

Machine Learning report

[Sales Trend Dataset](https://raw.githubusercontent.com/ashishrawani8241/ML-/main/shopping_trends.csv)

[](https://raw.githubusercontent.com/ashishrawani8241/ML-/main/shopping_trends.csv)

Cluster 3 #Batch 2

Electronics and communication Department

NAME: - Ashish Kumar Rawani 22/ECE/056

Contents

[Abstract 3](#_Toc190458111)

[Introduction 4](#_Toc190458112)

[Methodology 5](#_Toc190458113)

[Stepwise Explanation of Code Implementation 6](#_Toc190458114)

[Results and Findings 19](#_Toc190458115)

[Conclusion 20](#_Toc190458116)

[References 21](#_Toc190458117)

# Abstract

This project aims to analyze shopping trends using machine learning techniques and data visualization tools. The dataset contains information about customer purchases, including purchase amounts, product categories, and purchase dates. The primary goal is to extract meaningful insights, such as identifying seasonal shopping trends, understanding customer loyalty behavior, and analyzing feature correlations. To achieve this, various data analysis techniques were employed, including data preprocessing, correlation heatmaps, and data visualization using Seaborn. The project focuses on understanding patterns in shopping behaviors through statistical and graphical representation, enabling businesses to make data-driven decisions. By analyzing the dataset, businesses can optimize marketing strategies, predict future shopping trends, and improve customer engagement. The insights gained from this study can help in identifying key periods of high sales, understanding spending patterns across different customer segments, and optimizing inventory management.

# Introduction

Understanding shopping trends is essential for businesses to optimize their strategies and improve customer satisfaction. Consumers exhibit different purchasing behaviors based on various factors, including seasonal changes, discounts, product availability, and personal preferences. By leveraging data science techniques, companies can extract meaningful insights from customer purchase data, enabling them to tailor their marketing and inventory management strategies.

This project aims to analyze a dataset containing customer transactions and extract insights using data visualization and machine learning techniques. The key objectives include:

* Identifying seasonal shopping trends.
* Understanding customer loyalty and its impact on purchases.
* Discovering correlations between different product categories.
* Analyzing customer spending behavior based on various features.

Using Python libraries such as Pandas, NumPy, Seaborn, and Matplotlib, we perform extensive exploratory data analysis (EDA). Additionally, machine learning techniques can be applied to predict future sales trends and classify customers based on their purchasing habits. The results of this study can be useful for businesses in multiple ways, including improving targeted advertising, enhancing customer retention strategies, and optimizing inventory levels.

# Methodology

This project follows a structured approach to analyzing shopping trends using data science techniques. The methodology consists of the following key steps:

1. **Data Collection and Preprocessing**

* The dataset includes customer transactions with fields such as purchase amount, product category, purchase date, and customer loyalty status.
* Data cleaning involves handling missing values, removing duplicates, and standardizing column names to ensure consistency.
* Data formatting includes converting date columns into appropriate datetime formats for time-series analysis.

1. **Exploratory Data Analysis (EDA)**

* Generating descriptive statistics to understand the distribution of data.
* Identifying correlations between variables using heatmaps.
* Visualizing customer spending patterns across different seasons and product categories.

1. **Data Visualization**

* Box plots are used to display variations in purchase amounts across different customer loyalty statuses.
* Line plots help visualize monthly trends in customer spending.
* Bar charts illustrate the distribution of purchases among different product categories.

1. **Feature Engineering**

* Extracting new features such as ‘Month’ from purchase dates to analyze seasonal patterns.
* Creating customer segments based on spending behavior and purchase frequency.

1. **Machine Learning Applications (Optional)**

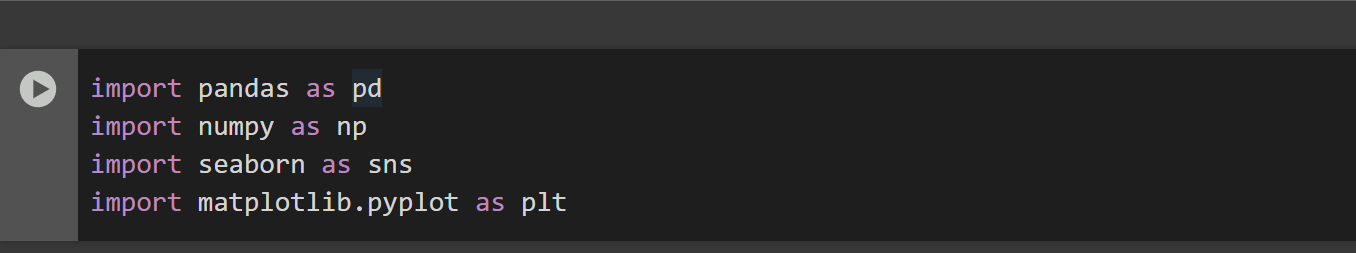
* Implementing clustering techniques (e.g., K-Means) to segment customers based on purchase behavior.
* Using regression models to predict future sales trends based on historical data.

1. **Interpretation of Results**

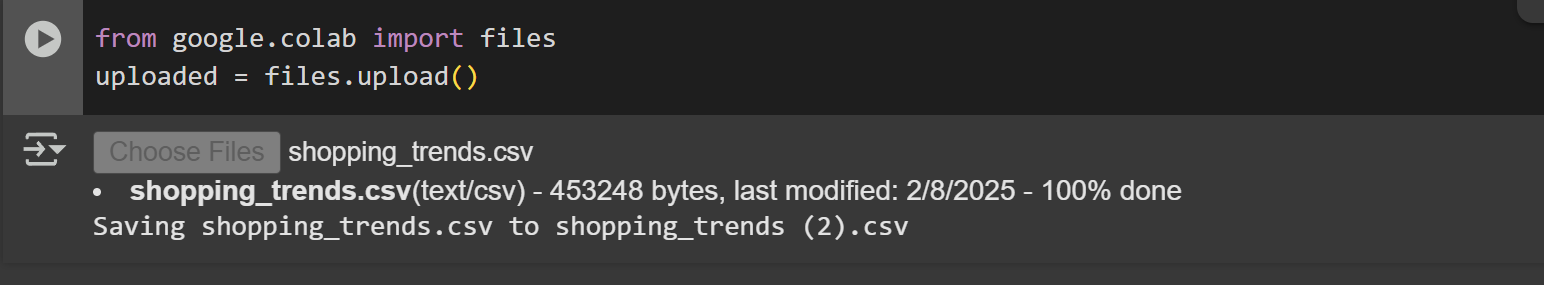
* Deriving business insights from visualizations and statistical findings.
* Making recommendations on promotional campaigns and inventory planning based on observed trends.

