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 prepositions 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:

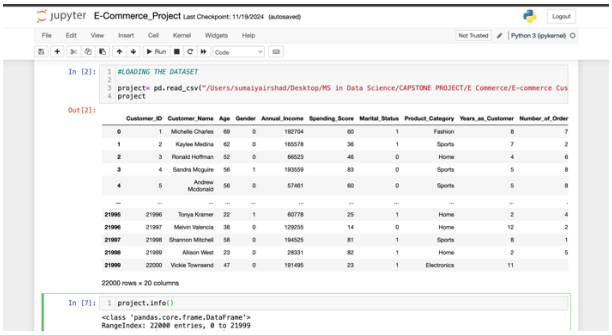
    - Why: Loyalty_Membership, and Preferred_Payment_Method indicates customer preferences and values.
    - Use case: Align loyalty programs or payment options with customer preferences.

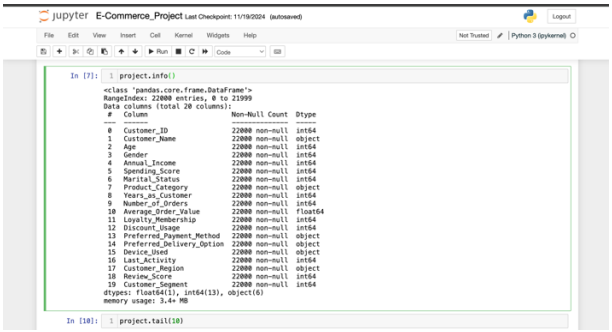
4. Hybrid Segmentation:

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

Data Preprocessing

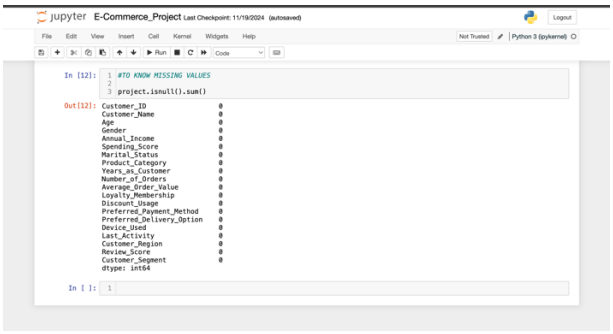
- Load the Dataset: Loading the dataset and getting to know the dataset.





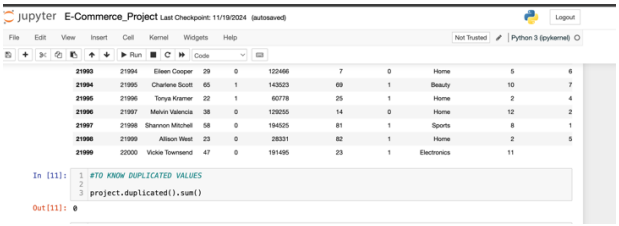
- Clean The Data: Here, I'll fix the missing data, remove duplicate values, and organize everything neatly.

1. Handling missing values



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

2. Handling duplicate values



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

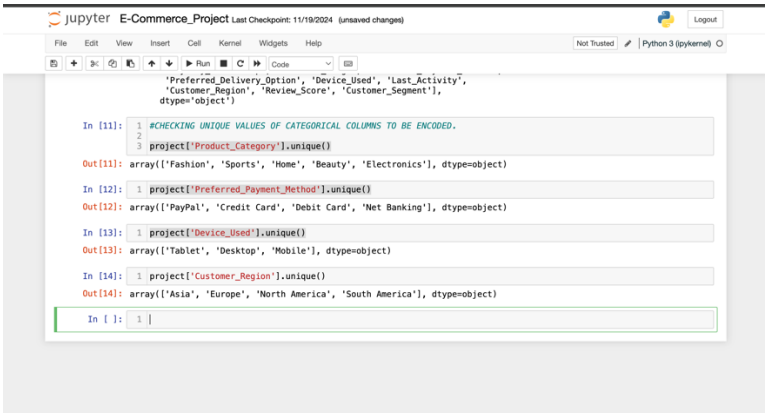
3. Datatypes: Here we'll observe the datatypes and convert them if necessary.

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

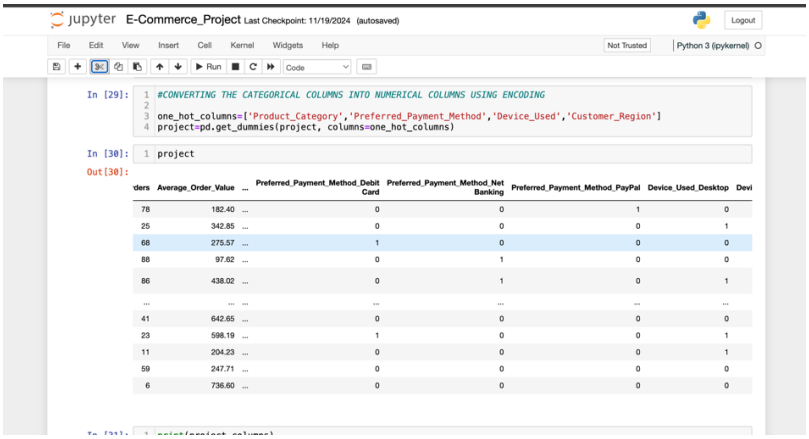
So, we'll need to convert the categorical columns Product_Category, Preferred_Payment_Method, Device_Used and Customer_Region into Numerical datatype using **encoding.** This conversion is done because Machine Learning model works best with numerical data.

Encoding Categorical Values

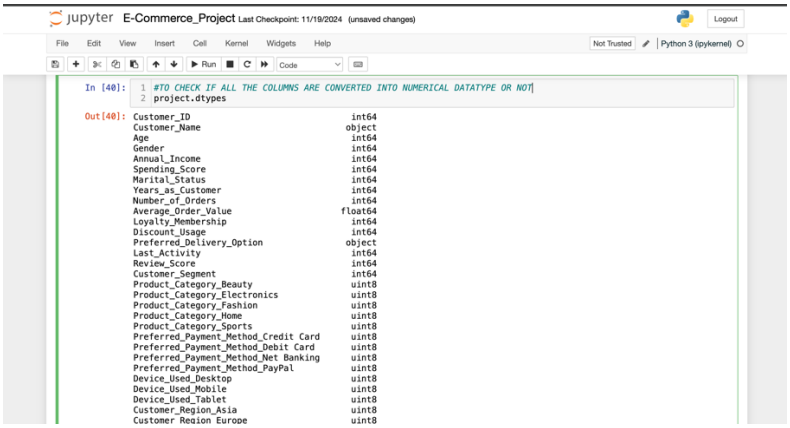
- To encode categorical values, we'll first check the categorical column for unique values as this will help in deciding the type of encoding (i.e., label-encoding or one-hot type encoding).



- 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.

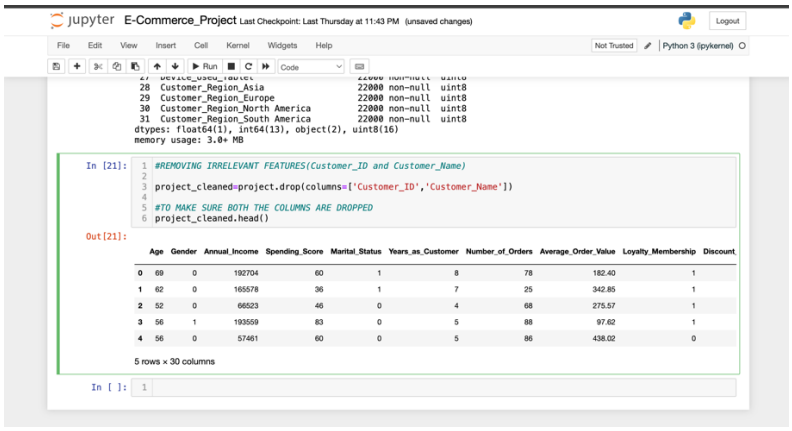


- Now to check if all the columns are converted into numerical datatype.



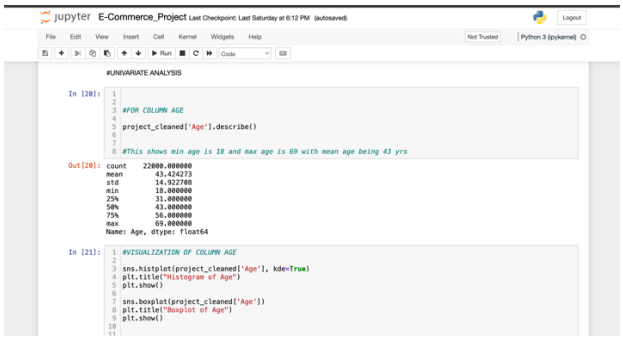
EDA-EXPLORATORY DATA ANALYSIS

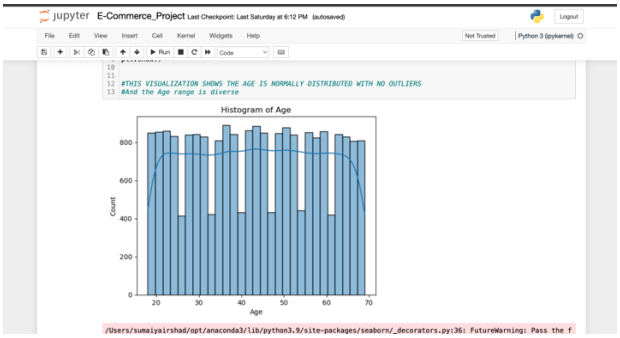
- Before performing EDA, I'll first remove the irrelevant columns (Customer_ID and Customer_Name) as it'll in no way contribute to meaningful patterns and insights.
- For this I'll use the drop method in pandas.

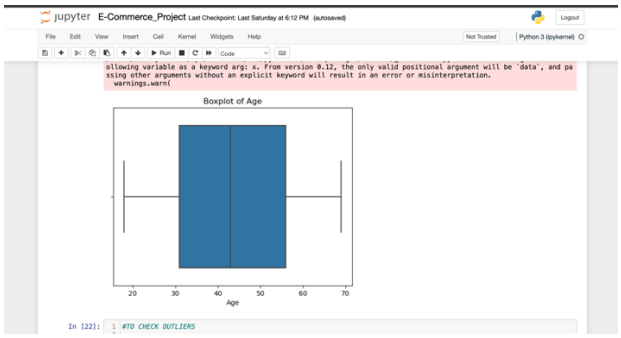


UNIVARIATE ANALYSIS

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

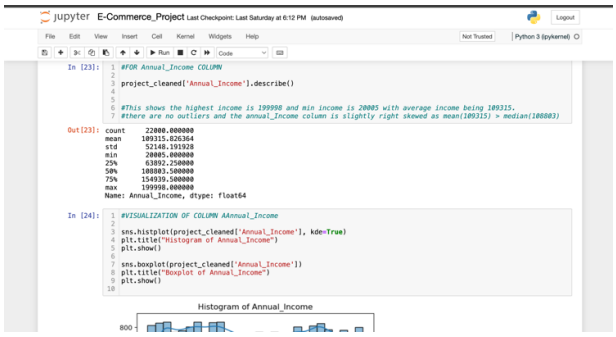


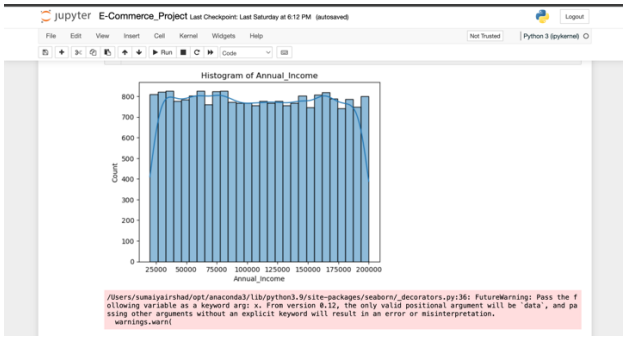




- This visualization shows that the age is distributed normally with no outliers. Also, the range of Age is diverse.
- We can also confirm that there are no outliers by using the below code.

