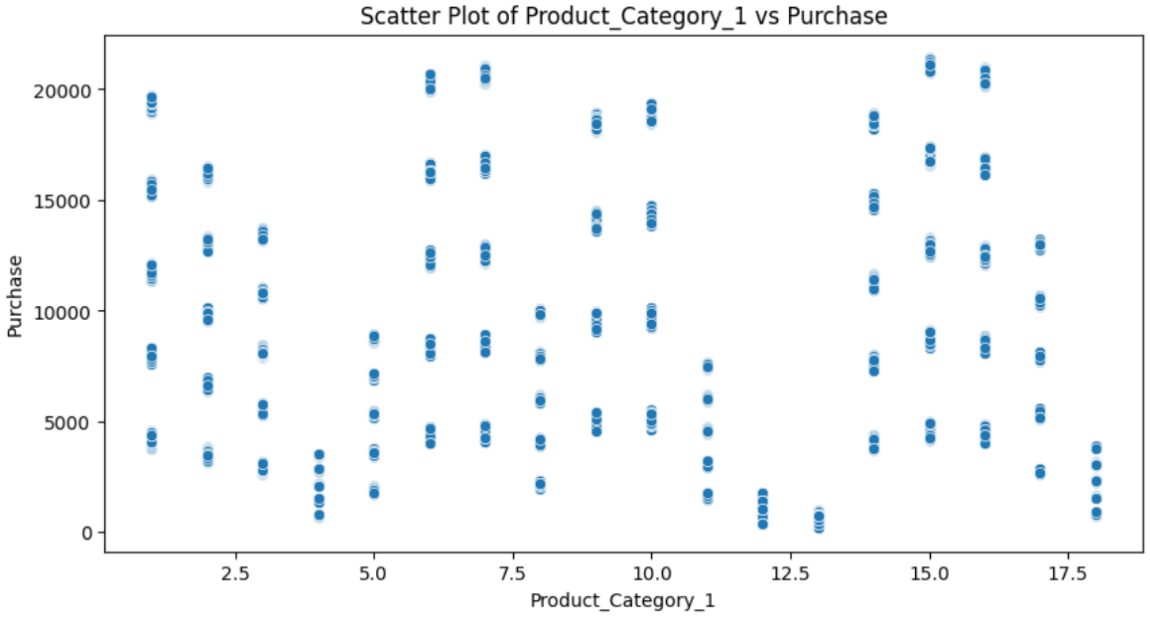
Pictorial representation for bivariate analysis of gender and purchase features using scatter plot.



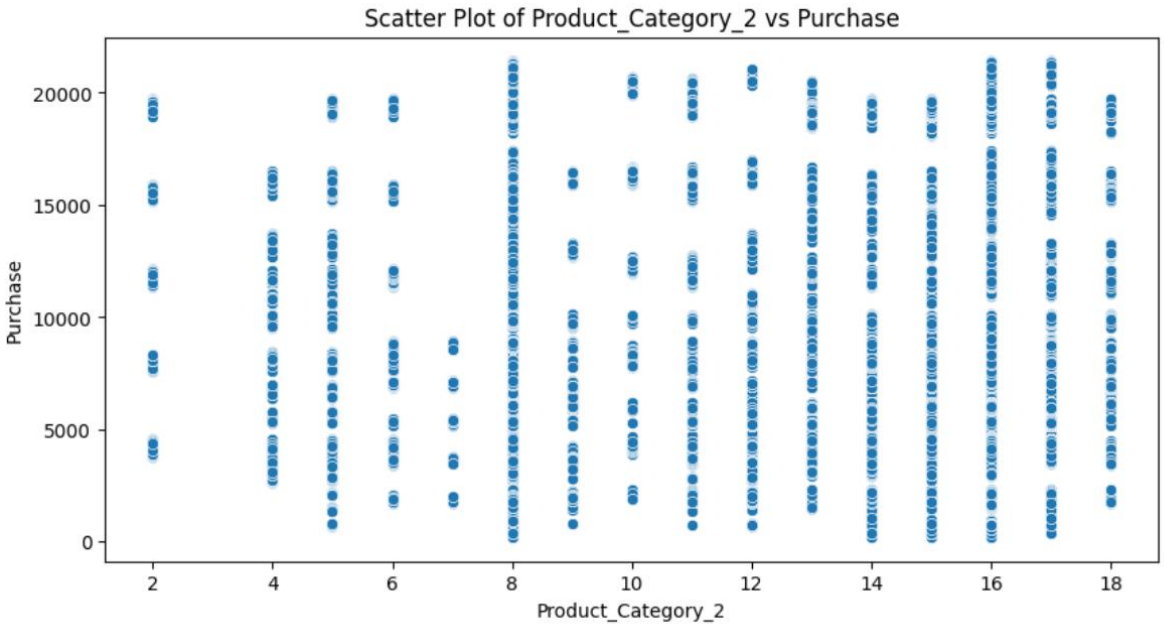
Pictorial representation for bivariate analysis of marital_status and purchase features using scatter plot.