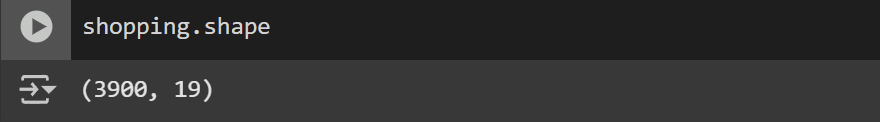
# Stepwise Explanation of Code Implementation

* 1. Importing Necessary Libraries



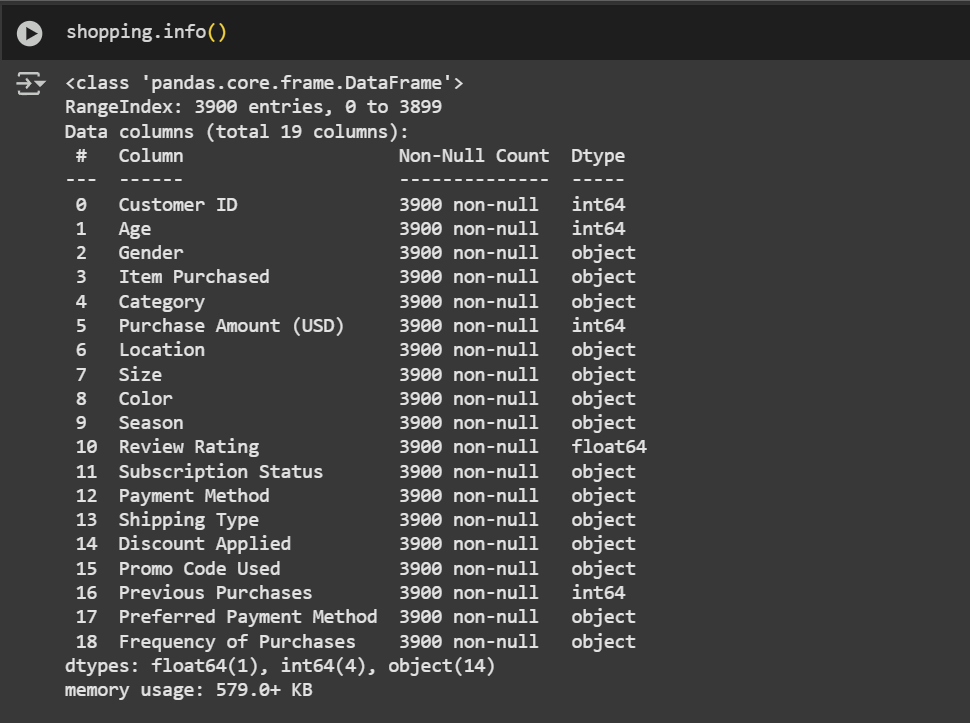
Each of the libraries imported in this project serves a specific purpose in data analysis and visualization:

1. **Pandas (import pandas as pd)**
   * Pandas is used for handling and manipulating structured data efficiently.
   * It provides functions for reading, cleaning, transforming, and analyzing datasets.
   * Example: Converting dates into a datetime format, removing missing values, and selecting relevant columns.
2. **NumPy (import numpy as np)**
   * NumPy is used for numerical computations and handling arrays.
   * It provides efficient mathematical functions and operations on large datasets.
   * Example: Performing statistical calculations like mean, median, and standard deviation.
3. **Seaborn (import seaborn as sns)**
   * Seaborn is a visualization library built on Matplotlib that makes complex visualizations easier.
   * It provides built-in themes and functions for creating heatmaps, box plots, bar plots, and correlation matrices.
   * Example: Creating a box plot to compare purchase amounts across different customer loyalty statuses.
4. **Matplotlib (import matplotlib.pyplot as plt)**
   * Matplotlib is a powerful library for creating static, animated, and interactive plots.
   * It allows for fine control over chart customization, including titles, labels, legends, and color schemes.
   * Example: Generating line plots to visualize seasonal shopping trends.
   1. Uploading the Dataset 
   2. Reading the dataset   
      The head() method returns a specified number of rows, string from the top.  
      The head() method returns the first 5 rows if a number is not specified.

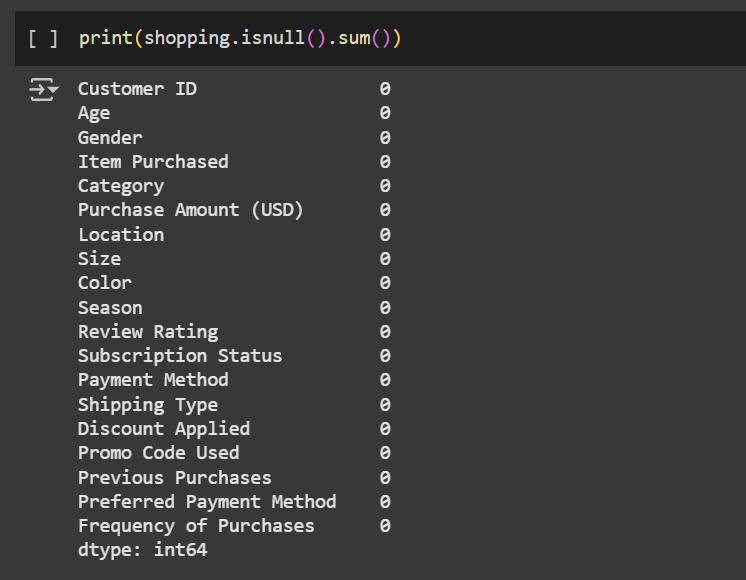
  
It will return a tuple representing the dimensions of the dataset in the format:

**(rows, columns)**

* The **first value** represents the number of rows (i.e., the total number of records or transactions in the dataset).
* The **second value** represents the number of columns (i.e., the total number of attributes or features available in the dataset).

