SSN COLLEGE OF ENGINEERING

KALAVAKKAM - 603 110

Department of Computer Science and Engineering

UCS2604 PRINCIPLES OF MACHINE LEARNING

Assignment 1

E-commerce Customer Data for Behavior Analysis

Madhumitha E- 3122 22 5001 068

Muthuselvi K - 3122 22 5001 074

Nandhine A - 3122 22 5001 076

Abstract:

In today's rapidly evolving digital marketplace, understanding customer behavior is essential for businesses to enhance customer experience, optimize sales strategies, and ensure sustainable growth. This project, "E-commerce Customer Behavior Analysis using Machine Learning," aims to analyze customer purchasing patterns, satisfaction levels, and key influencing factors through data-driven insights.

The dataset includes customer demographics, purchasing history, and engagement metrics, which undergo exploratory data analysis (EDA) and preprocessing techniques, including handling missing values, encoding categorical variables, and feature scaling. Various machine learning models, including Logistic Regression, Random Forest, Support Vector Machine (SVM), and are implemented to predict customer satisfaction levels. Feature selection techniques are applied to improve model performance, and class imbalance is addressed using SMOTE (Synthetic Minority Over-sampling Technique).
.This project demonstrates how machine learning can transform raw customer data into actionable insights, ultimately driving ethical, efficient, and customer-centric business decision

Introduction:

Problem Statement:

In the highly competitive e-commerce landscape, businesses must understand customer behavior to enhance user experience, improve customer retention, and optimize marketing strategies.
Customers have diverse purchasing behaviors influenced by factors such as age, gender, membership type, spending patterns, and past interactions with the platform. Additionally, customer satisfaction, a key metric for business success, is influenced by various parameters, making it difficult to predict and improve without data-driven methods.

This project aims to analyze e-commerce customer behavior using machine learning techniques to identify key factors affecting satisfaction levels.

Literature Survey:

The e-commerce industry has been rapidly growing, and customer behavior analysis is becoming increasingly important for businesses to stay competitive. Several studies have explored various aspects of customer behavior in the e-commerce domain, from predicting customer satisfaction to understanding the influence of personal and transactional data on customer engagement and retention.

A study by *Kim et al.* (2011) demonstrated how customer satisfaction can be predicted using various attributes such as the number of products purchased, time spent on the platform, and interaction with customer service. Machine learning techniques, such as Logistic Regression, Decision Trees, and Random Forest, have been used to develop predictive models for customer satisfaction in different industries, including e-commerce.

In a study by *Chandrashekar & Sahin (2014),* various feature selection techniques were reviewed for use in predictive models, including correlation-based methods, mutual information, and recursive feature elimination (RFE). In the context of e-commerce, this can help identify key attributes that influence customer satisfaction, such as age, membership type, and spending habits. Dimensionality reduction techniques like Principal Component Analysis (PCA) are also employed to reduce the feature space without losing important information.

Zhang et al. (2018) compared several models for predicting customer purchase behavior, including Logistic Regression, Support Vector Machines (SVM), and Random Forests. The study found that ensemble methods like Random Forest and Gradient Boosting provide more accurate predictions by aggregating the output of multiple models to minimize errors.

Development Environment:

The project is implemented using **Python**, which is widely used for machine learning and data analysis due to its extensive libraries and community support.

1. Programming Language

- **Python**: Python is chosen because of its ease of use, powerful libraries, and strong support for machine learning and data analysis. It allows efficient implementation of models like **Logistic Regression Random Forest and SVM** for customer behavior analysis.

2. Libraries Used

Data Handling and Preprocessing

- **NumPy**: Provides support for numerical computations, handling large datasets efficiently. It helps with operations like array manipulations and mathematical functions.
- **Pandas**: Used for data manipulation and analysis. It helps in loading, cleaning, and processing datasets in an easy-to-use **DataFrame** format.

Data Visualization

- **Matplotlib**: A plotting library used to create static, animated, and interactive visualizations. It helps in visualizing trends, relationships, and patterns in customer data.
- **Seaborn**: Built on Matplotlib, it provides more attractive and informative statistical graphics. It is useful for plotting heatmaps, distribution plots, and correlation matrices to understand customer behavior.

Proposed System:

1.Data Collection & Preprocessing

- Collect Data: Gather customer satisfaction-related data, including demographics, transaction history, and engagement metrics.
- Handle Missing Values: Use imputation techniques (mean/mode for numerical data, most frequent for categorical data).
- Encode Categorical Variables: Apply Label Encoding (for binary features) and One-Hot Encoding (for multi-category features).
- Handle Class Imbalance: Use SMOTE if there is an imbalance in satisfaction levels.

2.Feature Selection

3. Split the Data

- Train-Test Split: Divide the data into 80% training, 20% testing.
- Stratify the Split: Ensures balanced class distribution for classification tasks.

4. Train Models

- Baseline Model: Train a Logistic Regression model for comparison.
- Advanced Models: Train models such as Random Forest,Support Vector Machine (SVM)

5. Model Evaluation

- Performance Metrics: Use Accuracy, Precision, Recall, F1-Score
- Confusion Matrix: Analyze classification performance.

Dataset Description:

1. Total Records: 350 customer entries.
2. Total Attributes: 11 features (10 independent variables + 1 target variable).
3. Customer ID: Unique identifier for each customer (not useful for modeling).
4. Gender: Categorical variable (Male/Female).
5. Age: Numerical variable representing the customer's age.
6. City: Categorical variable indicating the customer's location.
7. Membership Type: Categorical variable representing different membership levels.
8. Total Spend: Numerical variable showing total money spent by the customer.
9. Items Purchased: Numerical variable representing the number of items bought.
10. Average Rating: Float value between 1 to 5 indicating customer feedback.
11. Discount Applied: Boolean variable (True/False) indicating if a discount was used.
12. Days Since Last Purchase: Numerical variable tracking customer engagement.
13. Satisfaction Level (Target Variable): Categorical variable with three classes (Low, Medium, High).

Implementation with screenshots (Data Collection, Data Preprocessing, Feature Engineering, ML Model, Comparison of ML Models)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

Load the dataset

```
#Load the dataset

file_path = '/content/drive/MyDrive/E-commerce Customer Behavior - Sheet1.csv'
df = pd.read_csv(file_path)
print(df.head())
print("\nShape: ",df.shape)
```

```
   Customer ID  Gender  Age           City Membership Type  Total Spend  \
0          101  Female   29       New York            Gold      1120.20   
1          102    Male   34    Los Angeles          Silver       780.50   
2          103  Female   43        Chicago          Bronze       510.75   
3          104    Male   30  San Francisco            Gold      1480.30   
4          105    Male   27          Miami          Silver       720.40   

   Items Purchased  Average Rating  Discount Applied  \
0               14             4.6              True   
1               11             4.1             False   
2                9             3.4              True   
3               19             4.7             False   
4               13             4.0              True   

   Days Since Last Purchase Satisfaction Level  
0                        25          Satisfied  
1                        18            Neutral  
2                        42        Unsatisfied  
3                        12          Satisfied  
4                        55        Unsatisfied  

Shape:  (350, 11)
```

ii) Data info,data describe and data columns

```
print("Data Info:")
print(df.info())

print("\nData Describe:")
print(df.describe())

print("\nData Columns:")
print(df.columns)
```

```
Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 350 entries, 0 to 349
Data columns (total 11 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer ID               350 non-null    int64
 1   Gender                    350 non-null    object
 2   Age                       350 non-null    int64
 3   City                      350 non-null    object
 4   Membership Type           350 non-null    object
 5   Total Spend               350 non-null    float64
 6   Items Purchased           350 non-null    int64
 7   Average Rating            350 non-null    float64
 8   Discount Applied          350 non-null    bool
 9   Days Since Last Purchase  350 non-null    int64
 10  Satisfaction Level        348 non-null    object
dtypes: bool(1), float64(2), int64(4), object(4)
memory usage: 27.8+ KB
None

Data Describe:
       Customer ID         Age  Total Spend  Items Purchased  Average Rating  \
count   350.000000  350.000000   350.000000       350.000000      350.000000
mean    275.500000   33.597143   845.381714        12.600000        4.019143
std     101.180532    4.870882   362.058695         4.155984        0.580539
min     101.000000   26.000000   410.800000         7.000000        3.000000
25%     188.250000   30.000000   502.000000         9.000000        3.500000
50%     275.500000   32.500000   775.200000        12.000000        4.100000
75%     362.750000   37.000000  1160.600000        15.000000        4.500000
max     450.000000   43.000000  1520.100000        21.000000        4.900000

       Days Since Last Purchase
count                350.000000
mean                  26.588571
std                   13.440813
min                    9.000000
25%                   15.000000
50%                   23.000000
75%                   38.000000
max                   63.000000
```

```
Data Columns:
Index(['Customer ID', 'Gender', 'Age', 'City', 'Membership Type',
        'Total Spend', 'Items Purchased', 'Average Rating', 'Discount Applied',
        'Days Since Last Purchase', 'Satisfaction Level'],
       dtype='object')
```

Exploratory Data Analysis

Check for missing values

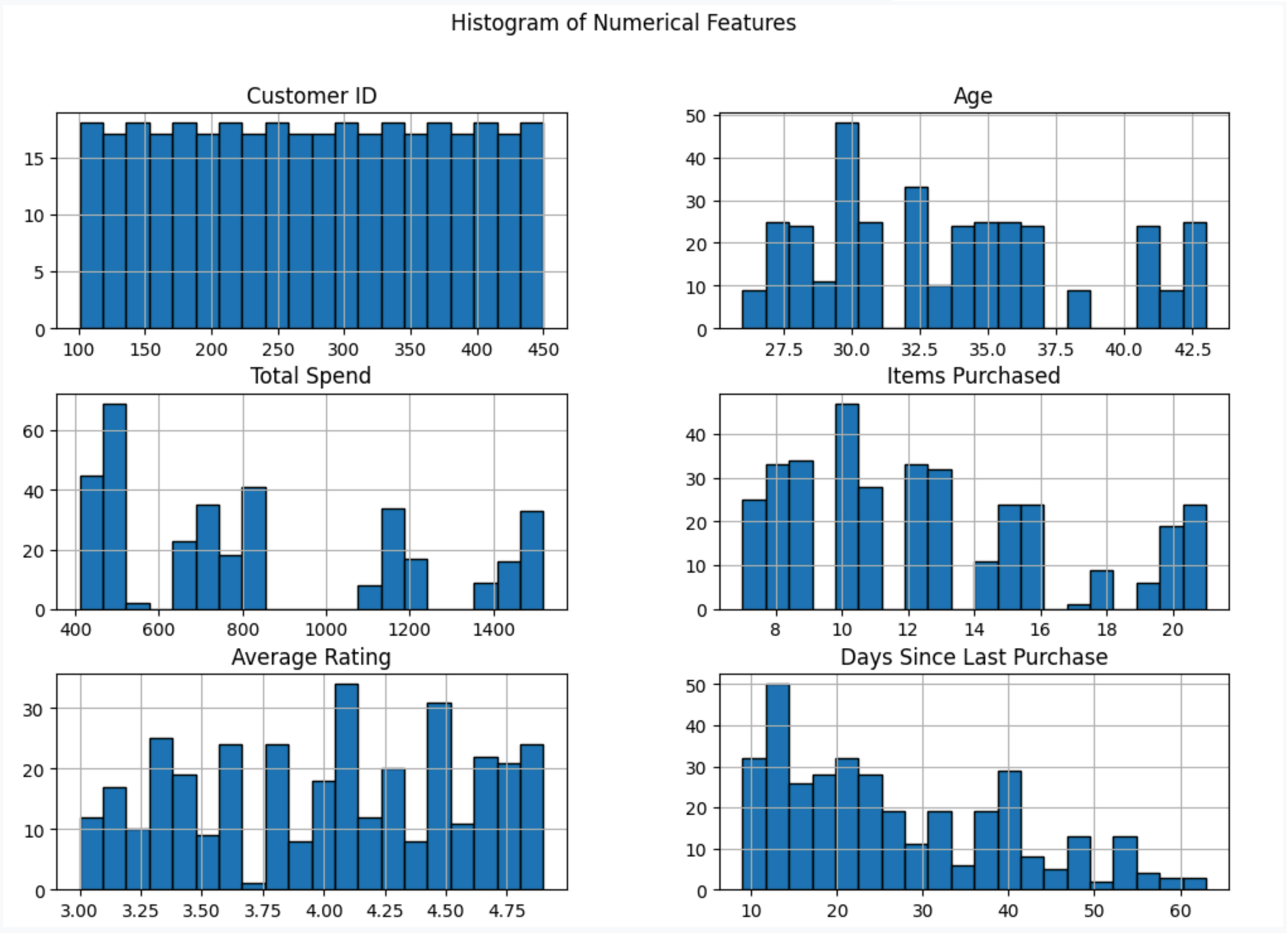
```
#Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())
```

```
Missing Values:
Customer ID                 0
Gender                      0
Age                         0
City                        0
Membership Type             0
Total Spend                 0
Items Purchased             0
Average Rating              0
Discount Applied            0
Days Since Last Purchase    0
Satisfaction Level          2
dtype: int64
```