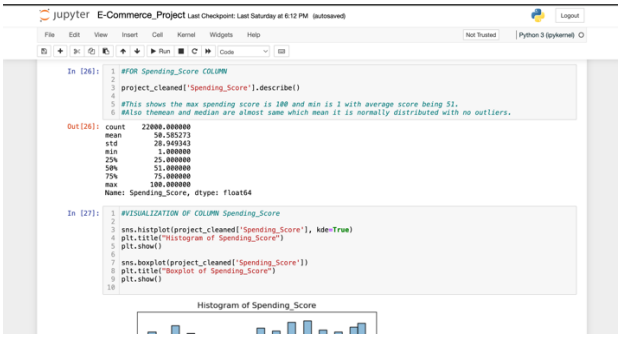
- For Column “ANNUAL_INCOME”

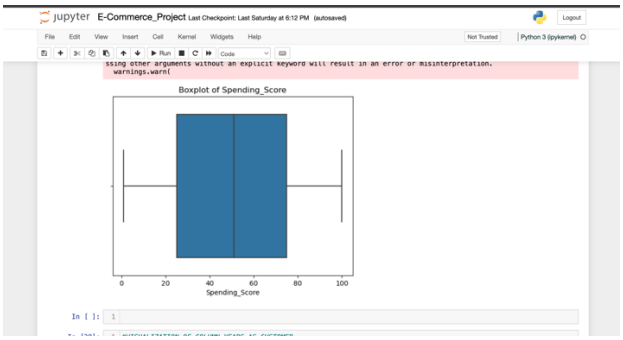




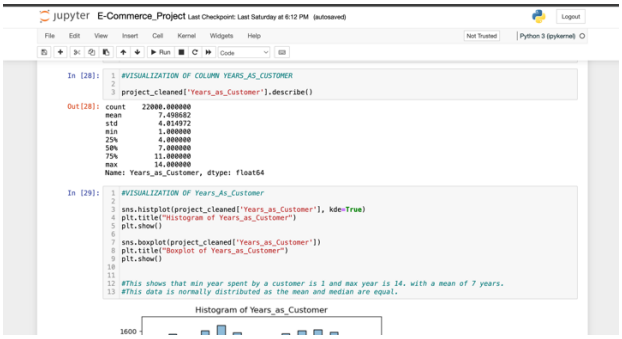
- This shows that the income ranges from 20005 USD to 199998 USD with average income being 109315 USD.
- Also, there are no outliers but the annual_income column is slightly right skewed as mean (109315)> median (108803).
- We can check the outliers by using the same code as above.

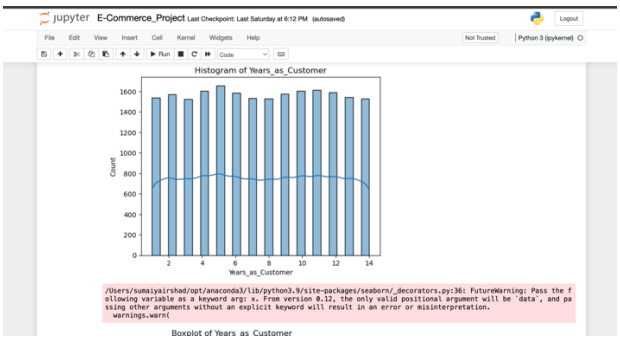
- For Column “SPENDING SCORE”

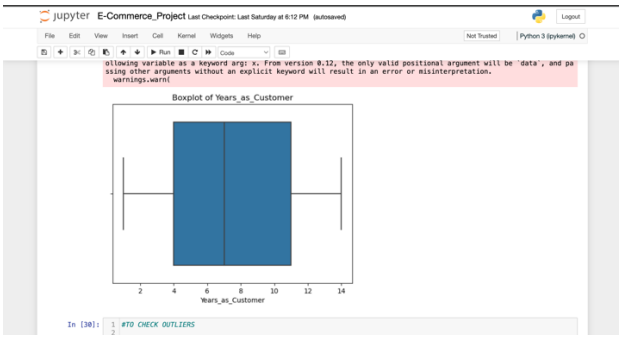




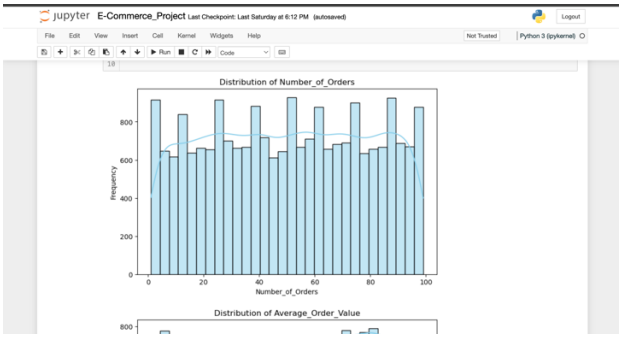
- This visualization shows that the maximum spending score is 100 and min score is 1 with an average of 51 spending score.
- The mean and median are almost same so this column is normally distributed with no outliers.
- Visualization of column " YEARS AS CUSTOMER"

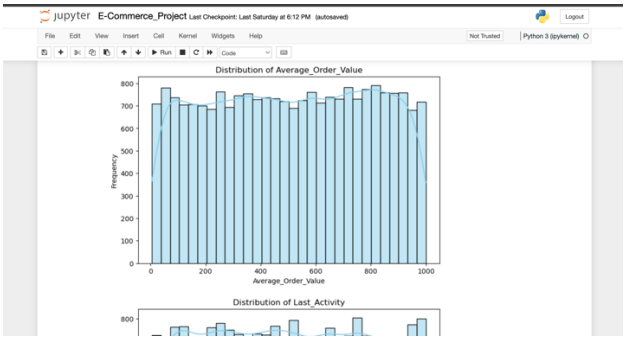


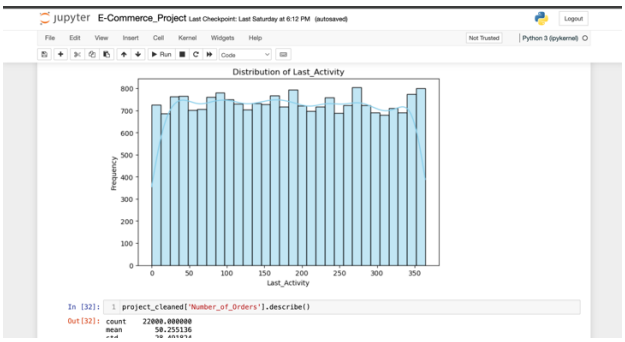


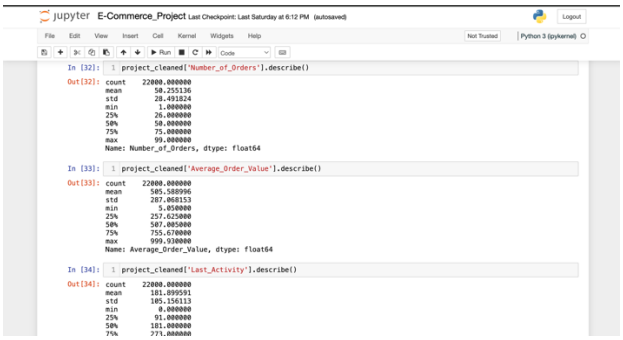


- This visualization shows that the minimum year spent by a customer is 1 and max year is 14 with the average year spent being 7 years.
- This column is normally distributed as mean and median are equal.
- Now, I'll do a combined code for the visualization of rest of the numerical columns.

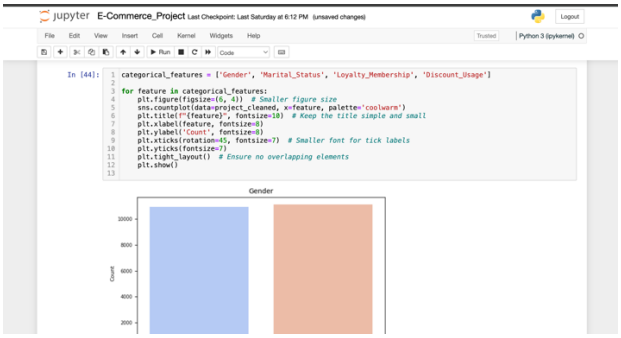


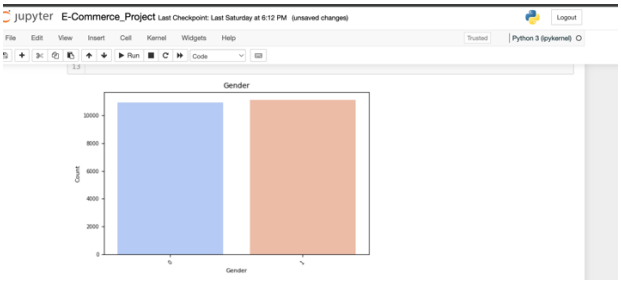


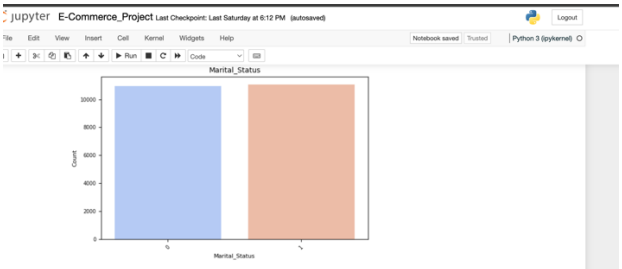




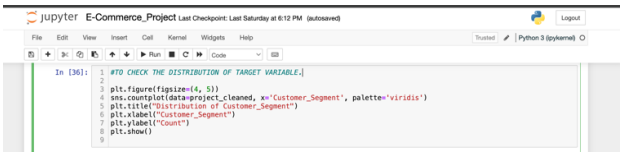
- Now, visualization of categorical features.

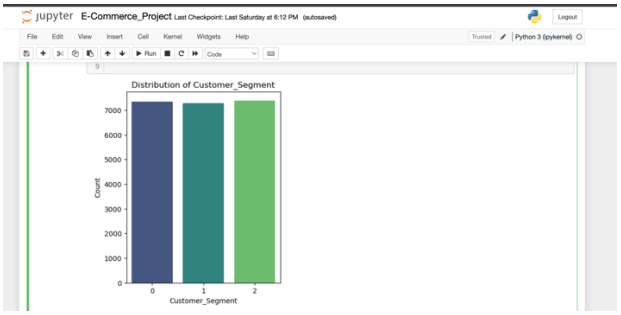






- Now, we'll see the distribution of Target Variable "CUSTOMER SEGMENT"

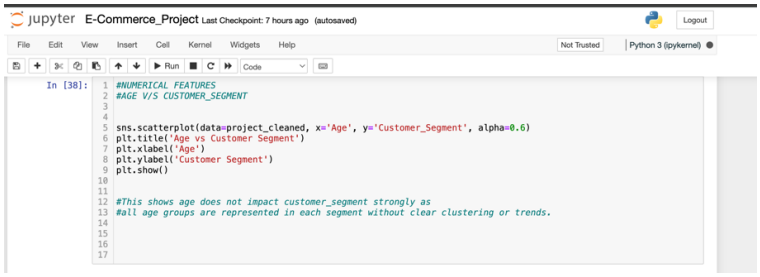


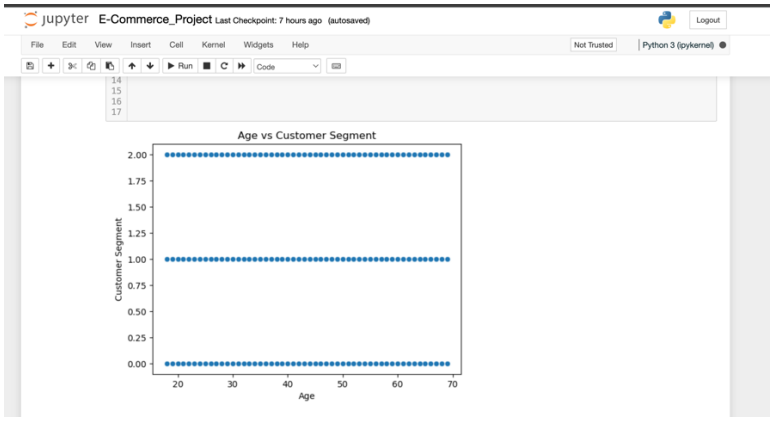


- The heights of the bars for segments 0,1 and 2 are almost equal implying segments are evenly distributed.
- Now, since the customer segments are well-balanced our predictive model will not be biased toward any specific segment due to class imbalance.
- Now, we can move ahead and perform bivariate analysis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