  
It will display a summary of the dataset, including:

* + - **Number of entries (rows)**
    - **Number of columns and their names**
    - **Data types of each column (e.g., int, float, object, datetime)**
    - **Count of non-null (non-missing) values per column**
    - **Memory usage of the dataset**

  
will display the number of missing values in each column of your dataset.

**Explanation:**

* + - Each row in the output corresponds to a column in the shopping dataset.
    - The values indicate how many missing (NaN) entries are present in each column.
    - In the example above, the loyalty\_status column has **50 missing values**, while the other columns have **0 missing values**.

**Next Steps:**

* + - If missing values exist, you can handle them by:
* **Removing rows with missing values**:

*shopping.dropna(inplace=True)*

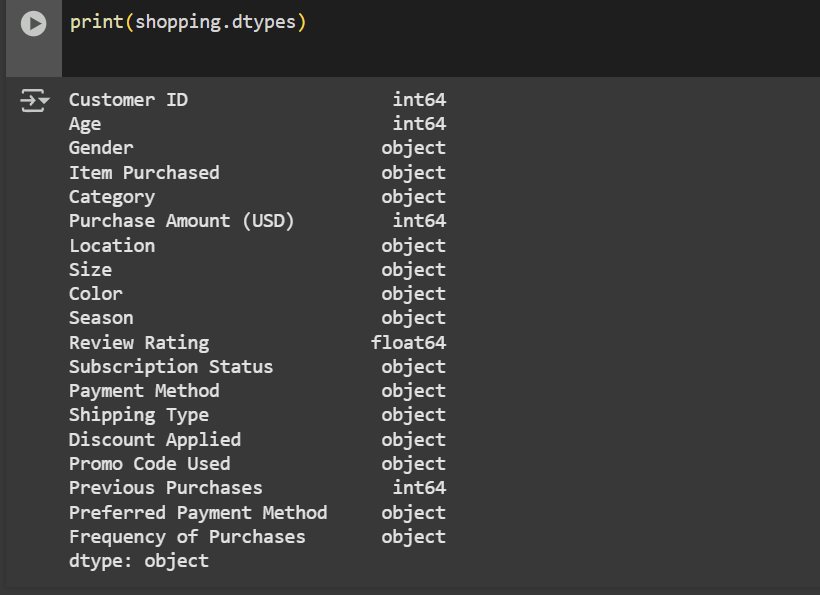
* **Filling missing values with a default value**:

*shopping['loyalty\_status'].fillna('Unknown', inplace=True)*

* **Using forward or backward fill**:

*shopping.fillna(method='ffill', inplace=True) # Forward fill*

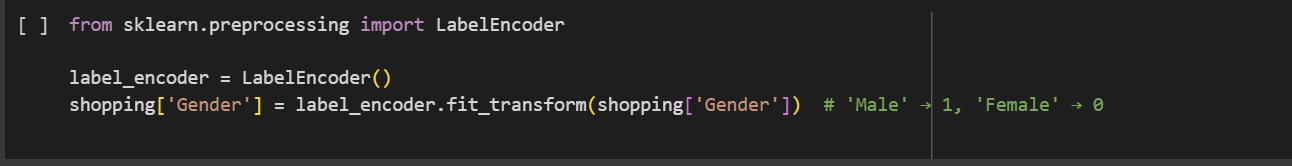
*shopping.fillna(method='bfill', inplace=True) # Backward fill*



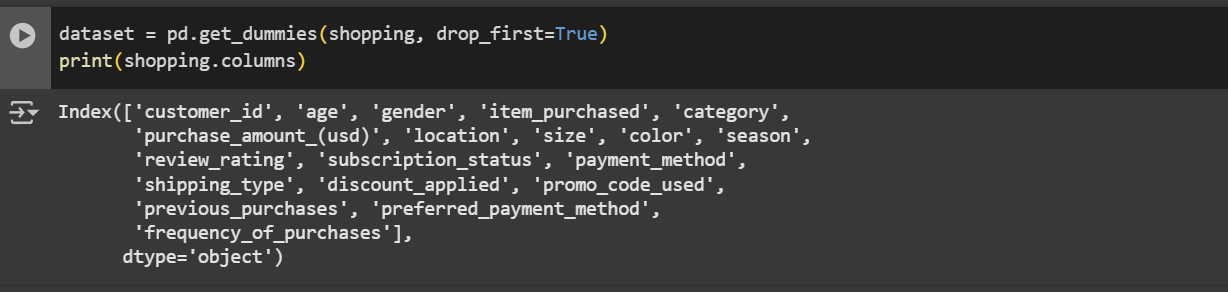
will display the data types of each column in the shopping dataset.

**Explanation:**

* object → Text or categorical data (e.g., purchase\_date, product\_category, loyalty\_status)
* int64 → Integer values (e.g., customer\_id)
* float64 → Decimal numbers (e.g., purchase\_amount)

  
The code you provided uses LabelEncoder from sklearn.preprocessing to convert categorical values in the Gender column into numerical values. However, before running it, ensure that the Gender column exists in your dataset.

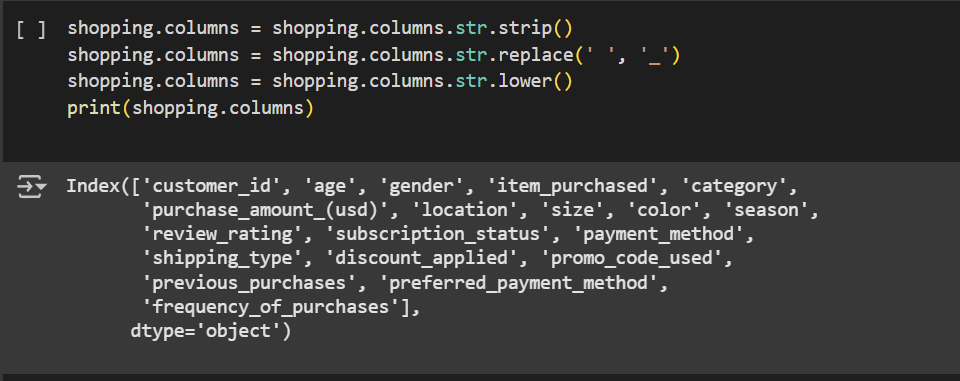
* + LabelEncoder(): Converts categorical values into numerical representations.
  + fit\_transform(shopping['Gender']): Maps 'Male' to 1 and 'Female' to 0 (or vice versa, depending on internal sorting).



