assigment2

November 9, 2023

Starbucks Promotional Campaign Data Analysis and Customer Segmentation

Introduction

This case study analyizes data provided by Starbucks that simulates their customer demographics and transactional activities during a promotional campaign. The campaign lasted for one month, during which customers received a variaty of offers. The purpose of this case study is to understand customer response to different offers in order to come up with better approaches to sending customers specific promotional deals. Customers are classified into segments based on their transactional activities, so that specific recommendations can be given regarding individual segments to improve customer stickiness, brand awareness and increase revenue in general. Customer segmentation also provides insights on new customer targeting

This code performs data analysis and builds a machine learning model to predict customer loyalty for Starbucks using a dataset. Here's a report summarizing the key steps and findings along with its implementation.

[110]:
```
#importing the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Data: Data is provided by Starbucks and available on Kaggle, and the I have read & displayed the contents on the csv file via pandas module

Data Preprocessing:

The code begins by importing necessary libraries, including NumPy, Pandas, Seaborn, and Matplotlib. Warnings are filtered to suppress warning messages. The dataset, presumably named "Starbucks satisfactory survey.csv," is loaded into a Pandas DataFrame. Column names are renamed to simplify them for ease of analysis.

[111]:
```
data=pd.read_csv("Starbucks satisfactory survey.csv")
data.head()
```

[111]:
```
                    Timestamp 1. Your Gender    2. Your Age  \
0  2019/10/01 12:38:43 PM GMT+8         Female  From 20 to 29
1  2019/10/01 12:38:54 PM GMT+8         Female  From 20 to 29
```

```
2  2019/10/01 12:38:56 PM GMT+8          Male  From 20 to 29
3  2019/10/01 12:39:08 PM GMT+8        Female  From 20 to 29
4  2019/10/01 12:39:20 PM GMT+8          Male  From 20 to 29

  3. Are you currently…? 4. What is your annual income?  \
0                Student            Less than RM25,000
1                Student            Less than RM25,000
2               Employed            Less than RM25,000
3                Student            Less than RM25,000
4                Student            Less than RM25,000

  5. How often do you visit Starbucks? 6. How do you usually enjoy Starbucks?  \
0                               Rarely                               Dine in
1                               Rarely                             Take away
2                              Monthly                               Dine in
3                               Rarely                             Take away
4                              Monthly                             Take away

  7. How much time do you normally  spend during your visit?  \
0                    Between 30 minutes to 1 hour
1                                Below 30 minutes
2                    Between 30 minutes to 1 hour
3                                Below 30 minutes
4                    Between 30 minutes to 1 hour

  8. The nearest Starbucks's outlet to you is…?  \
0                                    within 1km
1                                     1km - 3km
2                                 more than 3km
3                                 more than 3km
4                                     1km - 3km

  9. Do you have Starbucks membership card?  …  \
0                                       Yes  …
1                                       Yes  …
2                                       Yes  …
3                                        No  …
4                                        No  …

  11. On average, how much would you spend at Starbucks per visit?  \
0                                     Less than RM20
1                                     Less than RM20
2                                     Less than RM20
3                                     Less than RM20
4                                 Around RM20 - RM40

  12. How would you rate the quality of Starbucks compared to other brands
```

(Coffee Bean, Old Town White Coffee..) to be: \

0	4
1	4
2	4
3	2
4	3

13. How would you rate the price range at Starbucks? \

0	3
1	3
2	3
3	1
4	3

14. How important are sales and promotions in your purchase decision? \

0	5
1	4
2	4
3	4
4	4

15. How would you rate the ambiance at Starbucks? (lighting, music, etc…) \

0	5
1	4
2	4
3	3
4	2

16. You rate the WiFi quality at Starbucks as.. \

0	4
1	4
2	4
3	3
4	2

17. How would you rate the service at Starbucks? (Promptness, friendliness, etc..) \

0	4
1	5
2	4
3	3
4	3

18. How likely you will choose Starbucks for doing business meetings or hangout with friends? \

0	3

```
1                                                  2
2                                                  3
3                                                  3
4                                                  3

   19. How do you come to hear of promotions at Starbucks? Check all that apply.
\
0  Starbucks Website/Apps;Social Media;Emails;Dea…
1                    Social Media;In Store displays
2                      In Store displays;Billboards
3                 Through friends and word of mouth
4               Starbucks Website/Apps;Social Media

