

OBJECTIVE

Develop a predictive model to classify customer segments based on various E-Commerce related features. This project involves using Python for data preprocessing and exploratory data analysis (EDA), applying machine learning algorithms for model building, and utilizing Tableau for visualization and insights.

INTRODUCTION

WHY CUSTOMER SEGMENTATION IS IMPORTANT?

Segmenting customers in e-commerce is valuable because it helps businesses tailor their strategies to meet the specific needs, preferences, and behaviors of different customer groups.

Here's a brief overview of why it is important:

Personalized Marketing

- **Targeted Campaigns:** Segmentation enables businesses to design personalized marketing campaigns that resonate with specific customer groups, increasing engagement and conversion rates.
- **Relevant Recommendations:** By understanding customer preferences, businesses can recommend products more likely to appeal to each segment.

Improved Customer Experience

- **Customized Offers:** Offering personalized discounts or promotions based on customer behavior increases satisfaction and loyalty.
- **Tailored Communication:** Addressing customers in their preferred tone or medium improves the overall experience.

Resource Optimization

- Efficient allocation: Segmentation ensures marketing budgets and efforts are focused on high-value or high potential customer groups.
- Product Development: Insights from segments can guide the creation or improvement of products tailored to specific needs.

Enhanced Retention and Loyalty

- Retention Strategies: Identifying at risk customers through segmentation helps businesses implement retention strategies before they churn.
- Loyalty Programs: Tailoring loyalty programs to specific segments increase their effectiveness and encourages repeat purchases.

Actionable Insights

- Behavior Analysis: Understanding the difference in customer behavior across segments allows businesses to predict trends and plan accordingly.
- Profitability Analysis: Identifying which segments are most profitable helps prioritize efforts on high-return areas.

Competitive Advantage

- Better Understanding: A detailed understanding of customer segments gives businesses an edge in anticipating and meeting customer demands.
- Differentiation: Offering unique value propositions for each segment helps stand out in a competitive market.

Examples of Segmentation in E-commerce

- Demographic Segmentation: Age, gender, income level.
- Behavioral Segmentation: Purchase history, browsing patterns, response to promotions.
- Geographic Segmentation: Location-based strategies, such as local offers or region-specific campaigns.
- Psychographic Segmentation: Lifestyle, values, or interests influencing purchase decisions.

HOW WILL THIS PREDICTIVE MODEL HELP?

The predictive model for customer segmentation can significantly contribute to personalized marketing, improving customer retention and increasing sales in the following ways:

Personalized Marketing

Targeted Campaigns

- HOW: By identifying customer segments (eg. High-value buyers, occasional shoppers, etc), businesses can create campaign tailored to the unique needs and preferences of each group.
- IMPACT: Higher engagement rates, and better ROI on marketing campaigns.

Product Recommendations

- HOW: The model can suggest products based on segment-specific preferences or past-purchasing behavior.
- IMPACT: Personalized product suggestions encourage additional purchases and enhance customer satisfaction.

Customized Offers

- HOW: High-spending customers may receive exclusive discounts and new customers might get welcome offers.
- IMPACT: Incentivizes purchases by aligning offers with customer behavior.

Improving Customer Retention

Identifying at-risk customers

- HOW: Segments of Customers with reduced engagement (eg. Decreased purchase frequency) can be flagged as at-risk.
- IMPACT: Businesses can proactively engage these customers with retention strategies like reactivation emails or loyalty programs.

Loyalty programs

- HOW: The model helps design loyalty programs by identifying segments that respond well to rewards or points systems.
- IMPACT: Increased Customer Lifetime value (CLV) through consistent repeat purchases.

Enhanced Customer Support

- HOW: Segments with frequent complaints or low satisfaction scores can be prioritized for improved customer support.
- IMPACT: Builds trust and reduces churn by addressing pain points effectively.

Increasing Sales

Upselling and Cross-selling Opportunities

- HOW: Use segments like high-value buyers to suggest premium products or complementary items based on past purchases.
- IMPACT: Boosts Average Order Value and Total revenue.

Seasonal Promotions and Timings

- HOW: Segments can reveal when customers are more likely to purchase, enabling precise timing of promotions.
- IMPACT: Maximizing sales during peak purchasing window.

Optimizing Inventory and product offerings

- HOW: Insights from segmentation help identify fast-moving products for specific segments, ensuring stock availability and reducing overstock.
- IMPACT: Improves profitability by aligning inventory with demand patterns.

POTENTIAL PROJECT GOAL

The goal of the project is to:

1. Classify Customers into Segments: Predict the segment (Customer_Segment) a customer belongs to using other features.
2. Understand Segmentation Drivers: Identify the key factors influencing customer grouping to guide marketing and sales strategies.
3. Enable Strategic Insights: Use the segmentation to design targeted marketing, improve customer experiences, and boost revenue.

Segmentation Type based on the dataset

1. Behavioral Segmentation:

- Why: Features like Spending_Score, Years_as_customer, Number_of_orders, and Last_Activity indicate customer behaviors and interactions.
- Use Case: Classify customers into frequent shoppers, occasional buyers, and disengaged users.

2. Demographic Segmentation:

- Why: Features like Age, Gender, and Customer_Region helps identify groups based on demographic characteristics.
- Use Case: Tailor offerings for specific age groups or regions.

3. Psychographic Segmentation:

- Why: Loyalty_Membership, and Preferred_Payment_Method indicates customer preferences and values.
- Use case: Align loyalty programs or payment options with customer preferences.

4. Hybrid Segmentation:

- Why: Combining behavioral and demographic factors provides richer insights.
- Use Case: Create comprehensive profiles for strategic decision-making.

Data Preprocessing

- Load the Dataset: Loading the dataset and getting to know the dataset.

In [2]:

```
1 #LOADING THE DATASET
2 project= pd.read_csv("/Users/sumaiyahairshad/Desktop/MS In Data Science/CAPSTONE PROJECT/E Commerce/E-commerce Cus
3 project
```

Out[2]:

Customer_ID	Customer_Name	Age	Gender	Annual_Income	Spending_Score	Marital_Status	Product_Category	Years_as_Customer	Number_of_Orders
0	Michelle Charles	69	0	187054	60	1	Fashion	0	7
1	Keyla Medina	62	0	166578	36	1	Sports	7	2
2	3 Ronald Hoffman	52	0	166523	46	0	Home	4	6
3	4 Sandra McGuire	56	1	183559	83	0	Sports	5	8
4	Andrew McDonald	56	0	17481	60	0	Sports	5	8
...
21995	Tanya Kramer	22	1	80778	25	1	Home	2	4
21996	21997 Melvin Valencia	38	0	129255	14	0	Home	12	2
21997	21998 Shannon Mitchell	58	0	194525	81	1	Sports	6	1
21998	21999 Alison West	23	0	2831	82	1	Home	2	6
21999	22000 Vickie Townsend	47	0	191495	23	1	Electronics	11	

22000 rows × 20 columns

In [7]:

```
1 project.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22000 entries, 0 to 21999

In [7]:

```
1 project.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22000 entries, 0 to 21999
Data columns (total 20 columns):
 # & dtype
 0 Customer_ID int64
 1 Customer_Name object
 2 Age int64
 3 Gender int64
 4 Annual_Income int64
 5 Spending_Score int64
 6 Marital_Status int64
 7 Product_Category object
 8 Years_as_Customer int64
 9 Years_living int64
 10 Average_Order_Value float64
 11 Loyalty_Membership int64
 12 Income int64
 13 Preferred_Payment_Method object
 14 Preferred_Discovery_Option object
 15 Device_Used object
 16 Last_Activity int64
 17 Location_Region object
 18 Review_Score int64
 19 Total_Purchases int64
dtypes: float64(1), int64(13), object(6)
memory usage: 3.4+ MB

In [10]:

```
1 project.tail(10)
```

- Clean The Data: Here, I'll fix the missing data, remove duplicate values, and organize everything neatly.

1. Handling missing values

```
In [12]: 1 # TO KNOW MISSING VALUES
          2 project.isnull().sum()

Out[12]: Customer_ID      0
          Customer_Name     0
          Age                 0
          Annual_Income      0
          Spending_Score      0
          Marital_Status       0
          Product_Catgory     0
          Years_Ag_Customer    0
          Number_of_Oders       0
          Average_Order_Value   0
          Loyalty_Program       0
          Distance_Usage        0
          Preferred_Payment_Method 0
          Preferred_Delivery_Option 0
          Device_Used           0
          Last_Buy              0
          Customer_Region        0
          Review_Score           0
          Customer_Segment       0
          dtype: int64
```

This shows that there are no missing values in our dataset.

2. Handling duplicate values

```
In [11]: 1 # TO KNOW DUPLICATED VALUES
          2 project.duplicated().sum()

Out[11]: 0
```

This shows that we don't have any duplicate values as well.

3. Datatypes: Here we'll observe the datatypes and convert them if necessary.

- Numerical Columns (int64, float64): The numerical columns present in our dataset are Age, Annual_Income, Spending_Score and Number_of_Oders.
- Categorical Columns (Object): The categorical columns present in our dataset are Product_Category, Preferred_Payment_Method, Device_Used and Customer_Region.
- Target Variable: The target variable in our dataset is Customer_Segment which has numerical datatype and is already labelled.

So, we'll need to convert the categorical columns Product_Category, Preferred_Payment_Method, Device_Used and Customer_Region into Numerical datatype using **encoding**. This conversion is done because Machine Learning model works best with numerical data.

Encoding Categorical Values

- To encode categorical values, we'll first check the categorical column for unique values as this will help in deciding the type of encoding (i.e., label-encoding or one-hot type encoding).

The screenshot shows a Jupyter Notebook interface with the title "jupyter E-Commerce_Project Last Checkpoint: 11/19/2024 (unsaved changes)". The notebook has a Python 3 (ipykernel) kernel. The code in cell In [11] is:

```
1 #CHECKING UNIQUE VALUES OF CATEGORICAL COLUMNS TO BE ENCODED
2
3 project['Product_Category'].unique()
```

The output Out[11] is:

```
array(['Fashion', 'Sports', 'Home', 'Beauty', 'Electronics'], dtype=object)
```

Cells In [12], In [13], and In [14] show similar unique value checks for Preferred_Payment_Method, Device_Used, and Customer_Region respectively, with their respective outputs.

- For Product_Category, Preferred_Payment_Method, Device_Used and Customer_region I'll prefer one-hot encoding as all are nominal data(no order), the unique values are small, they have no inherent order and are independent categories.

The screenshot shows a Jupyter Notebook interface with the title "jupyter E-Commerce_Project Last Checkpoint: 11/19/2024 (autosaved)". The notebook has a Python 3 (ipykernel) kernel. The code in cell In [29] is:

```
1 #CONVERTING THE CATEGORICAL COLUMNS INTO NUMERICAL COLUMNS USING ENCODING
2
3 one_hot_columns=['Product_Category','Preferred_Payment_Method','Device_Used','Customer_Region']
4 project=pd.get_dummies(project, columns=one_hot_columns)
```

The output Out[29] shows the original project DataFrame.

Cell In [30] contains the command:

```
In [30]: 1 project
```

The output Out[30] displays the converted DataFrame with one-hot encoding applied to the categorical columns. The columns include users, Average_Order_Value, Preferred_Payment_Method_Debit_Card, Preferred_Payment_Method_Net_Banking, Preferred_Payment_Method_PayPal, Device_Used_Desktop, and Device_Used_Tablet.

users	Average_Order_Value	Preferred_Payment_Method_Debit_Card	Preferred_Payment_Method_Net_Banking	Preferred_Payment_Method_PayPal	Device_Used_Desktop	Device_Used_Tablet
78	182.40	0	0	1	0	0
25	342.85	0	0	0	1	0
68	275.57	1	0	0	0	0
88	97.62	0	1	0	0	0
86	438.02	0	1	0	1	0
...
41	642.65	0	0	0	0	0
23	598.19	1	0	0	1	0
11	204.23	0	0	0	1	0
59	247.71	0	0	0	0	0
6	736.60	0	0	0	0	0

- Now to check if all the columns are converted into numerical datatype.

```
In [48]: 1 #TO CHECK IF ALL THE COLUMNS ARE CONVERTED INTO NUMERICAL DATATYPE OR NOT
          project.dtypes
```

```
Out[48]: Customer_ID           int64
Customer_Name        object
Age                  int64
Gender                int64
Annual_Income         int64
Spending_Score       int64
Marital_Status        int64
Years_as_Customer    int64
Number_of_Orders     int64
Average_Order_Value   uint64
Loyalty_Membership   int64
Discount_Usage        object
Preferred_Delivery_Option int64
Last_Activity         int64
Review_Score          int64
Customer_Segment      int64
Product_Category_Beauty uint8
Product_Category_Electronics uint8
Product_Category_Fashion uint8
Product_Category_Home  uint8
Product_Category_Sports uint8
Preferred_Payment_Method_Credit_Card uint8
Preferred_Payment_Method_Debit_Card uint8
Preferred_Payment_Method_Net_Banking uint8
Preferred_Payment_Method_PayPal  uint8
Device_Used/Desktop  uint8
Device_Used/Mobile   uint8
Device_Used/Tablet   uint8
Customer_Region_Asia  uint8
Customer_Region_Europe uint8
```

EDA-EXPLORATORY DATA ANALYSIS

- Before performing EDA, I'll first remove the irrelevant columns (Customer_ID and Customer_Name) as it'll in no way contribute to meaningful patterns and insights.
- For this I'll use the drop method in pandas.

```
In [21]: 1 #REMOVING IRRELEVANT FEATURES(Customer_ID and Customer_Name)
          project_cleaned=project.drop(columns=['Customer_ID','Customer_Name'])
          project_cleaned
          5 #TO MAKE SURE BOTH THE COLUMNS ARE DROPPED
          project_cleaned.head(1)
```

```
Out[21]:   Age  Gender  Annual_Income  Spending_Score  Marital_Status  Years_as_Customer  Number_of_Orders  Average_Order_Value  Loyalty_Membership  Discount
0   69       0        192704            60             1              8               78            182.40                 1
1   62       0       169578            36             1              7               25            342.85                 1
2   52       0       166523            48             0              4               68            275.57                 1
3   56       1       193569            83             0              5               88            97.62                  1
4   56       0       57461             60             0              5               86            438.02                 0
```

5 rows × 30 columns

UNIVARIATE ANALYSIS

- Now, I'll perform Univariate Analysis to understand the data better.
 - This is for Column "AGE"

jupyter E-Commerce_Project Last Checkpoint: Last Saturday at 6:12 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3 (ipykernel) ○

UNIVARIATE ANALYSIS

In [26]:

```
1 #!/usr/bin/python3
2
3 #FOR COLUMN AGE
4
5 project_cleaned['Age'].describe()
6
7
8 #This shows min age is 18 and max age is 69 with mean age being 43 yrs
```

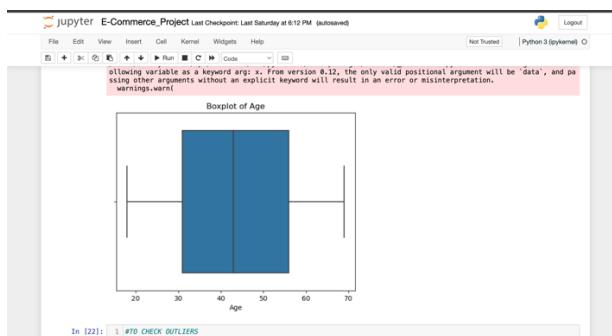
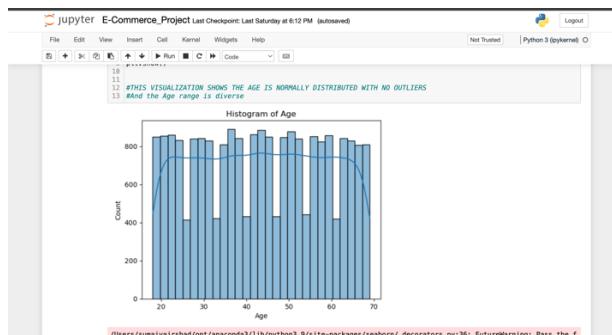
Out[26]:

```
count    22000.000000
mean        43.424273
std         14.770789
min         18.000000
25%         31.000000
50%         43.000000
75%         56.000000
max         69.000000
Name: Age, dtype: float64
```

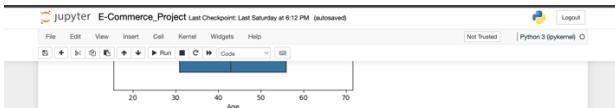
In [27]:

```
1 #VISUALIZATION OF COLUMN AGE
2
3 sns.histplot(project_cleaned['Age'], kde=True)
4 plt.title("Histogram of Age")
5 plt.show()
6
7 sns.boxplot(project_cleaned['Age'])
8 plt.title("Boxplot of Age")
9 plt.show()
```