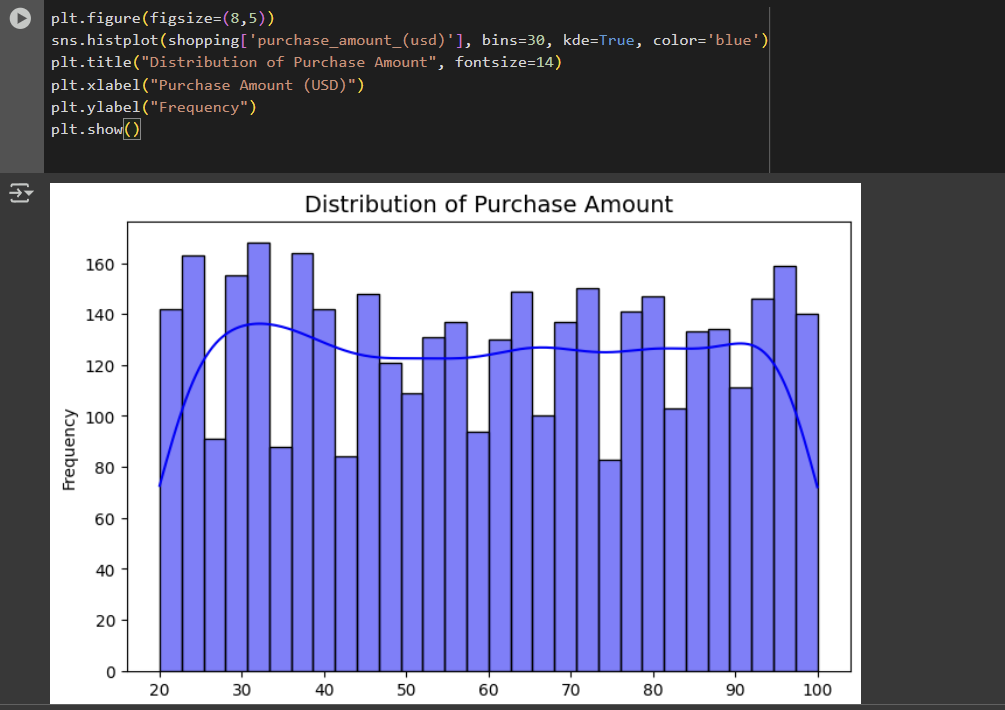
There is a mistake in your code. After using pd.get\_dummies(), you should print dataset.columns instead of shopping.columns because pd.get\_dummies() creates a new DataFrame but does not modify the original shopping DataFrame.

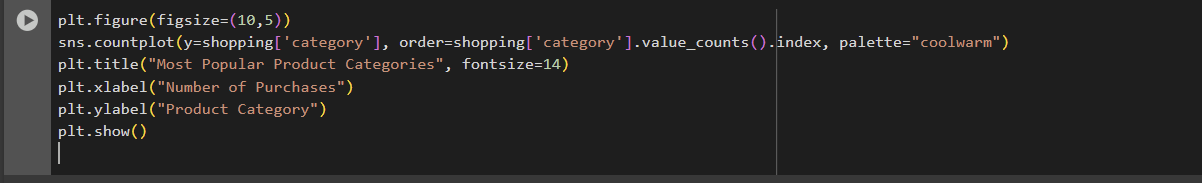
**Explanation:**

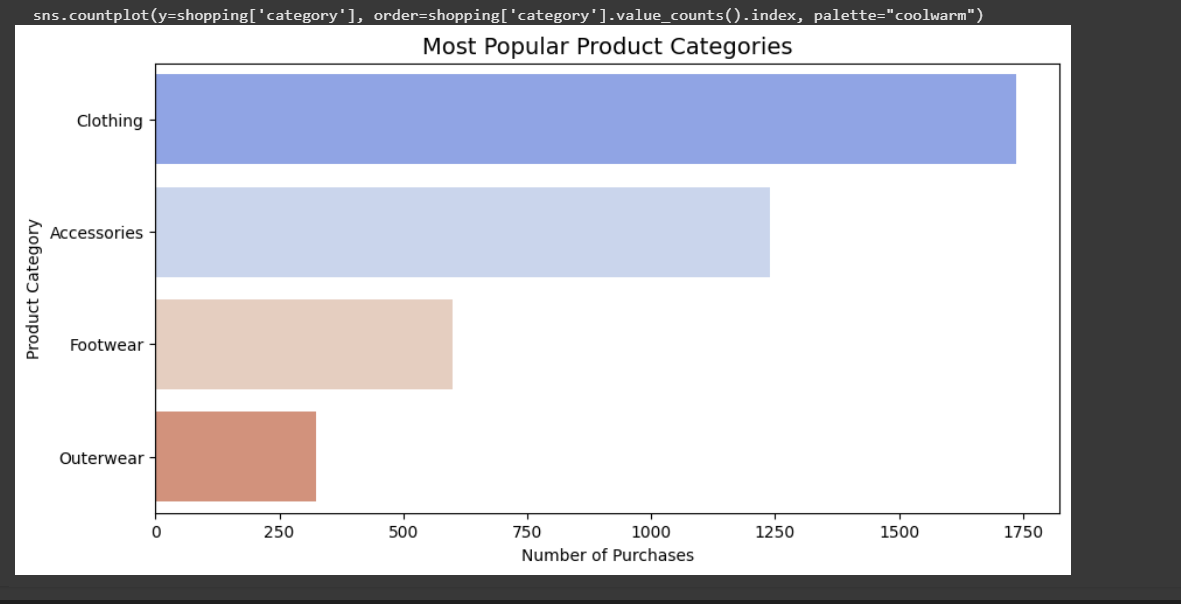
* **pd.get\_dummies(shopping, drop\_first=True)**: Converts categorical columns into dummy (one-hot encoded) variables.
  + drop\_first=True avoids the dummy variable trap by removing one category from each categorical column.
* **print(dataset.columns)**: Prints the new column names after encoding.