  20. Will you continue buying at Starbucks?
0                                        Yes
1                                        Yes
2                                        Yes
3                                         No
4                                        Yes

[5 rows x 21 columns]
```

[112]:

```
#renaming column names for simplicity
data.columns = ['Timestamp',
              'Gender',
              'Age',
              'Occupation',
              'Annual_Income',
              'Visit_Frequency',
              'Service_preferred',
              'Time_Spent_Frequency',
              'Nearest_Store_Distance',
              'Membership',
              'Frequent_Product',
              'Avg_Money_Spent',
              'Quality_Rating_vs_Other_Brands',
              'Price_Rating',
              'Sales_Promotion_Importance',
              'Ambiance_Rating',
              'WiFi_Rating',
              'Service_Rating',
              'Meetings_hangouts_preference',
              'Promotion_Source',
              'Loyalty'
             ]
data.head()
```

[112]:

```
                    Timestamp  Gender           Age Occupation  \
0  2019/10/01 12:38:43 PM GMT+8  Female  From 20 to 29    Student
1  2019/10/01 12:38:54 PM GMT+8  Female  From 20 to 29    Student
2  2019/10/01 12:38:56 PM GMT+8    Male  From 20 to 29   Employed
3  2019/10/01 12:39:08 PM GMT+8  Female  From 20 to 29    Student
4  2019/10/01 12:39:20 PM GMT+8    Male  From 20 to 29    Student

        Annual_Income Visit_Frequency Service_preferred  \
0  Less than RM25,000          Rarely           Dine in
1  Less than RM25,000          Rarely         Take away
2  Less than RM25,000         Monthly           Dine in
3  Less than RM25,000          Rarely         Take away
4  Less than RM25,000         Monthly         Take away

          Time_Spent_Frequency Nearest_Store_Distance Membership  …  \
0  Between 30 minutes to 1 hour             within 1km        Yes  …
1            Below 30 minutes              1km - 3km        Yes  …
2  Between 30 minutes to 1 hour         more than 3km        Yes  …
3            Below 30 minutes          more than 3km         No  …
4  Between 30 minutes to 1 hour              1km - 3km         No  …

       Avg_Money_Spent Quality_Rating_vs_Other_Brands  Price_Rating  \
0      Less than RM20                              4             3
1      Less than RM20                              4             3
2      Less than RM20                              4             3
3      Less than RM20                              2             1
4  Around RM20 - RM40                              3             3

  Sales_Promotion_Importance  Ambiance_Rating  WiFi_Rating  Service_Rating  \
0                          5                5            4               4
1                          4                4            4               5
2                          4                4            4               4
3                          4                3            3               3
4                          4                2            2               3

  Meetings_hangouts_preference  \
0                            3
1                            2
2                            3
3                            3
4                            3

                                  Promotion_Source Loyalty
0  Starbucks Website/Apps;Social Media;Emails;Dea…     Yes
1                     Social Media;In Store displays     Yes
2                       In Store displays;Billboards     Yes
3                    Through friends and word of mouth      No
```

```
4               Starbucks Website/Apps;Social Media     Yes

[5 rows x 21 columns]
```

[113]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 122 entries, 0 to 121
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Timestamp                       122 non-null    object
 1   Gender                          122 non-null    object
 2   Age                             122 non-null    object
 3   Occupation                      122 non-null    object
 4   Annual_Income                   122 non-null    object
 5   Visit_Frequency                 122 non-null    object
 6   Service_preferred               121 non-null    object
 7   Time_Spent_Frequency            122 non-null    object
 8   Nearest_Store_Distance          122 non-null    object
 9   Membership                      122 non-null    object
 10  Frequent_Product                122 non-null    object
 11  Avg_Money_Spent                 122 non-null    object
 12  Quality_Rating_vs_Other_Brands  122 non-null    int64
 13  Price_Rating                    122 non-null    int64
 14  Sales_Promotion_Importance      122 non-null    int64
 15  Ambiance_Rating                 122 non-null    int64
 16  WiFi_Rating                     122 non-null    int64
 17  Service_Rating                  122 non-null    int64
 18  Meetings_hangouts_preference    122 non-null    int64
 19  Promotion_Source                121 non-null    object
 20  Loyalty                         122 non-null    object
dtypes: int64(7), object(14)
memory usage: 20.1+ KB
```