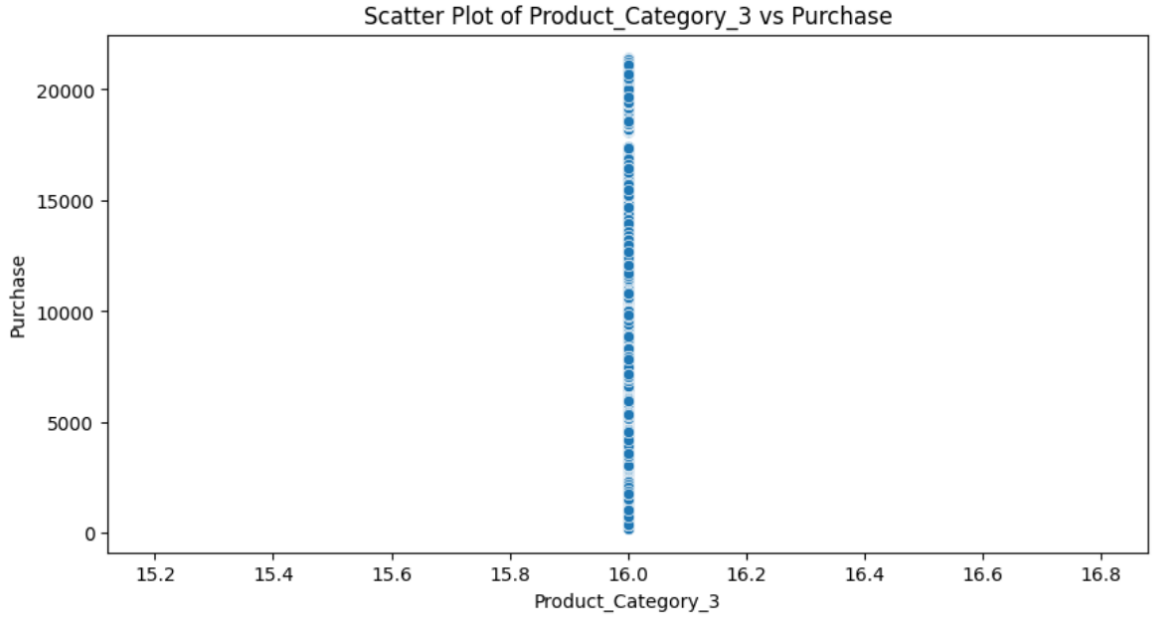
Pictorial representation for bivariate analysis of product_category_1 and purchase features using scatter plot.