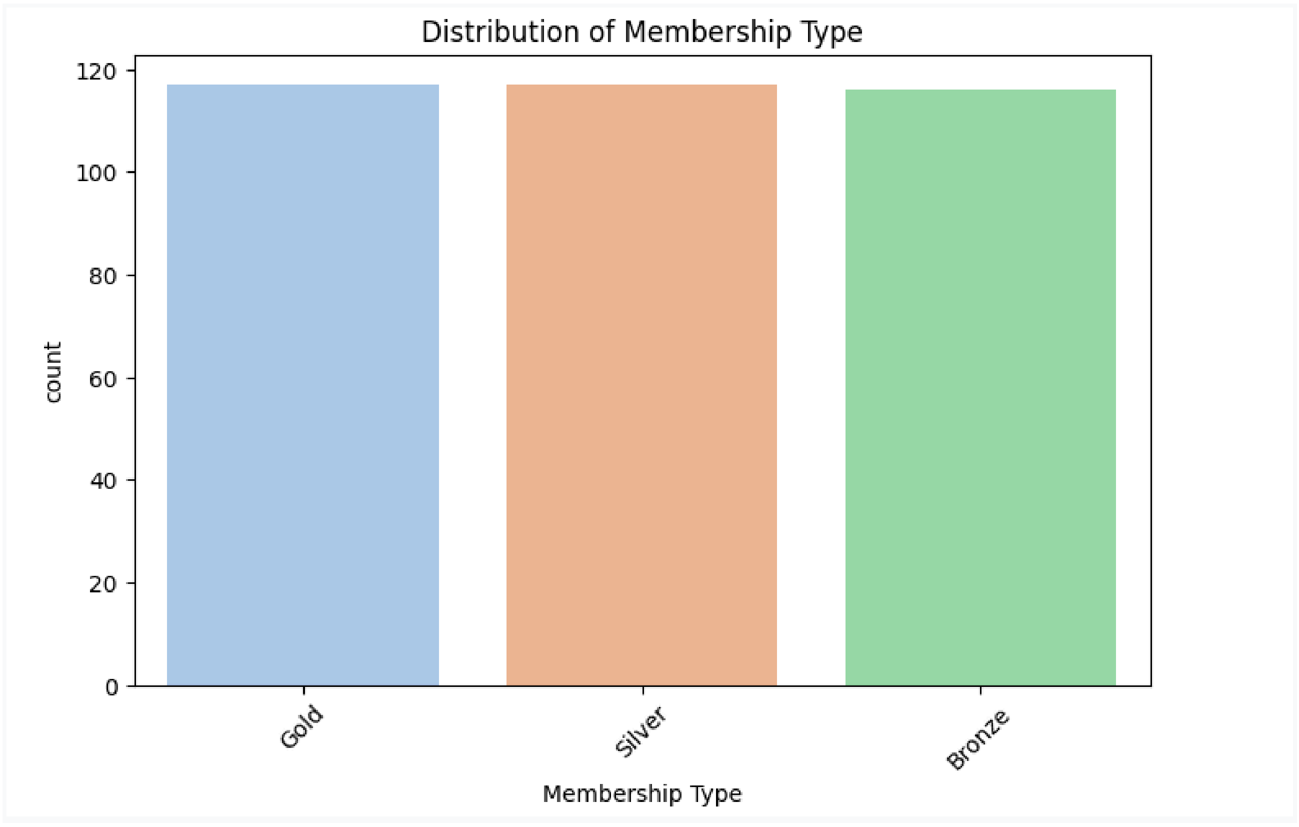
```
#Understanding Numerical and Categorical Data
print("\nStatistical Summary (Numerical Features):")
print(df.describe())

print("\nStatistical Summary (Categorical Features):")
print(df.describe(include='object'))
```

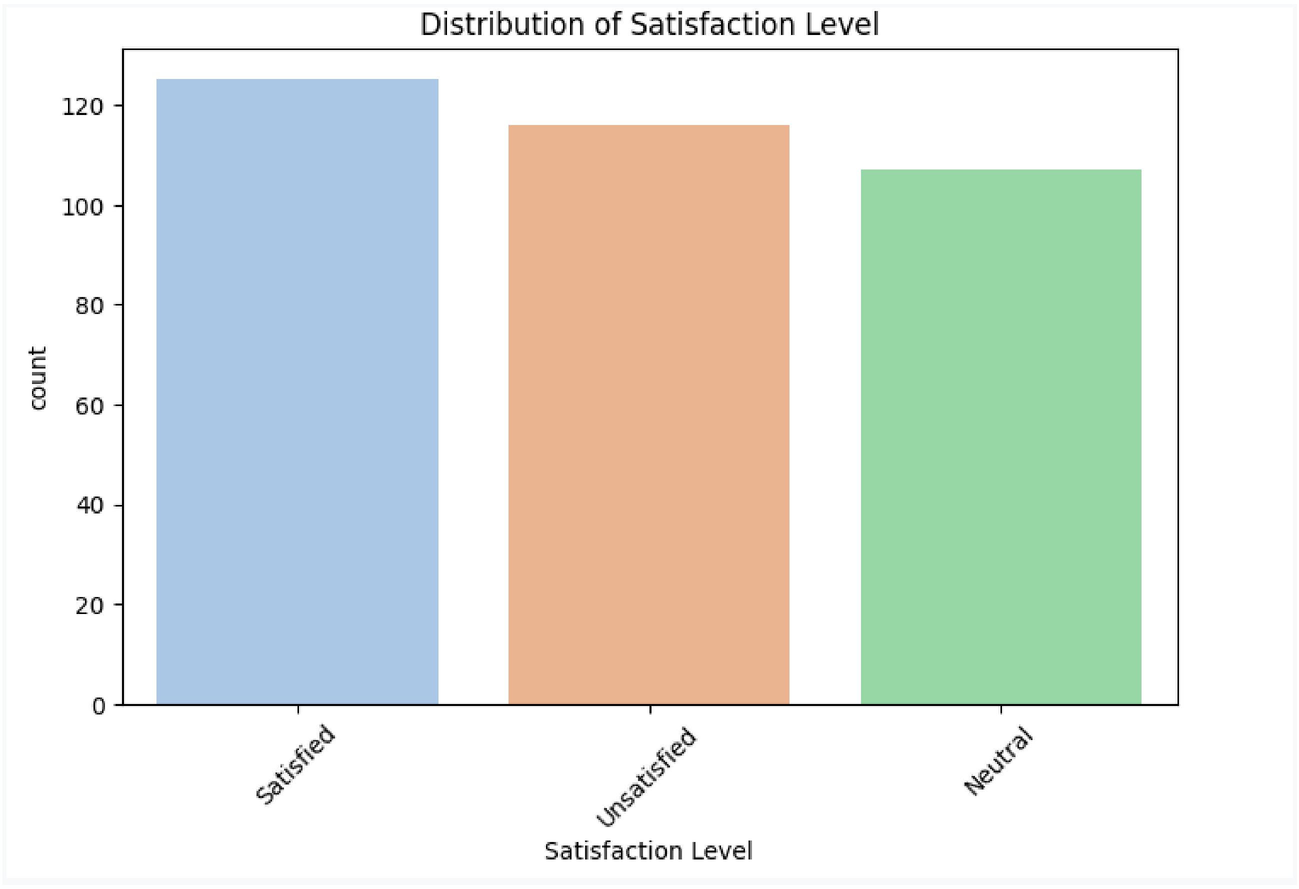
```
Statistical Summary (Numerical Features):
       Customer ID         Age  Total Spend  Items Purchased  Average Rating  \
count   350.000000  350.000000   350.000000       350.000000      350.000000
mean    275.500000   33.597143   845.381714        12.600000        4.019143
std     101.180532    4.870882   362.058695         4.155984        0.580539
min     101.000000   26.000000   410.800000         7.000000        3.000000
25%     188.250000   30.000000   502.000000         9.000000        3.500000
50%     275.500000   32.500000   775.200000        12.000000        4.100000
75%     362.750000   37.000000  1160.600000        15.000000        4.500000
max     450.000000   43.000000  1520.100000        21.000000        4.900000

       Days Since Last Purchase
count                350.000000
mean                  26.588571
std                   13.440813
min                    9.000000
25%                   15.000000
50%                   23.000000
75%                   38.000000
max                   63.000000

Statistical Summary (Categorical Features):
        Gender      City Membership Type Satisfaction Level
count      350       350             350                348
unique       2         6               3                  3
top     Female  New York            Gold          Satisfied
freq       175        59             117                125
```

```python
#Histogram for Numerical Features

df.hist(figsize=(12, 8), bins=20, edgecolor='black')
plt.suptitle("Histogram of Numerical Features")
plt.show()
```

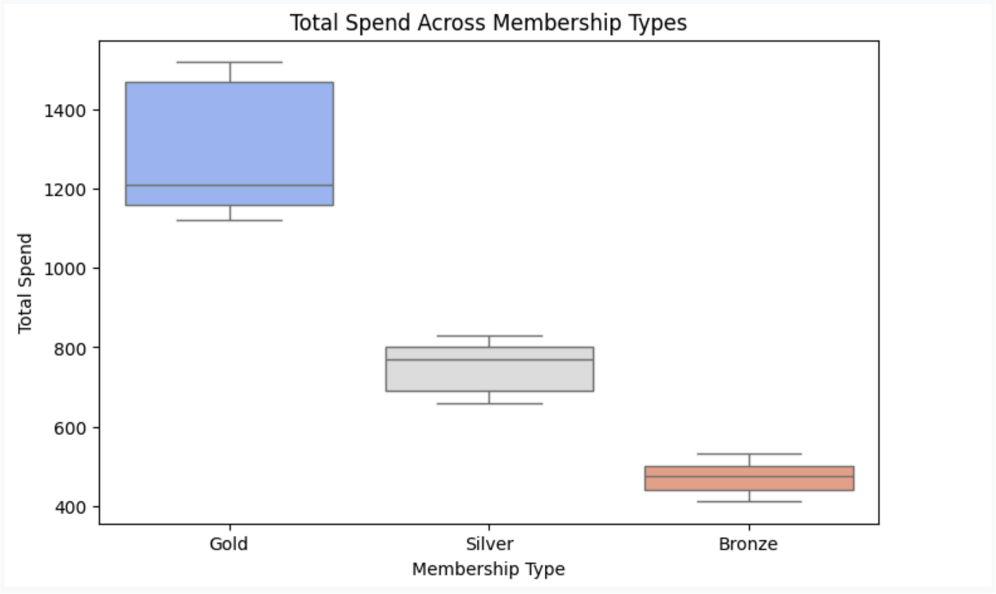


```python
#Categorical Feature Distribution

for col in ['Gender', 'City', 'Membership Type', 'Satisfaction Level']:
    plt.figure(figsize=(8, 5))
    sns.countplot(data=df, x=col, palette='pastel', order=df[col].value_counts().index)
    plt.title(f"Distribution of {col}")
    plt.xticks(rotation=45)
    plt.show()
```