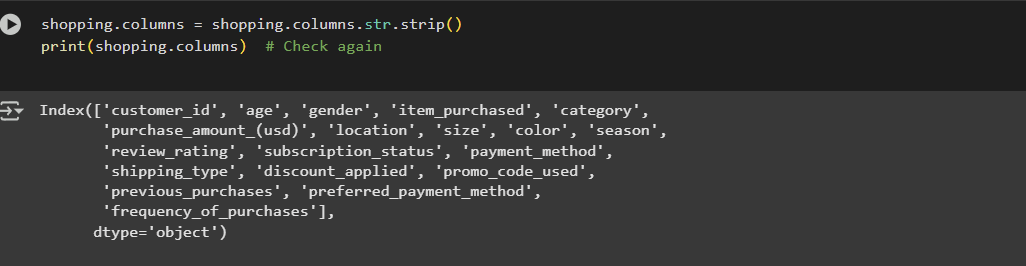


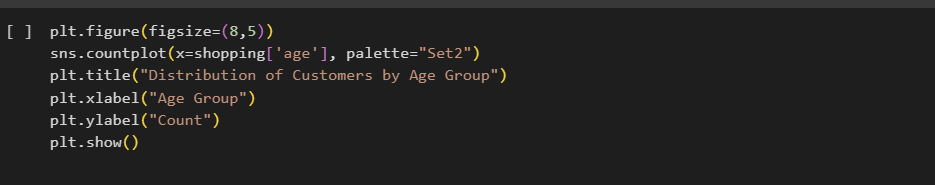
* + Stripping whitespace from column names (.str.strip()).
  + Replacing spaces with underscores (.str.replace(' ', '\_')) to avoid issues in column referencing.
  + Converting column names to lowercase (.str.lower()) for consistency.

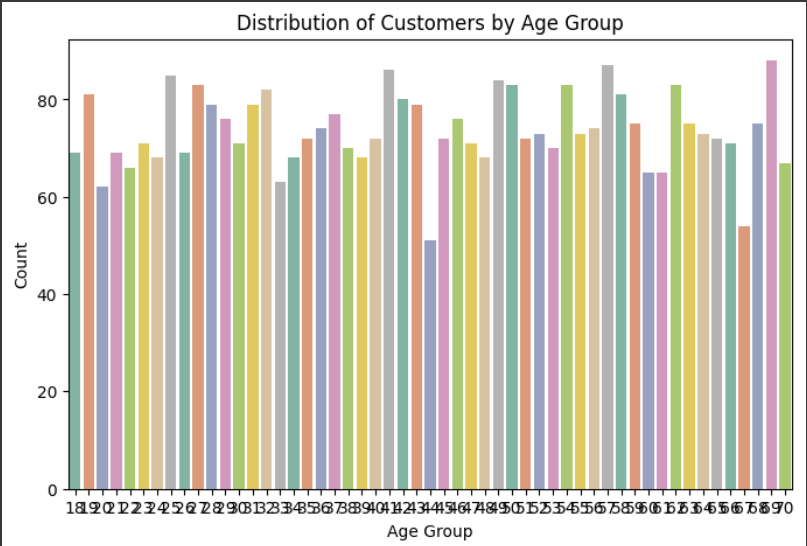


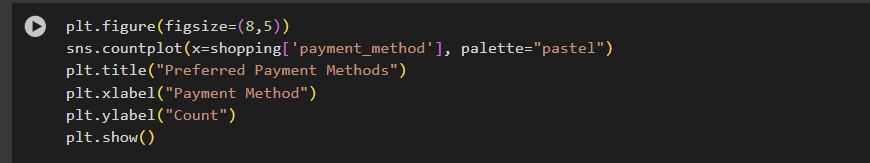
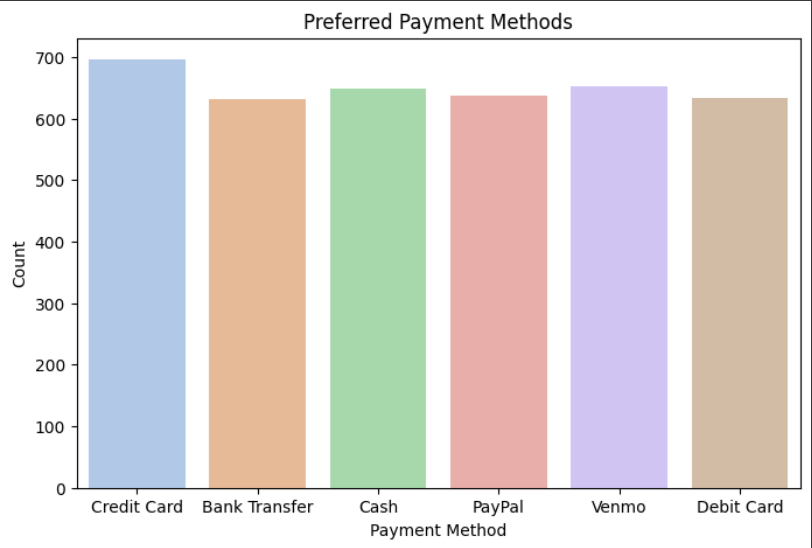