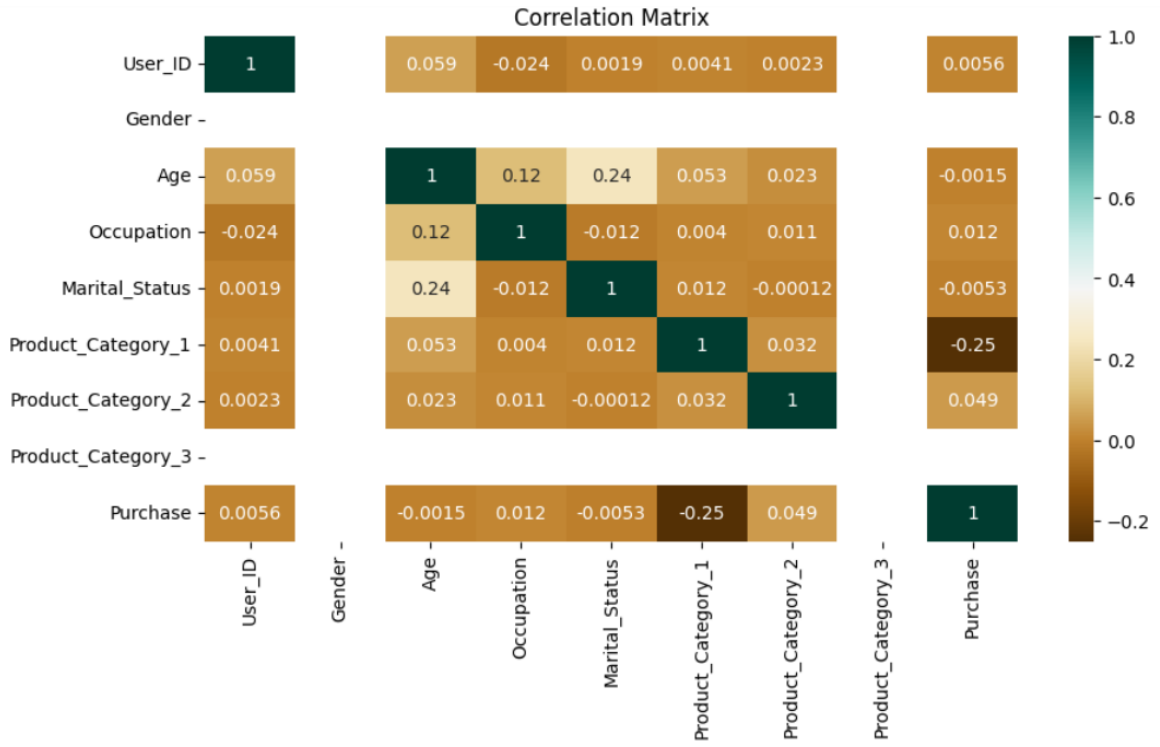
Pictorial representation for bivariate analysis of product_category_2 and purchase features using scatter plot.



Pictorial representation for bivariate analysis of product_category_3 and purchase features using scatter plot.



Correlation analysis using Heatmaps.



Feature selection:

Most important features were retrieved using the selectkbest function.

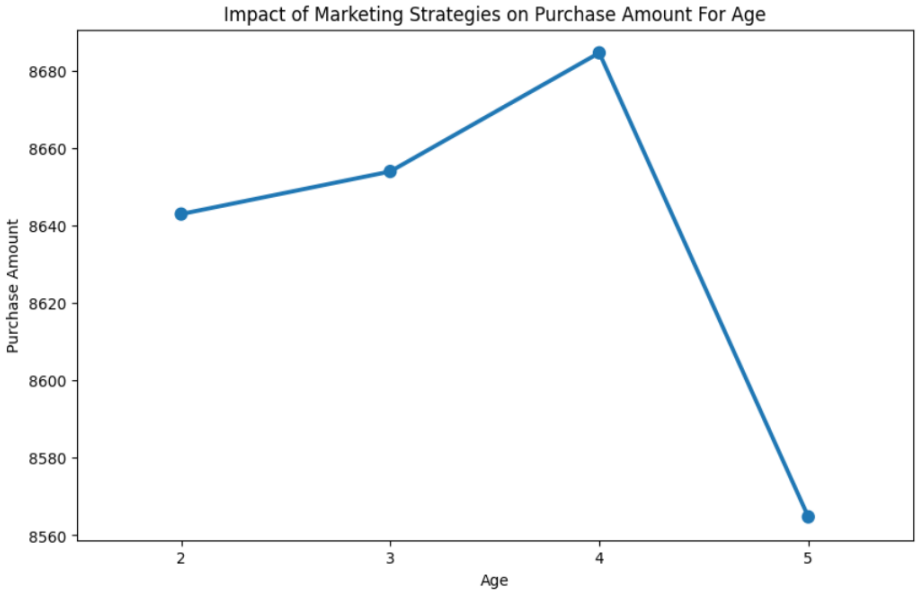
```
Most Important Features: Index(['Age', 'Occupation', 'City_Category', 'Marital_Status',
       'Product_Category_1', 'Product_Category_2', 'Purchase'],
      dtype='object')
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: UserWarning: Features [0 6 8] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:113: RuntimeWarning: divide by zero encountered in divide
  f = msb / msw
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:113: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
```

Final submission:

Here are the results retrieved when we analyzed the impact of marketing strategies.