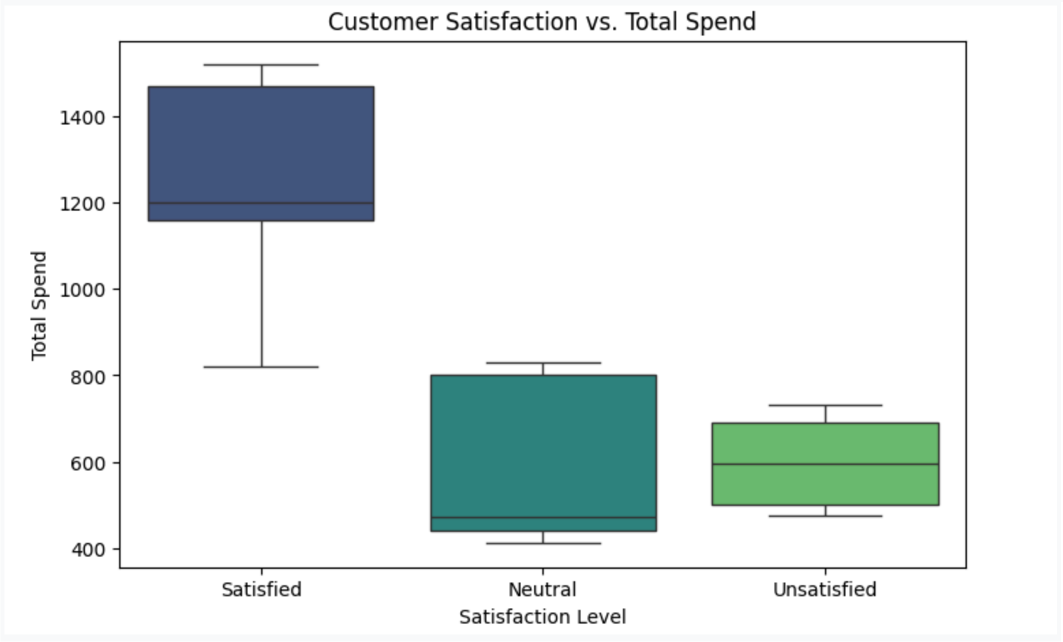

Distribution of Membership Type
count
0
20
40
60
80
100
120
Gold
Silver
Bronze
Membership Type


Distribution of Satisfaction Level
count
0
20
40
60
80
100
120
Satisfied
Unsatisfied
Neutral
Satisfaction Level

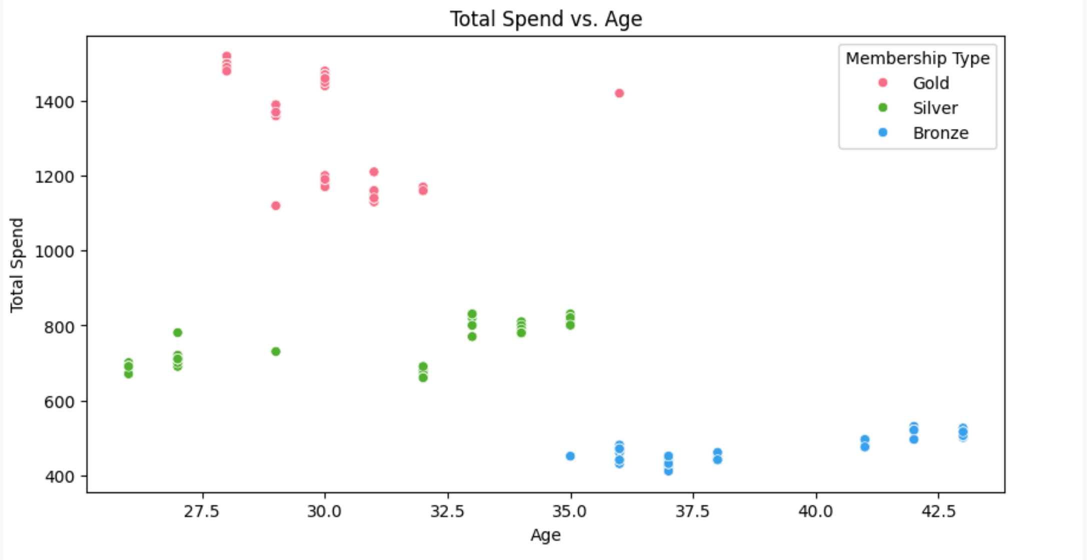
```python
#Boxplot for Spending Across Membership Type

plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='Membership Type', y='Total Spend', palette='coolwarm')
plt.title("Total Spend Across Membership Types")
plt.show()
```



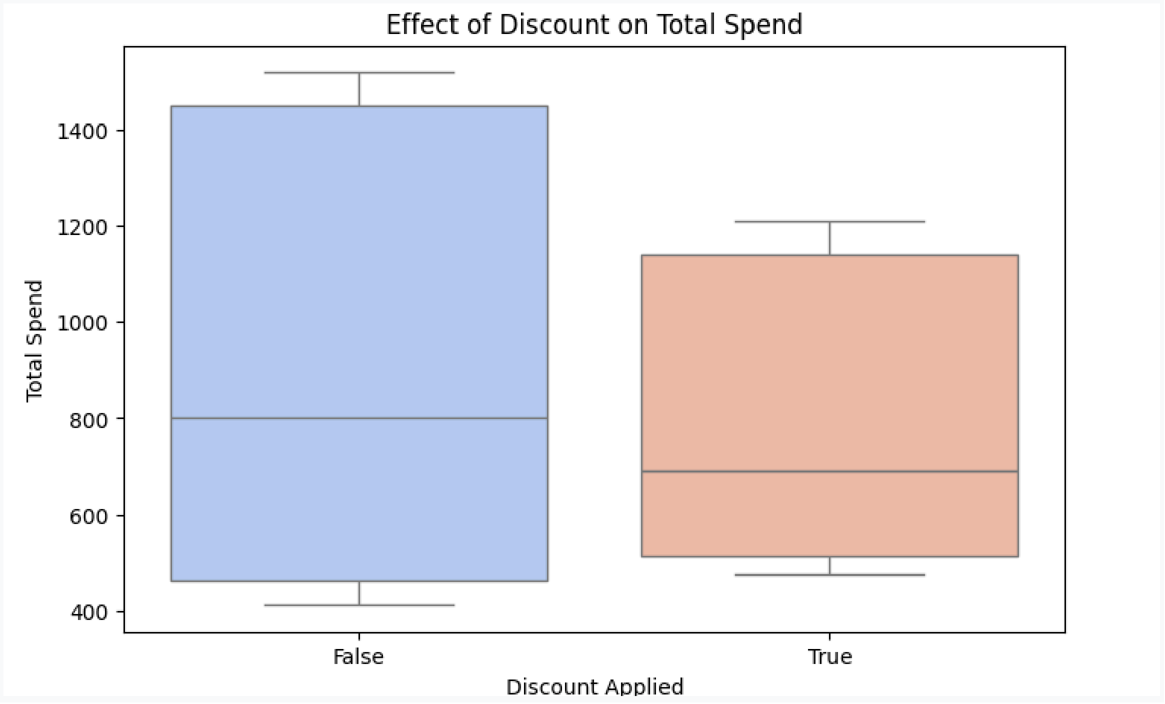
```python
#Customer Satisfaction vs. Total Spend- BOX PLOT

plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='Satisfaction Level', y='Total Spend', palette='viridis')
plt.title("Customer Satisfaction vs. Total Spend")
plt.show()
```



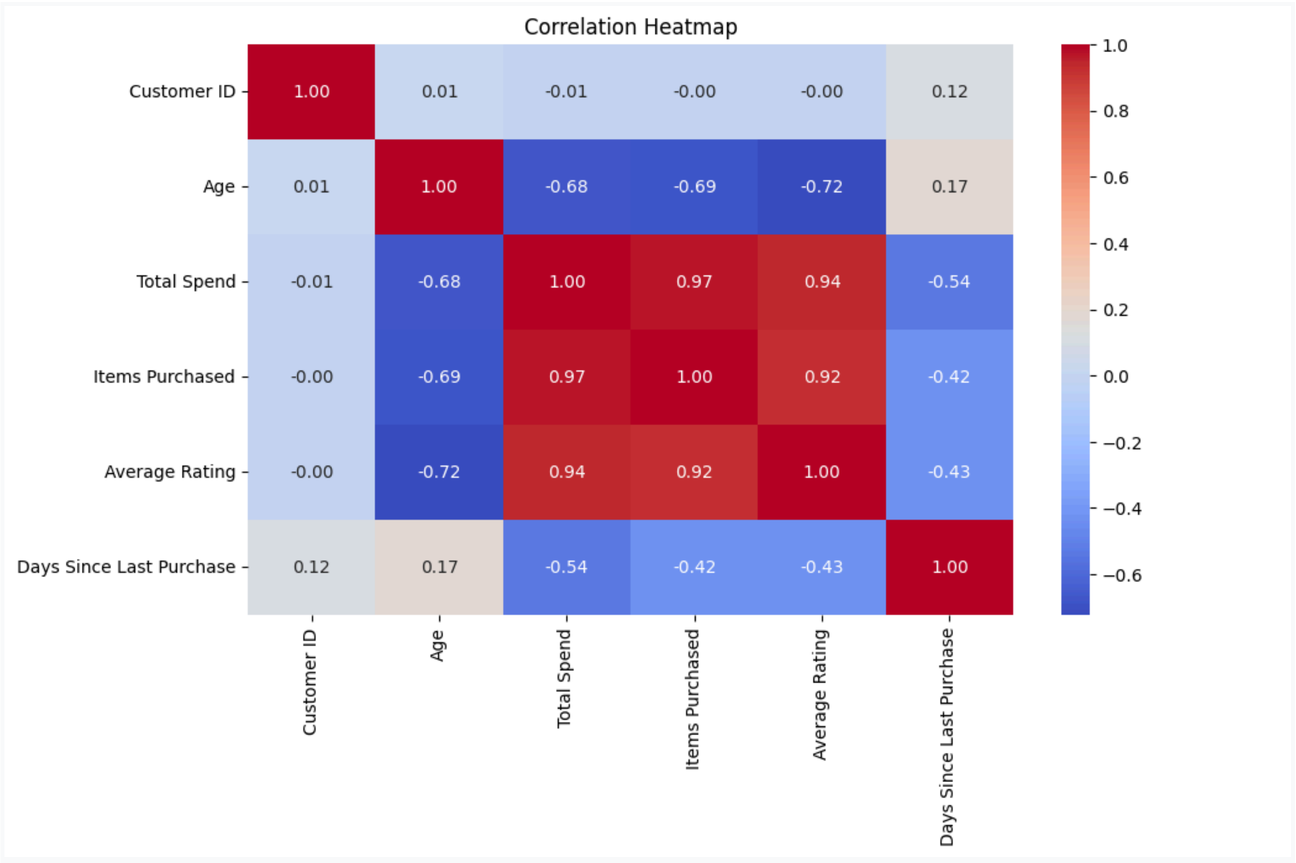
```python
#Spending Trends by Age - SCATTER PLOT
plt.figure(figsize=(10, 5))
sns.scatterplot(data=df, x='Age', y='Total Spend', hue='Membership Type', palette='husl')
plt.title("Total Spend vs. Age")
plt.show()
```



```python
#Impact of Discounts on Total Spend- BOX PLOT
plt.figure(figsize=(8, 5))
sns.boxplot(data=df, x='Discount Applied', y='Total Spend', palette='coolwarm')
plt.title("Effect of Discount on Total Spend")
plt.show()
```