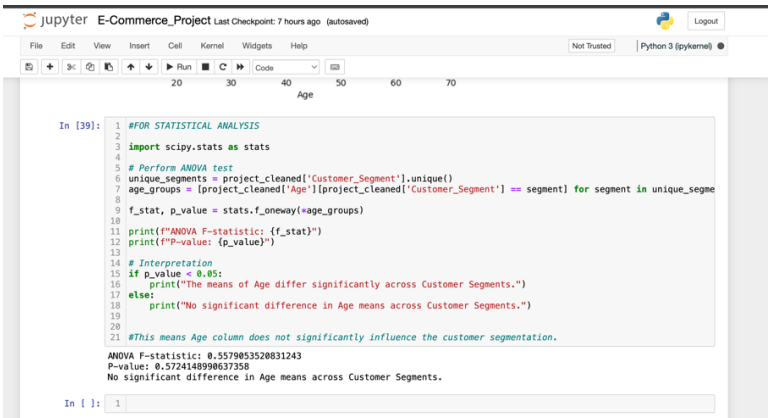
BIVARIATE ANALYSIS

- 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'

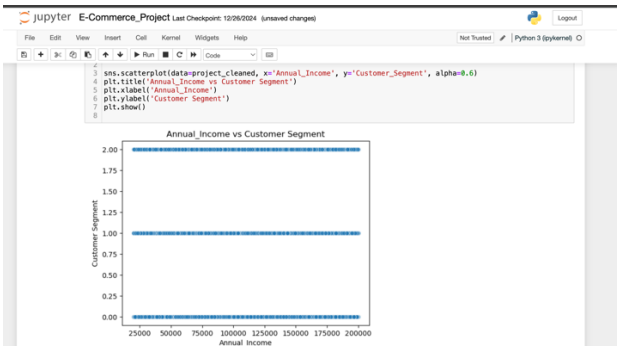




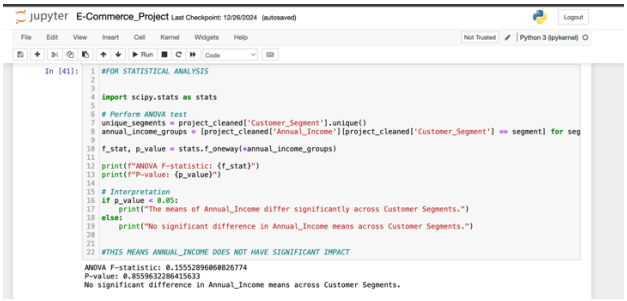
- I have done Statistical Analysis, to check how much age column affect the target variable.



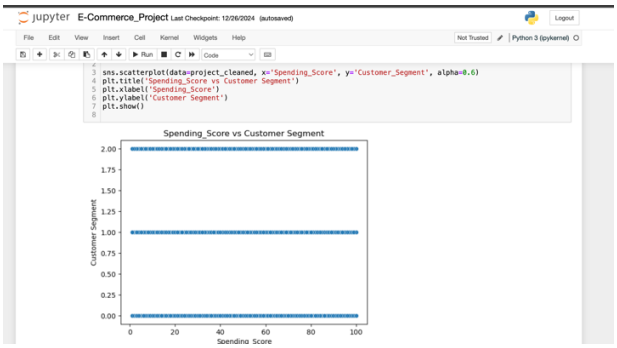
- Since the p-value is >0.05, this shows there is no significant influence of Age column on the target variable.
- ‘ANNUAL_INCOME V/S CUSTOMER SEGMENT’



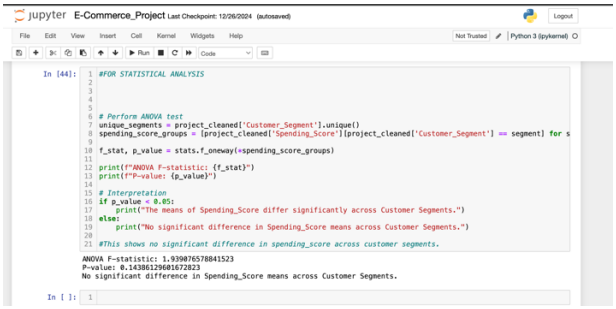
- For Statistical Analysis,



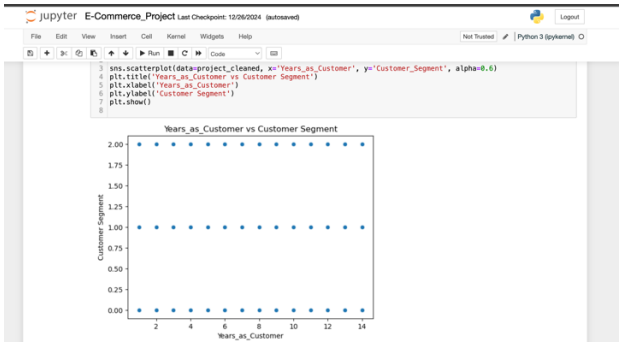
- Since p-value is >0.05, therefore there is no significant effect of Annual Income on the target Variable Customer Segment.
- ‘SPENDING_SCORE V/S CUSTOMER_SEGMENT’



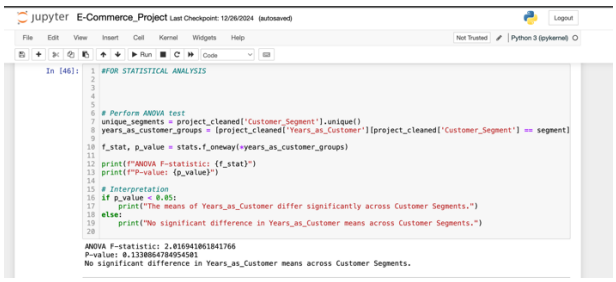
- For Statistical Analysis,



- 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'

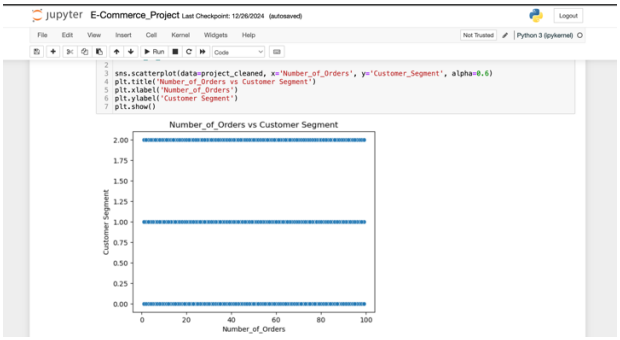


- For Statistical Analysis,

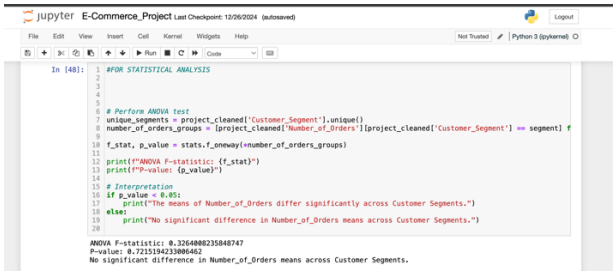


- Here, the p-value is > 0.05, hence no significant difference in years as customer-on-customer segment.

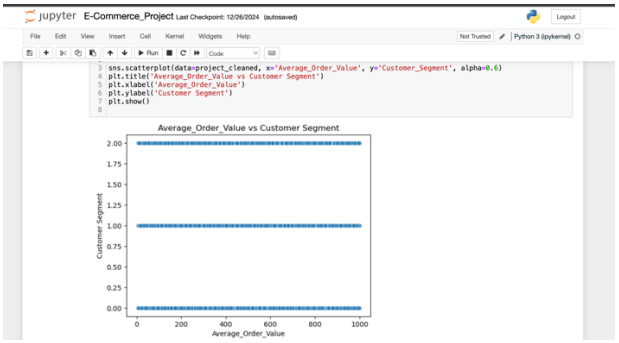
- ‘NUMBER_OF_ORDERS V/S CUSTOMER SEGMENT’



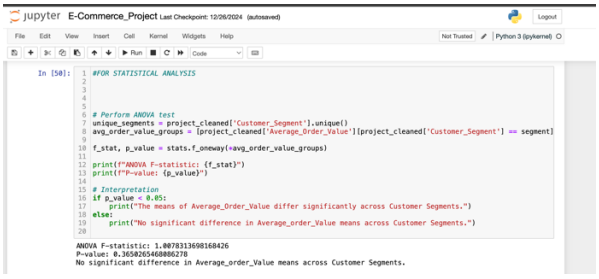
- For Statistical Analysis,



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

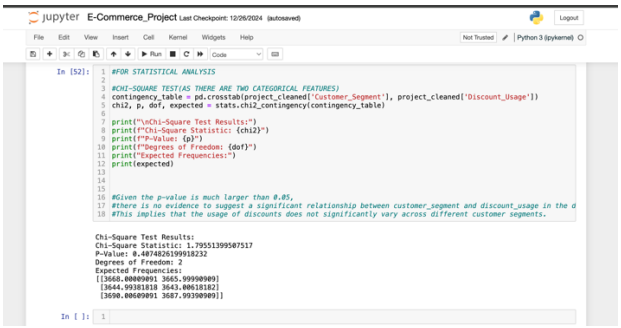


- For Statistical Analysis,

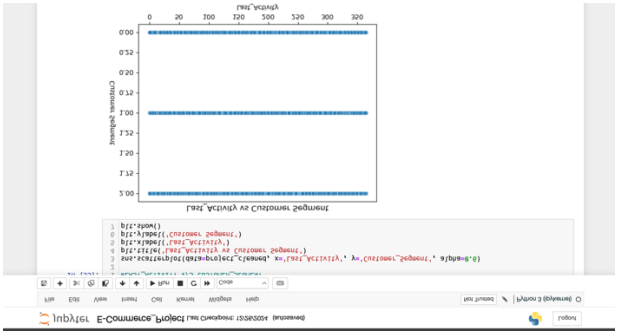


- The p-value is >0.05, hence no influence of average order value on the customer segment.
- ‘DISCOUNT_USAGE V/S CUSTOMER_SEGMENT’

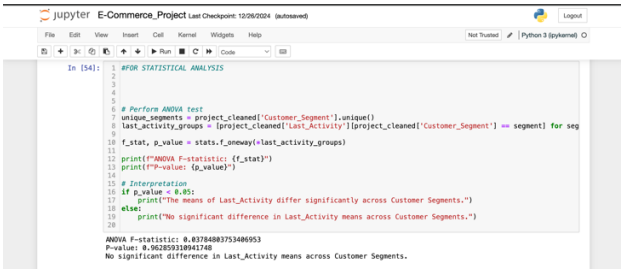
- For Statistical analysis, I’ll do Chi-Square test rather than Anova test as there are two categorical features.