10



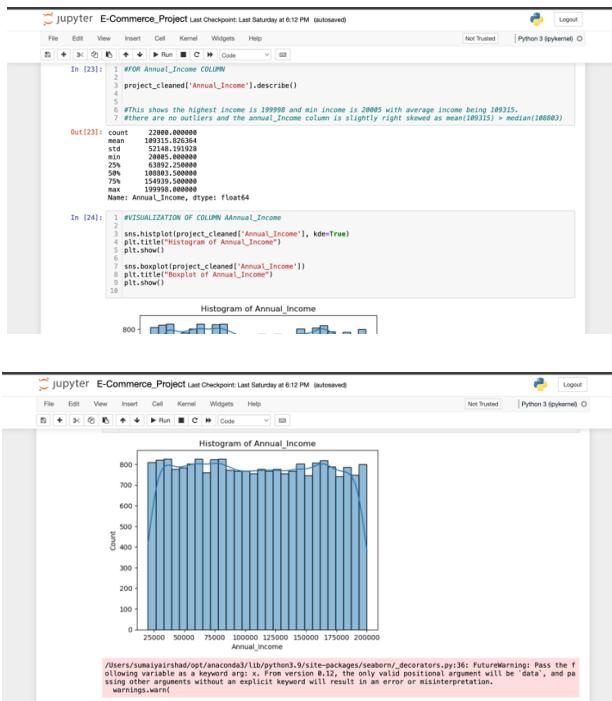
- This visualization shows that the age is distributed normally with no outliers. Also, the range of Age is diverse.
- We can also confirm that there are no outliers by using the below code.



```
In [22]: 1 #TO CHECK OUTLIERS
2 Q1 = project_cleaned['Age'].quantile(0.25) # First quartile
3 Q3 = project_cleaned['Age'].quantile(0.75) # Third quartile
4 IQR = Q3 - Q1 # Interquartile range
5
6 lower_bound = Q1 - 1.5 * IQR # Lower boundary for outliers
7 upper_bound = Q3 + 1.5 * IQR # Upper boundary for outliers
8
9 print("Lower Bound:", lower_bound)
10 print("Upper Bound:", upper_bound)
11
12 # Check for outliers
13 outliers = project_cleaned[(project_cleaned['Age'] < lower_bound) | (project_cleaned['Age'] > upper_bound)]
14
15 print("Number of Outliers:", len(outliers))
16
17
18 Lower_Bound: -6.5
19 Upper_Bound: 93.5
20 Number of Outliers: 0
```

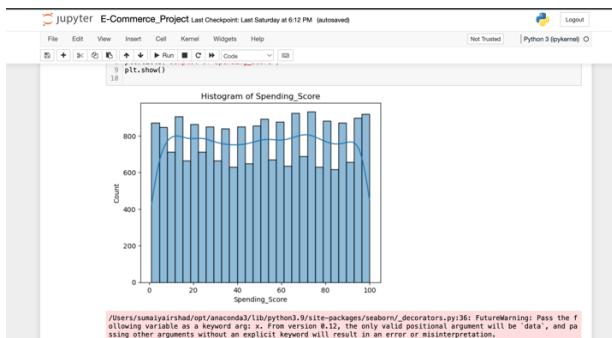
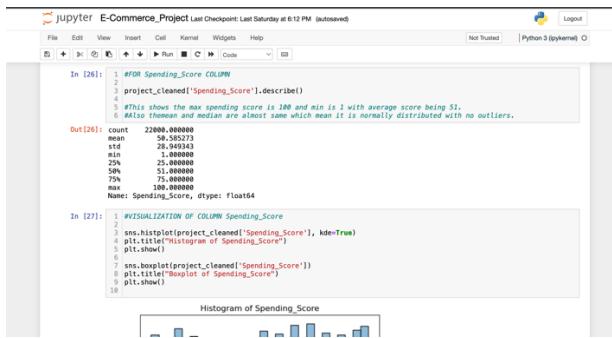
In [23]: 1 #FOR Annual_Income COLUMN

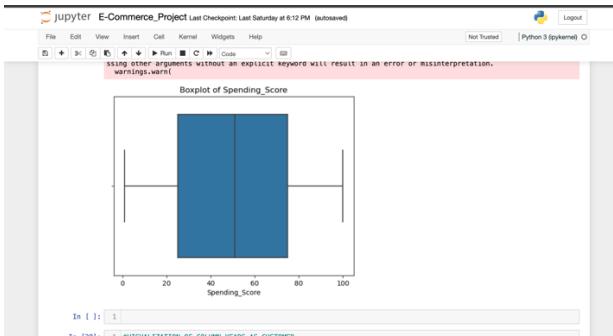
- For Column “ANNUAL_INCOME”



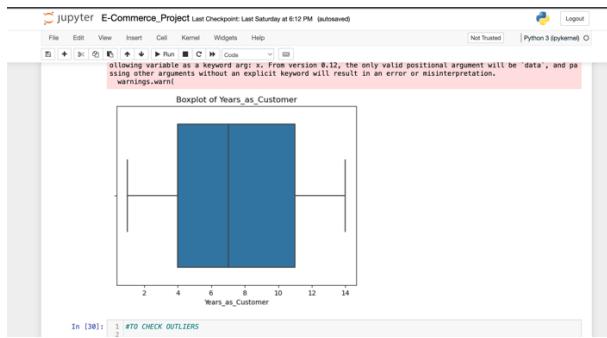
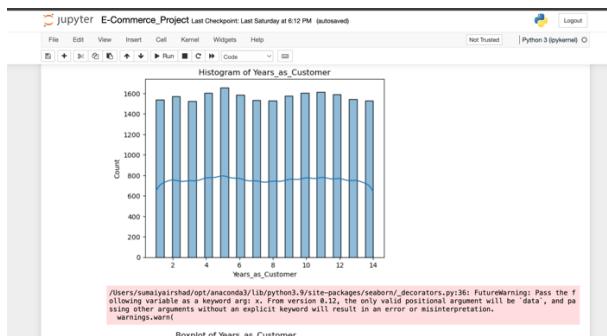
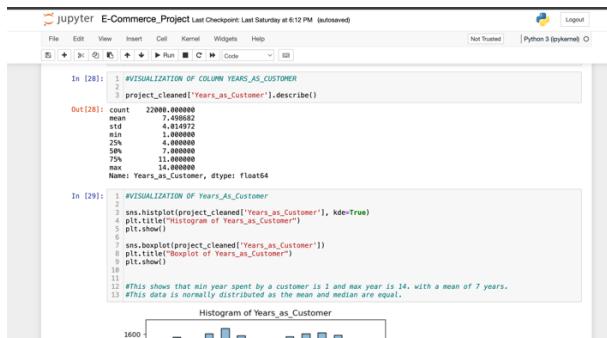


- This shows that the income ranges from 20005 USD to 199998 USD with average income being 109315 USD.
 - Also, there are no outliers but the annual_income column is slightly right skewed as mean (109315) > median (108803).
 - We can check the outliers by using the same code as above.
 - For Column “SPENDING SCORE”

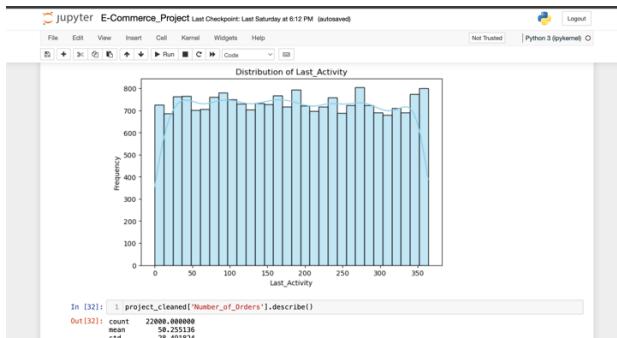
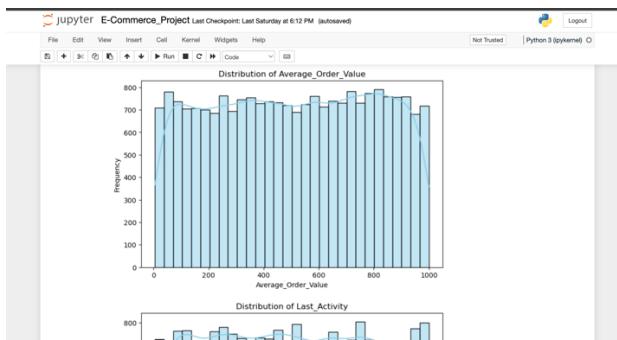
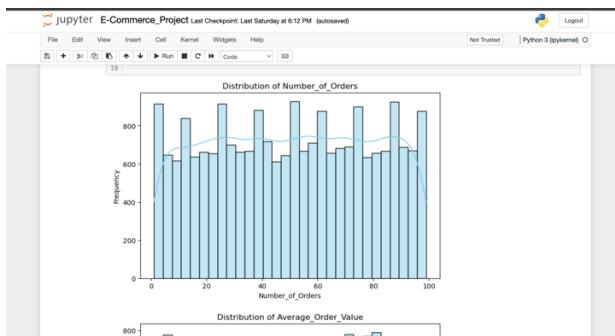
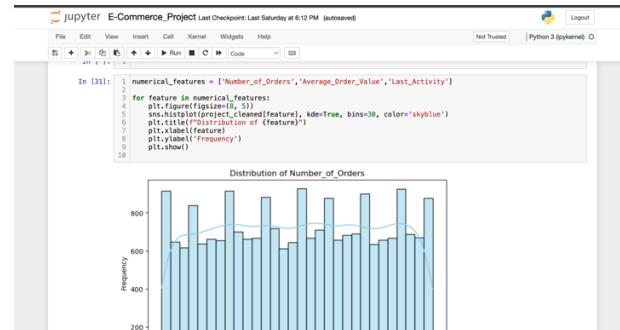




- This visualization shows that the maximum spending score is 100 and min score is 1 with an average of 51 spending score.
- The mean and median are almost same so this column is normally distributed with no outliers.
- Visualization of column “YEARS AS CUSTOMER”



- This visualization shows that the minimum year spent by a customer is 1 and max year is 14 with the average year spent being 7 years.
 - This column is normally distributed as mean and median are equal.
 - Now, I'll do a combined code for the visualization of rest of the numerical columns.



```

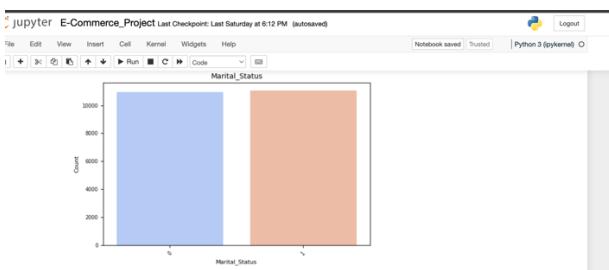
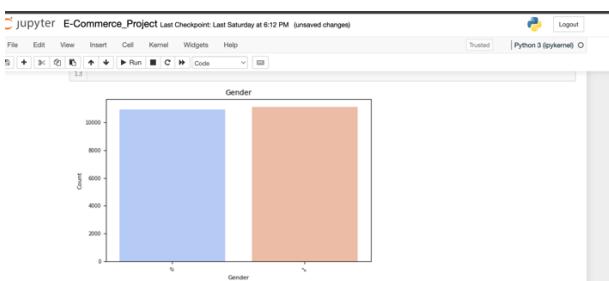
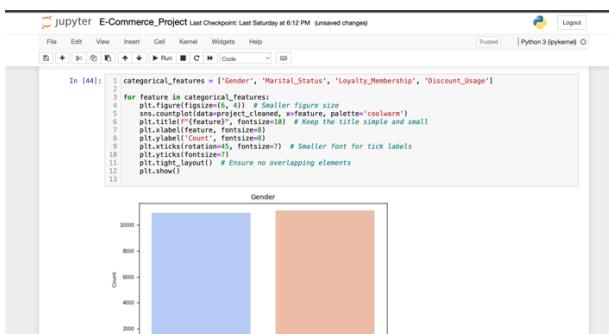
jupyter E-Commerce_Project Last Checkpoint: Last Saturday at 6:12 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [32]: project_cleaned['Number_of_Orders'].describe()
Out[32]:
count    22000.000000
mean     58.255130
std      28.491824
min      1.000000
25%     26.000000
50%     58.000000
75%     75.000000
max    999.930000
Name: Number_of_Orders, dtype: float64

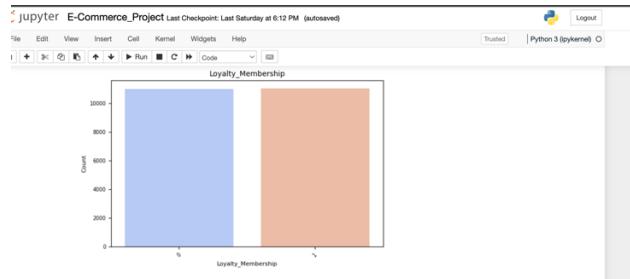
In [33]: project_cleaned['Average_Order_Value'].describe()
Out[33]:
count    22000.000000
mean     595.580000
std      287.060153
min      5.000000
25%     257.625000
50%     507.005000
75%     753.000000
max    999.930000
Name: Average_Order_Value, dtype: float64

In [34]: project_cleaned['Last_Activity'].describe()
Out[34]:
count    22000.000000
mean     185.156113
std      105.000000
min      0.000000
25%     100.000000
50%     181.000000
75%     271.000000
max    999.930000
Name: Last_Activity, dtype: float64

```

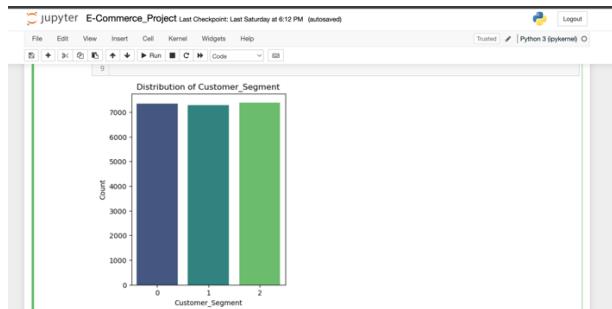
- Now, visualization of categorical features.





- Now, we'll see the distribution of Target Variable "CUSTOMER SEGMENT"

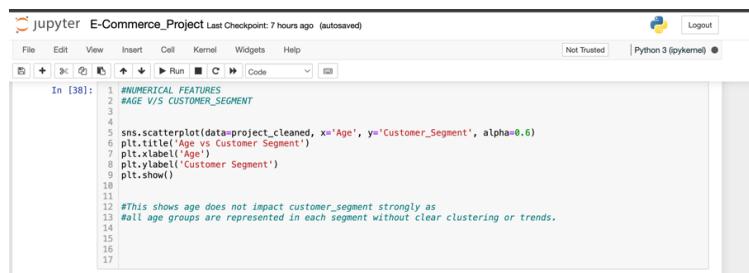
```
jupyter E-Commerce_Project Last Checkpoint: Last Saturday at 6:12 PM (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted Python 3 (ipykernel) O
In [36]: 1 #TO CHECK THE DISTRIBUTION OF TARGET VARIABLE.
2
3 plt.figure(figsize(4, 5))
4 sns.countplot(data=project_cleaned, x="Customer_Segment", palette='viridis')
5 plt.title("Distribution of Customer_Segment")
6 plt.xlabel("Customer_Segment")
7 plt.ylabel("Count")
8 plt.show()
```



- The heights of the bars for segments 0,1 and 2 are almost equal implying segments are evenly distributed.
- Now, since the customer segments are well-balanced our predictive model will not be biased toward any specific segment due to class imbalance.
- Now, we can move ahead and perform bivariate analysis with the target variable.

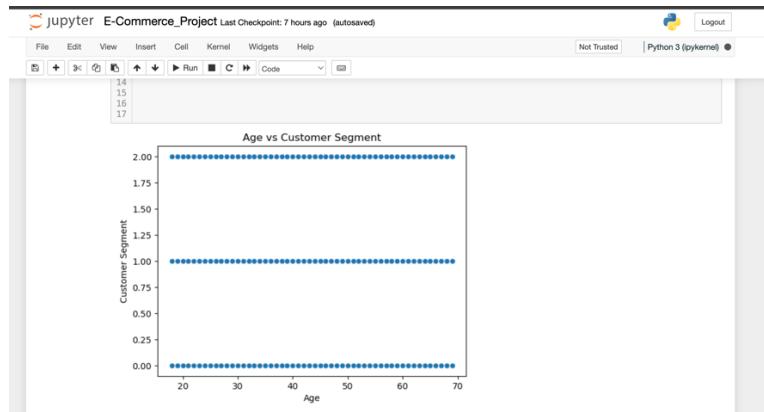
BIVARIATE ANALYSIS

- For Numerical features I'll use scatter plots to check the relationship between features and the target column. And I'll use boxplots to observe the distribution of numerical features across segments.
- 'AGE V/S CUSTOMER_SEGMENT'

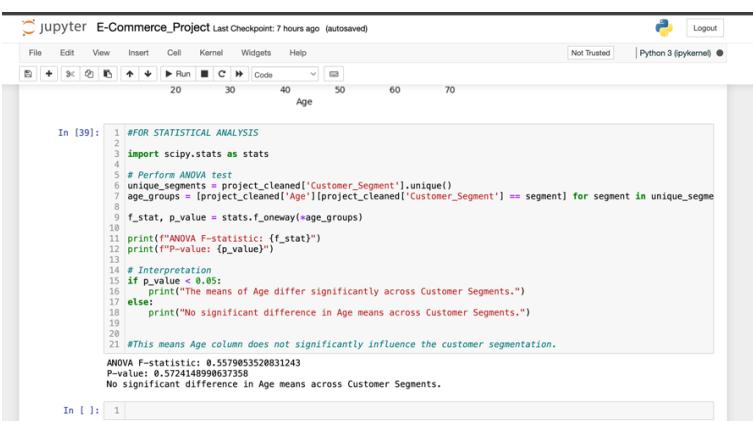


A screenshot of a Jupyter Notebook interface. The title bar says "jupyter E-Commerce_Project Last Checkpoint: 7 hours ago (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help. The toolbar has icons for file operations like Open, Save, Run, and Cell. The status bar shows "Not Trusted" and "Python 3 (pykernel)". A code cell titled "In [38]" contains the following Python code:

```
1 #NUMERICAL FEATURES
2
3 # AGE V/S CUSTOMER_SEGMENT
4
5 sns.scatterplot(data=project_cleaned, x="Age", y="Customer_Segment", alpha=0.6)
6 plt.title('Age vs Customer Segment')
7 plt.xlabel('Age')
8 plt.ylabel('Customer Segment')
9 plt.show()
10
11
12 #This shows age does not impact customer_segment strongly as
13 #all age groups are represented in each segment without clear clustering or trends.
14
15
16
17
```



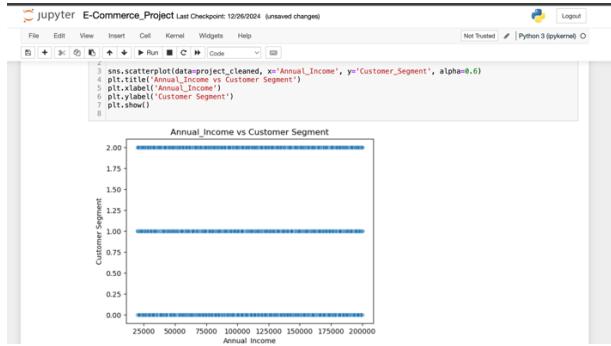
- I have done Statistical Analysis, to check how much age column affect the target variable.