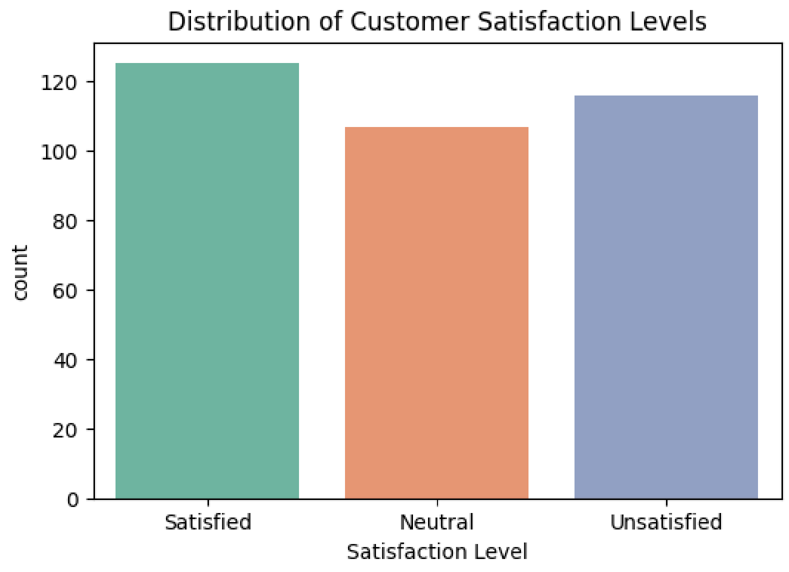


```python
#Correlation Heatmap for Numerical Features

numerical_cols = df.select_dtypes(include=np.number).columns

plt.figure(figsize=(10, 6))
sns.heatmap(df[numerical_cols].corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Correlation Heatmap")
plt.show()
```



	Customer ID	Age	Total Spend	Items Purchased	Average Rating	Days Since Last Purchase
Customer ID	1.00	0.01	-0.01	-0.00	-0.00	0.12
Age	0.01	1.00	-0.68	-0.69	-0.72	0.17
Total Spend	-0.01	-0.68	1.00	0.97	0.94	-0.54
Items Purchased	-0.00	-0.69	0.97	1.00	0.92	-0.42
Average Rating	-0.00	-0.72	0.94	0.92	1.00	-0.43
Days Since Last Purchase	0.12	0.17	-0.54	-0.42	-0.43	1.00

```python
#Understanding Class Distribution
plt.figure(figsize=(6, 4))
sns.countplot(data=df, x='Satisfaction Level', palette='Set2')
plt.title("Distribution of Customer Satisfaction Levels")
plt.show()

print(df['Satisfaction Level'].value_counts(normalize=True))
```



```
Satisfaction Level
Satisfied      0.359195
Unsatisfied    0.333333
Neutral        0.307471
Name: proportion, dtype: float64
```

Pre-Process the data:

Handling missing:

```
print(df.isnull().sum())

df['Satisfaction Level'].fillna(df['Satisfaction Level'].mode()[0],
inplace=True)
```

```
Customer ID                 0
Gender                      0
Age                         0
City                        0
Membership Type             0
Total Spend                 0
Items Purchased             0
Average Rating              0
Discount Applied            0
Days Since Last Purchase    0
Satisfaction Level          2
dtype: int64
Customer ID                 0
Gender                      0
Age                         0
City                        0
Membership Type             0
Total Spend                 0
Items Purchased             0
Average Rating              0
Discount Applied            0
Days Since Last Purchase    0
Satisfaction Level          0
```

Feature Selection & Target Definition

```
# Define Features & Target
X = df.drop(columns=['Satisfaction Level', 'Customer ID'])
y = df['Satisfaction Level']
```

Identifying Data Types categorical_features,Data Cleaning & Transformation Pipelines Numerical Pipeline

```
# Identify categorical & numerical features
categorical_features = ['Gender', 'City', 'Membership Type', 'Discount Applied']
numerical_features = ['Age', 'Total Spend', 'Items Purchased', 'Average Rating', 'Days Since Last Purchase']
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

cat_pipeline = Pipeline([
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer([
    ('num', num_pipeline, numerical_features),
    ('cat', cat_pipeline, categorical_features)
])

# Transform Data
X_processed = preprocessor.fit_transform(X)
```

Handling Class Imbalance Uses SMOTE (Synthetic Minority Over-sampling Technique)

```
# Handle Class Imbalance using SMOTE
smote = SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X_processed, y)
```

Preprocessing Steps :

One-Hot Encoding for Categorical Features

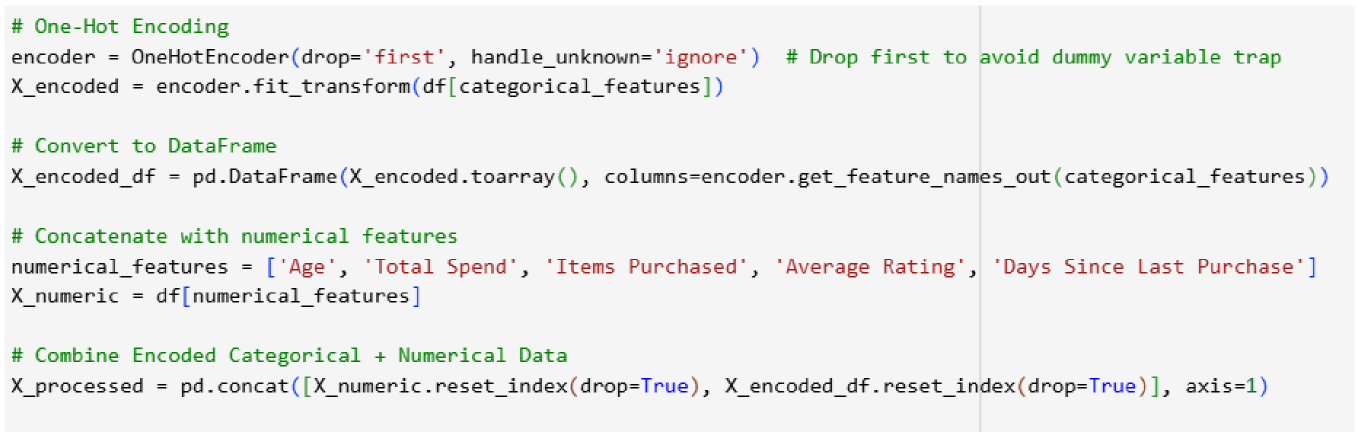
- Used OneHotEncoder with drop='first' to avoid the dummy variable trap (i.e., reducing multicollinearity).
- Set handle_unknown='ignore' to handle new/unseen categories in the test data.
- Transformed categorical features into binary columns.

Conversion to DataFrame

- The transformed categorical data (sparse matrix) was converted to a pandas DataFrame using pd.DataFrame(X_encoded.toarray()).
- Feature names were retrieved using encoder.get_feature_names_out(categorical_features).

Concatenation of Encoded Categorical Data with Numerical Features

- The numerical features were selected directly from the dataset.
- Both categorical and numerical features were combined using pd.concat().



```python
# One-Hot Encoding
encoder = OneHotEncoder(drop='first', handle_unknown='ignore')  # Drop first to avoid dummy variable trap
X_encoded = encoder.fit_transform(df[categorical_features])

# Convert to DataFrame
X_encoded_df = pd.DataFrame(X_encoded.toarray(), columns=encoder.get_feature_names_out(categorical_features))

# Concatenate with numerical features
numerical_features = ['Age', 'Total Spend', 'Items Purchased', 'Average Rating', 'Days Since Last Purchase']
X_numeric = df[numerical_features]

# Combine Encoded Categorical + Numerical Data
X_processed = pd.concat([X_numeric.reset_index(drop=True), X_encoded_df.reset_index(drop=True)], axis=1)
```

Split the data

```python
# Split Data
X_train, X_test, y_train, y_test = train_test_split(X_resampled,
y_resampled, test_size=0.2, random_state=42)
```

Correlation Heatmap

- Helps identify how strongly numerical variables are related.
- High correlation (close to 1 or -1) indicates a strong relationship between features.

Heatmap Analysis:

Strong Positive Correlations:

- Total Spend & Items Purchased (0.97) → More items purchased, higher spending.
- Total Spend & Average Rating (0.94) → High-spending customers give better ratings.
- Items Purchased & Average Rating (0.92) → Customers buying more items rate higher.

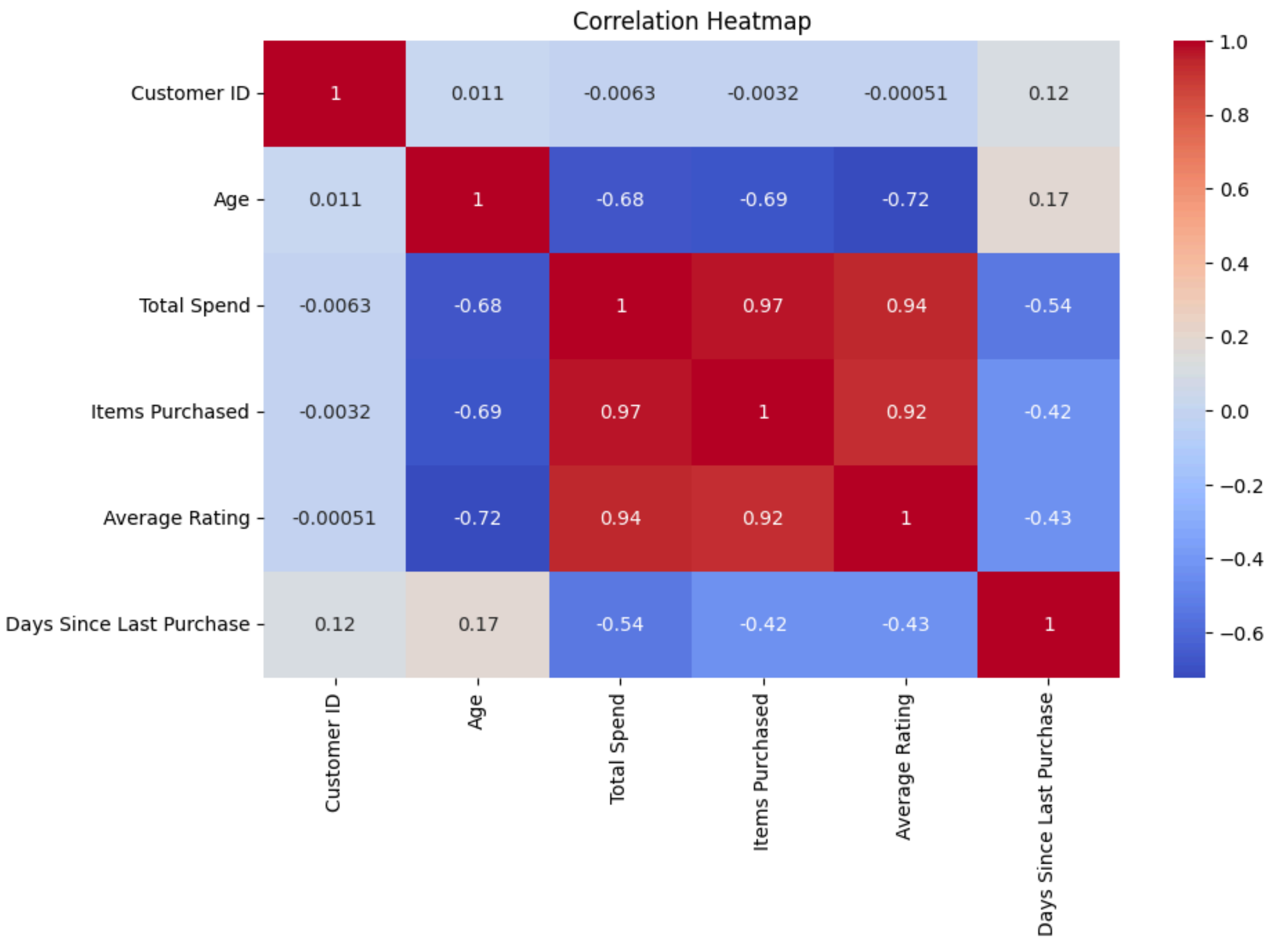
Strong Negative Correlations:

- Age & Total Spend (-0.68) → Older customers spend less.
- Age & Items Purchased (-0.69) → Older customers buy fewer items.
- Age & Average Rating (-0.72) → Older customers give lower ratings.
- Days Since Last Purchase & Total Spend (-0.54) → Longer gap → lower **spending.**

Weak or No Correlation:

Customer ID has no impact on other features

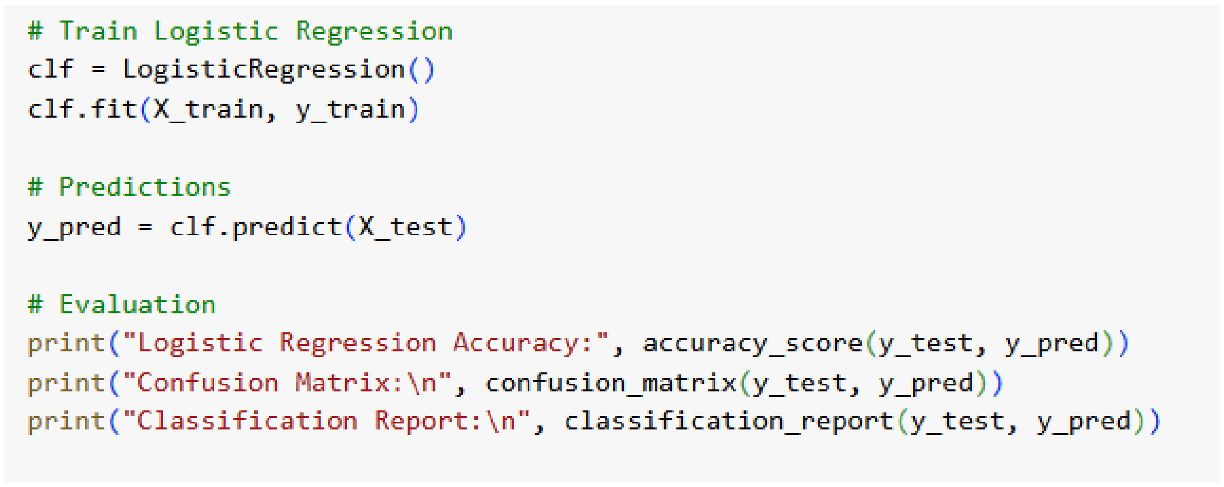
```
[54]  import seaborn as sns
      import matplotlib.pyplot as plt

      # Select only numerical columns
      df_numeric = df.select_dtypes(include=['int64', 'float64'])

      # Plot Heatmap
      plt.figure(figsize=(10, 6))
      sns.heatmap(df_numeric.corr(), annot=True, cmap="coolwarm")
      plt.title("Correlation Heatmap")
      plt.show()
```



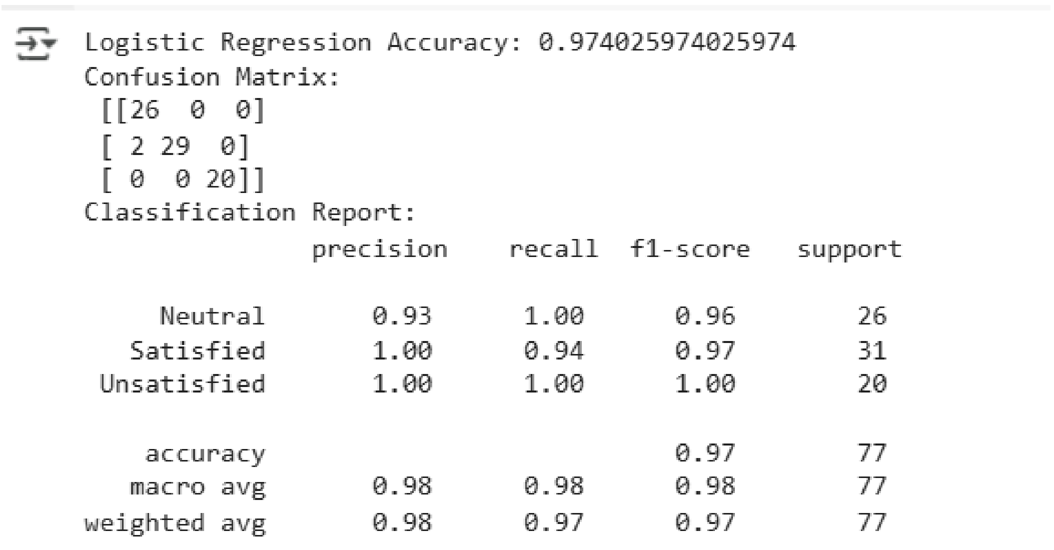
	Customer ID	Age	Total Spend	Items Purchased	Average Rating	Days Since Last Purchase
Customer ID	1	0.011	-0.0063	-0.0032	-0.00051	0.12
Age	0.011	1	-0.68	-0.69	-0.72	0.17
Total Spend	-0.0063	-0.68	1	0.97	0.94	-0.54
Items Purchased	-0.0032	-0.69	0.97	1	0.92	-0.42
Average Rating	-0.00051	-0.72	0.94	0.92	1	-0.43
Days Since Last Purchase	0.12	0.17	-0.54	-0.42	-0.43	1

Train and test the model:

1.Logistic Regression:



```
# Train Logistic Regression
clf = LogisticRegression()
clf.fit(X_train, y_train)

# Predictions
y_pred = clf.predict(X_test)

# Evaluation
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```



```
Logistic Regression Accuracy: 0.974025974025974
Confusion Matrix:
 [[26  0  0]
 [ 2 29  0]
 [ 0  0 20]]
Classification Report:
               precision    recall  f1-score   support

     Neutral       0.93      1.00      0.96        26
   Satisfied       1.00      0.94      0.97        31
 Unsatisfied       1.00      1.00      1.00        20

    accuracy                           0.97        77
   macro avg       0.98      0.98      0.98        77
weighted avg       0.98      0.97      0.97        77
```

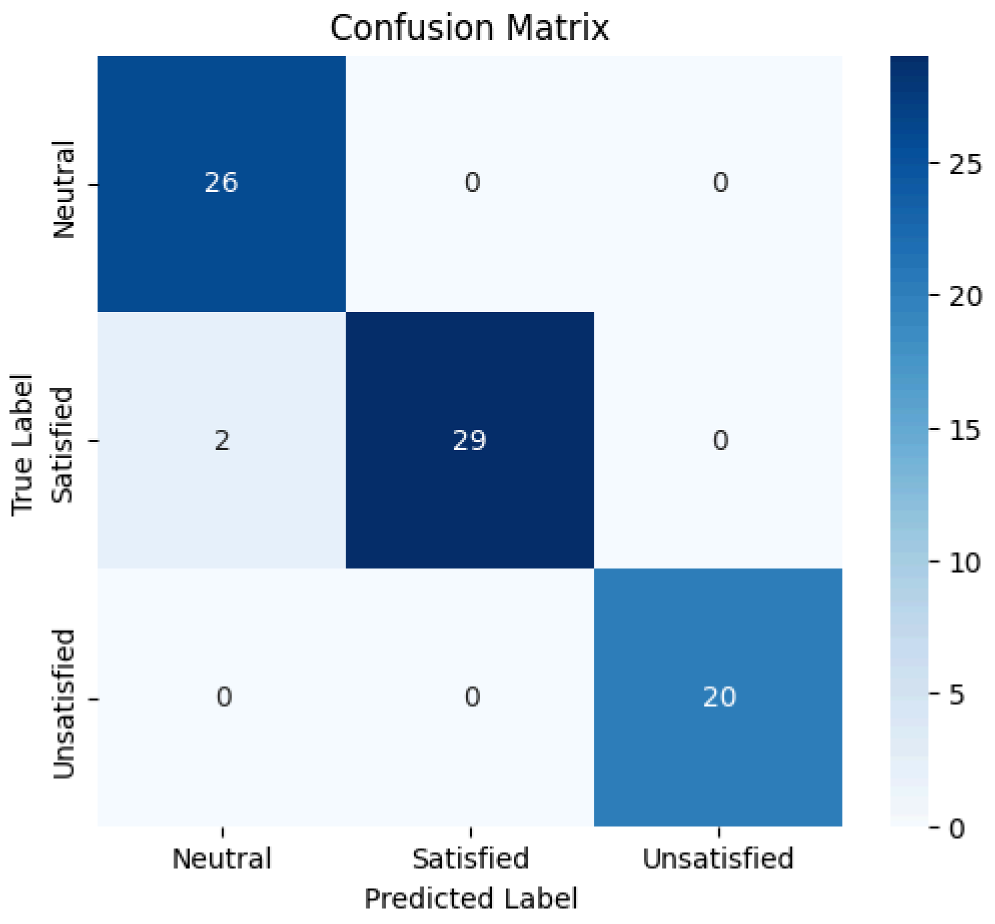
Why are we using Logistic Regression for this dataset?

- Logistic Regression is a classification algorithm used to predict categorical outcomes.
- Our dataset has customer satisfaction levels as target labels: Neutral, Satisfied, and Unsatisfied (multi-class classification).
- Logistic Regression is effective for interpretable, probabilistic classification and works well with structured data.

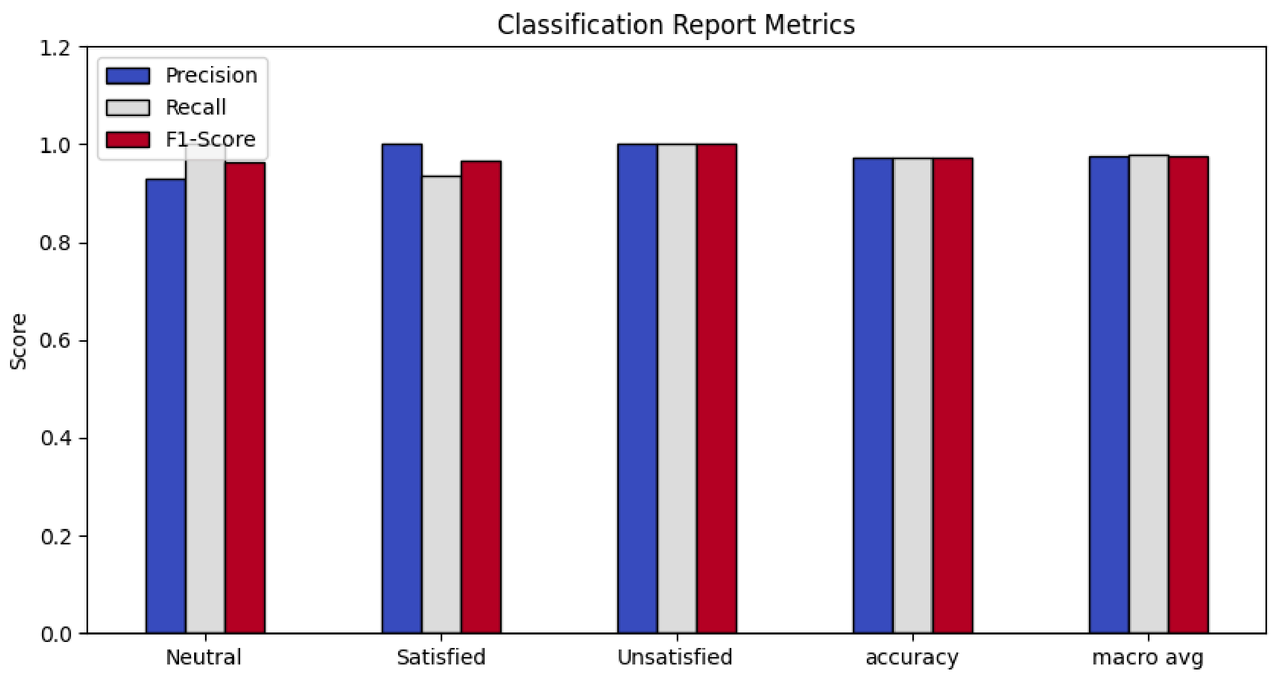
Evaluation Metrics & Interpretation:

- **Accuracy:** The model achieves 97.4% accuracy, meaning it correctly classifies 97.4% of test samples.
- **Confusion Matrix:** Shows actual vs. predicted classifications:
    1. 26 Neutral customers correctly classified, 2 misclassified as Satisfied.
    2. 29 Satisfied customers correctly classified, 2 misclassified as Neutral.
    3. 20 Unsatisfied customers correctly classified (perfect score).