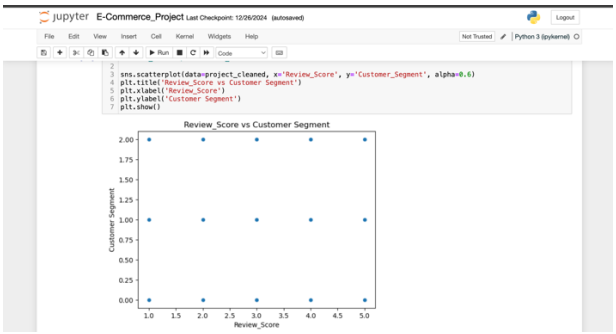
- The p-value is >0.05, hence discount usage also does not affect the target variable much.
- ‘LAST_ACTIVITY V/S CUSTOMER_SEGMENT’



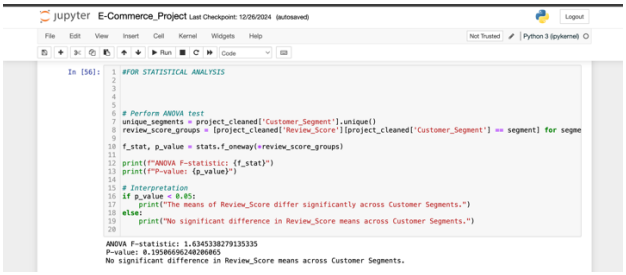
- For Statistical Analysis,



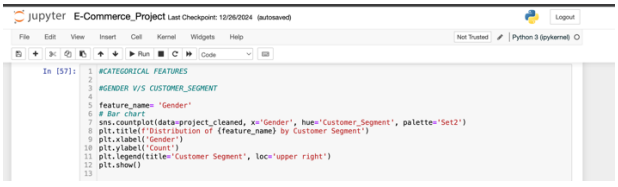
- The p-value is >0.05, hence there is no influence of Last Activity on the Customer Segment.
- 'REVIEW_SCORE V/S CUSTOMER_SEGMENT'

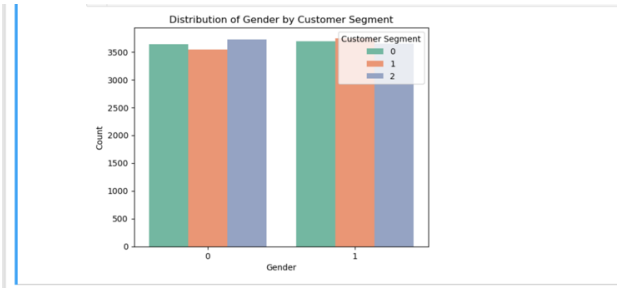


- For Statistical Analysis,

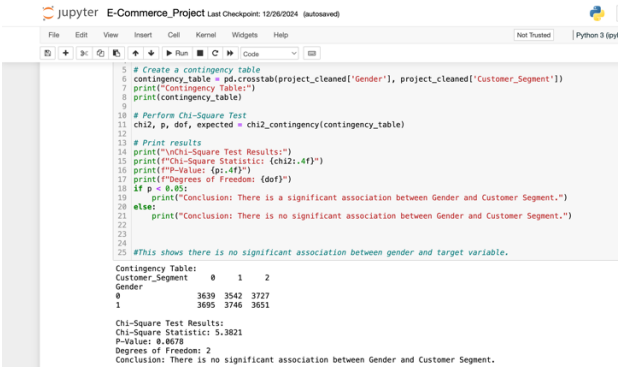


- The p-value is >0.05, then the review score also does not make much impact on the target variable.
- Now, I'll do the bivariate analysis of categorical features with Target Variable to understand more about the data.
- 'GENDER V/S CUSTOMER SEGMENT'

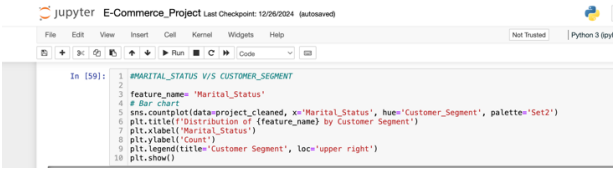


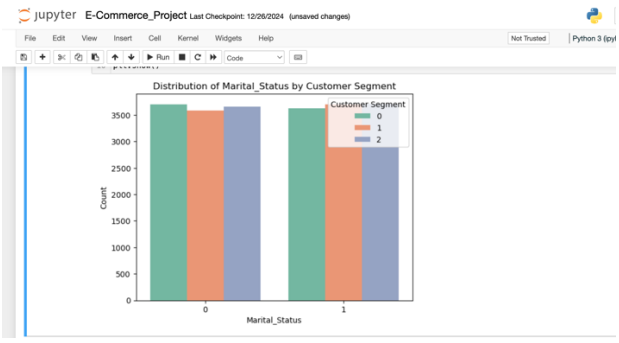


- For Statistical Analysis, I'll do the Chi-Square Test as we are dealing with two categorical features.

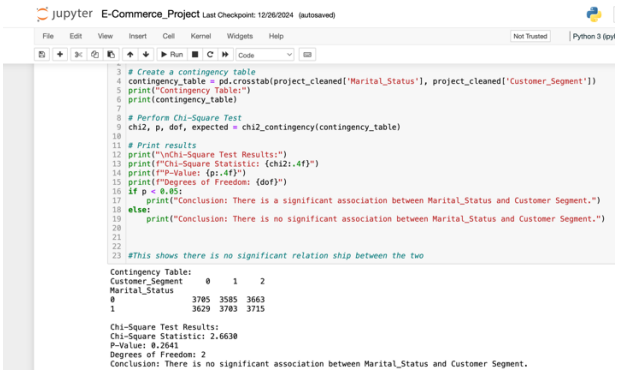


- This shows there is no significant association between Gender and Customer Segment as the p-value is >0.05.
- 'MARITAL_STATUS V/S CUSTOMER_SEGMNET'



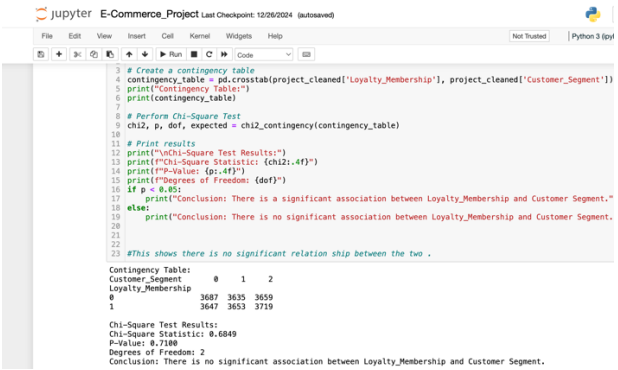


- For Statistical Analysis,



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

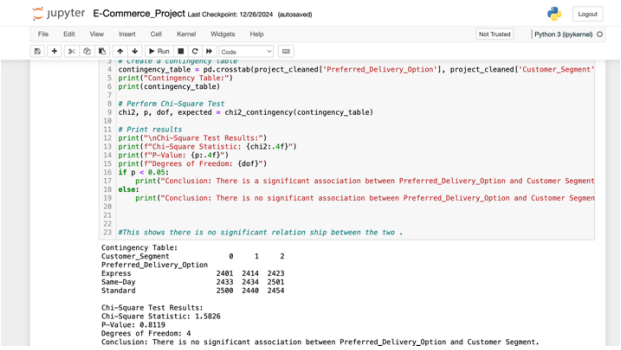
- For Statistical Analysis,



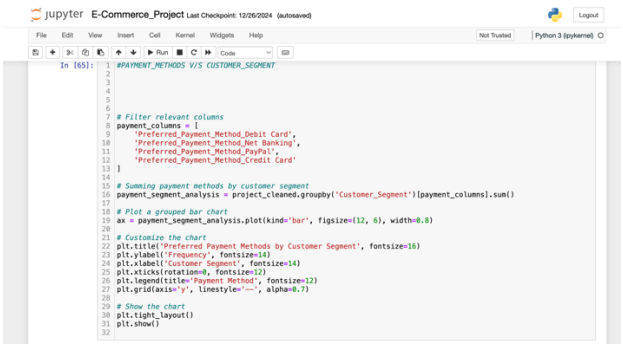
- Since the p-value is>0.05 there is no link between the loyalty membership and the target variable.

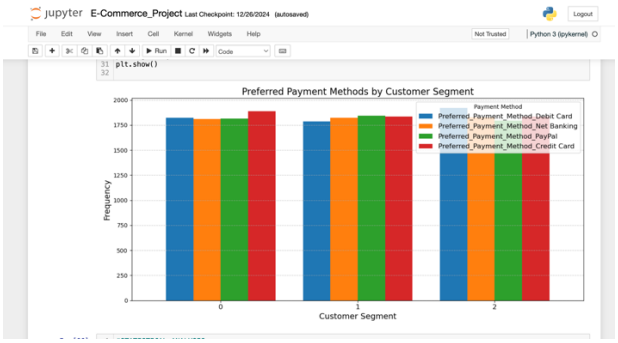
- ‘PREFERRED_DELIVERY_OPTION V/S CUSTOMER SEGMNET’

- For Statistical Analysis,

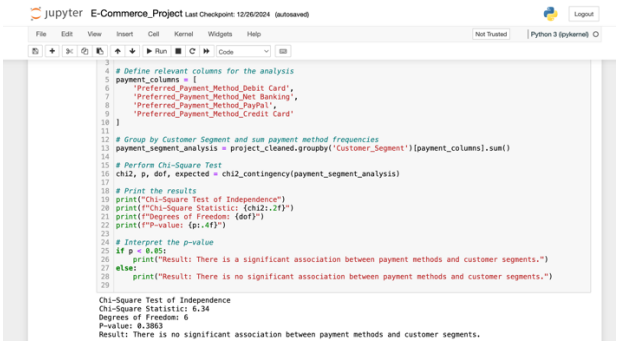


- Here also the p-value is >0.05, hence there is no association between preferred delivery option and the target variable.
- ‘PAYMENT_METHODS V/S CUSTOMER SEGMENT’

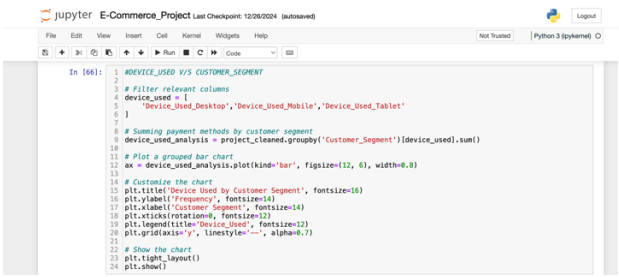


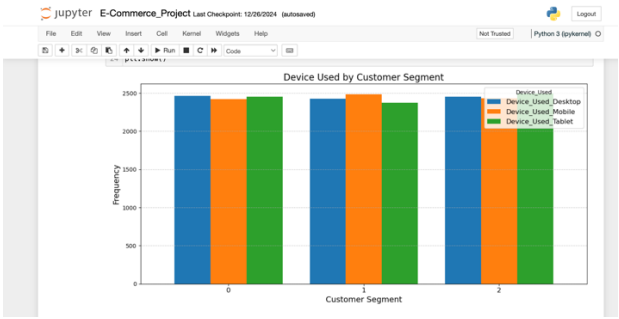


- For Statistical Analysis,

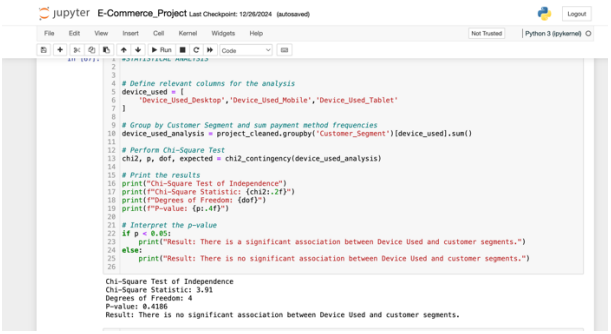


- This shows there is no association between preferred payment method and customer segment.
- ‘DEVICE_USED V/S CUSTOMER_SEGMENT’





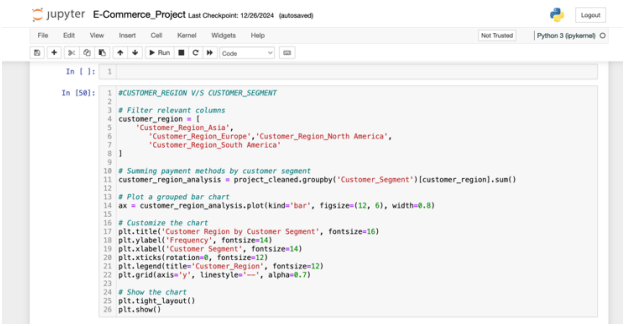
- For Statistical Analysis,



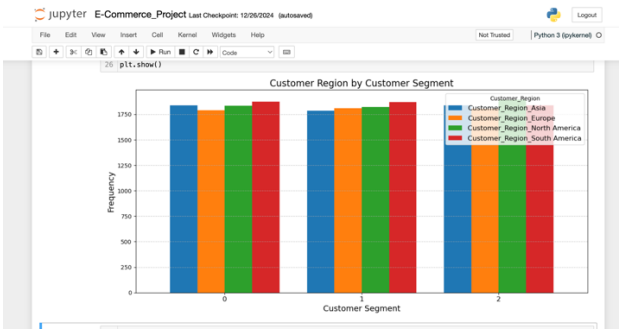
```
# Define relevant columns for the analysis
device_used = [
    'Device_Used_Desktop','Device_Used_Mobile','Device_Used_Tablet'
]

# Group by Customer Segment and sum payment method frequencies
device_used_analysis = project_cleaned.groupby('Customer_Segment')[device_used].sum()

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

# Print the results
print("Chi-Square Test of Independence")
print(f"Chi-Square Statistic: {chi2:.2f}")
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p:.4f}")

# Interpret the p-value
if p < 0.05:
    print("Result: There is a significant association between Device Used and customer segments.")
else:
    print("Result: There is no significant association between Device Used and customer segments.")
```

```
Chi-Square Test of Independence
Chi-Square Statistic: 3.91
Degrees of Freedom: 4
P-value: 0.4186
Result: There is no significant association between Device Used and customer segments.
```