[114]:
```
data.isna().sum()
#checking for any null values
```

[114]:
```
Timestamp                   0
Gender                      0
Age                         0
Occupation                  0
Annual_Income               0
Visit_Frequency             0
Service_preferred           1
Time_Spent_Frequency        0
Nearest_Store_Distance      0
```

```
Membership                         0
Frequent_Product                   0
Avg_Money_Spent                    0
Quality_Rating_vs_Other_Brands     0
Price_Rating                       0
Sales_Promotion_Importance         0
Ambiance_Rating                    0
WiFi_Rating                        0
Service_Rating                     0
Meetings_hangouts_preference       0
Promotion_Source                   1
Loyalty                            0
dtype: int64
```

[115]:
```
# since Service_preferred  & Promotion_Source  columns have only 1 row as null␣
↪delete the row!!
data=data[-data.Service_preferred.isnull()]
data=data[-data.Promotion_Source .isnull()]
```

[116]:
```
data.isna().sum()
#checking for NAN values in the data
```

[116]:
```
Timestamp                          0
Gender                             0
Age                                0
Occupation                         0
Annual_Income                      0
Visit_Frequency                    0
Service_preferred                  0
Time_Spent_Frequency               0
Nearest_Store_Distance             0
Membership                         0
Frequent_Product                   0
Avg_Money_Spent                    0
Quality_Rating_vs_Other_Brands     0
Price_Rating                       0
Sales_Promotion_Importance         0
Ambiance_Rating                    0
WiFi_Rating                        0
Service_Rating                     0
Meetings_hangouts_preference       0
Promotion_Source                   0
Loyalty                            0
dtype: int64
```

[117]:
```
data.describe()
```

[117]:

```
       Quality_Rating_vs_Other_Brands  Price_Rating  \
count                      121.000000    121.000000
mean                         3.685950      2.909091
std                          0.913173      1.072381
min                          1.000000      1.000000
25%                          3.000000      2.000000
50%                          4.000000      3.000000
75%                          4.000000      4.000000
max                          5.000000      5.000000

       Sales_Promotion_Importance  Ambiance_Rating  WiFi_Rating  \
count                  121.000000       121.000000   121.000000
mean                     3.818182         3.760331     3.256198
std                      1.064581         0.931171     0.962020
min                      1.000000         1.000000     1.000000
25%                      3.000000         3.000000     3.000000
50%                      4.000000         4.000000     3.000000
75%                      5.000000         4.000000     4.000000
max                      5.000000         5.000000     5.000000

       Service_Rating  Meetings_hangouts_preference
count      121.000000                    121.000000
mean         3.752066                      3.520661
std          0.829468                      1.033595
min          1.000000                      1.000000
25%          3.000000                      3.000000
50%          4.000000                      4.000000
75%          4.000000                      4.000000
max          5.000000                      5.000000
```

[118]:

```
data.isna().sum()
```

[118]:

```
Timestamp                         0
Gender                            0
Age                               0
Occupation                        0
Annual_Income                     0
Visit_Frequency                   0
Service_preferred                 0
Time_Spent_Frequency              0
Nearest_Store_Distance            0
Membership                        0
Frequent_Product                  0
Avg_Money_Spent                   0
Quality_Rating_vs_Other_Brands    0
Price_Rating                      0
Sales_Promotion_Importance        0
```

```
Ambiance_Rating                     0
WiFi_Rating                         0
Service_Rating                      0
Meetings_hangouts_preference        0
Promotion_Source                    0
Loyalty                             0
dtype: int64
```

Data Exploration:

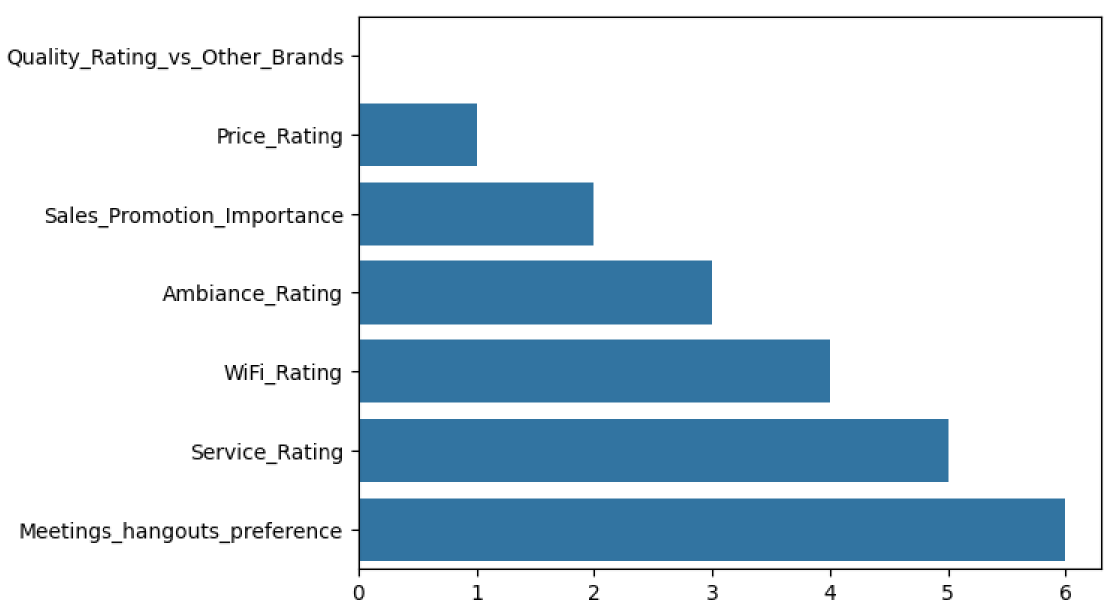
After renaming columns, the code checks for missing values using data.isna().sum(). It's observed that 'Service_preferred' and 'Promotion_Source' columns contain some null values, which are then removed using boolean indexing. The code utilizes Seaborn to create visualizations for data exploration. Numerical columns are visualized with bar plots and box plots to detect outliers. Categorical columns are analyzed using count plots to understand their distribution. Count plots are generated for each categorical variable to explore how they relate to customer loyalty. This helps identify trends and patterns.

[119]:
```
num_cols = data.select_dtypes(include='int64').columns
num_cols
#grouping numerical columns
```

[119]:
```
Index(['Quality_Rating_vs_Other_Brands', 'Price_Rating',
       'Sales_Promotion_Importance', 'Ambiance_Rating', 'WiFi_Rating',
       'Service_Rating', 'Meetings_hangouts_preference'],
      dtype='object')
```

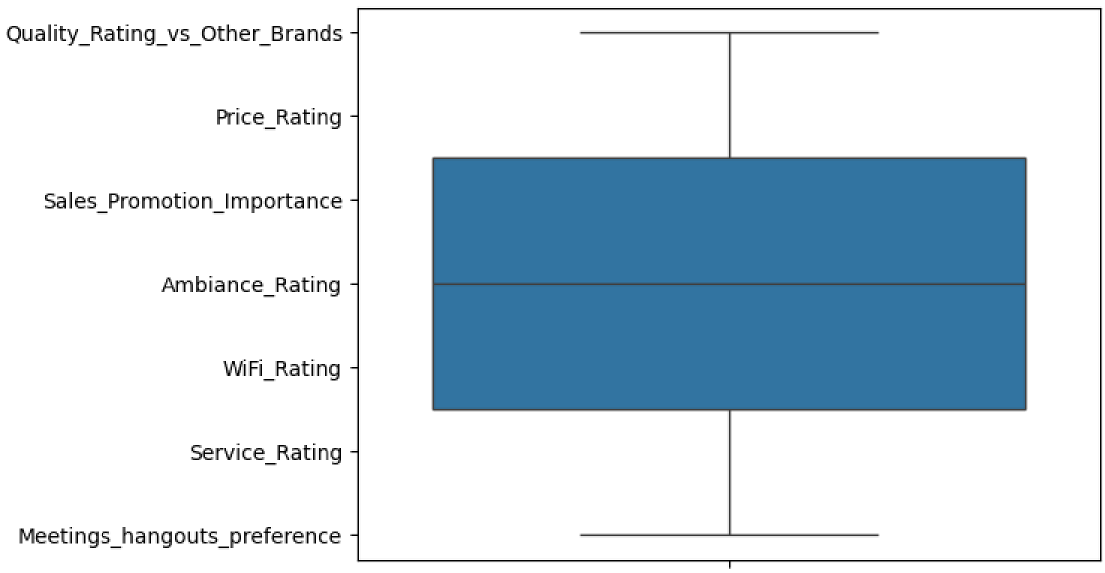
[120]:
```
sns.barplot(num_cols)
```

[120]: <Axes: >



[121]:
```
sns.boxplot(num_cols)
#used to check for any outliers present or not
```