```
# Compute Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
labels = ["Neutral", "Satisfied", "Unsatisfied"]


plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



```
report = classification_report(y_test, y_pred, output_dict=True)

# Convert to DataFrame for plotting
report_df = pd.DataFrame(report).T.iloc[:-1, :3]

# Plot the bar chart
report_df.plot(kind="bar", figsize=(10,5), colormap="coolwarm", edgecolor="black")
plt.title("Classification Report Metrics")
plt.ylabel("Score")
plt.xticks(rotation=0)
plt.ylim(0, 1.2)
plt.legend(["Precision", "Recall", "F1-Score"])
plt.show()
```



2.Random Forest:

Explanation of Random Forest Model

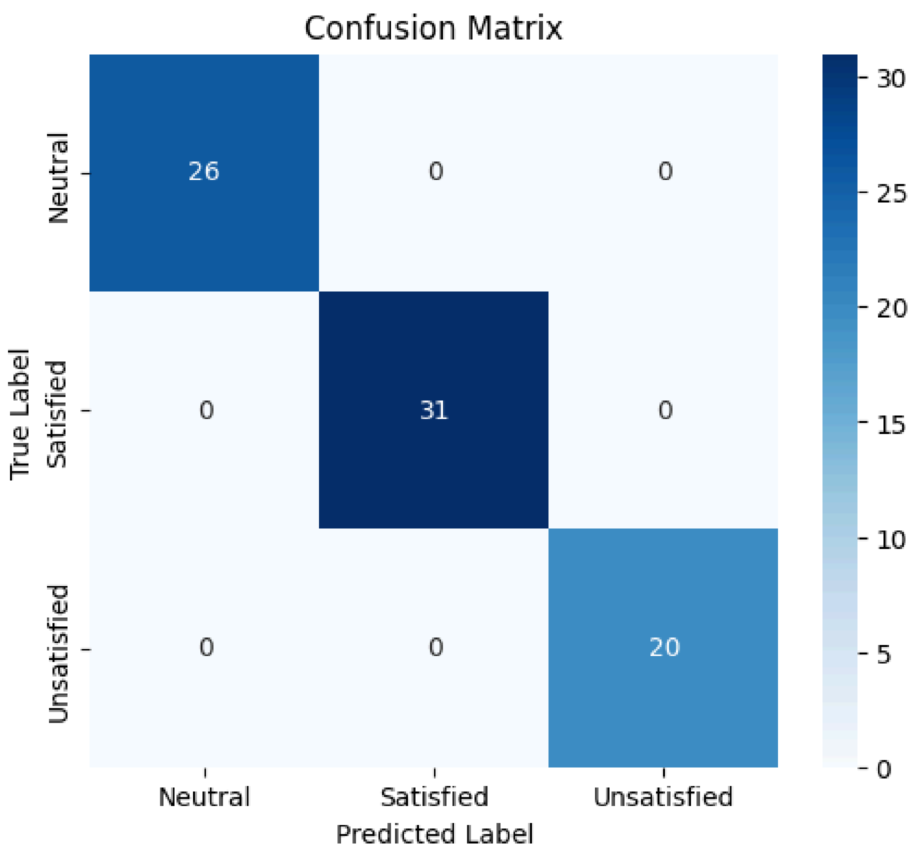
1. Why are we using Random Forest for this dataset?

- Random Forest is an ensemble learning method that builds multiple decision trees and averages their predictions for better accuracy and robustness.
- It is highly effective for classification tasks, handling non-linear relationships and feature interactions well.
- The dataset involves multi-class classification (Neutral, Satisfied, Unsatisfied), where Random Forest excels by reducing overfitting compared to single decision trees.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Initialize the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Predictions
y_pred_rf = rf_model.predict(X_test)

# Evaluation Metrics
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

```
Random Forest Accuracy: 1.0
Confusion Matrix:
 [[26  0  0]
 [ 0 31  0]
 [ 0  0 20]]
Classification Report:
               precision    recall  f1-score   support

     Neutral       1.00      1.00      1.00        26
   Satisfied       1.00      1.00      1.00        31
 Unsatisfied       1.00      1.00      1.00        20

    accuracy                           1.00        77
   macro avg       1.00      1.00      1.00        77
weighted avg       1.00      1.00      1.00        77
```

```python
# Compute Confusion Matrix
cm = confusion_matrix(y_test, y_pred_rf)
labels = ["Neutral", "Satisfied", "Unsatisfied"]

plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```



Model Performance and Results

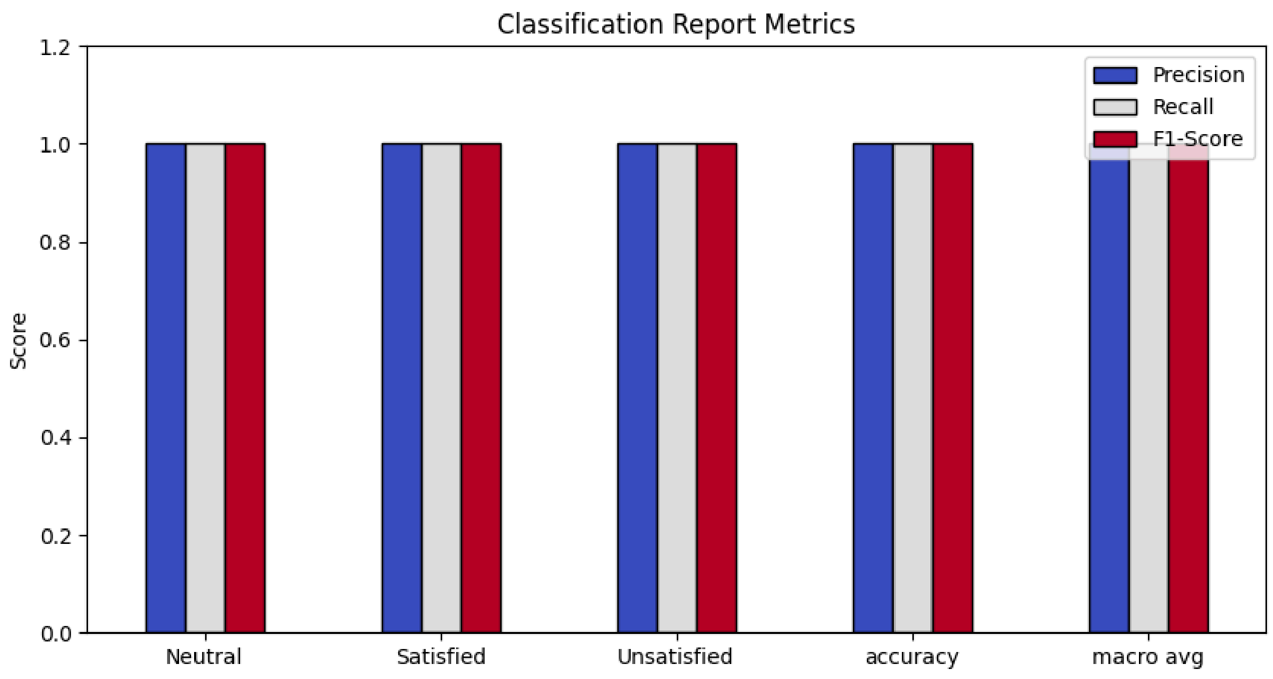
Confusion Matrix Analysis: The confusion matrix shows zero misclassifications, meaning every instance was correctly predicted.

Neutral: 26 correct

Satisfied: 31 correct

Unsatisfied: 20 correct

```
# classification report
report = classification_report(y_test, y_pred_rf, output_dict=True)

report_df = pd.DataFrame(report).T.iloc[:-1, :3]