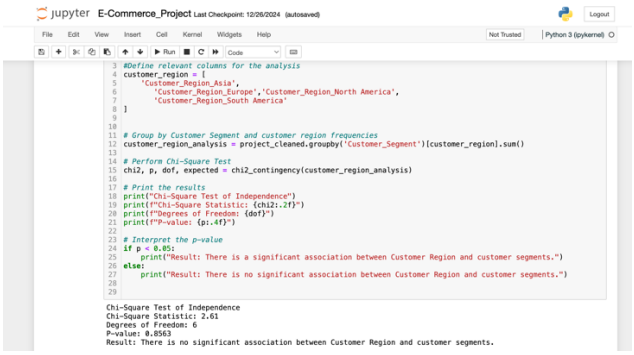
- The p-value is >0.05, hence there is no association between device used and customer segment.
- ‘CUSTOMER_REGION V/S CUSTOMER_SEGMENT’



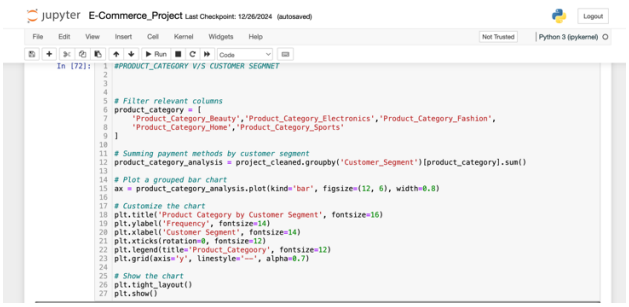
```
#CUSTOMER_REGION V/S CUSTOMER_SEGMENT

# Filter relevant columns
customer_region = [
    'Customer_Region_Asia',
        'Customer_Region_Europe','Customer_Region_North America',
        'Customer_Region_South America'
]

# Summing payment methods by customer segment
customer_region_analysis = project_cleaned.groupby('Customer_Segment')[customer_region].sum()

# Plot a grouped bar chart
ax = customer_region_analysis.plot(kind='bar', figsize=(12, 6), width=0.8)

# Customize the chart
plt.title('Customer Region by Customer Segment', fontsize=16)
plt.ylabel('Frequency', fontsize=14)
plt.xlabel('Customer Segment', fontsize=14)
plt.xticks(rotation=0, fontsize=12)
plt.legend(title='Customer_Region', fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show the chart
plt.tight_layout()
plt.show()
```

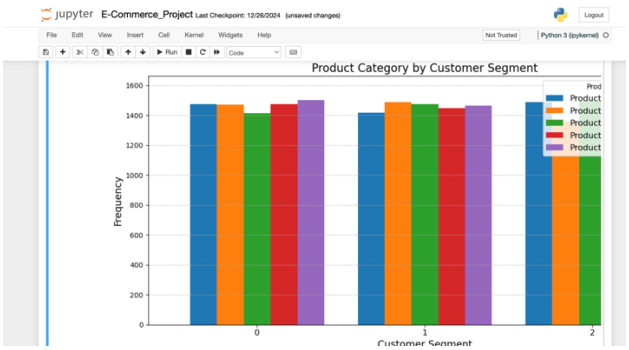


- For Statistical Analysis,

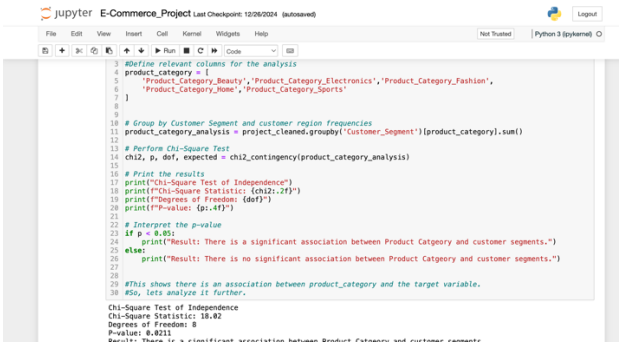


- The p-value is >0.05, hence there is no link between customer region and the target variable.
- ‘PRODUCT_CATEGORY V/S CUSTOMER_SEGMNET’

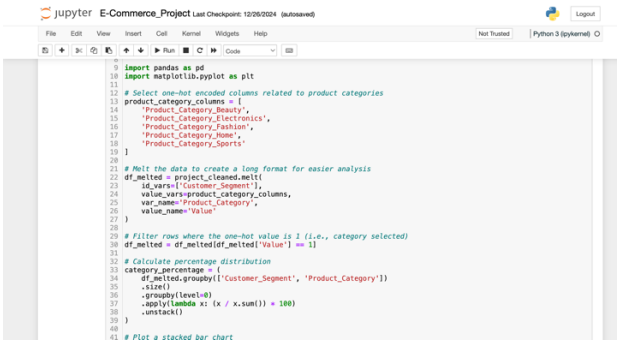


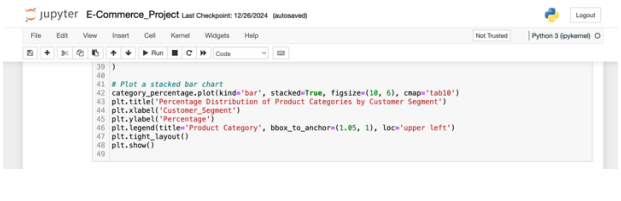


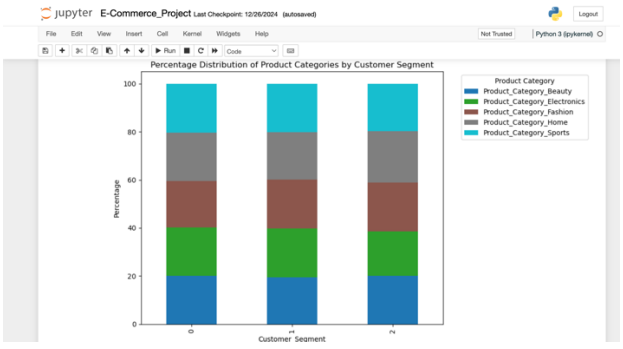
- For Statistical Analysis,



- 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.



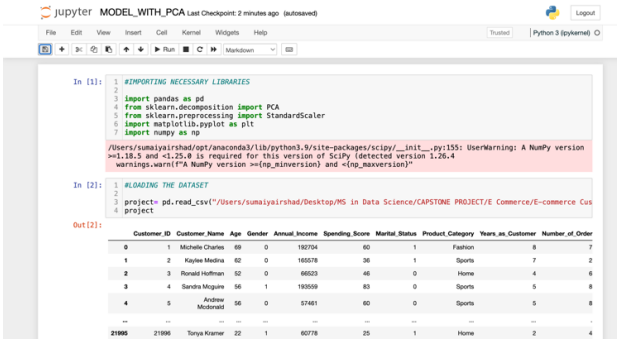




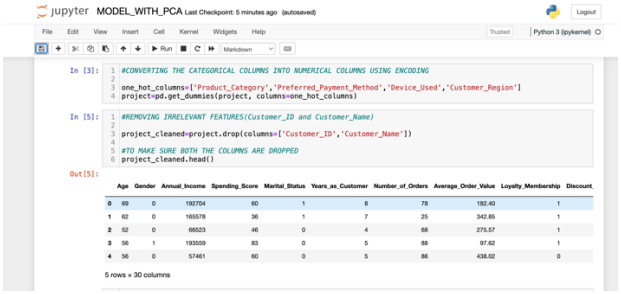
- After doing the Univariate and Bivariate Analysis I couldn't find any major affect of the features on the target variable.
- Now, I'll do PCA(Principal Component Analysis) and try to build a model first using Logistic Regression.

PCA (PRINCIPAL COMPONENT ANALYSIS)

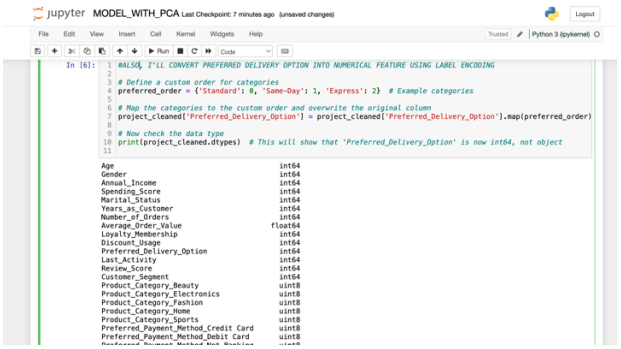
- Below is the steps I did to do PCA :
- First, I loaded the necessary libraries and the required dataset.



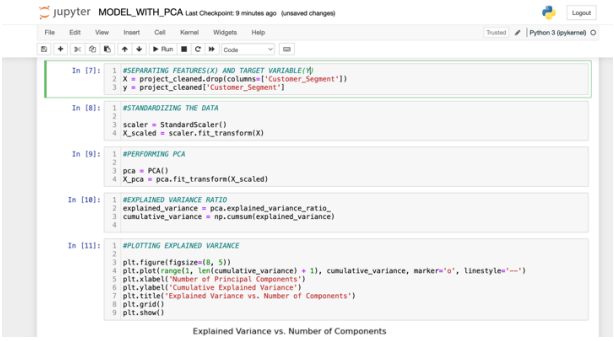
- Then, I did Encoding to convert categorical columns into numerical and also removed irrelevant columns.

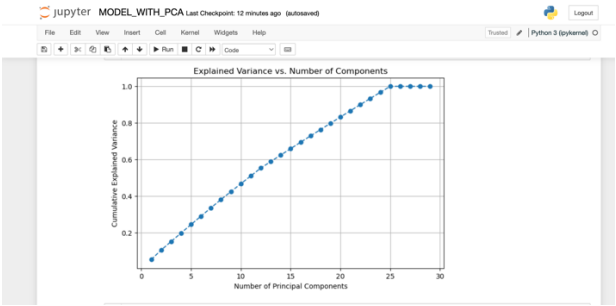


- Also, I converted Preferred_delivery_option into numerical feature using label encoding.

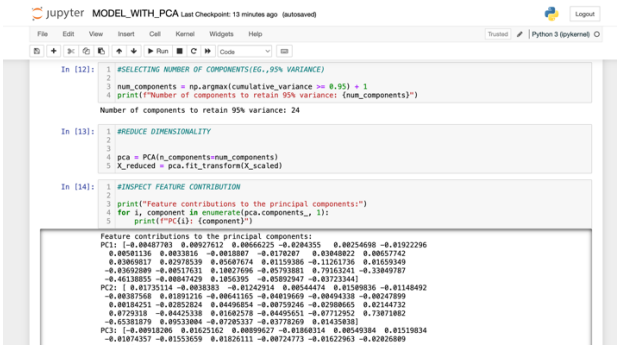


- Now, I'll separate the dataset into features and target variable, standardize it and perform PCA. I will also plot the explained variance ratio.

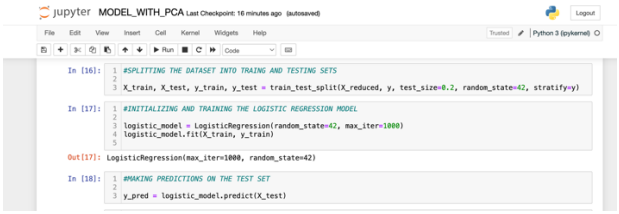




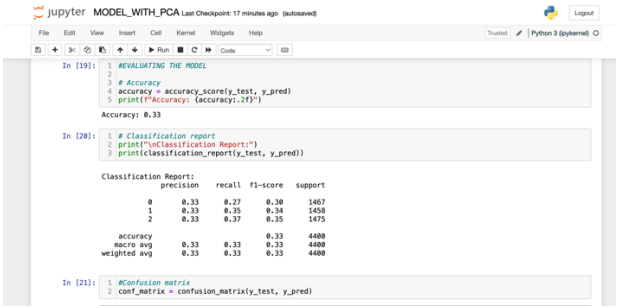
- Now, I'll select the number of components, reduce dimensionality and inspect feature contribution.