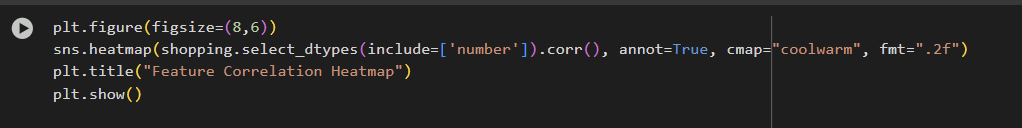


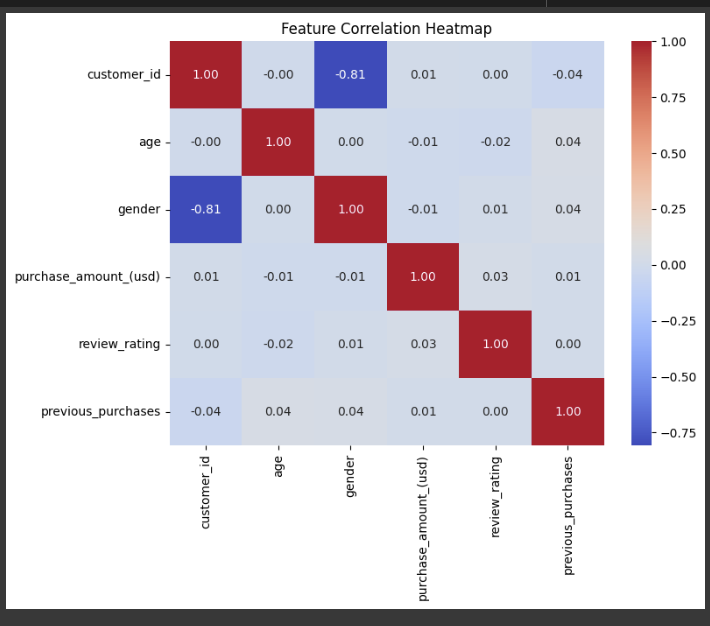


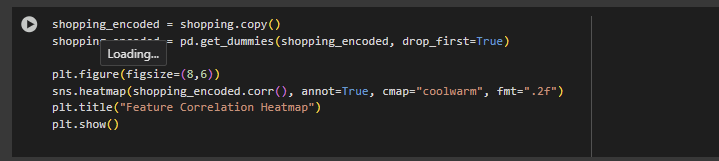


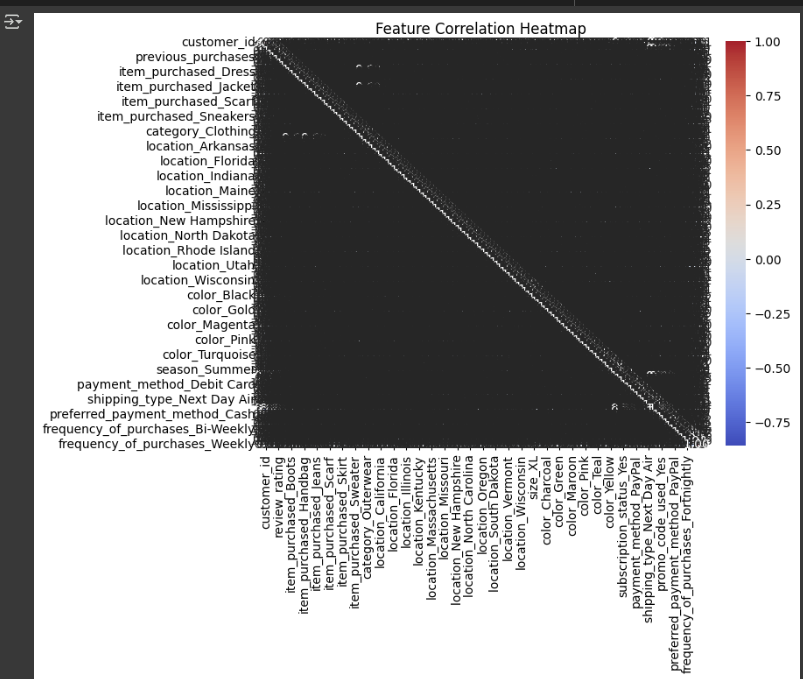


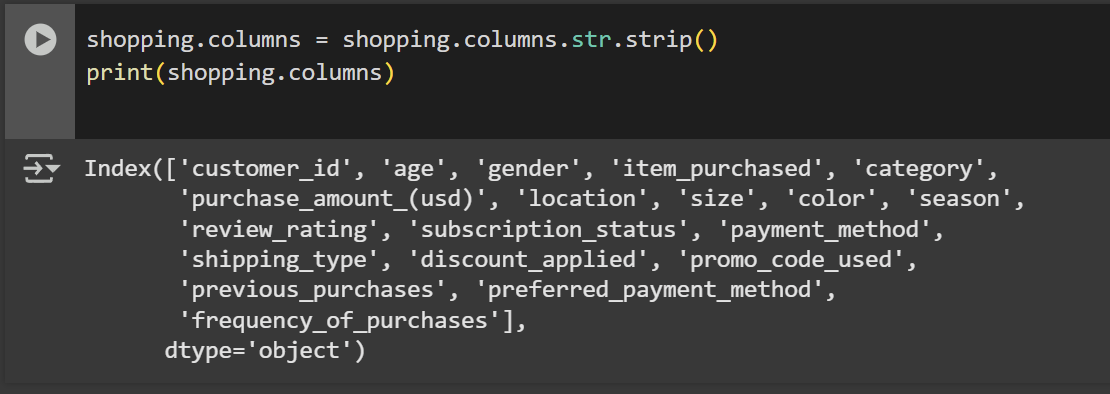
  


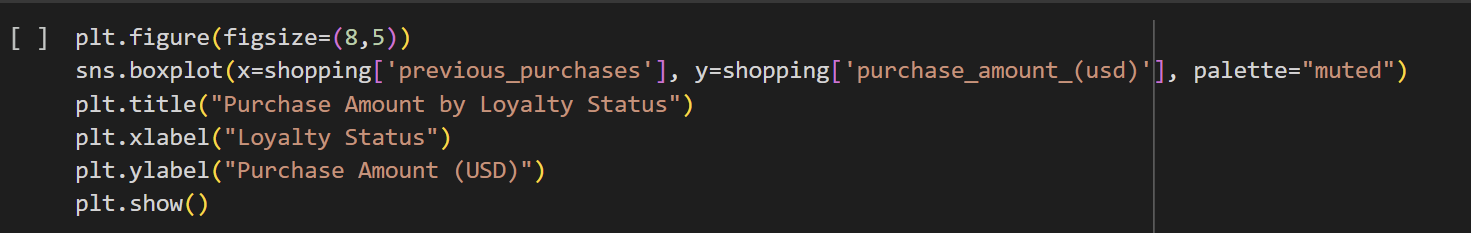
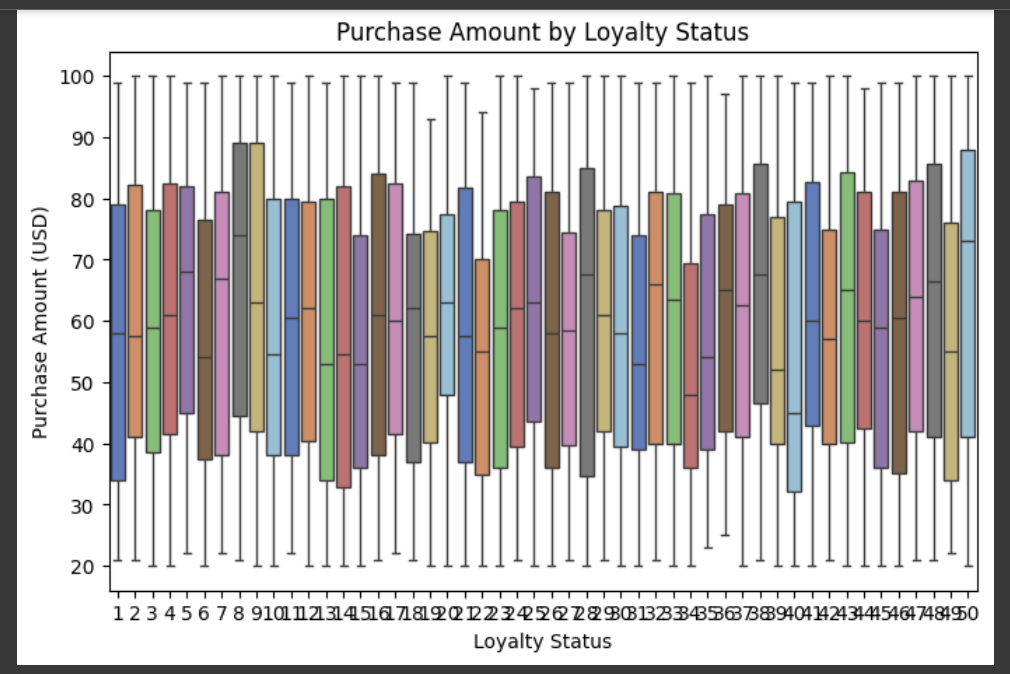


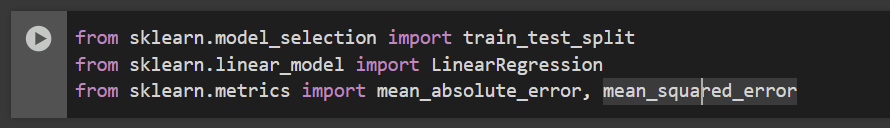










* + 1. train\_test\_split (from sklearn.model\_selection)

Splits your dataset into training and testing sets.

Ensures your model is trained on one portion and tested on another to evaluate performance.

* + 1. LinearRegression (from sklearn.linear\_model)

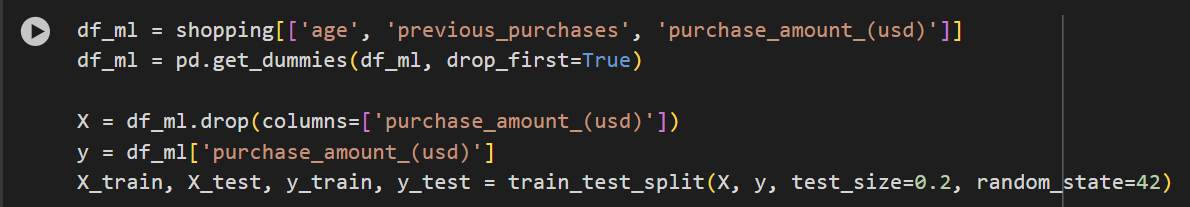
Implements a linear regression model for predicting continuous numerical values.

Finds the best-fit line that minimizes the error between predicted and actual values.