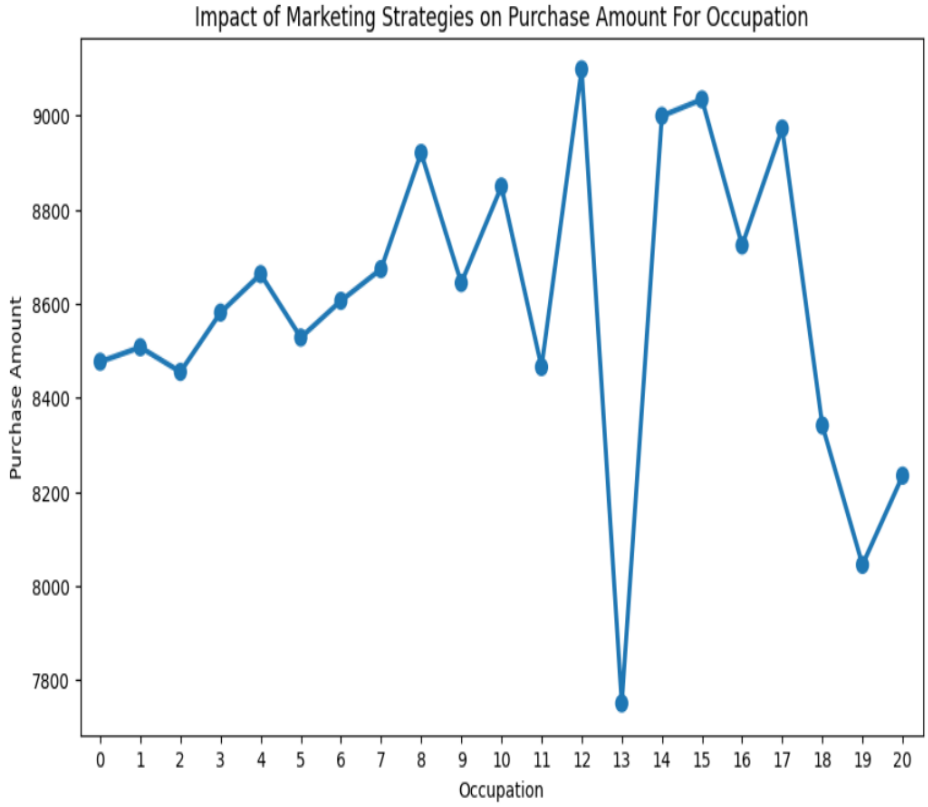
For Gender:

For Age:



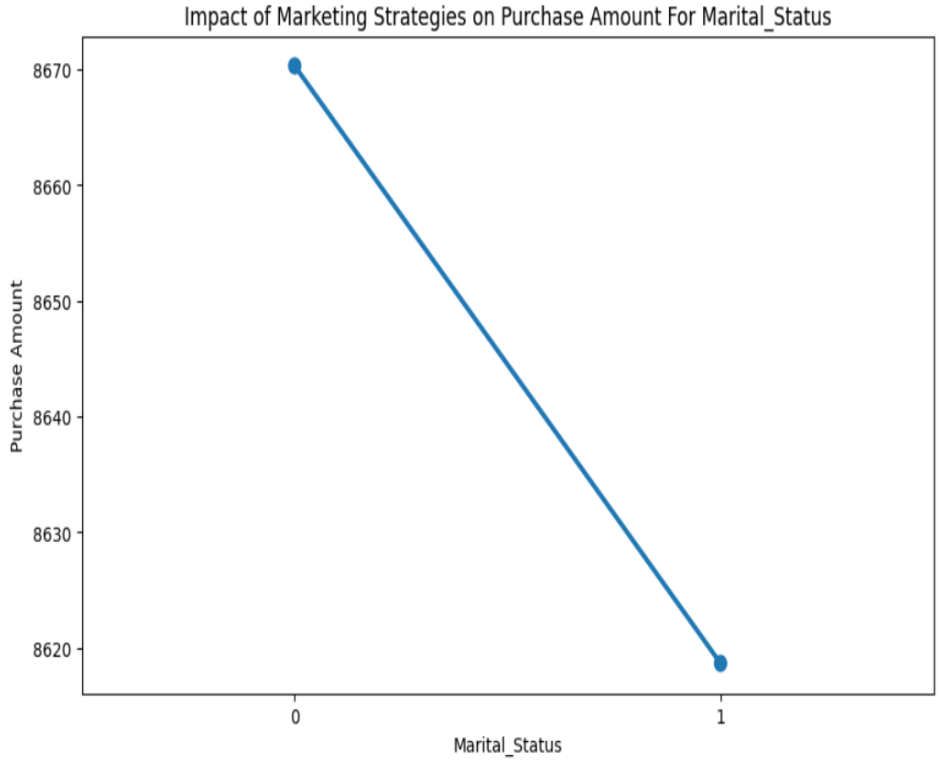
For city category attribute the impact on purchase is:

The marketing strategy on purchase amount for occupation:

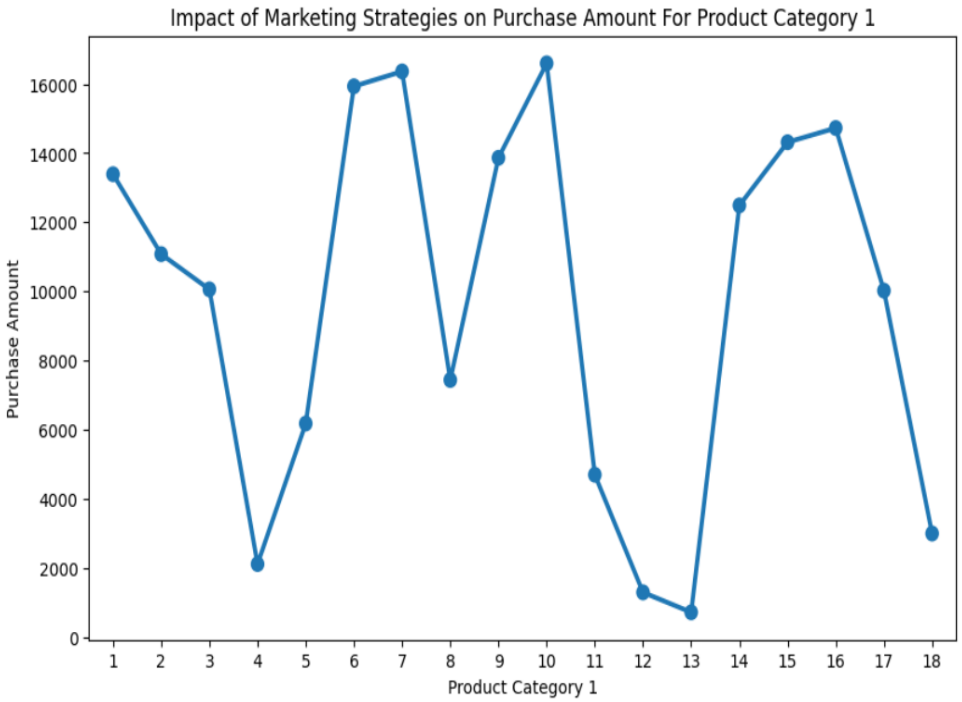


The Marketing strategy on purchase amount on stay in current city years in point plot as:

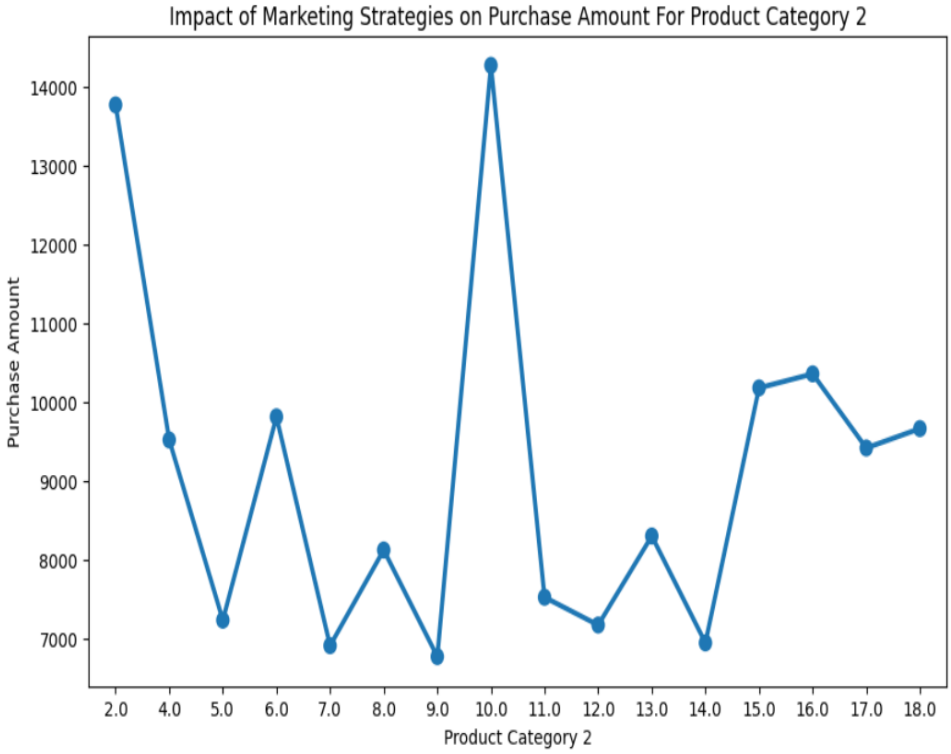
The Marketing strategy on purchase amount on marital status in point plot as:



The Impact of Marketing strategy on purchase amount for product category1 in point plot as:



The Impact of Marketing strategy on purchase amount for product category2 in point plot as:



The Impact of Marketing strategy on purchase amount for product category3 in point plot as:

The preliminary result for the Anova test is as follows:

```
One Way ANOVA Results:
F-Statistic: 661742.637878972
P-Value: 0.0
Reject the null hypothesis. There is a significant difference between means of the groups.
```

Final Submission:

Resampling with K-fold Cross Validation:

We used a K-fold Cross Validation method in this analysis to verify a few things such as model evaluation, whether the model is underfitting or overfitting and to calculate metrics such as mean absolute error and root mean squared absolute error.

We started by importing all the libraries required to perform the K-fold Cross Validation method and extracted the features of ‘x’ by dropping the “Purchase” column from the dataset and extracted features of ‘y’ from that column. Later used one-hot coding on the PurchaseDate column. After that selected the model as decision tree and to perform k-fold cross validation took numbers of splits as 5 and initialized the k-fold method and performed the cross validation and finally calculated and printed the mean squared error. Similarly, we performed root mean squared error too.

Importing the required libraries to perform K-Fold Cross Validation:

```
# @title Importing the required libraries to perfrom KFold cross validation
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
import pandas as pd
from numpy import mean
from numpy import absolute
from numpy import sqrt
```

Reading the black Friday sales dataset:

```
# @title Reading the Black Friday Sales Dataset
Sales = pd.read_csv('BLACK FRIDAY SALES.csv')
```

To evaluate the performance of the model using Mean Squared Error (MSE) we are performing K-Fold Cross Validation:

```
# @title To Evaluate the performance Of The Model We Are Performing KFold Cross Validation
# By dropping the purchase column i am extracting features
X = Sales.drop('Purchase', axis=1)
# Here we are extracting the target variable
y = Sales['Purchase']
# one-hot encoding for a categorical column
X = pd.get_dummies(X, columns=['Purchase Date'])
# Here we are building a model as Decision Tree
Dt_Model = DecisionTreeRegressor()
# Here we are setting the number of splits as 5
n_splits = 5
# Here we are defining the KFold crossvalidation method
kf_method = KFold(n_splits=n_splits, shuffle=True, random_state=42)
# Here we are evaluating the model using KFold Cross Validation
scores = cross_val_score(Dt_Model, X, y, cv=kf_method, scoring='neg_mean_absolute_error')
# Here we are calculating the mean absolute error
mean(absolute(scores))
```

```
2459.4418589729557
```