report_df.plot(kind="bar", figsize=(10,5), colormap="coolwarm", edgecolor="black")
plt.title("Classification Report Metrics")
plt.ylabel("Score")
plt.xticks(rotation=0)
plt.ylim(0, 1.2)
plt.legend(["Precision", "Recall", "F1-Score"])
plt.show()
```

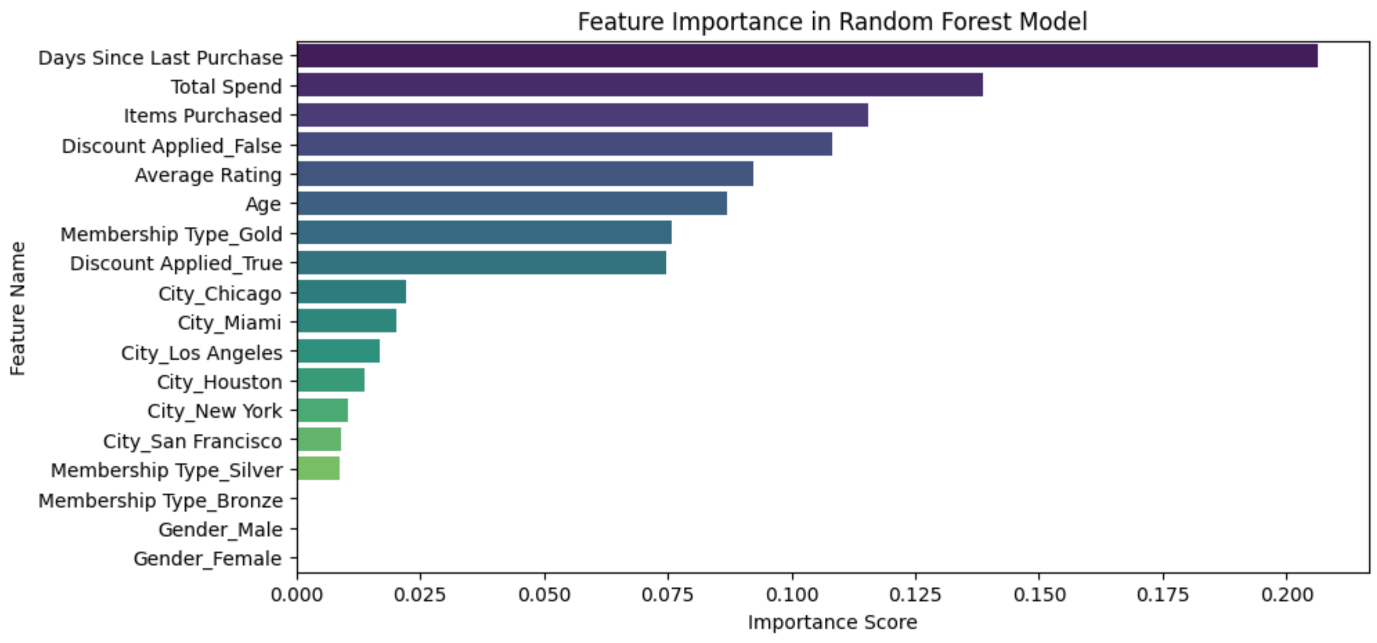


Analyzing Feature Importance in Random forest:

Why Are We Analyzing Feature Importance?

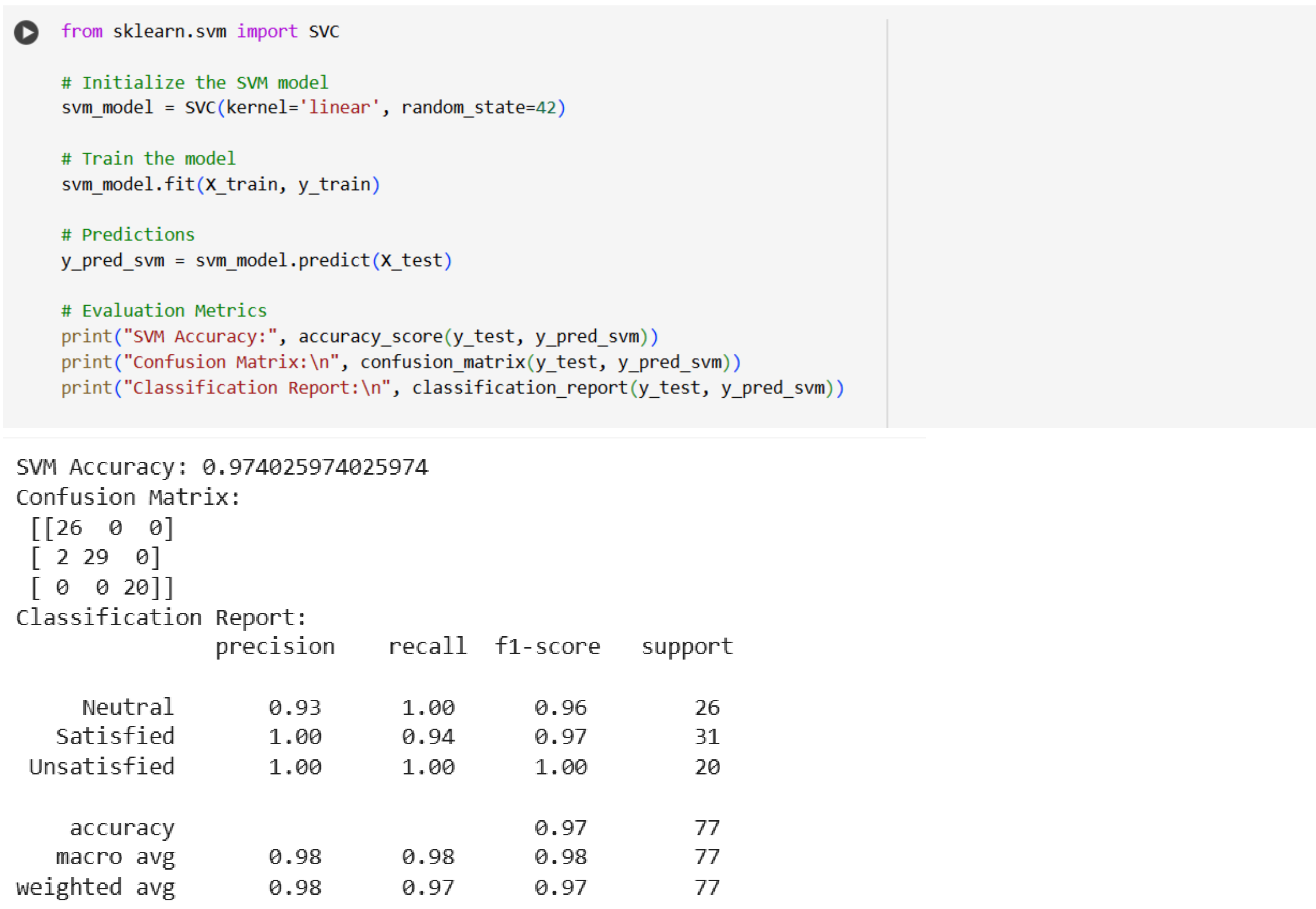
Feature importance helps identify which features influence predictions the most in a Random Forest model. It provides insights into the dataset, allowing us to focus on the most impactful variables.

```python
import numpy as np

importances = rf_model.feature_importances_
num_features = preprocessor.transformers_[0][2]
cat_features = preprocessor.transformers_[1][1]['encoder'].get_feature_names_out(categorical_features)
features = np.concatenate([num_features, cat_features])

# Create DataFrame for visualization
feat_imp_df = pd.DataFrame({"Feature": features, "Importance": importances})
feat_imp_df = feat_imp_df.sort_values(by="Importance", ascending=False)


plt.figure(figsize=(10,5))
sns.barplot(x="Importance", y="Feature", data=feat_imp_df, palette="viridis")
plt.title("Feature Importance in Random Forest Model")
plt.xlabel("Importance Score")
plt.ylabel("Feature Name")
plt.show()
```



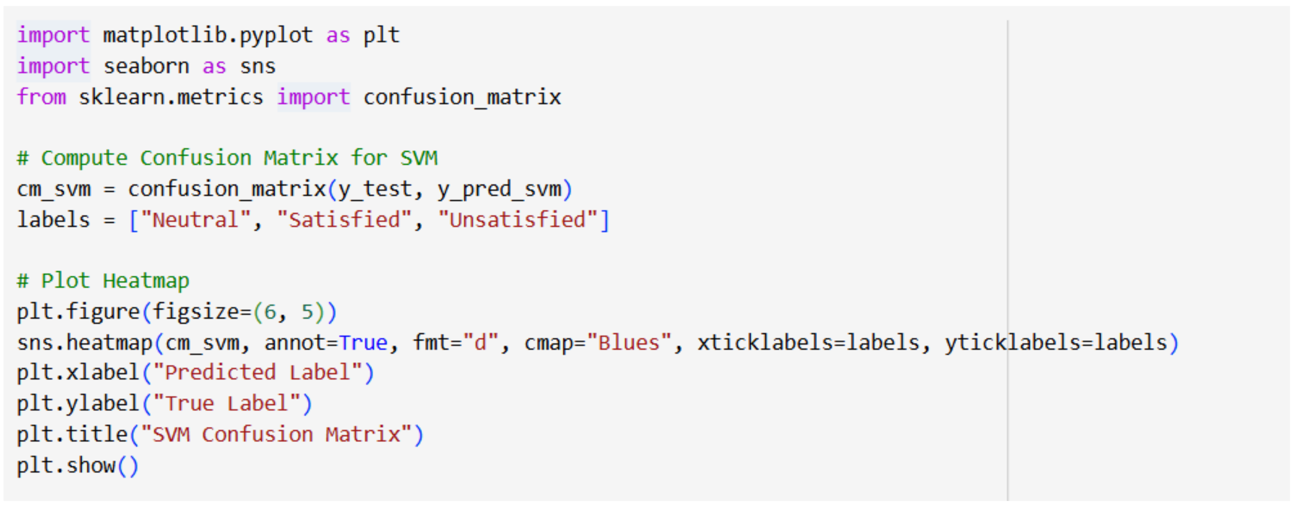
3.Support Vector Machine(SVM):

The SVM model is highly effective, especially in distinguishing Unsatisfied and Neutral customers. The minor misclassification (Satisfied – Neutral) suggests room for improvement, possibly by fine-tuning hyperparameters.

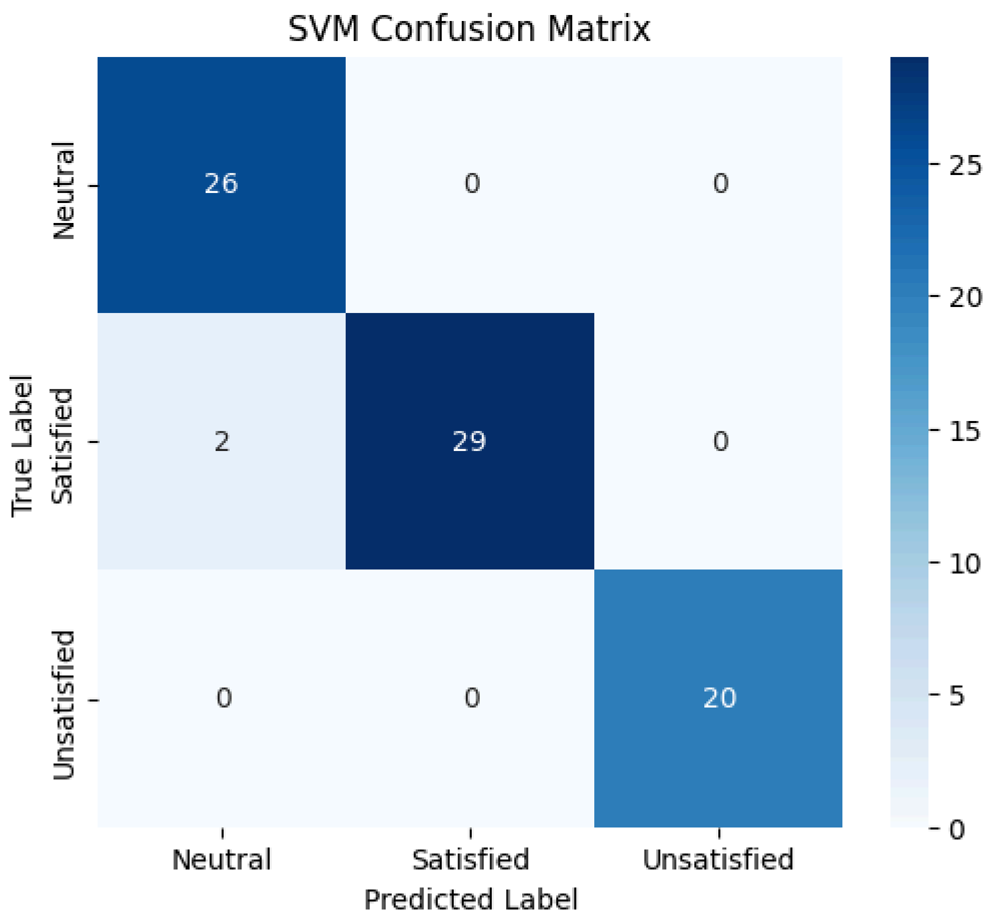


```python
from sklearn.svm import SVC

# Initialize the SVM model
svm_model = SVC(kernel='linear', random_state=42)

# Train the model
svm_model.fit(X_train, y_train)

# Predictions
y_pred_svm = svm_model.predict(X_test)

# Evaluation Metrics
print("SVM Accuracy:", accuracy_score(y_test, y_pred_svm))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
print("Classification Report:\n", classification_report(y_test, y_pred_svm))
```

```
SVM Accuracy: 0.974025974025974
Confusion Matrix:
 [[26  0  0]
 [ 2 29  0]
 [ 0  0 20]]
Classification Report:
               precision    recall  f1-score   support