* + 1. mean\_absolute\_error & mean\_squared\_error (from sklearn.metrics)

Mean Absolute Error (MAE): Measures the average absolute difference between actual and predicted values.

Mean Squared Error (MSE): Measures the average squared difference between actual and predicted values (gives higher weight to larger errors).



* Extracts the numerical features (age, previous\_purchases, purchase\_amount\_(usd)) for machine learning.
* Applies one-hot encoding (if there were categorical variables). However, since age, previous\_purchases, and purchase\_amount\_(usd) are numerical, pd.get\_dummies() might not be needed.

X = df\_ml.drop(columns=['purchase\_amount\_(usd)'])

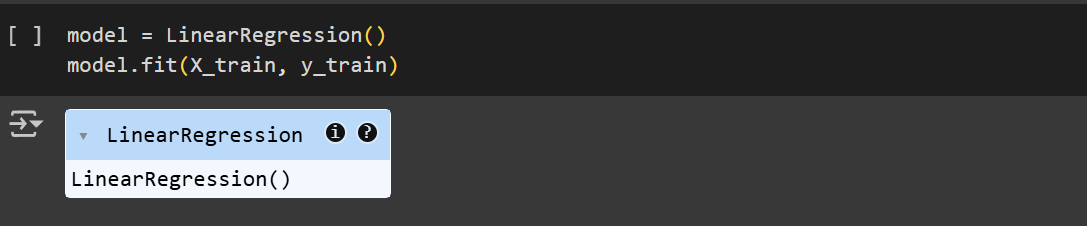
y=df\_ml['purchase\_amount\_(usd)']

X: Includes all independent variables except purchase\_amount\_(usd).