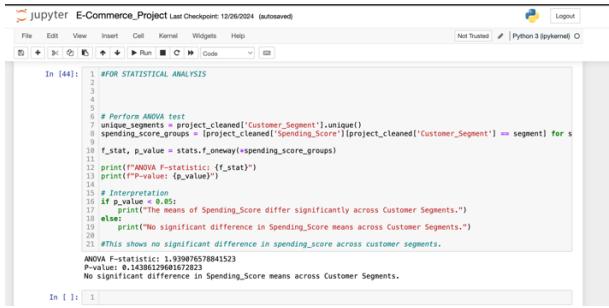
A screenshot of a Jupyter Notebook interface. The title bar says "jupyter E-Commerce_Project Last Checkpoint: 7 hours ago (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help. The toolbar has icons for file operations like Open, Save, Run, and Cell. The status bar shows "Not Trusted" and "Python 3 (pykernel)". A code cell titled "In [39]" contains the following Python code:

```
1 #FOR STATISTICAL ANALYSIS
2
3 import scipy.stats as stats
4
5 # Perform ANOVA test
6 unique_segments = project_cleaned['Customer_Segment'].unique()
7 age_groups = [project_cleaned['Age'][project_cleaned['Customer_Segment'] == segment] for segment in unique_segments]
8 f_stat, p_value = stats.f_oneway(*age_groups)
9
10 print("ANOVA F-statistic: {:.2f}".format(f_stat))
11 print("P-value: {:.2f}".format(p_value))
12
13 # Interpretation
14 if p_value < 0.05:
15     print("The means of Age differ significantly across Customer Segments.")
16 else:
17     print("No significant difference in Age means across Customer Segments.")
18
19
20
21 #This means Age column does not significantly influence the customer segmentation.
22
23 ANOVA F-statistic: 0.5579053520831243
24 P-value: 0.5724148990637358
25 No significant difference in Age means across Customer Segments.
```

- Since the p-value is >0.05 , this shows there is no significant influence of Age column on the target variable.
- ‘ANNUAL_INCOME V/S CUSTOMER SEGMENT’

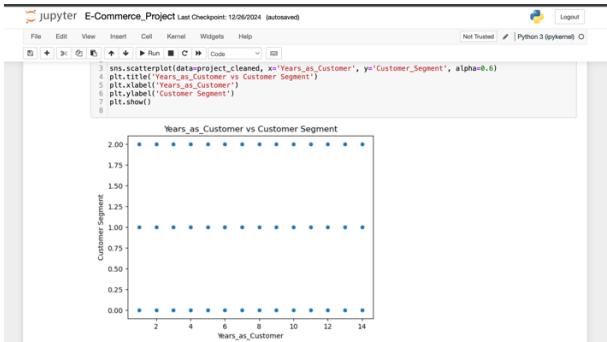


- For Statistical Analysis,

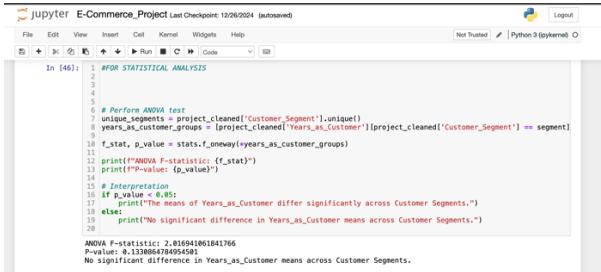


```
In [44]: 1 #FOR STATISTICAL ANALYSIS
2
3
4
5 # Perform ANOVA test
6 unique_segments = project_cleaned['Customer_Segment'].unique()
7 spending_score_groups = [project_cleaned['Spending_Score'][project_cleaned['Customer_Segment'] == segment] for s
8     in unique_segments]
9 f_stat, p_value = stats.f_oneway(spending_score_groups)
10 print("ANOVA F-statistic: ", f_stat)
11 print("P-value: ", p_value)
12 print("Interpretation")
13 if p_value < 0.05:
14     print("The means of Spending_Score differ significantly across Customer Segments.")
15 else:
16     print("No significant difference in Spending_Score means across Customer Segments.")
17
18 #This shows no significant difference in spending_score across customer segments.
19
20 ANOVA F-statistic: 1.93907578451523
21 P-value: 0.149852961673823
22 No significant difference in Spending_Score means across Customer Segments.
```

- Here also the p-value is >0.05 , hence Spending score also does not influence the target variable much.
- 'YEARS_AS_CUSTOMER V/S CUSTOMER SEGMENT'



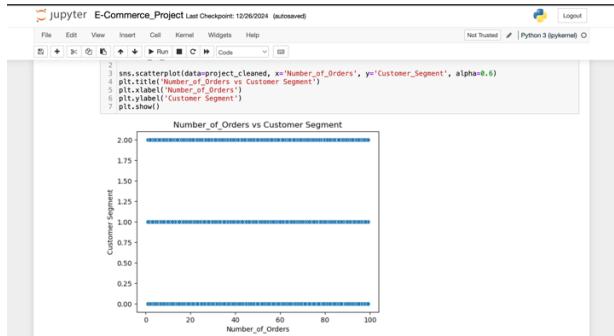
- For Statistical Analysis,



```
In [46]: 1 #FOR STATISTICAL ANALYSIS
2
3
4
5 # Perform ANOVA test
6 unique_segments = project_cleaned['Customer_Segment'].unique()
7 years_as_customer_groups = [project_cleaned['Years_as_Customer'][project_cleaned['Customer_Segment'] == segment] for s
8     in unique_segments]
9 f_stat, p_value = stats.f_oneway(years_as_customer_groups)
10 print("ANOVA F-statistic: ", f_stat)
11 print("P-value: ", p_value)
12 print("Interpretation")
13 if p_value < 0.05:
14     print("The means of Years_as_Customer differ significantly across Customer Segments.")
15 else:
16     print("No significant difference in Years_as_Customer means across Customer Segments.")
17
18 ANOVA F-statistic: 2.016941861841766
19 P-value: 0.339864795454921
20 No significant difference in Years_as_Customer means across Customer Segments.
```

- Here, the p-value is > 0.05 , hence no significant difference in years as customer-on-customer segment.

- **'NUMBER_OF_ORDERS V/S CUSTOMER SEGMENT'**



- For Statistical Analysis,

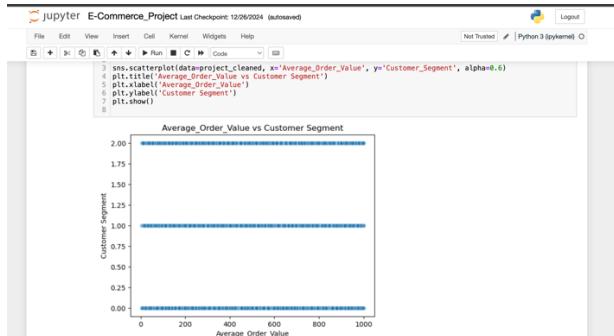
```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [48]: 1 #FOR STATISTICAL ANALYSIS
2
3
4
5
6 # Perform ANOVA test
7 unique_segments = project_cleaned['Customer_Segment'].unique()
8 number_of_orders_groups = [project_cleaned['Number_of_Orders'][project_cleaned['Customer_Segment'] == segment] for segment in unique_segments]
9 f_stat, p_value = stats.f_oneway(*number_of_orders_groups)
10 print("ANOVA F-statistic: ", f_stat)
11 print("p-value: ", p_value)
12 print("The means of Number_of_Orders differ significantly across Customer Segments.")
13 else:
14     print("No significant difference in Number_of_Orders means across Customer Segments.")
15
16 # Interpretation
17 if p_value < 0.05:
18     print("The means of Number_of_Orders differ significantly across Customer Segments.")
19 else:
20     print("No significant difference in Number_of_Orders means across Customer Segments.")

ANOVA F-statistic: 0.326488235848747
p-value: 0.725315232086462
No significant difference in Number_of_Orders means across Customer Segments.

```

- The p-value is >0.05 , hence no influence of number of orders on the target variable.
- **'AVERAGE_ORDER_VALUE V/S CUSTOMER SEGMENT'**



- For Statistical Analysis,

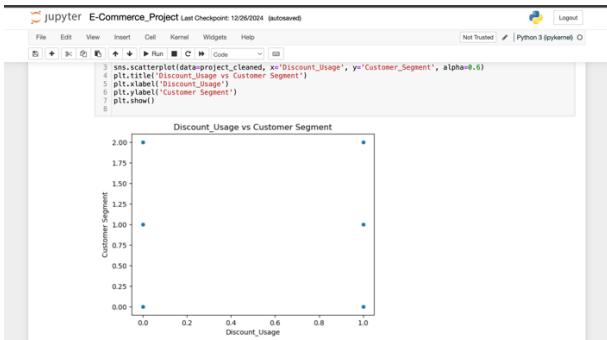
```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [50]: 1 #FOR STATISTICAL ANALYSIS
2
3
4
5
6 # Perform ANOVA test
7 unique_segments = project_cleaned['Customer_Segment'].unique()
8 avg_order_value_groups = [project_cleaned['Average_Order_Value'][project_cleaned['Customer_Segment'] == segment] for segment in unique_segments]
9 f_stat, p_value = stats.f_oneway(*avg_order_value_groups)
10 print("ANOVA F-statistic: ", f_stat)
11 print("p-value: ", p_value)
12 print("The means of Average_Order_Value differ significantly across Customer Segments.")
13 else:
14     print("No significant difference in Average_order_Value means across Customer Segments.")

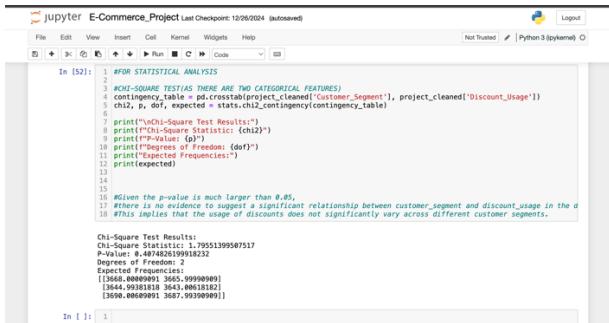
ANOVA F-statistic: 1.897313690168426
p-value: 0.369826546886278
No significant difference in Average_order_Value means across Customer Segments.

```

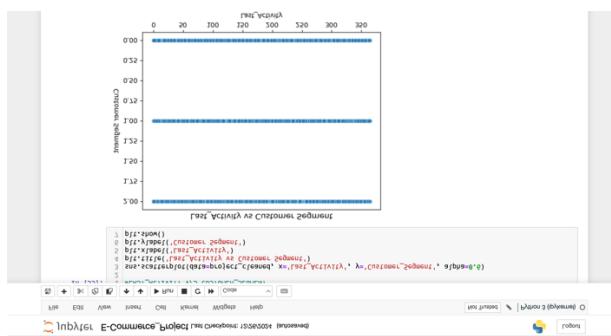
- The p-value is >0.05 , hence no influence of average order value on the customer segment.
 - ‘DISCOUNT_USAGE V/S CUSTOMER_SEGMENT’



- For Statistical analysis, I'll do Chi-Square test rather than Anova test as there are two categorical features.



- The p-value is >0.05 , hence discount usage also does not affect the target variable much.
 - ‘LAST ACTIVITY V/S CUSTOMER SEGMENT’



- For Statistical Analysis,

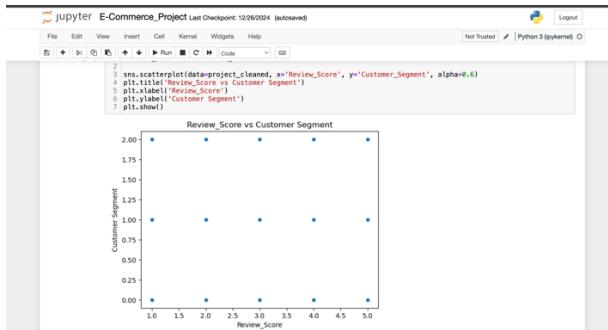
In [54]:

```
# PEARSON CORRELATION COEFFICIENTS
# Correlation matrix
# Correlation matrix for Last_Activity vs Customer_Segment
# Correlation matrix for Last_Activity vs Project_Segment
# Correlation matrix for Customer_Segment vs Project_Segment

# Perform ANOVA test
# Create groups based on project_cleaned['Customer_Segment'].unique()
last_activity_groups = [project_cleaned['Last_Activity'][project_cleaned['Customer_Segment'] == segment] for segment in project_cleaned['Customer_Segment'].unique()]
stats, p_value = stats.f_oneway(*last_activity_groups)
print("ANOVA F-statistic: ", f_stat)
print("P-value: ", p_value)

# Interpretation
if p_value < 0.05:
    print("The means of Last_Activity differ significantly across Customer Segments.")
else:
    print("No significant difference in Last_Activity Means across Customer Segments.")
```

- The p-value is >0.05 , hence there is no influence of Last Activity on the Customer Segment.
 - ‘REVIEW_SCORE V/S CUSTOMER_SEGMENT’



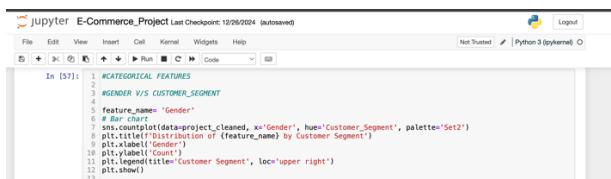
- For Statistical Analysis,

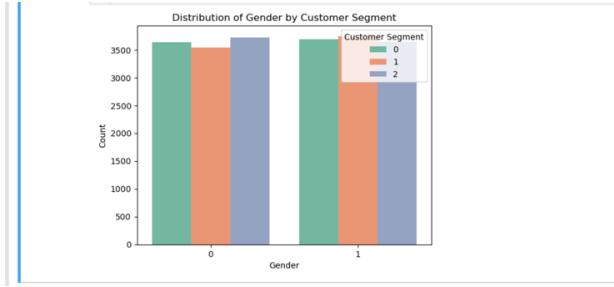
In [56]:

```
#FOR STATISTICAL ANALYSIS
1
2
3
4
5
6 # Perform ANOVA test
7 review_score_groups = [project_cleaned['Customer_Segment'].unique()]
8 review_score_groupby = [project_cleaned['Review_Score'][project_cleaned['Customer_Segment'] == segment] for segment
9     in stats]
10 f_statistic = stats.f_oneway(*review_score_groupby)
11 print("ANOVA F-statistic: ", f_statistic)
12 print("P-value: ", p_value)
13
14 # Interpretation
15 if p_value < 0.05:
16     print("The mean of Review_Score differ significantly across Customer Segments.")
17 else:
18     print("No significant difference in Review_Score means across Customer Segments.")
```

ANOVA F-statistic: 1.945138279135335
P-value: 0.1598660264290865
No significant difference in Review_Score means across Customer Segments.