To evaluate the performance of the model using Root Mean Squared Error (RMSE) we are performing K-Fold Cross Validation:

```
# @title To Evaluate the performance Of The Model Using Root Mean Squared Error(RMSE) We Are Performing KFold Cross Validation
# By dropping the purchase column i am extracting features
X = Sales.drop('Purchase', axis=1)
# Here we are extracting the target variable
y = Sales['Purchase']
# one-hot encoding for a categorical column
X = pd.get_dummies(X, columns=['Purchase Date'])
# Here we are building a model as Decision Tree
Dt_Model = DecisionTreeRegressor()
# Here we are setting the number of splits as 5
n_splits = 5
# Here we are defining the KFold cross validation method
kf_method = KFold(n_splits=n_splits, shuffle=True, random_state=42)
# Here we are evaluating the model using KFold Cross Validation
scores = cross_val_score(Dt_Model, X, y, cv=kf_method, scoring='neg_mean_absolute_error')
# Here we are calculating the root mean squared error
sqrt(mean(absolute(scores)))
```

```
49.59671206154227
```

Project Management:

Work Completed:

- **Description:**
  Chosen a Black Friday sales dataset from Kaggle and have observed the key features and quality of the data. Preprocessed the dataset by dropping irrelevant columns, removing the null values, duplicate values and replacing valueless categorical values with some mapping values.
  Later, Exploratory Data Analysis was performed using Univariate, Bivariate analyses and outlier detection to find the relations between the attributes and to retrieve the most vital feature of the data.
  For Feature selection SelectKBest is used as it selects the most relevant features. Selected Decision Tree classifier as model and trained 70% of the dataset and evaluated the trained model with accuracy, confusion matrix, classification report.

Final submission:

We have also analyzed the impact of marketing strategies on purchase for all the attributes including Gender, Age, Occupation etc. using the point plot for all the features. We have performed the Anova test to determine how the independent variables are affecting and are becoming a factor affecting the dependent variable Purchase. We have also selected to identify the specific gender that has bought more products using the heat maps to know which customer drives the sales more.
We have explored seasonal trends using the time series analysis and along with these we have performed Re sampling technique K - fold Cross validation on the data.

- **Responsibility:**
  Praharsha Mutyala - Exploratory Data Analysis, Detecting Outliers, Resampling Technique-KFold Cross Validation.
  Murali Manobhiram Penumutchu - Model Selection, Model Training and Model Evaluation, K nearest neighbors, Random Forest classifier, Decision Tree, Gaussian NB
  Likhitha Chandanati - Univariate, Bivariate and Correlation Analysis, analyzing specific gender bought more products using heat maps, Analyzing the Impact of Marketing Strategies Using Point Plot.
  Amarnadh Kari- Feature Selection, Exploring Seasonal Trends Using Time Series Analysis, One Way Anova.

- **Contributions:**
  Praharsha - 25%
  Murali Manobhiram - 25%
  Likhitha Chandanati - 25%
  Amarnadh Kari – 25%

References:

1.(https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial) : Kaggle tutorial by prashant for reference

2.(https://medium.com/@1512aliahmad/black-friday-sales-prediction-cd477ebae018) : publication by Ali Ahmad for reference

4. XGBoost, Decision Tree, and Random Forest Tutorials:

a. Towards DataScience: "Complete Guide to Parameter Tuning in XGBoost" - Available at: https://towardsdatascience.com/complete-guide-to-parameter-tuning-in-xgboost-w ith-codes-in-python-7f94d34bc058

b.Scikit learn Documentation: "Decision Trees" - Available at:

https://scikit-learn.org/stable/modules/tree.html.

GitHub Repository Link:

https://github.com/MuraliManobhiRam/Methods-in-Empirical-Analysis-Project-Group13