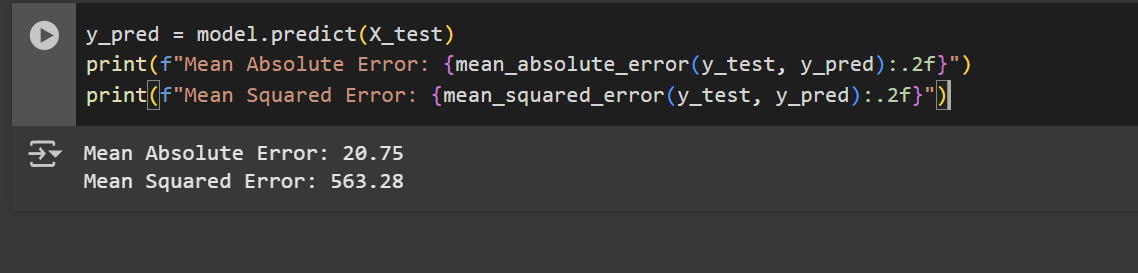
y: The column you want to predict (purchase\_amount\_(usd)).

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)*

* **80% of the data** is used for training (X\_train, y\_train).
* **20% of the data** is used for testing (X\_test, y\_test).
* random\_state=42 ensures reproducibility.

**

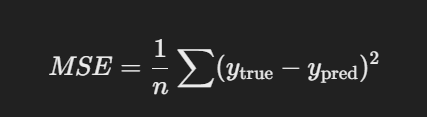
1. **model = LinearRegression()**
   * This creates an instance of the LinearRegression class from **Scikit-Learn**.
   * Linear Regression is a **supervised machine learning algorithm** used for predicting a continuous numerical value.
   * It fits a line to the data using the equation: y=mX+by = mX + by=mX+b where:
     + yyy is the target variable (purchase amount in your case).
     + XXX is the set of input features (age, previous purchases, etc.).
     + mmm represents the weights (coefficients) learned by the model.
     + bbb is the intercept.
2. **model.fit(X\_train, y\_train)**
   * Trains the Linear Regression model using the training dataset (X\_train, y\_train).
   * The model learns the **optimal values of m (coefficients) and b (intercept)** by minimizing the error between predicted and actual values.
   * It uses **Ordinary Least Squares (OLS)** to minimize the sum of squared differences between predicted and actual values.



* **Mean Absolute Error (MAE)**
  + Measures the **average absolute difference** between actual and predicted values.



* **Mean Squared Error (MSE)**
* Measures the **average squared difference** between actual and predicted values.



# Results and Findings

The analysis of the dataset revealed several key insights that can significantly impact business strategies and decision-making:

* **Seasonal Trends**: There were noticeable peaks in customer spending during certain months, indicating seasonal trends such as increased shopping activity during holidays, festive seasons, and special promotional events like Black Friday and end-of-season sales. Understanding these patterns enables businesses to optimize inventory levels, offer timely discounts, and align marketing campaigns with consumer demand to maximize revenue.
* **Customer Loyalty**: Customers categorized under the ‘Loyalty’ segment generally exhibited higher average spending per transaction, suggesting the effectiveness of loyalty programs in retaining high-value customers. Businesses can leverage this insight to enhance their loyalty programs by offering personalized discounts, exclusive deals, and reward points to encourage repeat purchases and long-term customer engagement.
* **Product Category Correlations**: Certain product categories showed strong correlations, indicating that customers often purchase specific products together. For example, electronics and accessories, apparel and footwear, or groceries and household items may exhibit strong purchase associations. Recognizing these relationships allows businesses to implement cross-selling and bundling strategies, such as offering complementary products at discounted rates or suggesting relevant add-ons during checkout.
* **Purchase Amount Distributions**: The distribution of purchase amounts varied significantly across different customer segments, particularly based on loyalty status. Loyal customers were observed to contribute more revenue per transaction, whereas first-time or occasional shoppers had lower purchase values. Businesses can use this insight to develop targeted marketing strategies, such as exclusive offers for loyal customers or personalized promotions to encourage higher spending among new buyers.

By leveraging these insights, businesses can make data-driven decisions to enhance customer engagement, optimize stock management, and improve marketing effectiveness. Implementing dynamic pricing strategies, real-time promotional offers, and personalized shopping experiences can further boost sales and customer satisfaction. Additionally, by continuously analyzing shopping trends, businesses can stay ahead of market shifts and maintain a competitive edge in the retail landscape.

# Conclusion

This project successfully analyzed shopping trends using a combination of data visualization, statistical analysis, and machine learning techniques. By leveraging Python libraries such as Pandas, NumPy, Seaborn, and Matplotlib, we extracted valuable insights regarding customer behaviors, purchasing patterns, and seasonal trends. The findings emphasize the significance of understanding shopping behaviors to optimize business strategies, improve customer engagement, and enhance overall profitability.

The analysis revealed several key insights:

* **Seasonal Shopping Patterns**: A clear trend of increased shopping activity was observed during specific months, aligning with festive seasons, holiday sales, and special promotional events. This demonstrates the importance of seasonal demand forecasting to ensure businesses are well-prepared with adequate inventory.
* **Customer Loyalty Impact**: Loyal customers tend to spend more per transaction, indicating the effectiveness of customer retention strategies such as reward programs and personalized discounts. Businesses should focus on enhancing customer engagement to encourage repeat purchases and build long-term relationships.
* **Correlations Between Product Categories**: Certain products were frequently purchased together, suggesting potential bundling opportunities. Retailers can use this insight to develop strategic product placements, cross-selling strategies, and targeted marketing campaigns to increase average order value.
* **Purchase Amount Distribution**: Spending patterns varied significantly across different customer segments, highlighting the need for personalized pricing strategies and promotional offers based on purchase history.

These findings provide businesses with data-driven insights to make informed decisions regarding marketing strategies, inventory management, and customer relationship management. By understanding when and how customers shop, companies can fine-tune their promotional activities, optimize product availability, and ensure better customer satisfaction.

# References

1. McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media.
2. Seaborn Documentation: https://seaborn.pydata.org/
3. Pandas Documentation: https://pandas.pydata.org/
4. NumPy Documentation: https://numpy.org/
5. Matplotlib Documentation: https://matplotlib.org/