[121]: <Axes: >

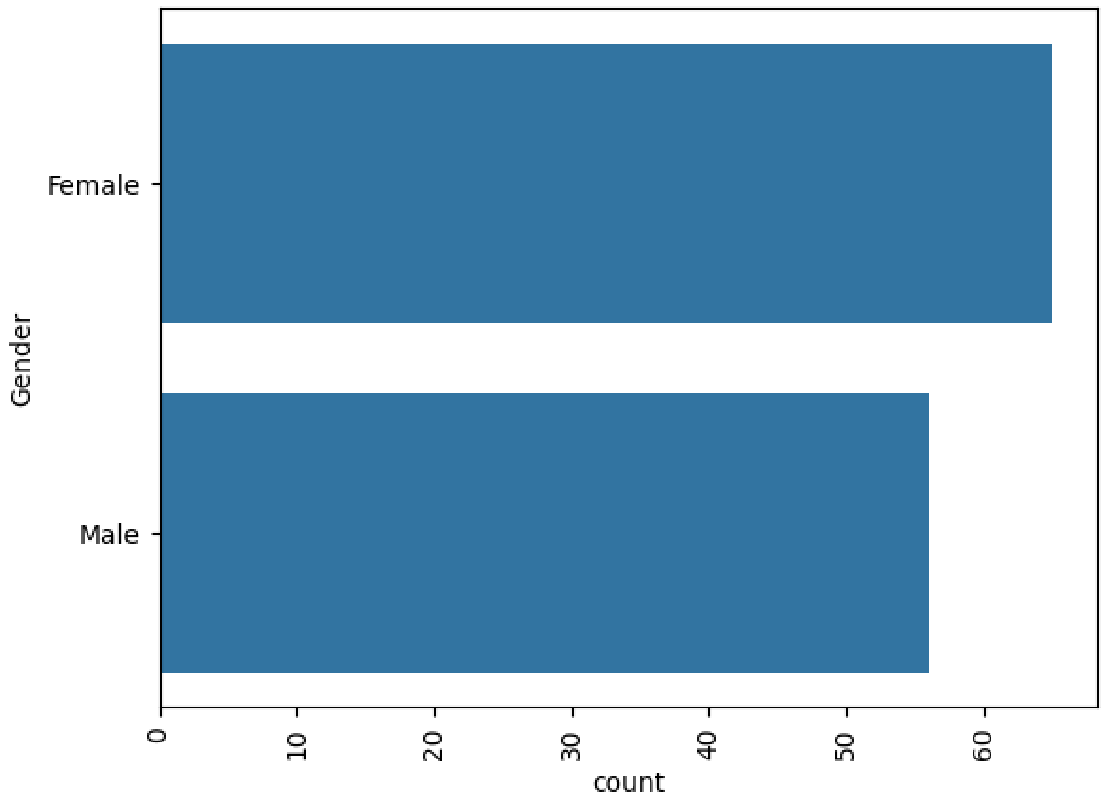


[122]:
```
cat_cols = data.select_dtypes(include='object').columns
cat_cols
#grouping categorical data columns
```