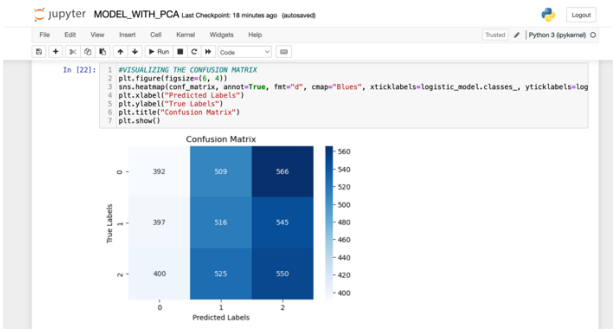


- I'll build a Model now using Logistic Regression. I 'll split the data into training and testing sets, initialize and train the logistic regression model and make predictions on the test set.

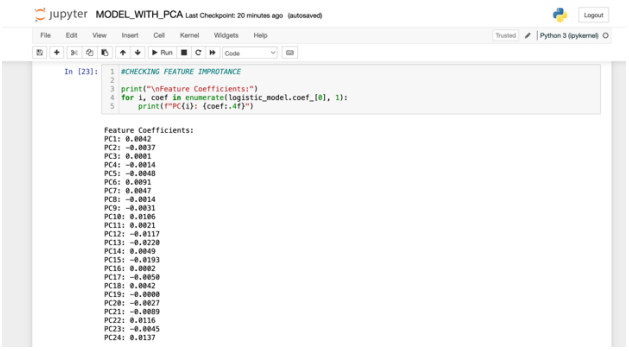


- Now, I'll evaluate the model and also visualize the confusion matrix.





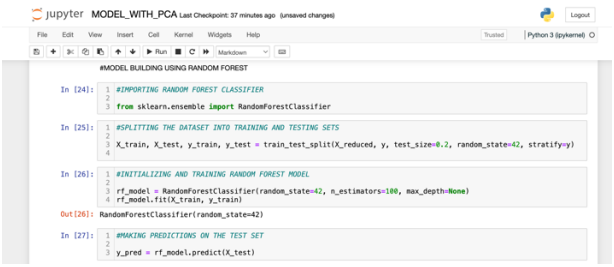
- Lastly, I'll check the Feature importance and interpret my results.

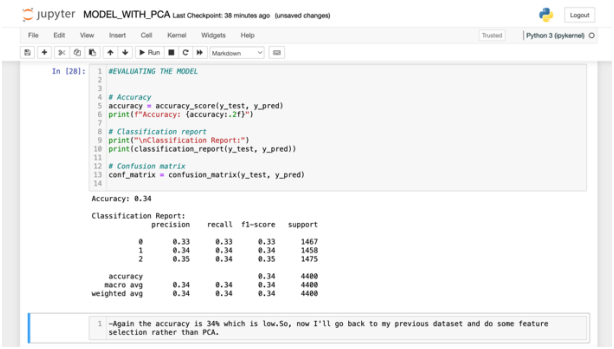


- So, this tells us that the model building with PCA has really low accuracy (33%) which means the dataset after applying PCA is not appropriate for the model.
- The reasons for low accuracy could be Class Imbalance, Over-dimensionality (too many features as even after PCA we might have unnecessary features),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

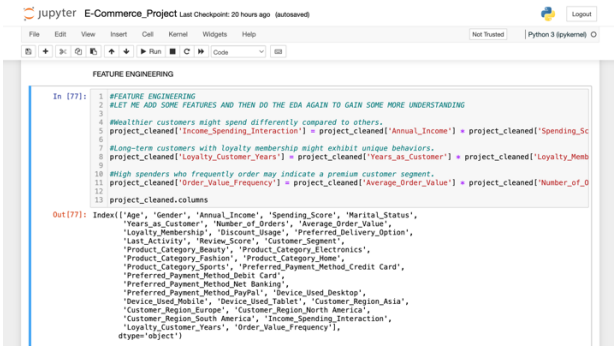




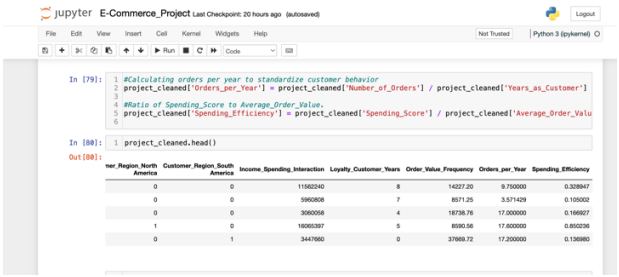
- The accuracy using Random Forest Classifier is also low (34%).So, now I'll skip checking with gradient booster and rather do some feature selection techniques on my previous data.
- I'll do some Feature Engineering before Selecting features using different Feature selecting methods.

FEATURE ENGINEERING

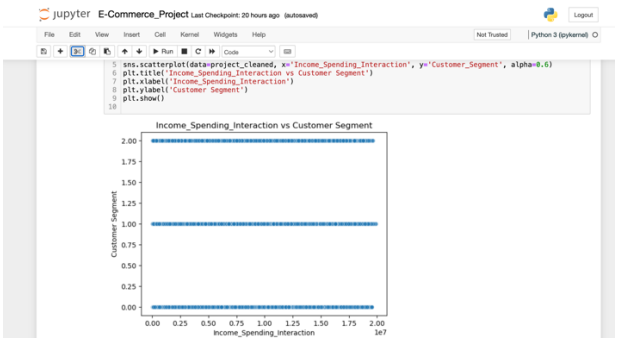
- I'll add 'Income_Spending_Interaction', 'Loyalty_Customer_Years' and 'Order_value_frequency'.



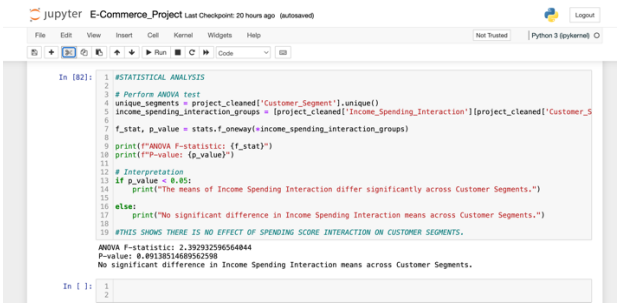
- I'll also add 'Orders_per_year' and 'Spending_Efficiency'.



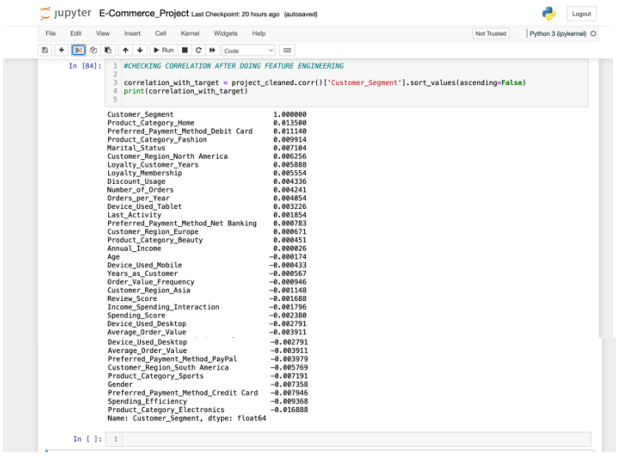
- Will again do some Eda with the new feature and the Target Variable to understand if there is any influence on the target variable.
- Income_Spending_Interaction v/s Customer_Segment



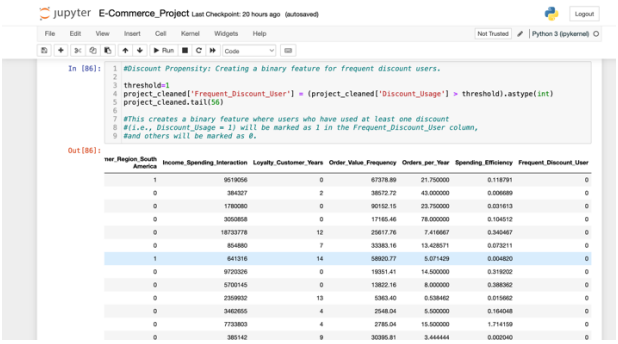
- The Statistical Analysis shows there is no association between income spending interaction and target variable.



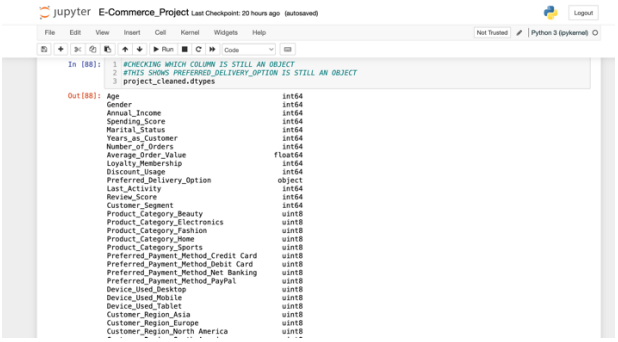
- I will check the correlation after feature Engineering.

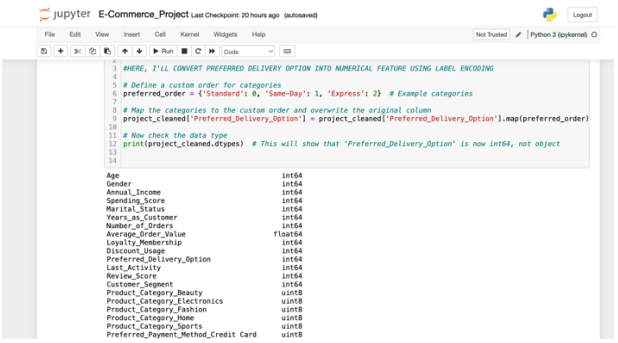


- Now, I'll create a binary feature for frequent discount users. Here, users who have used at least one discount will be marked as 1 in the Frequent_Discount_User column and others will be marked as 0.



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



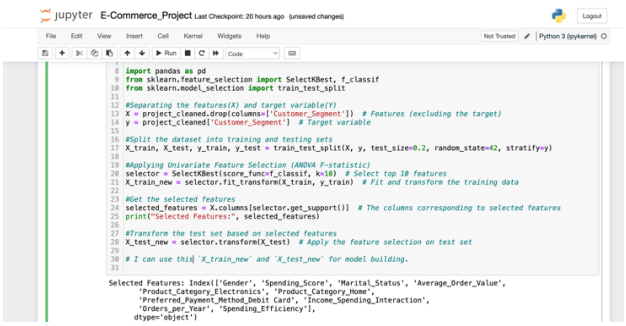


FEATURE SELECTION USING DIFFERENT METHODS

- 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:

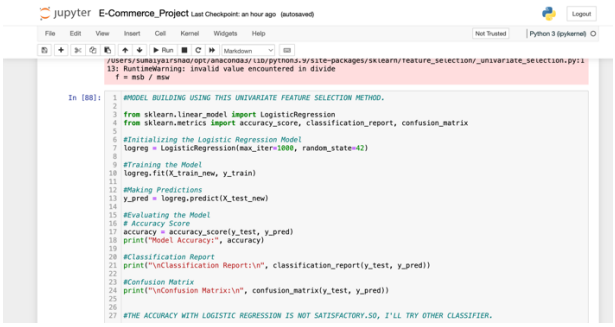
Univariate Feature Selection Method

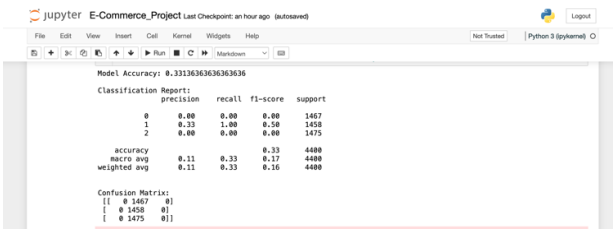
- This method is easy to implement and computationally efficient.
- The Anova F-test used in SelectKBest checks the relationship between each feature and the target variable.
- This helps in identifying the most relevant features for the classification problem.
- This method is also good for initial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



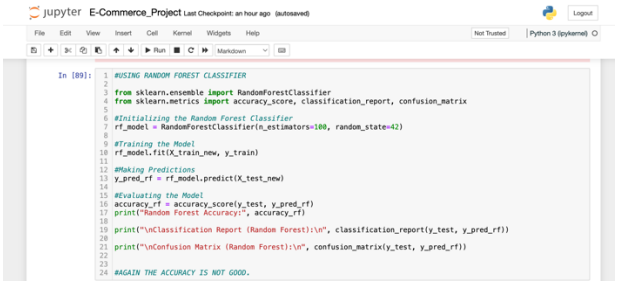
- The selected features here are 'Gender', 'Spending Score', 'Marital Status', 'Average Order Value', 'Product Category Electronics', 'Product Category Home', 'Preferred payment Method Debit Card', 'Income Spending Interaction', 'Orders Per Year', 'Spending Efficiency'.