- The p-value is >0.05 , then the review score also does not make much impact on the target variable.
 - Now, I'll do the bivariate analysis of categorical features with Target Variable to understand more about the data.
 - 'GENDER V/S CUSTOMER SEGMENT'





- For Statistical Analysis, I'll do the Chi-Square Test as we are dealing with two categorical features.

```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipy)
In [59]: 5 # Create a contingency table
6 contingency_table = pd.crosstab(project_cleaned['Gender'], project_cleaned['Customer_Segment'])
7 print(contingency_table)
8
9 # Perform Chi-Square Test
10 chi2, p, dof, expected = chi2_contingency(contingency_table)
11
12 # Print results
13 print("\nChi-Square Test Results:")
14 print(f"Chi-Square Statistic: {chi2:.4f}")
15 print(f"P-Value: {p:.4f}")
16 print(f"Degrees of Freedom: {dof}")
17 if p < 0.05:
18     print("Conclusion: There is a significant association between Gender and Customer Segment.")
19 else:
20     print("Conclusion: There is no significant association between Gender and Customer Segment.")
21
22
23
24
25 #This shows there is no significant association between gender and target variable.
Contingency Table:
Customer_Segment   0    1    2
Gender
0                3639  3542  3727
1                3639  3746  3651
Chi-Square Test Results:
Chi-Square Statistic: 5.3821
P-Value: 0.0648
Degrees of Freedom: 2
Conclusion: There is no significant association between Gender and Customer Segment.

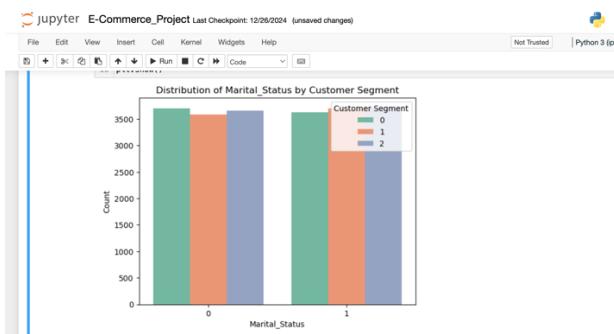
```

- This shows there is no significant association between Gender and Customer Segment as the p-value is >0.05 .
- 'MARITAL_STATUS V/S CUSTOMER_SEGMNET'

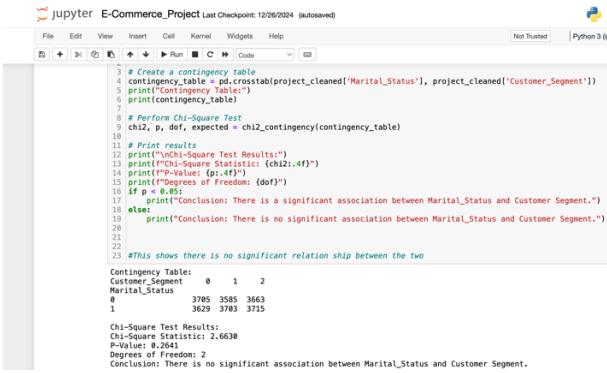
```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipy)
In [59]: 1 #MARITAL_STATUS V/S CUSTOMER_SEGMENT
2 feature_name= 'Marital_Status'
3 # Bar chart
4 sns.countplot(data=project_cleaned, x="Marital_Status", hue="Customer_Segment", palette='Set2')
5 plt.xlabel('Marital_Status')
6 plt.ylabel('Count')
7 plt.title('Customer Segment', loc='upper right')
8 plt.show()

```



- For Statistical Analysis,



```
# Create a contingency table
contingency_table = pd.crosstab(project_cleaned['Marital_Status'], project_cleaned['Customer_Segment'])

# Print Contingency Table
print(contingency_table)

# Perform Chi-Square Test
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Print results
print("\nChi-Square Test Results:")
print(f"Chi-Square Statistic: {chi2:.4f}")
print(f"P-Value: {p:.4f}")
print(f"Degrees of Freedom: {dof}")
if p < 0.05:
    print("Conclusion: There is a significant association between Marital_Status and Customer Segment.")
else:
    print("Conclusion: There is no significant association between Marital_Status and Customer Segment.")

# This shows there is no significant relationship between the two.

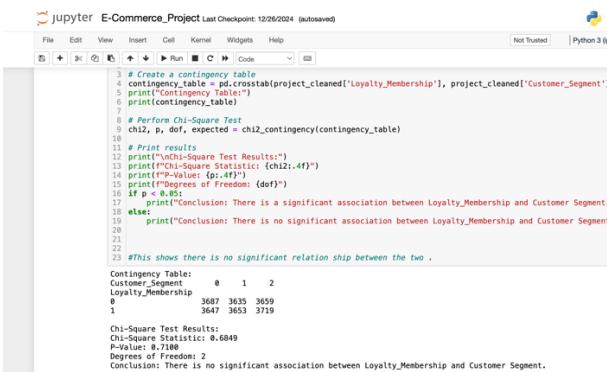
Contingency Table:
Customer_Segment      0      1      2
Marital_Status
W                     3765  3565  3663
M                     3629  3593  3715

Chi-Square Test Results:
Chi-Square Statistic: 2.6638
P-Value: 0.2641
Degrees of Freedom: 2
Conclusion: There is no significant association between Marital_Status and Customer Segment.
```

- The p-value is >0.05 , hence there is no association between Marital Status and customer segment.
- 'LOYALTY_MEMBERSHIP V/S CUSTOMER SEGMENT'**



- For Statistical Analysis,



```
# Create a contingency table
contingency_table = pd.crosstab(project_cleaned['Loyalty_Membership'], project_cleaned['Customer_Segment'])

# Print Contingency Table
print(contingency_table)

# Perform Chi-Square Test
chi2, p, dof, expected = chi2_contingency(contingency_table)

# Print results
print("\nChi-Square Test Results:")
print(f"Chi-Square Statistic: {chi2:.4f}")
print(f"P-Value: {p:.4f}")
print(f"Degrees of Freedom: {dof}")
if p < 0.05:
    print("Conclusion: There is a significant association between Loyalty_Membership and Customer Segment.")
else:
    print("Conclusion: There is no significant association between Loyalty_Membership and Customer Segment.")

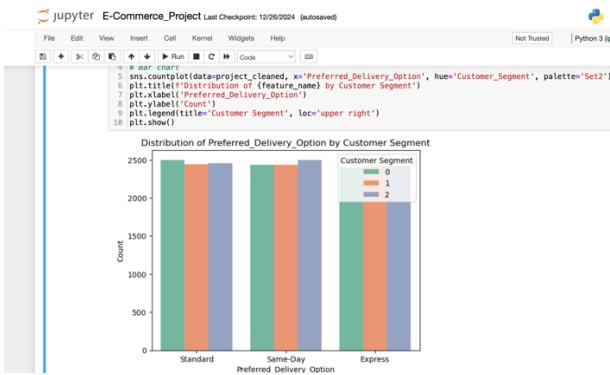
# This shows there is no significant relationship between the two.

Contingency Table:
Customer_Segment      0      1      2
Loyalty_Membership
0                     3687  3635  3659
1                     3647  3653  3719

Chi-Square Test Results:
Chi-Square Statistic: 0.6849
P-Value: 0.7108
Degrees of Freedom: 2
Conclusion: There is no significant association between Loyalty_Membership and Customer Segment.
```

- Since the p-value is >0.05 there is no link between the loyalty membership and the target variable.

- ‘PREFERRED_DELIVERY_OPTION V/S CUSTOMER SEGMENT’

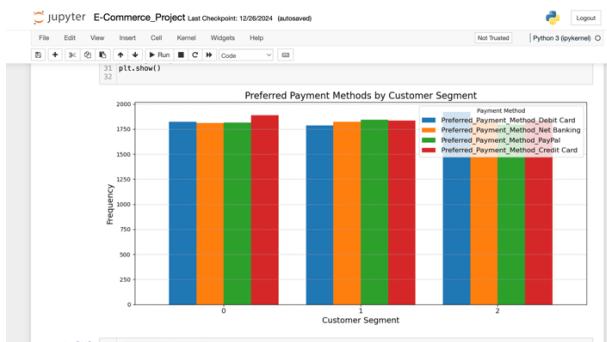


- For Statistical Analysis,

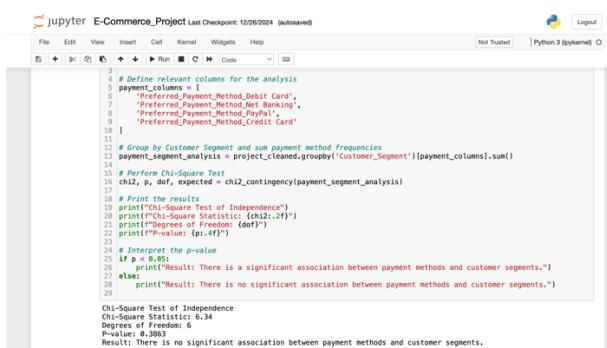
```
# Create a contingency table
1 contingency_table = pd.crosstab(project_cleaned['Preferred_Delivery_Option'], project_cleaned['Customer_Segment'])
2 print(contingency_table)
3 print(contingency_table.sum())
4 print(contingency_table)
5
# Perform Chi-Square Test
6 chis2, p, dof, expected = chisquare(contingency_table)
7
# Print results
8 print("Chi-Square Test Results:")
9 print("Chi-Square Statistic: (%.2f)" % chis2)
10 print("P-Value: (%.4f)" % p)
11 print("Degrees of Freedom: (%d)" % dof)
12 if p < 0.05:
13     print("Conclusion: There is a significant association between Preferred_Delivery_Option and Customer Segment")
14 else:
15     print("Conclusion: There is no significant association between Preferred_Delivery_Option and Customer Segment")
16
17 # This shows there is no significant relation ship between the two .
18
Contingency Table:
Customer_Segment      0      1      2
Preferred_Delivery_Option
Express                2401   2414   2423
Same-Day               2433   2434   2581
Standard               2580   2446   2454
Chi-Square Test Results:
Chi-Square Statistic: 1.5826
P-Value: 0.8119
Degrees of Freedom: 4
Conclusion: There is no significant association between Preferred_Delivery_Option and Customer Segment.
```

- Here also the p-value is >0.05, hence there is no association between preferred delivery option and the target variable.
- ‘PAYMENT_METHODS V/S CUSTOMER SEGMENT’

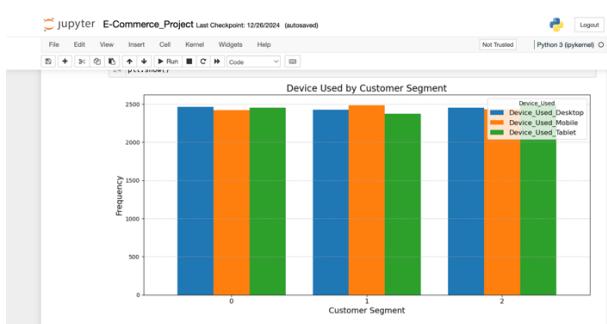
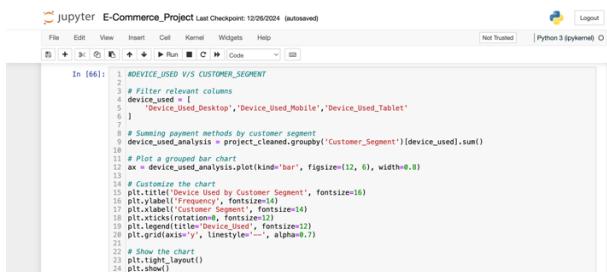
```
# PAYMENT_METHODS V/S CUSTOMER_SEGMENT
1 payment_columns = [
2     'Preferred_Payment_Method_Credit Card',
3     'Preferred_Payment_Method_Debit Card',
4     'Preferred_Payment_Method_Net Banking',
5     'Preferred_Payment_Method_PayPal',
6     'Preferred_Payment_Method_Credit Card'
7 ]
8
9 # Summing payment methods by customer segment
10 payment_segment_analysis = project_cleaned.groupby('Customer_Segment')[payment_columns].sum()
11
12 # Plot a grouped bar chart
13 ax = payment_segment_analysis.plot(kind='bar', figsize=(12, 6), width=0.8)
14
15 # Customize the chart
16 plt.title("Payment Methods by Customer Segment", fontsize=16)
17 plt.xlabel("Customer Segment", fontsize=14)
18 plt.ylabel("Frequency", fontsize=14)
19 plt.grid(True)
20 plt.ylim(0, 100)
21 plt.legend(title="Payment Method", fontsize=12)
22 plt.grid(axis="y", linestyle="--", alpha=0.7)
23
24 # Show the chart
25 plt.tight_layout()
26 plt.show()
```



- For Statistical Analysis,



- This shows there is no association between preferred payment method and customer segment.
- ‘DEVICE_USED V/S CUSTOMER_SEGMENT’



- For Statistical Analysis,

```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted | Python 3 (ipykernel) O
In [1]: 2
3 # Define relevant columns for the analysis
4 device_used = [
5     'Device_Used/Desktop', 'Device_Used/Mobile', 'Device_Used/Tablet'
6 ]
7
8 # Group by Customer Segment and sum payment method frequencies
9 device_used_analysis = project_cleaned.groupby('Customer_Segment')[device_used].sum()
10
11 # Perform Chi-Square Test
12 chis2, p, dof, expected = chi2_contingency(device_used_analysis)
13
14 # Print the results
15 print("Chi-Square Test of Independence")
16 print(f"Chi-Square Statistic: {chis2:.2f}")
17 print(f"Degrees of Freedom: {dof}")
18 print(f"P-value: {p:.4f}")
19
20 # Interpret the p-value
21 if p < 0.05:
22     print("Result: There is a significant association between Device Used and customer segments.")
23 else:
24     print("Result: There is no significant association between Device Used and customer segments.")
25
26
Chi-Square Test of Independence
Chi-Square Statistic: 3.91
Degrees of Freedom: 4
P-value: 0.416
Result: There is no significant association between Device Used and customer segments.

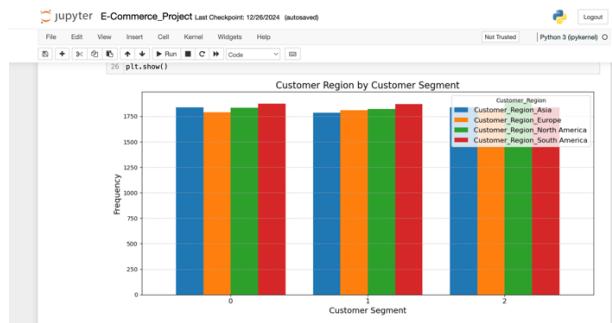
```

- The p-value is >0.05 , hence there is no association between device used and customer segment.
- ‘CUSTOMER_REGION V/S CUSTOMER_SEGMENT’

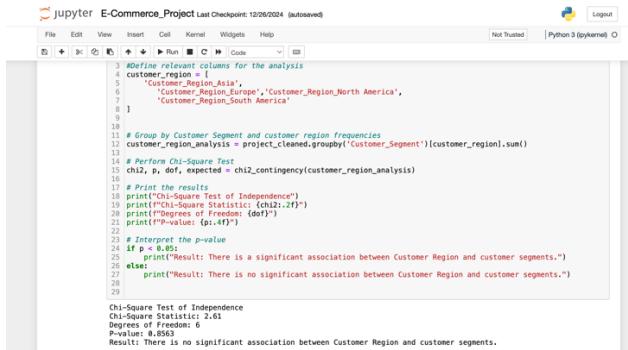
```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted | Python 3 (ipykernel) O
In [1]: 1
2 # CUSTOMER_REGION V/S CUSTOMER_SEGMENT
3
4 # Filter relevant columns
5 customer_region = [
6     'Customer_Region_Asia',
7     'Customer_Region_Europe',
8     'Customer_Region_North America',
9     'Customer_Region_South America'
10
11 # Summing payment methods by customer segment
12 customer_region_analysis = project_cleaned.groupby('Customer_Segment')[customer_region].sum()
13
14 # Plot a grouped bar chart
15 ax = customer_region_analysis.plot(kind='bar', figsize=(12, 6), width=0.8)
16
17 # Customize the chart
18 plt.title('Customer Region by Customer Segment', fontsize=16)
19 plt.xlabel('Customer Segment', fontsize=14)
20 plt.ylabel('Frequency', fontsize=14)
21 plt.xticks(rotation=0, fontsize=12)
22 plt.yticks([0, 250, 500, 750, 1000, 1250, 1500, 1750], fontsize=12)
23 plt.grid(axis='y', linestyle='--', alpha=0.7)
24
25 # Show the chart
26 plt.tight_layout()
27 plt.show()

```



- For Statistical Analysis,

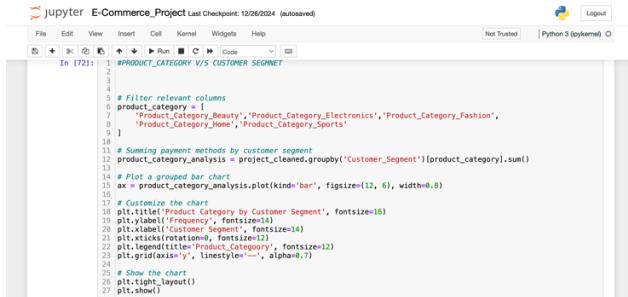


```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3 (ipykernel) Logout
In [1]: 3 #Define relevant columns for the analysis
4 customer_region = [
5     'Customer_Region_Asia',
6     'Customer_Region_Europe','Customer_Region_North_America',
7     'Customer_Region_South_America'
8 ]
9
10
11 # Group by Customer Segment and customer region frequencies
12 customer_region_analysis = project_cleaned.groupby('Customer_Segment')[customer_region].sum()
13
14 # Perform Chi-Square Test
15 chi2, p, dof, expected = chi2_contingency(customer_region_analysis)
16
17 print("Chi-Square Test results")
18 print("Chi-Square Test of Independence")
19 print(f"Chi-Square Statistic: {chi2:.2f}")
20 print(f"Degrees of Freedom: {dof}")
21 print(f"P-value: {p:.4f}")
22
23 # Interpret the p-value
24 if p < 0.05:
25     print("Result: There is a significant association between Customer Region and customer segments.")
26 else:
27     print("Result: There is no significant association between Customer Region and customer segments.")
28
29
30
31 Chi-Square Test of Independence
32 Chi-Square Statistic: 2.61
33 Degrees of Freedom: 6
34 P-value: 0.8562
35 Result: There is no significant association between Customer Region and customer segments.