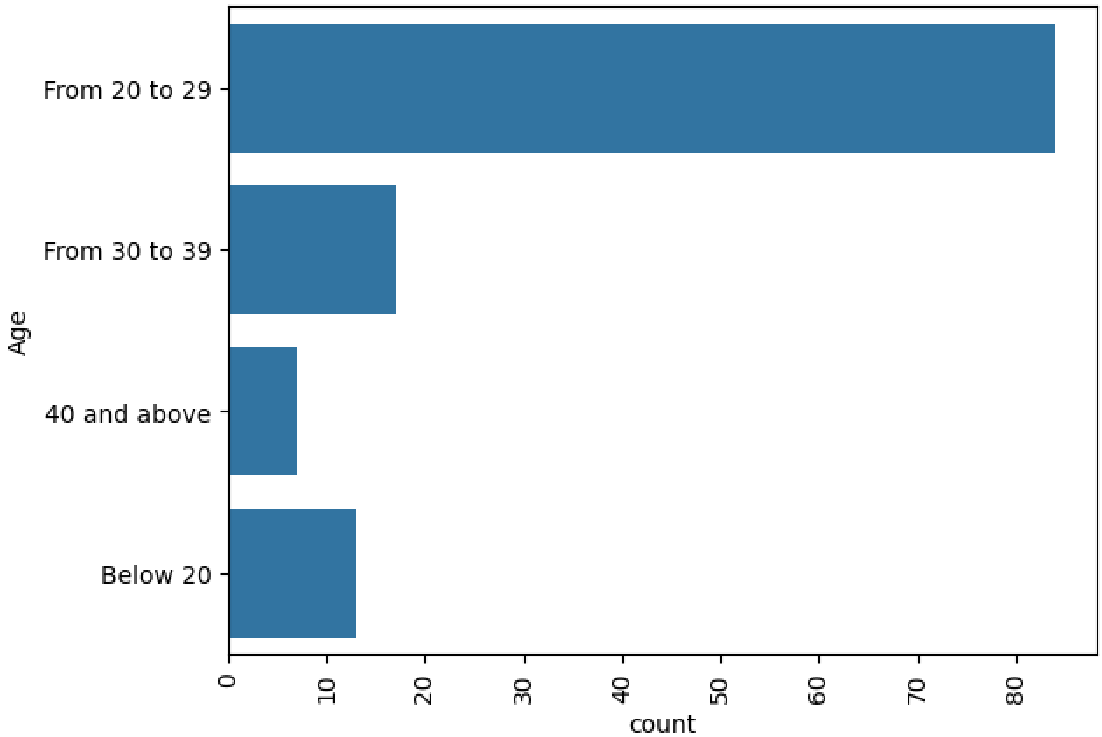
[122]:
```
Index(['Timestamp', 'Gender', 'Age', 'Occupation', 'Annual_Income',
       'Visit_Frequency', 'Service_preferred', 'Time_Spent_Frequency',
       'Nearest_Store_Distance', 'Membership', 'Frequent_Product',
       'Avg_Money_Spent', 'Promotion_Source', 'Loyalty'],
      dtype='object')
```

[123]:
```
for i in cat_cols[1:]:
    print("CountPlot for the column: "+ i)
    sns.countplot(data[i])
    plt.xticks(rotation=90)
    plt.show()
```

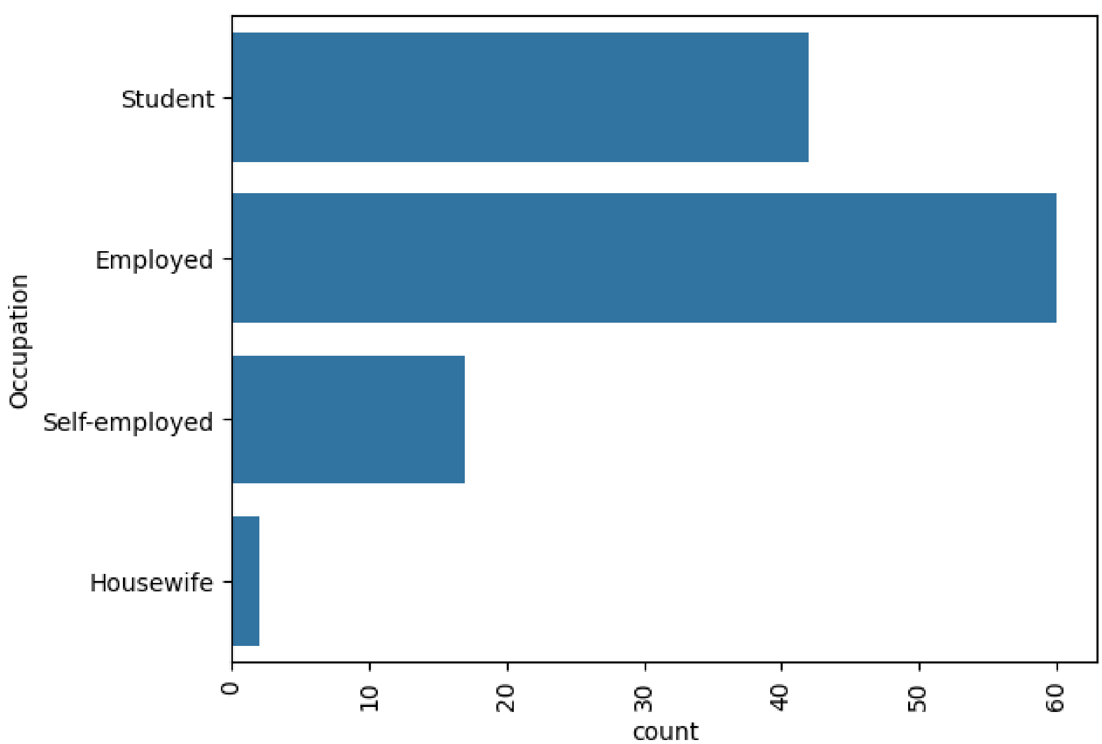
```
CountPlot for the column: Gender
```