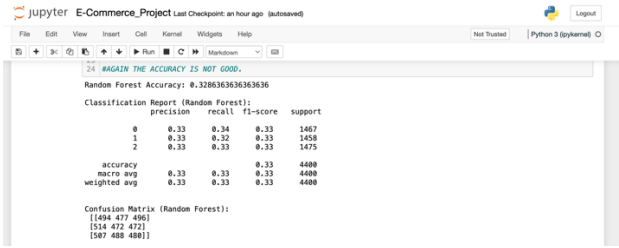
- Now, I'll build a model using these features.





- 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.





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

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

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

In [89]:

```
#USING GRADIENT BOOSTING CLASSIFIER

from sklearn.ensemble import GradientBoostingClassifier

# Step 1: Initialize the Gradient Boosting Classifier
gb_model = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, random_state=42)

# Step 2: Train the Model
gb_model.fit(X_train_new, y_train)

# Step 3: Make Predictions
y_pred_gb = gb_model.predict(X_test_new)

# Step 4: Evaluate the Model
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print("Gradient Boosting Accuracy:", accuracy_gb)

print("\nClassification Report (Gradient Boosting):\n", classification_report(y_test, y_pred_gb))

print("\nConfusion Matrix (Gradient Boosting):\n", confusion_matrix(y_test, y_pred_gb))

#THE ACCURACY IS NOT SATISFACTORY.
```

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

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

```
Gradient Boosting Accuracy: 0.3325

Classification Report (Gradient Boosting):
              precision    recall  f1-score   support

           0       0.32      0.30      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]
 [474 455 529]
 [447 459 569]]
```

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

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

In [90]:

```
#USING SVM CLASSIFIER

from sklearn.svm import SVC

# Step 1: Initialize the SVM Classifier
svm_model = SVC(kernel='rbf', C=1, gamma='scale', random_state=42)

# Step 2: Train the Model
svm_model.fit(X_train_new, y_train)

# Step 3: Make Predictions
y_pred_svm = svm_model.predict(X_test_new)

# Step 4: Evaluate the Model
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print("SVM Accuracy:", accuracy_svm)

print("\nClassification Report (SVM):\n", classification_report(y_test, y_pred_svm))

print("\nConfusion Matrix (SVM):\n", confusion_matrix(y_test, y_pred_svm))

#ACCURACY IS NOT SATISFACTORY.
```

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

File Edit View Insert Cell Kernel Widgets Help — Not Trusted

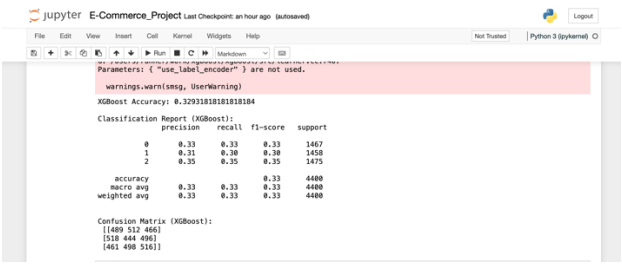
```
SVM Accuracy: 0.3275

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)

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

In [91]:

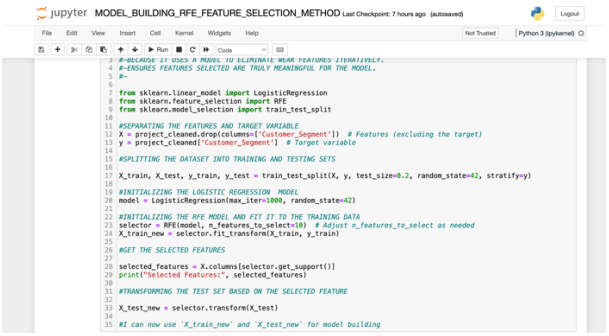
```
#USING XGBOOST



from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Initialize the XGBoost Classifier
xgb_model = XGBClassifier(random_state=42, use_label_encoder=False, eval_metric='mlogloss')

# Step 2: Train the Model
xgb_model.fit(X_train_new, y_train)

# Step 3: Make Predictions
y_pred_xgb = xgb_model.predict(X_test_new)

# Step 4: Evaluate the Model
# Accuracy Score
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print("XGBoost Accuracy:", accuracy_xgb)

# Classification Report
print("\nClassification Report (XGBoost):\n", classification_report(y_test, y_pred_xgb))

# Confusion Matrix
print("\nConfusion Matrix (XGBoost):\n", confusion_matrix(y_test, y_pred_xgb))

#THE ACCURACY OF XGBOOST IS ALSO AROUND 33%
```



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

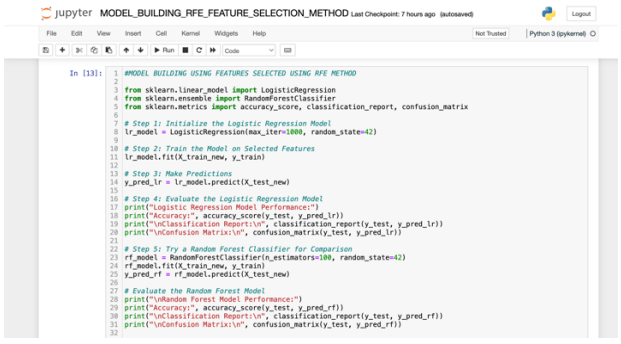
RECURSIVE FEATURE ELIMINTAION METHOD

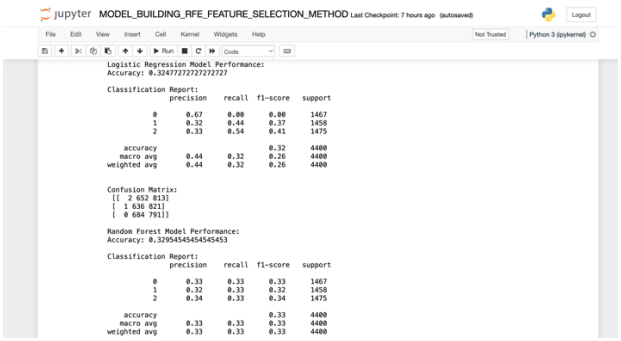
- I'll use this method because it uses a model to eliminate weak features iteratively.
- Also, it ensures features selected are truly meaningful for the model.
- Below is the code I used for feature selection using this method.

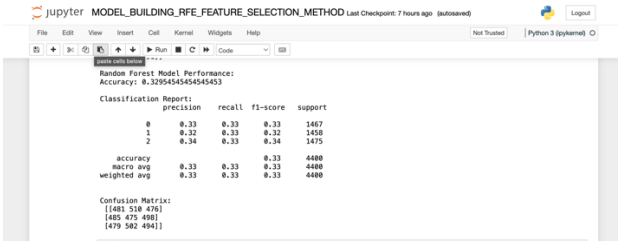