```

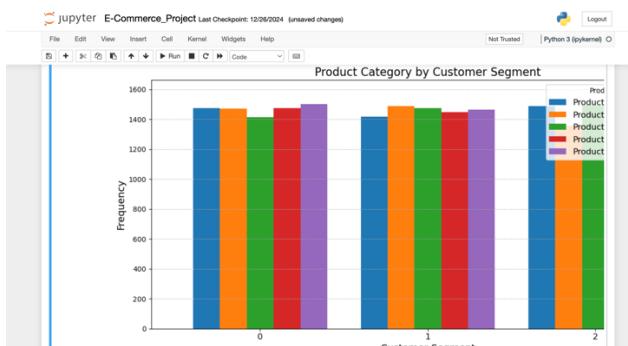
- The p-value is >0.05, hence there is no link between customer region and the target variable.
- ‘PRODUCT_CATEGORY V/S CUSTOMER_SEGMNET’



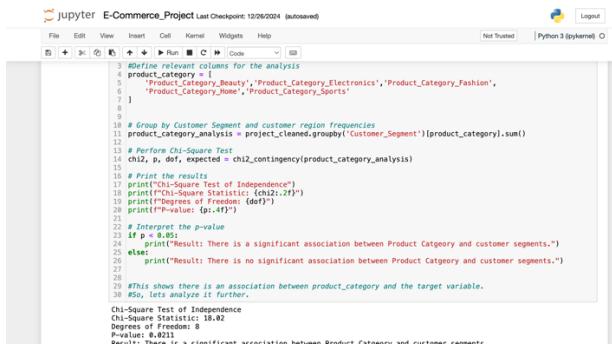
```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted Python 3 (ipykernel) Logout
In [72]: #PRODUCT_CATEGORY V/S CUSTOMER SEGMENT
1
2
3
4
5 # Filter relevant columns
6 product_category = [
7     'Product_Category_Beauty','Product_Category_Electronics','Product_Category_Fashion',
8     'Product_Category_Home','Product_Category_Sports'
9 ]
10
11 # Summing payment methods by customer segment
12 payment_method_analysis = project_cleaned.groupby('Customer_Segment')[product_category].sum()
13
14 # Plot grouped bar chart
15 ax = product_category_analysis.plot(kind='bar', figsize=(12, 6), width=0.8)
16
17 # Customize the chart
18 plt.title('Product Category by Customer Segment', fontsize=16)
19 plt.ylabel('Frequency', fontsize=14)
20 plt.xlabel('Customer Segment', fontsize=14)
21 plt.xticks(rotation=0, fontsize=12)
22 plt.yticks(fontsize=12)
23 plt.grid(axis='y', linestyle='--', alpha=0.7)
24
25 # Show the chart
26 plt.tight_layout()
27 plt.show()

```



- For Statistical Analysis,

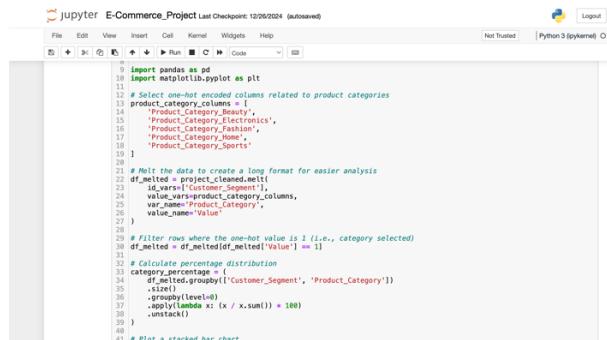


```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O
3 # Define relevant columns for the analysis
4 product_category_analysis = project_cleaned.groupby(['Customer_Segment']).sum()
5
6 # Group by Customer Segment and customer_region frequencies
7 product_category_analysis = project_cleaned.groupby(['Customer_Segment'])['Product_Category'].sum()
8
9
10 # Perform Chi-Square Test
11 chis2, p, dof, expected = chi2_contingency(product_category_analysis)
12
13 # Print the results
14 print("Chi-Square Test of Independence")
15 print(f"Chi-Square Statistic: {chis2:.2f}")
16 print(f"Degrees of Freedom: {dof}")
17 print(f"P-value: {p:.4f}")
18
19 # Interpret the p-value
20 if p < 0.05:
21     print("There is a significant association between Product Category and customer segments.")
22 else:
23     print("There is no significant association between Product Category and customer segments.")
24
25 # This shows there is an association between product_category and the target variable.
26 # So, let's analyze it further.
27
28 Chi-Square Test of Independence
29 Chi-Square Statistic: 18.02
30 Degrees of Freedom: 8
31 P-value: 0.0212
32
33 # There is a significant association between Product Category and customer segments.

```

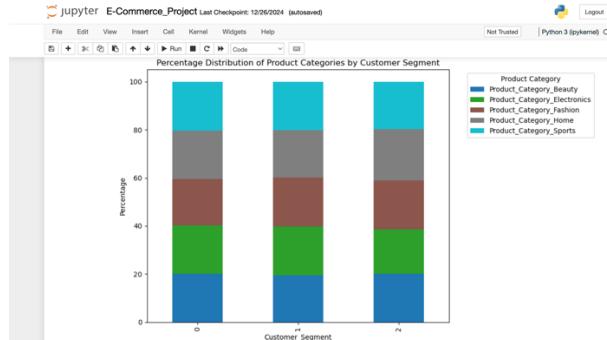
- The p-value is <0.05 , hence there is an association between the Product Category and the Target variable.
- Since there is an association between the product category and the customer segment, let's calculate the percentage breakdown using a stacked bar chart.



```

jupyter E-Commerce_Project Last Checkpoint: 12/26/2024 (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # Select one-hot-encoded columns related to product categories
5 product_category_columns = [
6     'Product_Category_Beauty',
7     'Product_Category_Electronics',
8     'Product_Category_Fashion',
9     'Product_Category_Home',
10    'Product_Category_Sports'
11 ]
12
13 # Melt the data to create a long format for easier analysis
14 df_melted = project_cleaned.melt(
15     id_vars='Customer_Segment',
16     value_vars=product_category_columns,
17     var_name='Product_Category',
18     value_name='Value'
19 )
20
21 # Filter rows where the one-hot value is 1 (i.e., category selected)
22 df_melted = df_melted[df_melted['Value'] == 1]
23
24 # Calculate percentage distribution
25 category_percentage = (
26     df_melted.groupby(['Customer_Segment', 'Product_Category'])
27     .size()
28     .groupby(level=0)[('Customer_Segment', 'Product_Category')]
29     .apply(lambda x: (x / x.sum()) * 100)
30     .unstack()
31 )
32
33 # Plot a stacked bar chart
34 category_percentage.plot(kind='bar', stacked=True, figsize=(10, 6), cmap='tab10')
35 plt.title('Percentage Distribution of Product Categories by Customer Segment')
36 plt.xlabel('Customer Segment')
37 plt.ylabel('Percentage')
38 plt.legend(['Product Category', 'bbox_to_anchor=(1.05, 1), loc='upper left'])
39 plt.tight_layout()
40 plt.show()
41
42 # Plot a stacked bar chart
43 category_percentage.plot(kind='bar', stacked=True, figsize=(10, 6), cmap='tab10')
44 plt.title('Percentage Distribution of Product Categories by Customer Segment')
45 plt.xlabel('Customer Segment')
46 plt.ylabel('Percentage')
47 plt.legend(['Product Category', 'bbox_to_anchor=(1.05, 1), loc='upper left'])
48 plt.tight_layout()
49 plt.show()

```



- After doing the Univariate and Bivariate Analysis I couldn't find any major affect of the features on the target variable.
- Now, I'll do PCA(Principal Component Analysis) and try to build a model first using Logistic Regression.

PCA (PRINCIPAL COMPONENT ANALYSIS)

- Below is the steps I did to do PCA :
- First, I loaded the necessary libraries and the required dataset.

```
In [1]: 1 #IMPORTING NECESSARY LIBRARIES
2
3 import pandas as pd
4 from sklearn.decomposition import PCA
5 from sklearn.preprocessing import StandardScaler
6 import numpy as np
7 import warnings
8 warnings.filterwarnings('ignore')

In [2]: 1 #LOADING THE DATASET
2
3 project=pd.read_csv("/Users/sumayyahashed/Desktop/MS in Data Science/CAPSTONE PROJECT/E Commerce/E-commerce Cus
4 project

Out[2]: Customer_ID Customer_Name Age Gender Annual_Income Spending_Score Marital_Status Product_Category Years_as_Customer Number_of_Order
0 69 Michael Charles 49 0 180704 60 1 Fashion 8 7
1 62 Keisha Medina 62 0 165078 36 1 Sports 7 2
2 52 Ronald Hoffman 62 0 66523 46 0 Home 4 6
3 56 Sandra McGuire 56 1 193559 83 0 Sports 5 8
4 5 Andrew McDonald 46 0 57461 60 0 Sports 5 8
...
21995 21995 Tonya Kramer 22 1 60778 25 1 Home 2 4
```

- Then, I did Encoding to convert categorical columns into numerical and also removed irrelevant columns.

```
In [3]: 1 #CONVERTING THE CATEGORICAL COLUMNS INTO NUMERICAL COLUMNS USING ENCODING
2
3 one_hot_columns=['Product_Category','Preferred_Payment_Method','Device_Used','Customer_Region']
4 project=pd.get_dummies(project,columns=one_hot_columns)

In [5]: 1 #REMOVING IRRELEVANT FEATURES(Customer_ID and Customer_Name)
2
3 project_cleaned=project.drop(columns=['Customer_ID','Customer_Name'])
4
5 #TO MAKE SURE BOTH THE COLUMNS ARE DROPPED
6
7 project_cleaned.head()

Out[5]: Age Gender Annual_Income Spending_Score Marital_Status Years_as_Customer Number_of.Orders Average_Order_Value Loyalty_Membership Discount
0 69 0 180704 60 1 8 78 182.40 1
1 62 0 165078 36 1 7 25 342.85 1
2 52 0 66523 46 0 4 68 275.57 1
3 56 1 193559 83 0 5 88 97.62 1
4 56 0 57461 60 0 5 86 438.02 0
```

- Also, I converted Preferred_delivery_option into numerical feature using label encoding.

```
In [6]: 1 #ALSO, I'LL CONVERT PREFERRED_DELIVERY_OPTION INTO NUMERICAL FEATURE USING LABEL ENCODING
2
3 # define a custom order for categories
4 preferred_order = ('Standard': 0, 'Same-Day': 1, 'Express': 2) # Example categories
5
6 # Map the categories to the custom order and overwrite the original column
7 project_cleaned['Preferred_Delivery_Option'] = project_cleaned['Preferred_Delivery_Option'].map(preferred_order)
8
9 # Now check the data type
10 print(project_cleaned.dtypes) # This will show that 'Preferred_Delivery_Option' is now int64, not object
11

Age int64
Gender int64
Annual_Income int64
Spending_Score int64
Marital_Status int64
Years_as_Customer int64
Number_of.Orders int64
Average_Order_Value float64
Loyalty_Membership int64
Discount int64
Preferred_Delivery_Option int64
Last_Activity int64
Device_Used int64
Customer_Segment int64
Product_Category_Beauty uint8
Product_Category_Electronics uint8
Product_Category_Fashion uint8
Product_Category_Home uint8
Product_Category_Sports uint8
Preferred_Payment_Method_Credit_Card uint8
Preferred_Payment_Method_Debit_Card uint8
... uint8
```

- Now, I'll separate the dataset into features and target variable, standardize it and perform PCA. I will also plot the explained variance ratio.

jupyter MODEL_WITH_PCA Last Checkpoint: 9 minutes ago (unsaved changes) Trusted Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Help

In [7]:

```
1 #SEPARATING FEATURES AND TARGET VARIABLE
2 X = project_cleaned.drop(columns=['Customer_Segment'])
3 y = project_cleaned['Customer_Segment']
```

In [8]:

```
1 #STANDARDIZING THE DATA
2 from sklearn.preprocessing import StandardScaler
3 scaler = StandardScaler()
4 X_scaled = scaler.fit_transform(X)
```

In [9]:

```
1 #PERFORMING PCA
2 from sklearn.decomposition import PCA
3 pca = PCA()
4 X_pca = pca.fit_transform(X_scaled)
```

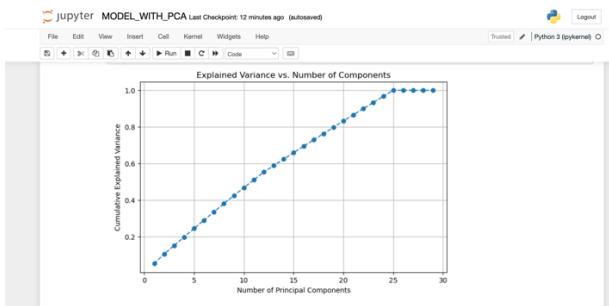
In [10]:

```
1 #EXPLAINED VARIANCE RATIO
2 explained_variance = pca.explained_variance_ratio_
3 cumulative_variance = np.cumsum(explained_variance)
4
```

In [11]:

```
1 #PLOTTING EXPLAINED VARIANCE
2 plt.figure(figsize=(8, 5))
3 plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance, marker='o', linestyle='--')
4 plt.xlabel('Number of Principal Components')
5 plt.ylabel('Cumulative Explained Variance')
6 plt.title('Explained Variance vs. Number of Components')
7 plt.show()
```

Explained Variance vs. Number of Components



- Now, I'll select the number of components, reduce dimensionality and inspect feature contribution.

- I'll build a Model now using Logistic Regression. I'll split the data into training and testing sets, initialize and train the logistic regression model and make predictions on the test set.

```

In [16]: 1 #SPLITTING THE DATASET INTO TRAINING AND TESTING SETS
2 X_train, X_test, y_train, y_test = train_test_split(X_reduced, y, test_size=0.2, random_state=42, stratify=y)
3
In [17]: 1 #INITIALIZING AND TRAINING THE LOGISTIC REGRESSION MODEL
2 logistic_model = LogisticRegression(random_state=42, max_iter=1000)
3 logistic_model.fit(X_train, y_train)
4
Out[17]: LogisticRegression(max_iter=1000, random_state=42)

In [18]: 1 #MAKING PREDICTIONS ON THE TEST SET
2 y_pred = logistic_model.predict(X_test)
3

```

- Now, I'll evaluate the model and also visualize the confusion matrix.

```

In [19]: 1 #EVALUATING THE MODEL
2 accuracy = accuracy_score(y_test, y_pred)
3 print(f"Accuracy: {accuracy:.2f}")
4
Accuracy: 0.33

In [20]: 1 #Classification report
2 print("Vc classification report:")
3 print(classification_report(y_test, y_pred))

Classification Report:
precision    recall    f1-score   support
          0       0.33      0.27      0.30      1467
          1       0.33      0.35      0.34      1458
          2       0.33      0.37      0.35      1475

accuracy                           0.33      4400
macro avg       0.33      0.33      0.33      4400
weighted avg    0.33      0.33      0.33      4400

In [21]: 1 #Confusion matrix
2 conf_matrix = confusion_matrix(y_test, y_pred)
3

```

Confusion Matrix

		Predicted Labels		
		0	1	2
True Labels	0	392	509	566
	1	397	516	545
2	0	400	525	550
	1			

- Lastly, I'll check the Feature importance and interpret my results.

```

In [23]: 1 #CHECKING FEATURE IMPORTANCE
2 print("Feature Coefficients:")
3 for i, coef in enumerate(logistic_model.coef_[0], 1):
4     print(f"PC{i}: {coef:.4f}")

Feature Coefficients:
PC1: 0.0042
PC2: -0.0037
PC3: 0.0038
PC4: -0.0018
PC5: 0.0009
PC6: 0.0091
PC7: 0.0047
PC8: -0.0004
PC9: -0.0031
PC10: 0.0021
PC11: 0.0021
PC12: -0.0117
PC13: 0.0020
PC14: 0.0049
PC15: -0.0103
PC16: 0.0020
PC17: -0.0052
PC18: 0.0022
PC19: -0.0008
PC20: -0.0027
PC21: 0.0009
PC22: 0.0116
PC23: -0.0045
PC24: 0.0137

```

- So, this tells us that the model building with PCA has really low accuracy (33%) which means the dataset after applying PCA is not appropriate for the model.
- The reasons for low accuracy could be Class Imbalance, Over-dimensionality (too many features as even after PCA we might have unnecessary features), Weak features (as PCA

does reduce the dimensionality but does not guarantee features are strongly related with target variable are retained.

- Low accuracy could also be due to Logistic regression limitations as logistic regression assumes linear relationships and if the data is not linearly separable the model might underperform.
 - So, I'll try a more powerful classifiers like Random Forest and Gradient Boosting just to check if the accuracy improves or not.
 - Here, is the code using Random Forest Classifier.

Jupyter MODEL_WITH_PCA Last Checkpoint: 37 minutes ago (unsaved changes) Trusted Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help

In [24]: # MODEL BUILDING USING RANDOM FOREST

```
1 #IMPORTING RANDOM FOREST CLASSIFIER
2
3 from sklearn.ensemble import RandomForestClassifier
```

In [25]: #SPLITTING THE DATASET INTO TRAINING AND TESTING SETS

```
1 X_train, X_test, y_train, y_test = train_test_split(X_reduced, y, test_size=0.2, random_state=42, stratify=y)
```

In [26]: #INITIALIZING AND TRAINING RANDOM FOREST MODEL

```
1 rf_model = RandomForestClassifier(random_state=42, n_estimators=100, max_depth=None)
2 rf_model.fit(X_train, y_train)
```

Out[26]: RandomForestClassifier(random_state=42)

In [27]: #MAKING PREDICTIONS ON THE TEST SET

```
1
2 y_pred = rf_model.predict(X_test)
```

Jupyter MODEL_WITH_PCA Last Checkpoint: 38 minutes ago (unsaved changes) Trusted Python 3 (ipykernel) Logout

File Edit View Insert Cell Kernel Widgets Help

In [28]: #VALIDATING THE MODEL

```
1
2 # Accuracy
3 accuracy = accuracy_score(y_test, y_pred)
4 print(f"Accuracy: {accuracy:.2f}%")
5
6 # Classification report
7 print("\nClassification Report")
8 print(classification_report(y_test, y_pred))
9
10 # Confusion matrix
11 conf_matrix = confusion_matrix(y_test, y_pred)
12
13 print(conf_matrix)
```

Accuracy: 34

Classification Report:

	precision	recall	f1-score	support
0	0.33	0.33	0.33	1467
1	0.34	0.32	0.32	1458
2	0.35	0.34	0.35	1457
3	0.34	0.34	0.34	1449

accuracy macro avg 0.34 0.34 0.34 4400
weighted avg 0.34 0.34 0.34 4400

1 —Again the accuracy is 34% which is low, so now I'll go back to my previous dataset and do some feature selection rather than PCA.