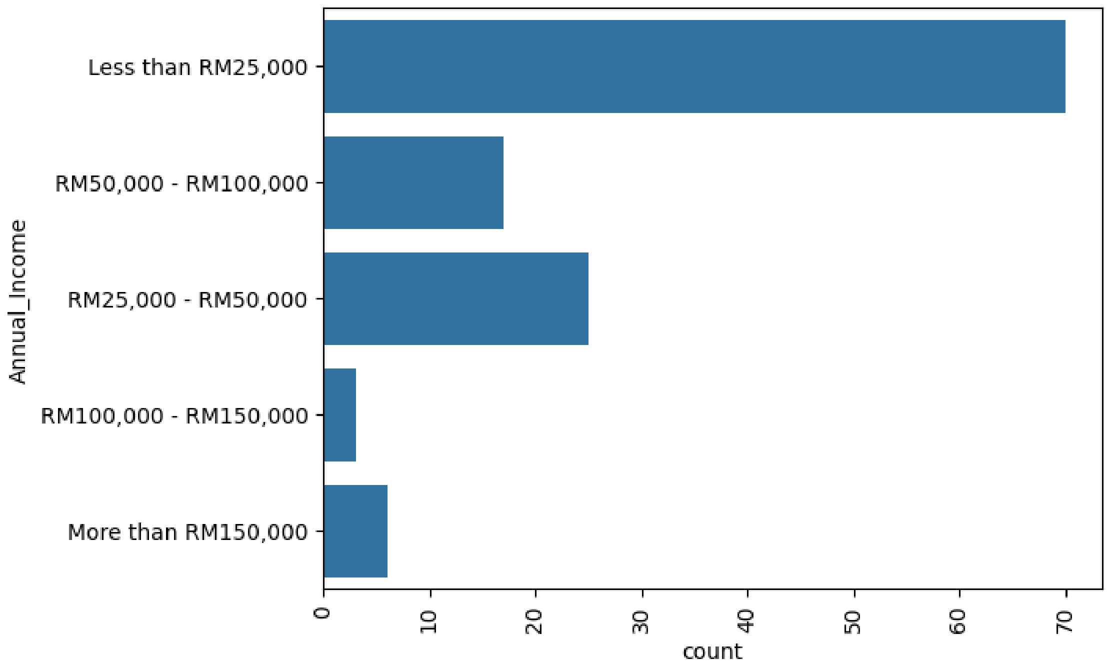
CountPlot for the column: Age



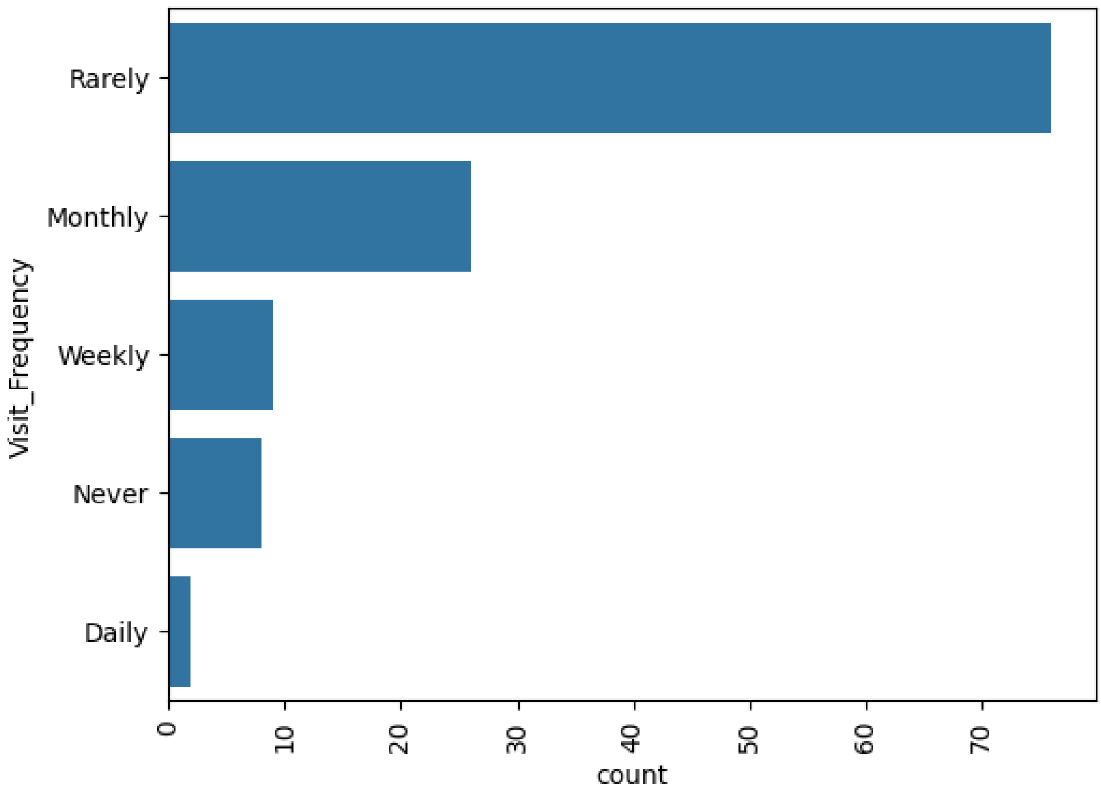