```
Selected Features: Index(['Age', 'Annual_Income', 'Spending_Score', 'Years_as_Customer',
       'Number_of_Orders', 'Average_Order_Value', 'Last_Activity',
       'Income_Spending_Interaction', 'Order_Value_Frequency',
       'Orders_per_Year'],
      dtype='object')
```

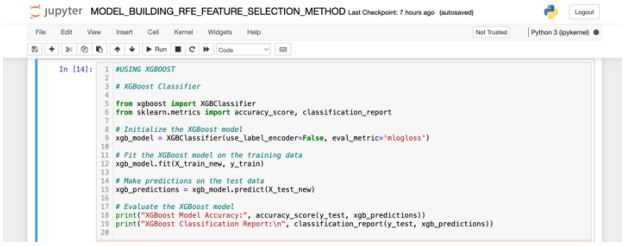
- The features selected through this method are 'Age', 'Spending score', 'Years as customer', 'Number of orders', Average order value', 'Last Activity', 'Income Spending Interaction', 'Order Value Frequency', 'Orders per year'.
- Now, I'll build a model using these features.

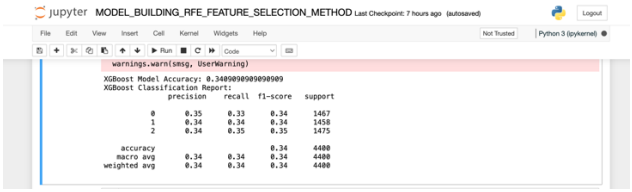


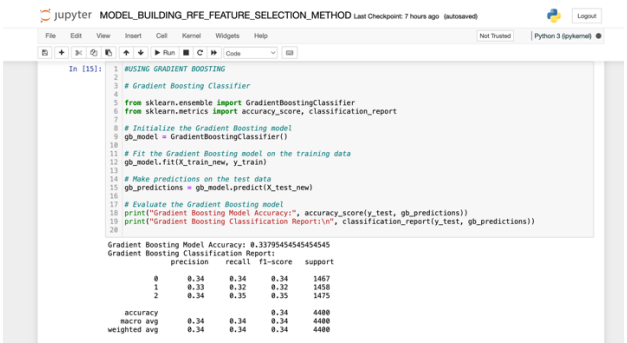




- Since the accuracy using random forest classifier and Logistic regression is 33% only, I'll try other classifiers as well like XGBoost, Gradient Boosting and SVM.



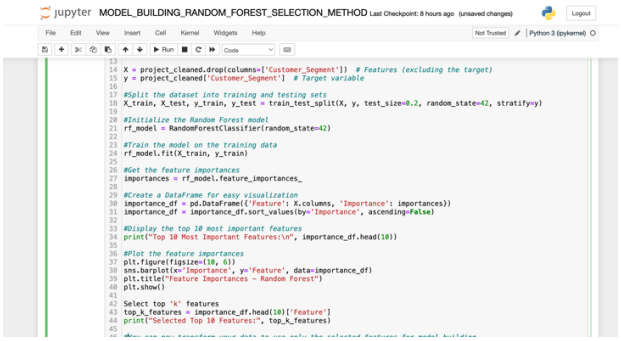


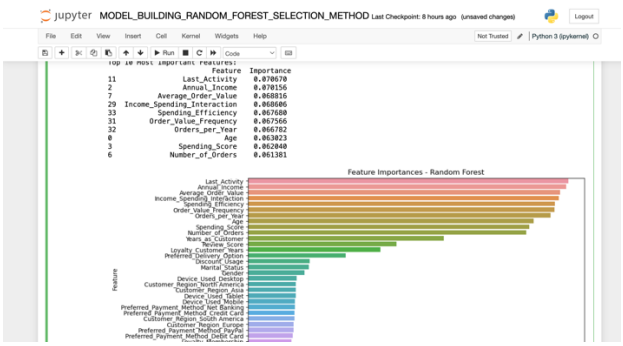


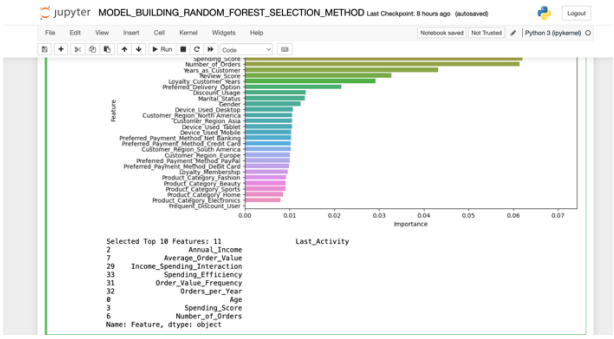
- This shows the accuracy using other classifiers is also somewhere around 33%.

Random Forest Method

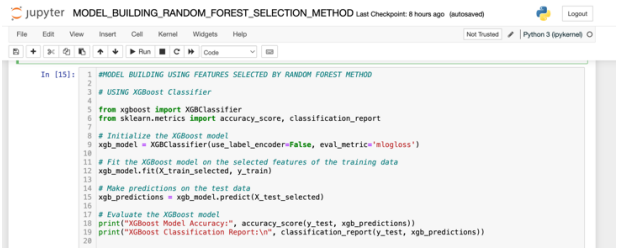
- Tree-based methods like Random Forest are great for classification tasks because they can handle both linear and non-linear relationships between features and the target variable.
- Random Forest naturally ranks features based on their importance which is calculated using how much each feature reduces 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.

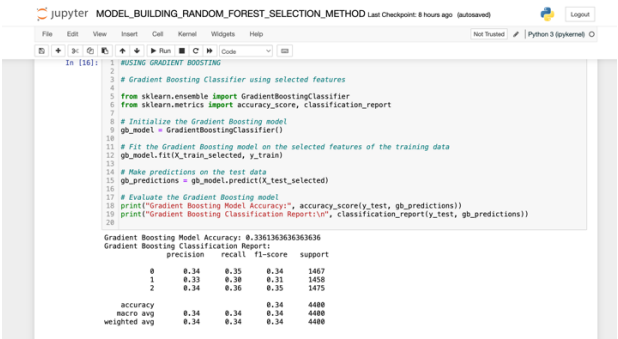




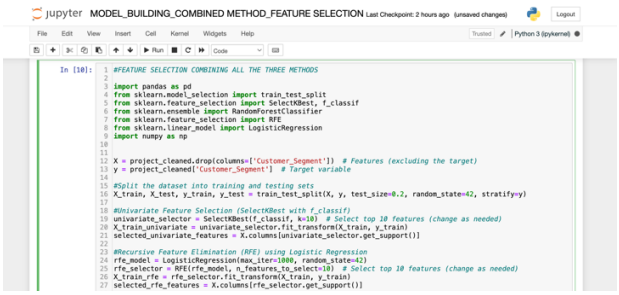


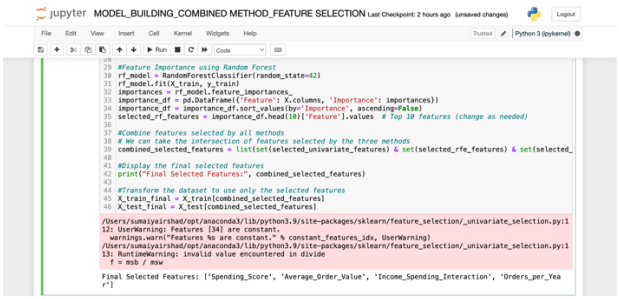
- The features selected using this method are ‘Annual Income’, ‘Average order value’,’Income Spending Interaction’, ‘Spending Efficiency’, ’Order Value Frequency’, ‘Orders per year’, ‘Age’, ‘Spending score’, ‘No of Orders’.
- Now, I’ll build Model using this set of features. Here, Also I’ll use different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



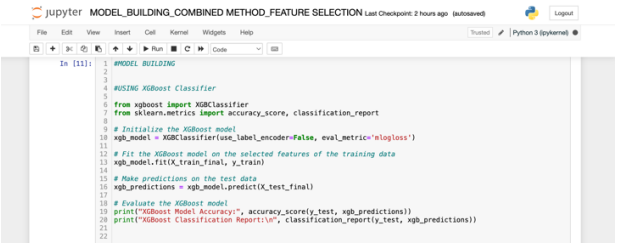


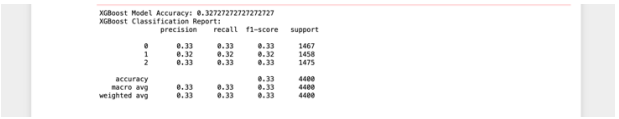
- The accuracy using all of these different classifiers is still 33%.
- Now, I'll combine all these 3 methods and then do feature selection to check if accuracy improves.

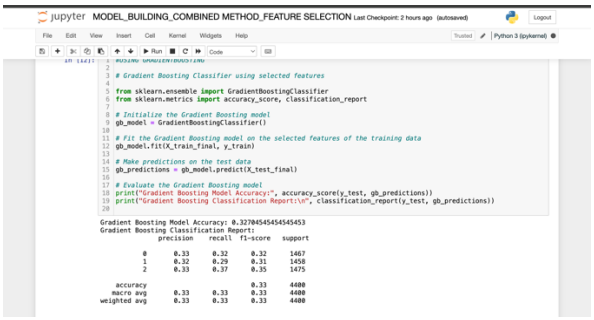


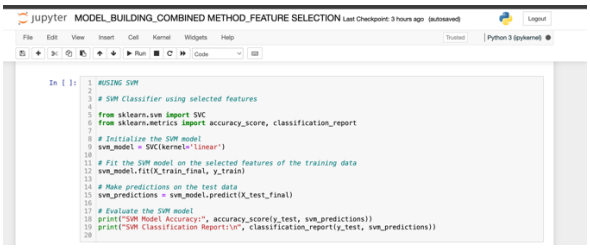


- The selected features using this method are 'Spending Score', 'Average Order Value', 'Income Spending Interaction', 'Orders per year'.
- Now, I'll build a model using these features and see how efficient the model is.Again I'll use three different classifiers, XGBoost, Gradient Boosting and SVM.

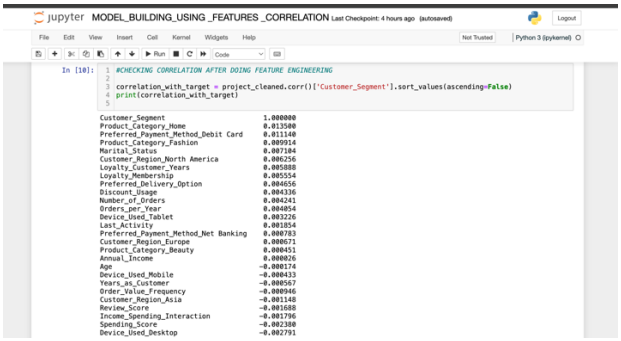


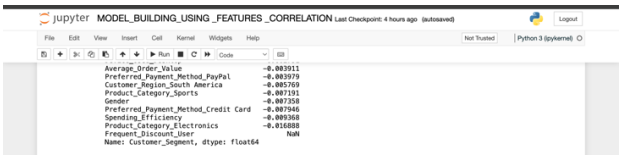




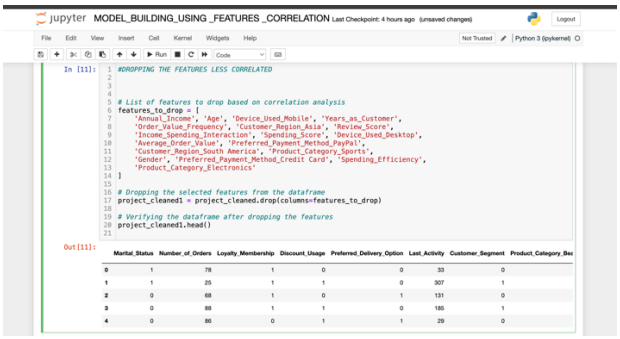


- The accuracy of the model using all the three classifiers is somewhere around 33 % only.
- Now since models build using features selected from different methods still does not have a satisfactory accuracy, I'll try building a model by selecting features through correlation.





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

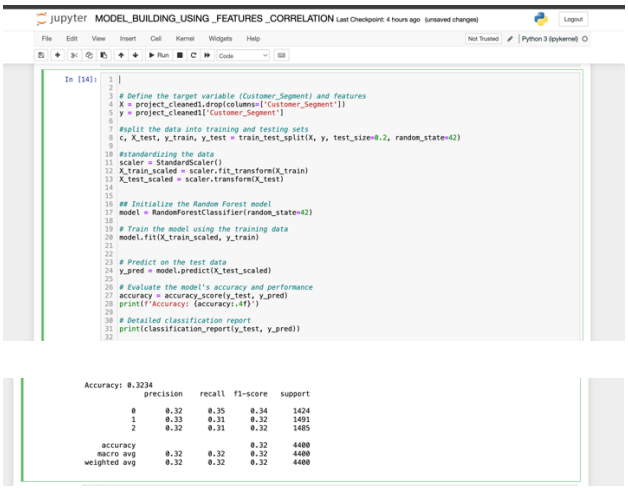


```
In [11]:
#DROPPING THE FEATURES LESS CORRELATED


# List of features to drop based on correlation analysis
features_to_drop = [
    'Annual_Income', 'Age', 'Device_Used_Mobile', 'Years_as_Customer',
    'Order_Value_Frequency', 'Customer_Region_Asia', 'Review_Score',
    'Income_Spending_Interaction', 'Spending_Score', 'Device_Used_Desktop',
    'Average_Order_Value', 'Preferred_Payment_Method_PayPal',
    'Customer_Region_South America', 'Product_Category_Sports',
    'Gender', 'Preferred_Payment_Method_Credit Card', 'Spending_Efficiency',
    'Product_Category_Electronics'
]

# Dropping the selected features from the dataframe
project_cleaned1 = project_cleaned.drop(columns=features_to_drop)

# Verifying the dataframe after dropping the features
project_cleaned1.head()
```

Out[11]:

	Marital_Status	Number_of_Orders	Loyalty_Membership	Discount_Usage	Preferred_Delivery_Option	Last_Activity	Customer_Segment	Product_Category_Bea
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	88	1	1	0	185	1	
4	0	86	0	1	1	29	0	

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



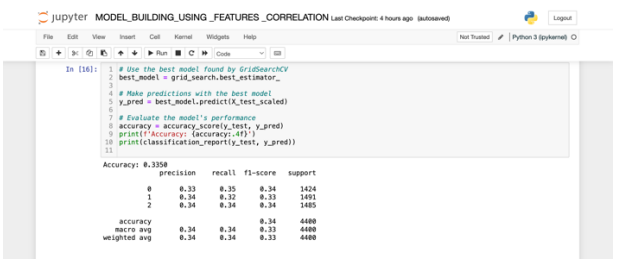
```
In [14]:

# Define the target variable (Customer_Segment) and features
X = project_cleaned1.drop(columns=['Customer_Segment'])
y = project_cleaned1['Customer_Segment']

#split the data into training and testing sets
c, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#standardizing the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


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

# Train the model using the training data
model.fit(X_train_scaled, y_train)


# Predict on the test data
y_pred = model.predict(X_test_scaled)

# Evaluate the model's accuracy and performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')

# Detailed classification report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.3234
              precision    recall  f1-score   support

           0       0.32      0.35      0.34      1424
           1       0.33      0.31      0.32      1491
           2       0.32      0.31      0.32      1485

    accuracy                           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

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import StandardScaler


# Define hyperparameter grid
param_grid = {
    'n_estimators': [100, 200],  # Number of trees in the forest
    'max_depth': [None, 10],        # Maximum depth of the tree
    'min_samples_split': [2, 5],        # Minimum samples required to split a node
    'min_samples_leaf': [1, 2],         # Minimum samples required at a leaf node
    'max_features': ['auto'],  # Number of features to consider when looking for the best split
    'bootstrap': [True, False]          # Whether bootstrap samples are used when building trees
}


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

# Initialize GridSearchCV with Random Forest and the parameter grid
grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=2, scoring='accuracy')

# Train the model with grid search to find the best parameters
grid_search.fit(X_train_scaled, y_train)

# Get the best parameters from the grid search
print("Best Parameters: ", grid_search.best_params_)
```

```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best Parameters:  {'bootstrap': True, 'max_depth': None, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_sample
s_split': 2, 'n_estimators': 200}
```



```
In [16]:
# Use the best model found by GridSearchCV
best_model = grid_search.best_estimator_

# Make predictions with the best model
y_pred = best_model.predict(X_test_scaled)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.4f}')
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.3350
              precision    recall  f1-score   support

           0       0.33      0.35      0.34      1424
           1       0.34      0.32      0.33      1491
           2       0.34      0.34      0.34      1485

    accuracy                           0.34      4400
   macro avg       0.34      0.34      0.33      4400
weighted avg       0.34      0.34      0.33      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.