- The accuracy using Random Forest Classifier is also low (34%). So, now I'll skip checking with gradient booster and rather do some feature selection techniques on my previous data.
 - I'll do some Feature Engineering before Selecting features using different Feature selecting methods.

FEATURE ENGINEERING

- I'll add 'Income_Spending_Interaction', 'Loyalty_Customer_Years' and 'Order_value_frequency'.

```
In [77]: #FEATURE ENGINEERING
#LET ME ADD SOME FEATURES AND THEN DO THE EDA AGAIN TO GAIN SOME MORE UNDERSTANDING
#Healthier customers might spend differently compared to others.
project_cleaned['Income_Spending_Interaction'] = project_cleaned['Annual_Income'] * project_cleaned['Spending_Score']
#Long-term customers with loyalty membership might exhibit unique behaviors.
project_cleaned['Loyalty_Customer_Years'] = project_cleaned['Years_at_Customer'] * project_cleaned['Loyalty_Member']
#High spenders who frequently order may indicate a premium customer segment.
project_cleaned['Order_Value_Frequency'] = project_cleaned['Average_Order_Value'] * project_cleaned['Number_of_Orders']

Out[77]: Index(['Age', 'Gender', 'Annual_Income', 'Spending_Score', 'Marital_Status',
       'Education', 'Number_of.Orders', 'Average_Order_Value',
       'Loyalty_Membership', 'Discount_Use', 'Preferred_Delivery_Option',
       'Last_Activity', 'Review_Score', 'Customer_Segment',
       'Product_Cat_Electronics', 'Product_Cat_Fashion', 'Product_Cat_Home',
       'Product_Cat_Vehicle', 'Preferred_Payment_Method_Credit_Card',
       'Preferred_Payment_Method_Debit_Card',
       'Preferred_Payment_Method_Net_Banking',
       'Preferred_Payment_Method_PayPal',
       'Device_Used_Desktop', 'Device_Used_Tablet', 'Customer_Region_Asia',
       'Customer_Region_Europe', 'Customer_Region_North_America',
       'Customer_Region_South_America', 'Income_Spending_Interaction',
       'Loyalty_Customer_Years', 'Order_Value_Frequency'],
      dtype='object')
```

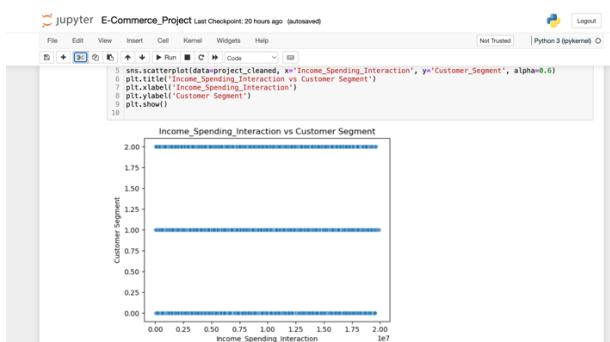
- I'll also add 'Orders_per_year' and 'Spending_Efficiency'.

```
In [79]: #Calculating orders per year to standardize customer behavior
project_cleaned['Orders_per_Year'] = project_cleaned['Number_of_Orders'] / project_cleaned['Years_as_Customer']

Out[79]: #Ratio of Spending_Score to Average_Order_Value
project_cleaned['Spending_Efficiency'] = project_cleaned['Spending_Score'] / project_cleaned['Average_Order_Value']

In [80]: project_cleaned.head()
Out[80]:   user_region North_America Customer_Region_South_America Income_Spending_Interaction Loyalty_Customer_Years Order_Value_Frequency Orders_per_Year Spending_Efficiency
0               0                 0                         8           14227.20          9.000000        0.000000        0.000000        0.000000
0               0                 0                         7           8671.35          5.500000        0.000000        0.000000        0.000000
0               0                 0                         6           5178.76          3.500000        0.000000        0.000000        0.000000
1               0                 0                         4           1738.76          17.000000        0.000000        0.000000        0.000000
1               0                 0                         5           6500.56          17.000000        0.000000        0.000000        0.000000
```

- Will again do some Eda with the new feature and the Target Variable to understand if there is any influence on the target variable.
- Income_Spending_Interaction v/s Customer_Segment



- The Statistical Analysis shows there is no association between income spending interaction and target variable.

```

jupyter E-Commerce_Project Last Checkpoint: 20 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [82]: 1 #STATISTICAL ANALYSIS
2
3 # Perform ANOVA Test
4 unique_segments = project_cleaned['Customer_Segment'].unique()
5 income_spending_interaction_groups = [project_cleaned['Income_Spending_Interaction']] + [project_cleaned['Customer_Segment']]
6 f_stat, p_value = stats.f_oneway(*income_spending_interaction_groups)
7 print("ANOVA F-statistic: {f_stat}")
8 print("P-value: {p_value}")
9
10 if p_value <= 0.05:
11     print("The means of Income Spending Interaction differ significantly across Customer Segments.")
12 else:
13     print("No significant difference in Income Spending Interaction means across Customer Segments.")
14
15 #THIS SHOWS THERE IS NO EFFECT OF SPENDING SCORE INTERACTION ON CUSTOMER SEGMENTS.

```

ANOVA F-statistic: 2.399312906640444
P-value: 0.09138514689562598
No significant difference in Income Spending Interaction means across Customer Segments.

- I will check the correlation after feature Engineering.

```

jupyter E-Commerce_Project Last Checkpoint: 20 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [84]: 1 #CHECKING CORRELATION AFTER DOING FEATURE ENGINEERING
2
3 correlation_with_target = project_cleaned.corr()['Customer_Segment'].sort_values(ascending=False)
4 print(correlation_with_target)

```

	Customer_Segment	Product_Category_Home	Preferred_Payment_Method_Debit_Card	Product_Category_Fashion	Marital_Status	Customer_Region_North_America	Loyalty_Member_Years	Loyalty_Membership	Device_Used_Tablet	Last_Activity	Preferred_Payment_Method_Net_Banking	Customer_Region_Europe	Product_Category_Beauty	Annual_Income	Age	Device_Used_Mobile	Years_at_Customer	Customer_Region_Asia	Review_Score	Income_Spending_Interaction	Spending_Score	Device_Used_Laptop	Average_Order_Value	Device_Used/Desktop	Device_Used_Mobile	Preferred_Payment_Method_PayPal	Customer_Region_South_America	Product_Category_Technology	Gender	Preferred_Payment_Method_Credit_Card	Spending_Efficiency	Product_Category_Electronics	Name: Customer_Segment, dtype: float16			
Customer_Segment	1.000000																																			
Product_Category_Home	0.015040	1.000000																																		
Preferred_Payment_Method_Debit_Card	0.011540	0.011540	1.000000																																	
Product_Category_Fashion	0.000192	0.000192	0.000192	1.000000																																
Marital_Status	0.007184	0.007184	0.007184	0.007184	1.000000																															
Customer_Region_North_America	0.006256	0.006256	0.006256	0.006256	0.006256	1.000000																														
Loyalty_Member_Years	0.000388	0.000388	0.000388	0.000388	0.000388	0.000388	1.000000																													
Loyalty_Membership	0.005554	0.005554	0.005554	0.005554	0.005554	0.005554	0.005554	1.000000																												
Device_Used_Tablet	0.000230	0.000230	0.000230	0.000230	0.000230	0.000230	0.000230	0.000230	1.000000																											
Last_Activity	0.001854	0.001854	0.001854	0.001854	0.001854	0.001854	0.001854	0.001854	0.001854	1.000000																										
Preferred_Payment_Method_Net_Banking	0.000673	0.000673	0.000673	0.000673	0.000673	0.000673	0.000673	0.000673	0.000673	0.000673	1.000000																									
Customer_Region_Europe	0.000671	0.000671	0.000671	0.000671	0.000671	0.000671	0.000671	0.000671	0.000671	0.000671	0.000671	1.000000																								
Product_Category_Beauty	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	0.000651	1.000000																							
Annual_Income	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	0.000826	1.000000																						
Age	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	-0.000174	1.000000																					
Device_Used_Mobile	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	-0.000153	1.000000																				
Years_at_Customer	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	-0.000567	1.000000																			
Customer_Region_Asia	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	-0.001590	1.000000																		
Review_Score	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	-0.001680	1.000000																	
Income_Spending_Interaction	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	-0.002796	1.000000																
Spending_Score	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	-0.002388	1.000000														
Device_Used_Laptop	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	-0.002792	1.000000													
Average_Order_Value	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911	-0.003911										
Device_Used/Desktop	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791										
Device_Used/Desktop	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791										
Device_Used/Desktop	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791	-0.002791										
Preferred_Payment_Method_PayPal	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979	-0.003979										
Customer_Region_South_America	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769	-0.005769										
Product_Category_Technology	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731	-0.005731										
Gender	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358	-0.007358										
Preferred_Payment_Method_Credit_Card	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346	-0.009346										
Spending_Efficiency	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368	-0.009368										
Product_Category_Electronics	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888	-0.016888										
Name: Customer_Segment, dtype: float16																																				

- Now, I'll create a binary feature for frequent discount users. Here, users who have used at least one discount will be marked as 1 in the Frequent_Discount_User column and others will be marked as 0.

jupyter E-Commerce_Project Last checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel) |

In [86]:

```
#Discount Propensity: Creating a binary feature for frequent discount users.
1 threshold = 5
2 project_cleaned['Frequent_Discount_User'] = (project_cleaned['Discount_Usage'] > threshold).astype(int)
3 project_cleaned.tail(56)
```

#This creates a binary feature where users who have used at least one discount 4 #!#, Discount_Usage == 1 will be marked as 1 in the Frequent_Discount_User column, 5 and others will be marked as 0.

Out[86]:

Merchandise	Region	Social_America	Income	Spending	Interaction	Loyalty	Customer_Years	Order_Value	Frequency	Orders_per_Year	Spending_Efficiency	Frequent_Discount_User
1		9519056	0	67379.89	21.750000	0	0	57187.72	43.000000	0	0.118791	0
0		384327	2	36872.72	23.750000	0	0	90102.72	23.750000	0	0.036689	0
0		1780000	0	90102.72	23.750000	0	0	0	0	0	0.031613	0
0		3050858	0	17165.46	78.000000	0	0	0	0	0	0.050452	0
0		1873778	12	29617.76	7.416667	0	0	0	0	0	0.340497	0
0		854880	7	33383.16	13.420631	0	0	0	0	0	0.073211	0
1		641316	14	58602.77	5.071429	0	0	0	0	0	0.050480	0
0		9729209	0	18831.44	14.000000	0	0	0	0	0	0.035030	0
0		1044940	0	18831.44	14.000000	0	0	0	0	0	0.036862	0
0		2359920	13	5363.40	5.538440	0	0	0	0	0	0.016662	0
0		3462655	4	2564.00	5.500000	0	0	0	0	0	0.150448	0
0		7738303	4	2786.04	15.500000	0	0	0	0	0	0.174159	0
0		385124	9	3038.81	3.444444	0	0	0	0	0	0.020040	0

- I forgot to encode preferred_Delivery_option into numerical feature, so, I'll do this before heading towards feature selection.

jupyter E-Commerce_Project Last checkpoint: 20 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [88]:

```
1 #CHECKING WHICH COLUMN IS STILL AN OBJECT
2 #THIS SHOWS PREFERRED_DELIVERY_OPTION IS STILL AN OBJECT
3 project_cleaned.dtypes
```

Out [88]:

Gender	int64
Annual_Income	int64
Spouse_Age	int64
Marital_Status	int64
Years_with_Customer	int64
Number_of_Orders	int64
Average_Order_Value	float64
User_Membership	int64
Discount_Usage	int64
Pref_delivery_Option	object
Last_Activity	int64
Review_Score	int64
Customer_Segment	int64
Product_Category_Beauty	uint8
Product_Category_Electronics	uint8
Product_Category_Fashion	uint8
Product_Category_Home	uint8
Preferred_Payment_Method_Credit_Card	uint8
Preferred_Payment_Method_Debit_Card	uint8
Preferred_Payment_Method_Net_Banking	uint8
Preferred_Payment_Method_PayPal	uint8
Device_Used_Desktop	uint8
Device_Used_Mobile	uint8
Device_Used_Tablet	uint8
Customer_Region_Asia	uint8
Customer_Region_Europe	uint8
Customer_Region_North_America	uint8



The screenshot shows a Jupyter Notebook interface with the title "jupyter E-Commerce_Project Last Checkpoint: 20 hours ago (autosaved)". The code cell contains Python code for data processing:

```
# WHERE I'LL CONVERT PREFERRED_DELIVERY_OPTION INTO NUMERICAL FEATURE USING LABEL ENCODING
# Define a custom order for categories
preferred_order = {'Standard': 0, 'Same-Day': 1, 'Express': 2} # Example categories

# Map the categories to the custom order and overwrite the original column.
project_cleaned['Preferred_Delivery_Option'] = project_cleaned['Preferred_Delivery_Option'].map(preferred_order)

# Now check the data type
print(project_cleaned.dtypes) # This will show that 'Preferred_Delivery_Option' is now int64, not object
```

Below the code cell, the notebook lists the data types for each column:

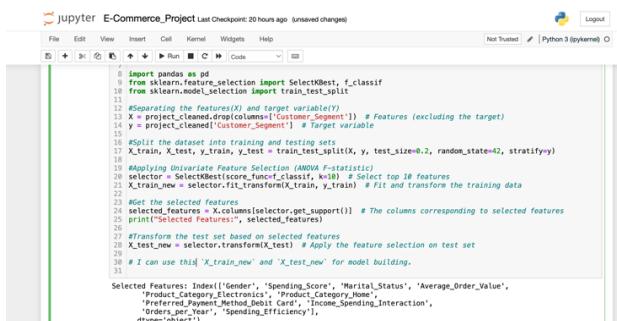
Age	int64
Gender	int64
Annual_Income	int64
Spending_Score	int64
Marital_Status	int64
Years_at_Customer	int64
Number_Cards	int64
Average_Order_Value	float64
Loyalty_Membership	int64
Days_since_Last_Purchase	int64
Preferred_Delivery_Option	int64
Latitude	int64
Review_Score	int64
CustomerSegment	int64
Product_Category_Beauty	uint8
Product_Category_Electronics	uint8
Product_Category_Food	uint8
Product_Category_Home	uint8
Product_Category_Pet	uint8
Preferred_Payment_Method_Credit_Card	uint8

FEATURE SELECTION USING DIFFERENT METHODS

- For feature selection I'll use 3 methods and build a model with the selected features of each method and check the accuracy of the model.
- I'll also do feature selection by combining all the three methods and build a model using that set of features.
- The methods I'll be using for feature Selection are:

Univariate Feature Selection Method

- This method is easy to implement and computationally efficient.
- The Anova F-test used in SelectKBest checks the relationship between each feature and the target variable.
- This helps in identifying the most relevant features for the classification problem.
- This method is also good for initial exploration.
- Below is the code for univariate feature selection method:



The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code for feature selection. The code uses the ANOVA F-test to select the top 10 features from a dataset. It includes importing necessary libraries, separating features and target, splitting the data into training and testing sets, applying the selector, fitting and transforming the training data, getting the selected features, and transforming the test set based on the selected features. The selected features are listed at the bottom of the code cell.

```
8 import pandas as pd
9 from sklearn.feature_selection import SelectKBest, f_classif
10 from sklearn.model_selection import train_test_split
11
12 #Separating the features(X) and target variable(y)
13 X = project_cleaned.drop(columns=['Customer_Segment']) # Features (excluding the target)
14 y = project_cleaned['Customer_Segment'] # Target variable
15
16 #Split the dataset into training and testing
17 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
18
19 #Applying Univariate Feature Selection (ANOVA F-statistic)
20 selector = SelectKBest(score_func=f_classif, k=10) # Select top 10 features
21 X_train_new = selector.fit_transform(X_train, y_train) # Fit and transform the training data
22
23 #Get the selected features
24 selected_features = [feature for feature, support in selector.get_support().items() if support]
25 print("Selected Features:", selected_features)
26
27 #Transform the test set based on selected features
28 X_test_new = selector.transform(X_test) # Apply the feature selection on test set
29
30 #I can use this 'X_train_new' and 'X_test_new' for model building.
31
```

Selected Features: Index(['Gender', 'Spending_Score', 'Marital_Status', 'Average_Order_Value', 'Product_Category_Electronics', 'Product_Category_Home', 'Preferred_Payment_Method_Debit_Card', 'Income_Spending_Interaction', 'Orders_per_Year', 'Spending_Efficiency'], dtype='object')

- The selected features here are 'Gender', 'Spending Score', 'Marital Status', 'Average Order Value', 'Product Category Electronics', 'Product Category Home', 'Preferred payment Method Debit Card', 'Income Spending Interaction', 'Orders Per Year', 'Spending Efficiency'.