     Neutral       0.93      1.00      0.96        26
   Satisfied       1.00      0.94      0.97        31
 Unsatisfied       1.00      1.00      1.00        20

    accuracy                           0.97        77
   macro avg       0.98      0.98      0.98        77
weighted avg       0.98      0.97      0.97        77
```



```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Compute Confusion Matrix for SVM
cm_svm = confusion_matrix(y_test, y_pred_svm)
labels = ["Neutral", "Satisfied", "Unsatisfied"]

# Plot Heatmap
plt.figure(figsize=(6, 5))
sns.heatmap(cm_svm, annot=True, fmt="d", cmap="Blues", xticklabels=labels, yticklabels=labels)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("SVM Confusion Matrix")
plt.show()
```



Support Vector Machine (SVM) model:

we can conclude the following:

- High Accuracy (97.4%)
- The model correctly predicts 97.4% of the test samples, indicating strong overall performance.

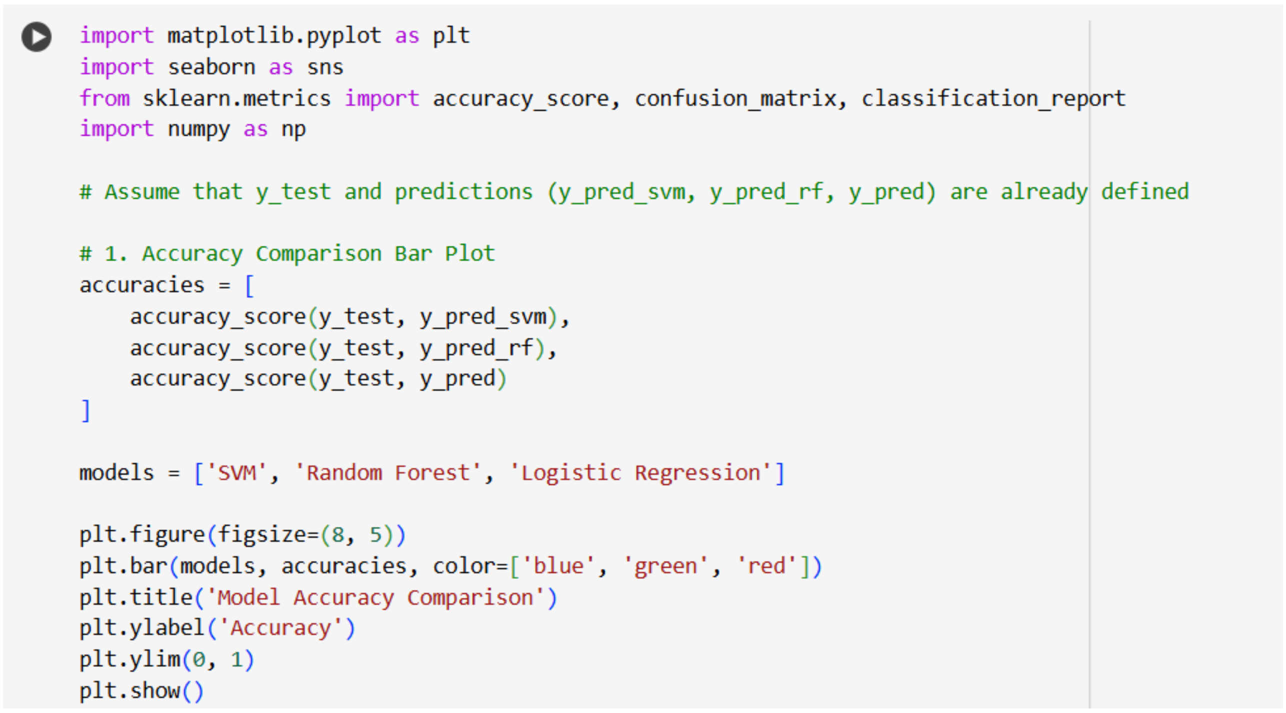
Confusion Matrix:

- Neutral: All 26 samples were correctly classified (100% recall).
- Satisfied: 29 out of 31 were correctly classified, with only 2 misclassified as Neutral.
- Unsatisfied: Perfect classification (100% recall and precision).

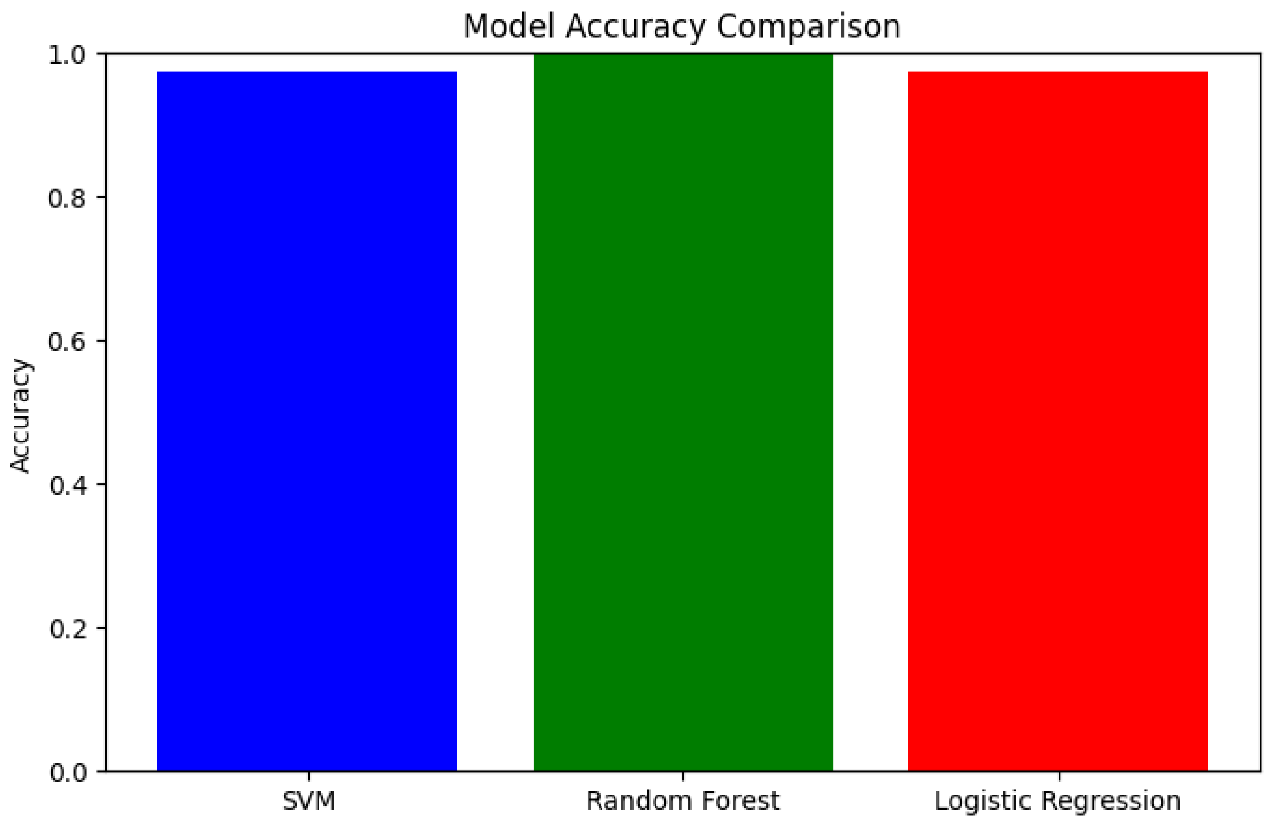
COMPARISON:

1. Accuracy Comparison

bar plot to compare the accuracy of each model.



```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import numpy as np

# Assume that y_test and predictions (y_pred_svm, y_pred_rf, y_pred) are already defined

# 1. Accuracy Comparison Bar Plot
accuracies = [
    accuracy_score(y_test, y_pred_svm),
    accuracy_score(y_test, y_pred_rf),
    accuracy_score(y_test, y_pred)
]

models = ['SVM', 'Random Forest', 'Logistic Regression']

plt.figure(figsize=(8, 5))
plt.bar(models, accuracies, color=['blue', 'green', 'red'])
plt.title('Model Accuracy Comparison')
plt.ylabel('Accuracy')
plt.ylim(0, 1)
plt.show()
```



E-commerce Model Comparison (SVM, Logistic Regression, Random Forest)

Aspect	SVM	Logistic Regression	Random Forest
Model type	Linear/Non-linear Classifier (Linear Kernel)	Linear Classifier	Ensemble of Decision Trees
EDA	Describe,countplot, histograms	Describe,countplot, histograms	Describe,countplot, histograms
Preprocessing	Handling Missing values,Encode	Handling Missing values,Encode	Handling Missing values,Encode
Class Imbalance Handling	- Use SMOTE to balance class distribution	- Use SMOTE to balance class distribution	- Use SMOTE to balance class distribution
Training vs Test results	- Train Accuracy: 97.4% - Test Accuracy: 97.4%	- Train Accuracy: 97.4% - Test Accuracy: 97.4%	- Train Accuracy: 100% - Test Accuracy: 100%

Impact of project on human , societal ,ethical and sustainable development

i)Societal Impact:

Enhanced Customer Experience:

- Helps businesses personalize recommendations, improve customer service, and create better shopping experiences.
- Reduces frustration by providing relevant products, leading to customer satisfaction.

Business Growth & Employment Opportunities:

- Supports small businesses by optimizing marketing strategies and pricing decisions.
- Generates jobs in data analytics, AI, and customer relationship management.

Consumer Protection & Fraud Detection:

- Identifies fraudulent transactions and unusual purchasing behavior, ensuring safe transactions for users.
- Helps prevent scams, fake reviews, and unauthorized transactions.

ii)Ethical Impact:

Data Privacy & Security:

- Handling personal data responsibly is crucial. Ethical concerns arise if data is misused, sold, or exposed to breaches.
- Ensuring compliance with GDPR (General Data Protection Regulation) and CCPA (California Consumer Privacy Act) is important.

Bias in AI Models:

- If training data is biased, AI may favor certain demographics, leading to unfair recommendations.
- Ensuring fairness and diversity in data collection and model training is critical.

Consumer Manipulation:

- Overuse of predictive analytics can lead to aggressive marketing strategies, exploiting consumer habits.
- Ethical AI should prioritize transparency and allow users control over recommendations.

iii)Sustainable Development Impact

Reducing Waste & Overproduction:

Predicting customer preferences accurately helps businesses reduce overproduction and inventory waste, leading to sustainable consumption.

Optimized Logistics & Carbon Footprint Reduction:

AI-driven demand forecasting helps optimize supply chain management, reducing unnecessary transportation and storage, which lowers CO_2 emissions.

Sustainable Marketing Practices:

Helps businesses target eco-conscious consumers by recommending sustainable products and avoiding mass, wasteful advertising campaigns.

Future works to improve the model:

Data Expansion:
Collect more customer data to increase the sample size and improve the model's ability to generalize across different customer behaviors and characteristics.

Feature Engineering:
Create new features from existing data, like recency, frequency, and monetary value (RFM), which could provide additional insight into customer satisfaction and spending behaviors.

Time Series Analysis:
Implement time-series forecasting techniques to predict customer behavior over time, such as predicting when a customer will make the next purchase or the likelihood of churn based on past purchasing data.

References

Dataset- https://www.kaggle.com/datasets/uom190346a/e-commerce-customer-behavior-dataset

Researchpaper-https://www.researchgate.net/profile/Akhilesh-Waoo-2/publication/369967977_Customer_Behavior_Analysis_in_E-Commerce_using_Machine_Learning_Approach_A_Survey/links/6436e13a609c170a13111400/Customer-Behavior-Analysis-in-E-Commerce-using-Machine-Learning-Approach-A-Survey.pdf?utm_source=chatgpt.com

SVM-https://www.ibm.com/think/topics/support-vector-machine