- Now, I'll build a model using these features.

```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Logout Not Trusted Python 3 (ipykernel) O
In [88]: 1 #MODEL BUILDING USING THIS UNIVARIATE FEATURE SELECTION METHOD.
2
3 from sklearn.linear_model import LogisticRegression
4 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
5
6 #Initializing the Logistic Regression Model
7 logreg = LogisticRegression(max_iter=1000, random_state=42)
8
9 #Training the Model
10 logreg.fit(X_train_new, y_train)
11
12 #Model Predictions
13 y_pred_lr = logreg.predict(X_test_new)
14
15 #Evaluating the Model
16 # Accuracy Score
17 accuracy_lr = accuracy_score(y_test, y_pred_lr)
18 print("Model Accuracy:", accuracy_lr)
19
20 #Classification Report
21 print("\nClassification Report\n", classification_report(y_test, y_pred_lr))
22
23 #Confusion Matrix
24 print("\nConfusion Matrix\n", confusion_matrix(y_test, y_pred_lr))
25
26
27 #THE ACCURACY WITH LOGISTIC REGRESSION IS NOT SATISFACTORY, SO, I'LL TRY OTHER CLASSIFIER.

```

```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Logout Not Trusted Python 3 (ipykernel) O
In [88]: Model Accuracy: 0.33136363636363636
Classification Report:
precision    recall   f1-score   support
          0       0.00     0.00      0.00    1467
          1       0.33     1.00      0.50    1458
          2       0.00     0.00      0.00    1475

accuracy                           0.33    4400
macro avg       0.11     0.33      0.16    4400
weighted avg    0.11     0.33      0.16    4400

Confusion Matrix:
[[ 0 1467  0]
 [ 0 1458  0]
 [ 0 1475  0]]

```

- The accuracy with logistic regression is just 33% which is quite low, so I'll try other classifiers such as Random forest, Gradient Boosting, SVM and XG boost as well.

```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Logout Not Trusted Python 3 (ipykernel) O
In [89]: 1 #USING RANDOM FOREST CLASSIFIER
2
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
5
6 #Initializing the Random Forest Classifier
7 rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
8
9 #Training the Model
10 rf_model.fit(X_train_new, y_train)
11
12 #Making Predictions
13 y_pred_rf = rf_model.predict(X_test_new)
14
15 #Evaluating the Model
16 accuracy_rf = accuracy_score(y_test, y_pred_rf)
17 print("Random Forest Accuracy:", accuracy_rf)
18
19
20 print("\nClassification Report (Random Forest):\n", classification_report(y_test, y_pred_rf))
21 print("\nConfusion Matrix (Random Forest):\n", confusion_matrix(y_test, y_pred_rf))
22
23
24 #AGAIN THE ACCURACY IS NOT GOOD.

```

```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Logout Not Trusted Python 3 (ipykernel) O
In [89]: 24 #AGAIN THE ACCURACY IS NOT GOOD.
Random Forest Accuracy: 0.32656363636363636
Classification Report (Random Forest):
precision    recall   f1-score   support
          0       0.33     0.32      0.33    1467
          1       0.33     0.32      0.33    1458
          2       0.33     0.33      0.33    1475

accuracy                           0.33    4400
macro avg       0.33     0.33      0.33    4400
weighted avg    0.33     0.33      0.33    4400

Confusion Matrix (Random Forest):
[[494 477 496]
 [514 472 472]
 [587 488 489]]

```

- The accuracy of random forest classifier is also not good.

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)

```
In [89]: 1 #USING GRADIENT BOOSTING CLASSIFIER
2
3 from sklearn.ensemble import GradientBoostingClassifier
4
5 # Step 1: Initialize the Gradient Boosting Classifier
6 gbm_model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)
7
8 # Step 2: Train the Model
9 gbm_model.fit(X_train_new, y_train)
10
11 # Step 3: Make Predictions
12 y_pred_gb = gbm_model.predict(X_test_new)
13
14 # Step 4: Evaluate the Model
15 accuracy_gb = accuracy_score(y_test, y_pred_gb)
16 print("Gradient Boosting Accuracy:", accuracy_gb)
17
18 print("\nClassification Report (Gradient Boosting):\n", classification_report(y_test, y_pred_gb))
19
20 print("\nConfusion Matrix (Gradient Boosting):\n", confusion_matrix(y_test, y_pred_gb))
21
22 #THE ACCURACY IS NOT SATISFACTORY.
```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)

```
Gradient Boosting Accuracy: 0.325
Classification Report (Gradient Boosting):
precision    recall   f1-score   support
          0       0.33      0.31      0.31     1467
          1       0.33      0.31      0.32     1458
          2       0.34      0.39      0.36     1475

accuracy                           0.33    4400
macro avg                           0.33    0.33    0.33    4400
weighted avg                          0.33    0.33    0.33    4400

Confusion Matrix (Gradient Boosting):
[[439 466 562]
 [474 455 520]
 [447 459 569]]
```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)

```
In [90]: 1 #USING SVM CLASSIFIER
2
3 from sklearn.svm import SVC
4
5 # Step 1: Initialize the SVM Classifier
6 svm_model = SVC(kernel='rbf', C=1, gamma='scale', random_state=42)
7
8 # Step 2: Train the Model
9 svm_model.fit(X_train_new, y_train)
10
11 # Step 3: Make Predictions
12 y_pred_svm = svm_model.predict(X_test_new)
13
14 # Step 4: Evaluate the Model
15 accuracy_svm = accuracy_score(y_test, y_pred_svm)
16 print("SVM Accuracy:", accuracy_svm)
17
18 print("\nClassification Report (SVM):\n", classification_report(y_test, y_pred_svm))
19
20 print("\nConfusion Matrix (SVM):\n", confusion_matrix(y_test, y_pred_svm))
21
22 #ACCURACY IS NOT SATISFACTORY.
```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)

```
SVM Accuracy: 0.325
Classification Report (SVM):
precision    recall   f1-score   support
          0       0.00      0.00      0.00     1467
          1       0.32      0.31      0.32     1458
          2       0.33      0.67      0.44     1475

accuracy                           0.33    4400
macro avg                           0.22    0.33    0.25    4400
weighted avg                          0.22    0.33    0.25    4400

Confusion Matrix (SVM):
[[ 0 461 1006]
 [ 0 452 1006]
 [ 0 486 989]]
```

jupyter E-Commerce_Project Last Checkpoint: an hour ago (autosaved)

```
In [91]: 1 #USING XGBOOST
2
3
4 from xgboost import XGBClassifier
5 from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
6
7 # Step 1: Initialize the XGBoost Classifier
8 xgb_model = XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='mlogloss')
9
10 # Step 2: Train the Model
11 xgb_model.fit(X_train_new, y_train)
12
13 # Step 3: Make Predictions
14 y_pred_xgb = xgb_model.predict(X_test_new)
15
16 # Step 4: Evaluate the Model
17 # Accuracy Score
18 accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
19 print("XGBoost Accuracy:", accuracy_xgb)
20
21 # Classification Report
22 print("\nClassification Report (XGBoost):\n", classification_report(y_test, y_pred_xgb))
23
24 # Confusion Matrix
25 print("\nConfusion Matrix (XGBoost):\n", confusion_matrix(y_test, y_pred_xgb))
26
27 #THE ACCURACY OF XGBOOST IS ALSO AROUND 33%
```

```
jupyter E-Commerce_Project Last Checkpoint: an hour ago [automated] Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipython) ▾


```

Parameters: {'use_label_encoder': True} are not used.

warnings.warn(msg, UserWarning)

XGBboost Accuracy: 0.29311818181818184

Classification Report (XGBboost):
precision recall f1-score support
 0 0.33 0.33 0.33 1467
 1 0.31 0.30 0.30 1455
 2 0.35 0.35 0.35 1475

accuracy 0.33 4480
macro avg 0.33 0.33 0.33 4480
weighted avg 0.33 0.33 0.33 4480

Confusion Matrix (XGBboost):
[[489 512 466]
 [518 444 496]
 [461 498 518]]

```


```

- The accuracy using XGBoost, SVM and gradient boost is all somewhere around 33% which is not a good score for a good model.

RECURSIVE FEATURE ELIMINATION METHOD

- I'll use this method because it uses a model to eliminate weak features iteratively.
 - Also, it ensures features selected are truly meaningful for the model.
 - Below is the code I used for feature selection using this method.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter MODEL_BUILDING_RFE_FEATURE_SELECTION_METHOD Last Checkpoint: 7 hours ago (automated)
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help
- Status Bar:** Not Trusted | Python 3 (ipykernel) O

The notebook cell contains the following Python code:

```
P=SCALE IT DOES A PRUNE TO ELIMINATE WEAK FEATURES IMMEDIATELY.
P=ENSURES FEATURES SELECTED ARE TRULY MEANINGFUL FOR THE MODEL.

1 # Importing the libraries
2 import numpy as np
3 import pandas as pd
4
5 from sklearn.linear_model import LogisticRegression
6 from sklearn.feature_selection import RFE
7 from sklearn.model_selection import train_test_split
8
9 #SEPARATING THE FEATURES AND TARGET VARIABLE
10 x = project['cleaned'].drop(['Customer_Segment']) # Features (excluding the target)
11 y = project['cleaned']['Customer_Segment'] # Target variable
12
13 #SPLITTING THE DATASET INTO TRAINING AND TESTING SETS
14
15 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
16
17 #INITIALIZING THE LOGISTIC REGRESSION MODEL
18 model = LogisticRegression(max_iter=1000, random_state=42)
19
20 #INITIALIZING THE RFE MODEL AND FIT IT TO THE TRAINING DATA
21 selector = RFE(model, n_features_to_select=10) # Adjust n_features_to_select as needed
22 X_train_new = selector.fit_transform(X_train, y_train)
23
24 #GET THE SELECTED FEATURES
25 selected_features = X.columns[selector.get_support()]
26 print("Selected Features:", selected_features)
27
28 #TRANSFORMING THE TEST SET BASED ON THE SELECTED FEATURE
29
30 X_test_new = selector.transform(X_test)
31
32 #Can now use 'X_train_new' and 'X_test_new' for model building
33
34 #
```

- The features selected through this method are 'Age', 'Spending score', 'Years as customer', 'Number of orders', Average order value', 'Last Activity', 'Income Spending Interaction', 'Order Value Frequency', 'Orders per year'.
 - Now, I'll build a model using these features.

In [13]:

```
# MODEL BUILDING USING FEATURES SELECTED USING RFE METHOD
1 #from sklearn.linear_model import LogisticRegression
2 #from sklearn.ensemble import RandomForestClassifier
3 #from sklearn.metrics import classification_report, confusion_matrix
4
5 # Step 1: Initialize the Logistic Regression Model
6 lr_model = LogisticRegression(max_iter=1000, random_state=42)
7
8 # Step 2: Train the model on Selected Features
9 lr_model.fit(X_train_new, y_train)
10
11 # Step 3: Make Predictions
12 y_pred_lr = lr_model.predict(X_test_new)
13
14 # Step 4: Evaluate the Logistic Regression Model
15 print("Random Forest Model Performance")
16 print("Accuracy", accuracy_score(y_test, y_pred_lr))
17 print("Classification Report", classification_report(y_test, y_pred_lr))
18 print("Confusion Matrix", confusion_matrix(y_test, y_pred_lr))
19
20 # Step 5: Train a Random Forest Classifier for Comparison
21 rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
22 rf_model.fit(X_train_new, y_train)
23 y_pred_rf = rf_model.predict(X_test_new)
24
25 # Evaluate the Random Forest Model
26 print("Random Forest Model Performance:")
27 print("Accuracy", accuracy_score(y_test, y_pred_rf))
28 print("Classification Report", classification_report(y_test, y_pred_rf))
29 print("Confusion Matrix", confusion_matrix(y_test, y_pred_rf))
```

```

jupyter MODEL_BUILDING_RFE_FEATURE_SELECTION_METHOD Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
Logistic Regression Model Performance:
Accuracy: 0.32477272727272727

Classification Report:
precision recall f1-score support
0 0.67 0.89 0.89 1467
1 0.33 0.33 0.33 1458
2 0.33 0.54 0.41 1475

accuracy macro avg weighted avg
0.44 0.44 0.44
Confusion Matrix:
[[ 2 652 813]
 [ 1 632 112]
 [ 0 684 7911]]
Random Forest Model Performance:
Accuracy: 0.32954545454545453

Classification Report:
precision recall f1-score support
0 0.33 0.33 0.33 1467
1 0.34 0.32 0.32 1458
2 0.34 0.33 0.34 1475

accuracy macro avg weighted avg
0.33 0.33 0.33
Confusion Matrix:
[[480 510 476]
 [479 501 490]
 [479 502 494]]

```

```

jupyter MODEL_BUILDING_RFE_FEATURE_SELECTION_METHOD Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
Random Forest Model Performance:
Accuracy: 0.32954545454545453

Classification Report:
precision recall f1-score support
0 0.33 0.33 0.33 1467
1 0.34 0.32 0.32 1458
2 0.34 0.33 0.34 1475

accuracy macro avg weighted avg
0.33 0.33 0.33
Confusion Matrix:
[[480 510 476]
 [479 501 490]
 [479 502 494]]

```

- Since the accuracy using random forest classifier and Logistic regression is 33% only, I'll try other classifiers as well like XGBoost, Gradient Boosting and SVM.

```

jupyter MODEL_BUILDING_RFE_FEATURE_SELECTION_METHOD Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [14]: 1 #USING XGBOOST
2
3 # XGBoost Classifier
4 from xgboost import XGBClassifier
5 from sklearn.metrics import accuracy_score, classification_report
6
7 # Initialize the XGBoost model
8 xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
9
10 # Fit the XGBoost model on the training data
11 xgb_model.fit(X_train_new, y_train)
12
13 # Make predictions on the test data
14 xgb_predictions = xgb_model.predict(X_test_new)
15
16 # Evaluate the XGBoost model
17 print("XGBoost Model Accuracy:", accuracy_score(y_test, xgb_predictions))
18 print("XGBoost Classification Report:", classification_report(y_test, xgb_predictions))
20

```

```

jupyter MODEL_BUILDING_RFE_FEATURE_SELECTION_METHOD Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
warnings.warn("DeprecationWarning")
XGBoost Model Accuracy: 0.3409999999999999
XGBoost Classification Report:
precision recall f1-score support
0 0.35 0.33 0.34 1467
1 0.34 0.34 0.34 1458
2 0.34 0.35 0.33 1475

accuracy macro avg weighted avg
0.34 0.34 0.34
Confusion Matrix:
[[480 510 476]
 [479 501 490]
 [479 502 494]]

```

```

jupyter MODEL_BUILDING_RFE_FEATURE_SELECTION_METHOD Last Checkpoint: 7 hours ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) Logout
In [15]: 1 #USING GRADIENT BOOSTING
2
3 # Gradient Boosting Classifier
4 from sklearn.ensemble import GradientBoostingClassifier
5 from sklearn.metrics import accuracy_score, classification_report
6
7 # Initialize the Gradient Boosting model
8 gb_model = GradientBoostingClassifier()
9
10 # Fit the Gradient Boosting model on the training data
11 gb_model.fit(X_train_new, y_train)
12
13 # Make predictions on the test data
14 gb_predictions = gb_model.predict(X_test_new)
15
16 # Evaluate the Gradient Boosting model
17 print("Gradient Boosting Model Accuracy:", accuracy_score(y_test, gb_predictions))
18 print("Gradient Boosting Classification Report:", classification_report(y_test, gb_predictions))
20
Gradient Boosting Model Accuracy: 0.33795454545454545
Gradient Boosting Classification Report:
precision recall f1-score support
0 0.36 0.36 0.34 1467
1 0.33 0.32 0.32 1458
2 0.34 0.35 0.35 1475

accuracy macro avg weighted avg
0.34 0.34 0.34
Confusion Matrix:
[[480 510 476]
 [479 501 490]
 [479 502 494]]

```

- This shows the accuracy using other classifiers is also somewhere around 33%.

Random Forest Method

- Tree-based methods like Random Forest are great for classification tasks because they can handle both linear and non-linear relationships between features and the target variable.
 - Random Forest naturally ranks features based on their importance which is calculated using how much each feature reduces impurity.
 - Also, Random Forest can handle complex interactions between features without needing explicit specification from the user.
 - Below is the code for Random Forest Feature Selection Method.

jupyter MODEL_BUILDING_RANDOM_FOREST_SELECTION_METHOD Last Checkpoint: 8 hours ago (un.saved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel) O

```
# X = project_cleaned.drop(columns=['Customer_Segment']) # Features (excluding the target)
# y = project_cleaned['Customer_Segment'] # Target variable

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

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

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

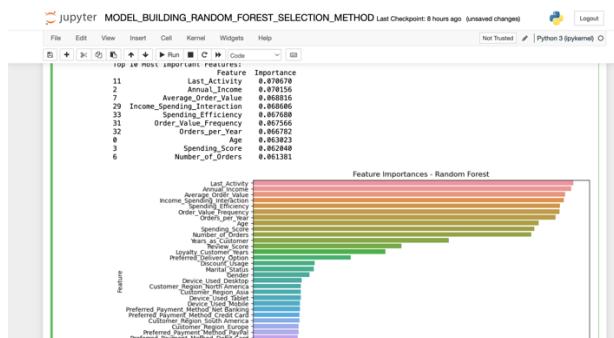
#Get the feature importances
feature_importances = rf_model.feature_importances_

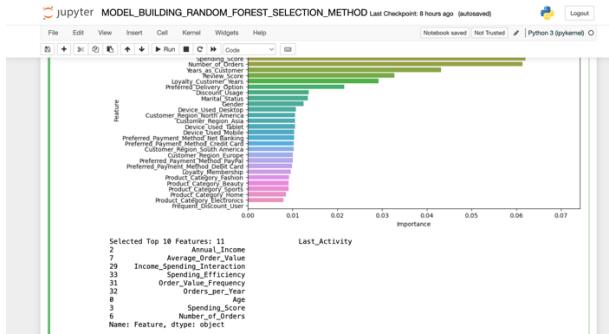
#Create a DataFrame for easy visualization
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': importance})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

#Display the top 10 most important features
print("Top 10 Most Important Features:\n", importance_df.head(10))

#Plot feature importance
sns.set()
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importances - Random Forest')
plt.show()

#Select top 'K' features
top_K_features = importance_df.head(10)['Feature']
print("Selected Top 10 Features: ", top_K_features)
```





- The features selected using this method are ‘Annual Income’, ‘Average order value’, ‘Income Spending Interaction’, ‘Spending Efficiency’, ‘Order Value Frequency’, ‘Orders per year’, ‘Age’, ‘Spending score’, ‘No of Orders’.
 - Now, I’ll build Model using this set of features. Here, Also I’ll use different classifiers to get the best accuracy.



In [15]:

```
1 #MODEL BUILDING USING FEATURES SELECTED BY RANDOM FOREST METHOD
2
3 # USING XGBoost Classifier
4
5 from xgboost import XGBClassifier
6 from sklearn.metrics import accuracy_score, classification_report
7
8 # Initialize the XGBoost model
9 xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
10
11 # Fit the XGBoost model on the selected features of the training data
12 xgb_model.fit(X_train_selected, y_train)
13
14 # Make predictions on the test data
15 xgb_predictions = xgb_model.predict(X_test_selected)
16
17 # Evaluate the XGBoost model
18 print("XGBoost Model Accuracy:", accuracy_score(y_test, xgb_predictions))
19 print("XGBoost Classification Report:\n", classification_report(y_test, xgb_predictions))
```

```
jupyter MODEL_BUILDING_RANDOM_FOREST_SELECTION_METHOD Last Checkpoint: 8 hours ago (autosaved) Log out
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 [ipykernel] 0
 Run Cell Code
warnings.warn(msg, UserWarning)
XGBoost Model Accuracy: 0.3116363636363636
XGBoost Classification Report:
precision    recall   f1-score   support
          0       0.33     0.32      0.32    1467
          1       0.32     0.32      0.32    1458
          2       0.34     0.36      0.35    1475

accuracy                           0.33    4400
macro avg       0.33     0.33      0.33    4400
weighted avg    0.33     0.33      0.33    4400
```

jupyter MODEL_BUILDING_RANDOM_FOREST_SELECTION_METHOD Last Checkpoint: 8 hours ago (autoseved)                                                                                                                                                                                                                                                                          <img alt="blue square icon" data-bbox="3165 10 3170

- The accuracy using all of these different classifiers is still 33%.
- Now, I'll combine all these 3 methods and then do feature selection to check if accuracy improves.

```
In [18]: 1 #FEATURE SELECTION COMBINING ALL THE THREE METHODS
2
3 import pandas as pd
4 from sklearn.model_selection import train_test_split
5 from sklearn.feature_selection import SelectKBest, f_classif
6 from sklearn.ensemble import RandomForestClassifier
7 from sklearn.linear_model import LogisticRegression
8 from sklearn.linear_model import Ridge
9 import numpy as np
10
11 X = project_cleaned.drop(columns=['Customer_Segment']) # Features (excluding the target)
12 y = project_cleaned['Customer_Segment'] # Target variable
13
14 #split the dataset into training and testing sets
15 Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
16
17 #univariate Feature Selection (SelectKBest with f_classif)
18 #univariate_selection = SelectKBest(f_classif, k=10) #Select top 10 features (change as needed)
19 #Xtrain_univariate = univariate_selection.fit_transform(Xtrain, y_train)
20 #selected_univariate_features = X.columns[univariate_selection.get_support()]
21
22 #Recursive Feature Elimination (RFE) using Logistic Regression
23 #rfe = LogisticRegression(max_iter=1000, C=1, random_state=42)
24 #rfe_selector = RFECV(rfe, n_features_to_select=10) # Select top 10 features (change as needed)
25 #Xtrain_rfe = rfe_selector.fit_transform(Xtrain, y_train)
26 #selected_rfe_features = X.columns[rfe_selector.get_support()]
27 selected_rfe_features = X.columns[rfe_selector.get_support()]

In [19]: 1 #Feature Importance using Random Forest
2 rf_model = RandomForestClassifier(random_state=42)
3 rf_model.fit(Xtrain, y_train)
4 importance_rf = rf_model.feature_importances_
5 importance_rf = pd.DataFrame({'Feature': X.columns, 'Importance': importance_rf})
6 importance_rf = importance_rf.sort_values('Importance', ascending=False)
7 selected_rf_features = importance_rf.head(10)['Feature'].values # Top 10 features (change as needed)

#Combine features selected by all methods
#We can take the intersection of features selected by the three methods
8 combined_selected_features = set(selected_univariate_features) & set(selected_rfe_features) & set(selected_rf_features)
9
10 #Display the final selected features
11 print("Final Selected Features:", combined_selected_features)
12
13 #Transform the dataset to use only the selected features
14 X_train_final = Xtrain[combined_selected_features]
15 Xtest_final = Xtest[combined_selected_features]

/Users/susmitayashash/Desktop/Downloads/lib/python3.6/site-packages/sklearn/feature_selection/_univariate_selection.py:17: UserWarning: Features [34] are constant.
warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/Users/susmitayashash/Desktop/Downloads/lib/python3.6/site-packages/sklearn/feature_selection/_univariate_selection.py:13: RuntimeWarning: invalid value encountered in divide
f_msb / msw
Final Selected Features: ['Spending_Score', 'Average_Order_Value', 'Income_Spending_Interaction', 'Orders_per_Year']
```

- The selected features using this method are ‘Spending Score’, ‘Average Order Value’, ‘Income Spending Interaction’, ‘Orders per year’.
- Now, I'll build a model using these features and see how efficient the model is. Again I'll use three different classifiers, XGBoost, Gradient Boosting and SVM.

```
In [11]: 1 #MODEL BUILDING
2
3 #USING XGBoost Classifier
4
5 from xgboost import XGBClassifier
6 from sklearn.metrics import accuracy_score, classification_report
7
8 #Initialize the XGBoost model
9 xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss')
10
11 #Fit the XGBoost model on the selected features of the training data
12 xgb_model.fit(Xtrain_final, y_train)
13
14 #Make predictions on the test data
15 xgb_predictions = xgb_model.predict(Xtest_final)
16
17 #Evaluate the XGBoost model
18 print("XGBoost Model Accuracy:", accuracy_score(y_test, xgb_predictions))
19 print("XGBoost Classification Report:\n", classification_report(y_test, xgb_predictions))
20
21
22
XGBoost Model Accuracy: 0.32727272727272727
XGBoost Classification Report:
precision    recall   f1-score   support
          0       0.33     0.33      0.33    1467
          1       0.32     0.32      0.32    1250
          2       0.33     0.33      0.33    1475

accuracy                           0.33    4400
macro avg       0.33     0.33      0.33    4400
weighted avg    0.33     0.33      0.33    4400
```

```

jupyter MODEL_BUILDING_COMBINED METHOD_FEATURE SELECTION Last Checkpoint: 2 hours ago (autosaved) Python 3 (ipykernel) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
In [14]: 1 # Using GBM
2
3 # Gradient Boosting classifier using selected features
4
5 from sklearn.ensemble import GradientBoostingClassifier
6 from sklearn.metrics import accuracy_score, classification_report
7
8 # Initialize the Gradient Boosting model
9 gbm = GradientBoostingClassifier(n_estimators=100, learning_rate=0.05)
10
11 # Fit the Gradient Boosting model on the selected features of the training data
12 gbm.fit(X_train_final, y_train)
13
14 # Make predictions on the test data
15 gbm_predictions = gbm.predict(X_test_final)
16
17 # Evaluate the Gradient Boosting model
18 print("Gradient Boosting Model Accuracy: ", accuracy_score(y_test, gbm_predictions))
19 print("Gradient Boosting Classification Report:\n", classification_report(y_test, gbm_predictions))
20
21 Gradient Boosting Model Accuracy: 0.32784545454545453
22 Gradient Boosting Classification Report:
23      precision    recall   f1-score   support
24
25          0       0.33     0.32     0.32      1467
26          1       0.33     0.32     0.32      1458
27          2       0.33     0.37     0.35      1475
28
29   accuracy                           0.33    4488
30   macro avg       0.33     0.33     0.33    4488
31   weighted avg    0.33     0.33     0.33    4488

```

```

jupyter MODEL_BUILDING_COMBINED METHOD_FEATURE SELECTION Last Checkpoint: 3 hours ago (autosaved) Python 3 (ipykernel) Logout
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)
In [1]: 1 # Using SVM
2
3 # SVM classifier using selected features
4
5 from sklearn.svm import SVC
6 from sklearn.metrics import accuracy_score, classification_report
7
8 # Initialize the SVM model
9 svm_model = SVC(kernel='linear')
10
11 # Fit the SVM model on the selected features of the training data
12 svm_model.fit(X_train_final, y_train)
13
14 # Make predictions on the test data
15 svm_predictions = svm_model.predict(X_test_final)
16
17 # Evaluate the SVM model
18 print("SVM Model Accuracy: ", accuracy_score(y_test, svm_predictions))
19 print("SVM Classification Report:\n", classification_report(y_test, svm_predictions))
20
21 SVM Model Accuracy: 0.32784545454545453
22 SVM Classification Report:
23      precision    recall   f1-score   support
24
25          0       0.33     0.32     0.32      1467
26          1       0.33     0.32     0.32      1458
27          2       0.33     0.37     0.35      1475
28
29   accuracy                           0.33    4488
30   macro avg       0.33     0.33     0.33    4488
31   weighted avg    0.33     0.33     0.33    4488

```

- The accuracy of the model using all the three classifiers is somewhere around 33 % only.
- Now since models build using features selected from different methods still does not have a satisfactory accuracy, I'll try building a model by selecting features through correlation.

```

jupyter MODEL_BUILDING_USING_FEATURES_CORRELATION Last Checkpoint: 4 hours ago (autosaved) Python 3 (ipykernel) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
In [10]: 1 #CHECKING CORRELATION AFTER DOING FEATURE ENGINEERING
2
3 correlation_with_target = project_cleaned.corr()['Customer_Segment'].sort_values(ascending=False)
4 print(correlation_with_target)
5
6
7 Customer_Segment
8
9 Age           0.000000
10 Product_Category_Home  0.013988
11 Preferred_Payment_Method_Debit_Card  0.011148
12 Product_Category_Electronics  0.009094
13 Marital_Status  0.007184
14 Customer_Age  0.006426
15 Loyalty_Customer_Years  0.005888
16 Loyalty_Membership  0.005534
17 Preferred_Payment_Method_Credit_Card  0.004605
18 Preferred_Payment_Method_BankCard  0.004336
19 Discount_Usage  0.004241
20 Number_of_Children  0.004241
21 Orders_per_Year  0.004054
22 Device_Used_Tablet  0.003954
23 Lot_Type  0.003954
24 Preferred_Payment_Method_Net_Banking  0.003783
25 Current_Payment_Term  0.003601
26 Product_Category_Beauty  0.000451
27 Annual_Income  0.000360
28 Age           -0.000374
29 Device_Used_Mobile  0.000413
30 Years_of_Loyalty  0.000457
31 Order_Value_Frequency  0.000446
32 Customer_Domination_Asia  0.001248
33 Review_Score  0.001688
34 Income_Spending_Interaction  0.001796
35 Spending_Efficiency  0.001798
36 Device_Used_Desktop  0.002791
37 Device_Used/Desktop  0.002791
38
39 Name: Customer_Segment, dtype: float64

```

```

jupyter MODEL_BUILDING_USING_FEATURES_CORRELATION Last Checkpoint: 4 hours ago (autosaved) Python 3 (ipykernel) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
In [10]: 1 #CHECKING CORRELATION AFTER DOING FEATURE ENGINEERING
2
3 correlation_with_target = project_cleaned.corr()['Customer_Segment'].sort_values(ascending=False)
4 print(correlation_with_target)
5
6
7 Customer_Segment
8
9 Average_Order_Value  -0.003911
10 Preferred_Payment_Method_PayPal  -0.003979
11 Current_Payment_Term  -0.003959
12 Product_Category_Sports  -0.007191
13 Gender  -0.007258
14 Preferred_Payment_Method_Credit_Card  -0.000360
15 Spending_Efficiency  -0.000368
16 Product_Category_Electronics  -0.010088
17 Frequent_Discount_User  NaN
18
19 Name: Customer_Segment, dtype: float64

```

- After checking the correlation I'll select only those features with positive correlation value and drop the rest.

```
In [11]: 1 #DROPPING THE FEATURES LESS CORRELATED
2
3
4 # List of features to drop based on correlation analysis
5 features_to_drop = [
6     'Annual_Income', 'Age', 'Device_Used_Mobile', 'Years_as_Customer',
7     'Order_Volume_Frequency', 'Customer_Role_Asia', 'Review_Score',
8     'Order_Volume_Frequency', 'Customer_Role_Asia', 'Review_Score',
9     'Order_Volume_Frequency', 'Customer_Role_Asia', 'Review_Score',
10    'Average_Order_Value', 'Preferred_Payment_Method_PayPal',
11    'Gender', 'Preferred_Payment_Method_Credit_Card', 'Spending_Efficiency',
12    'Gender', 'Preferred_Payment_Method_Credit_Card', 'Spending_Efficiency',
13    'Product_Category_Electronics'
14 ]
15
16 # Dropping the selected features from the dataframe
17 project_cleaned = project_cleaned.drop(columns=features_to_drop)
18
19 # Verifying the dataframe after dropping the features
20 project_cleaned.head()
21
```

Out[11]:

	Marital_Status	Number_of_Orders	Loyalty_Membership	Discount_Use	Preferred_Delivery_Option	Last_Activity	Customer_Segment	Product_Category_Beer
0	1	78	1	0	0	33	0	
1	1	25	1	1	0	307	1	
2	0	68	1	0	1	131	0	
3	0	68	0	1	0	165	1	
4	0	86	0	1	1	29	0	

- Now, I'll build a model using these features.

```
In [14]: 1
2
3 # Define the target variable (Customer_Segment) and features
4 X = project_cleaned.drop(columns=['Customer_Segment'])
5 y = project_cleaned['Customer_Segment']
6
7 # Split the data into training and testing sets
8 c, X_train, y_train, X_test, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
9
10 # Standardizing the data
11 scaler = StandardScaler()
12 X_train_scaled = scaler.fit_transform(X_train)
13 X_test_scaled = scaler.transform(X_test)
14
15
16 ## Initialize the Random Forest model
17 model = RandomForestClassifier(random_state=42)
18
19 # Train the model using the training data
20 model.fit(X_train_scaled, y_train)
21
22
23 # Predict on the test data
24 y_pred = model.predict(X_test_scaled)
25
26 # Evaluate the model's accuracy and performance
27 accuracy = accuracy_score(y_test, y_pred)
28 print(f'Accuracy: {accuracy:.4f}')
29
30 # Detailed classification report
31 print(classification_report(y_test, y_pred))
32
```

Accuracy: 0.3224

	precision	recall	f1-score	support
0	0.32	0.35	0.34	1224
1	0.33	0.33	0.32	1491
2	0.32	0.31	0.32	1485

accuracy

	0.32	0.32	0.32	4400
--	------	------	------	------

macro avg

	0.32	0.32	0.32	4400
--	------	------	------	------

weighted avg

	0.32	0.32	0.32	4400
--	------	------	------	------

- With these features also I am getting almost similar accuracy.
- I'll try doing hyperparameter tuning

```
In [15]: 1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn.metrics import accuracy_score, classification_report
3 from sklearn.preprocessing import StandardScaler
4
5
6 # Define hyperparameter grid
7 parameters = {
8     'n_estimators': [100, 200], # Number of trees in the forest
9     'max_depth': [None, 10], # Maximum depth of the tree
10    'min_samples_leaf': [1, 2], # Minimum samples required to split a node
11    'min_samples_split': [1, 2], # Minimum samples required at a leaf node
12    'max_features': ['auto', 'sqrt'], # Number of features to consider when looking for the best split
13    'bootstrap': [True, False]
14 }
15
16 # Initialize Random Forest model
17 rf_model = RandomForestClassifier(random_state=42)
18
19 # Initialize GridSearchCV
20 grid_search = GridSearchCV(estimator=rf_model, param_grid=parameters,
21                           cv=5, n_jobs=-1, verbose=2, scoring='accuracy')
22
23 # Train the model with grid search to find the best parameters
24 grid_search.fit(X_train_scaled, y_train)
25
26 # Get the best parameters from the grid search
27 grid_search.best_params_
28
29
30 Fitting 5 folds for each of 32 candidates, totalling 160 fits
31 Best Parameters: {'bootstrap': True, 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_sample_ssplit': 2, 'n_estimators': 200}
```

```
In [16]: 1 # Use the best model found by GridSearchCV
2 best_model = grid_search.best_estimator_
3
4 # Make predictions with the best model
5 y_pred = best_model.predict(X_test_scaled)
6
7 # Evaluate the model's performance
8 accuracy = accuracy_score(y_test, y_pred)
9 print(f'Accuracy: {accuracy:.4f}')
10 print(classification_report(y_test, y_pred))
11
```

Accuracy: 0.3358

	precision	recall	f1-score	support
0	0.34	0.35	0.34	1224
1	0.34	0.32	0.33	1491
2	0.34	0.34	0.34	1485

accuracy

	0.34	0.34	0.34	4400
--	------	------	------	------

macro avg

	0.34	0.34	0.34	4400
--	------	------	------	------

weighted avg

	0.34	0.34	0.34	4400
--	------	------	------	------

- Even after hyperparameter tuning, I am getting the same accuracy which is 33%.
- Now, I'll create datasets selecting each set of features obtained from different feature Selecting methods and visualize it on Tableau for better understanding.