CountPlot for the column: Occupation



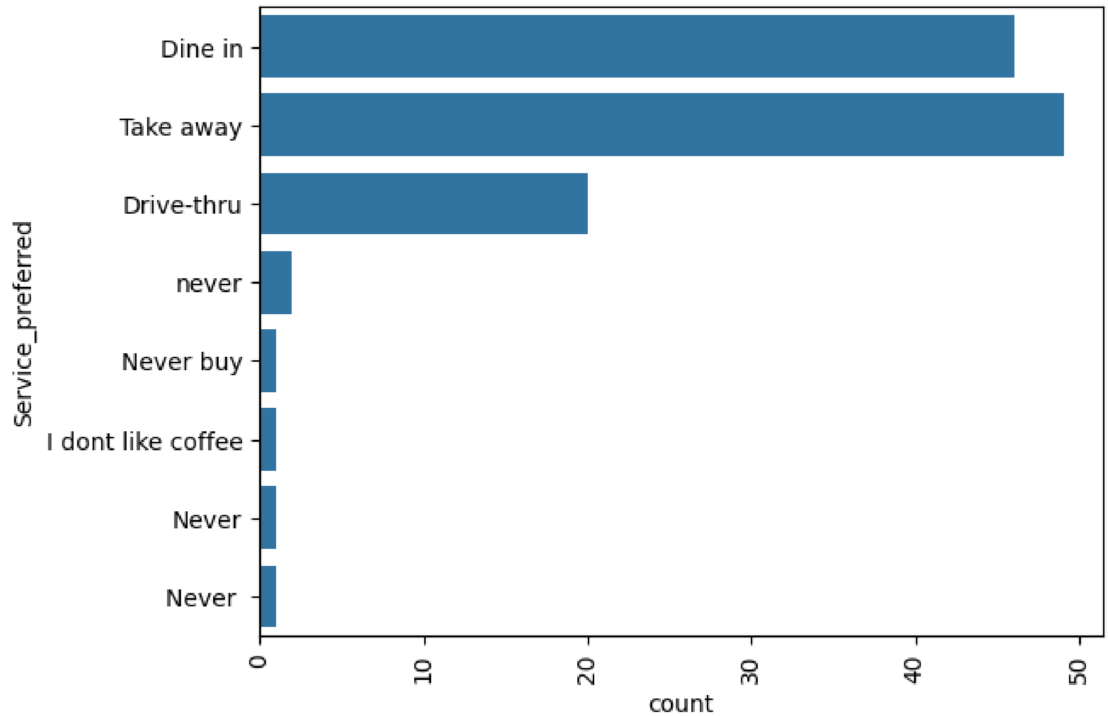
CountPlot for the column: Annual_Income



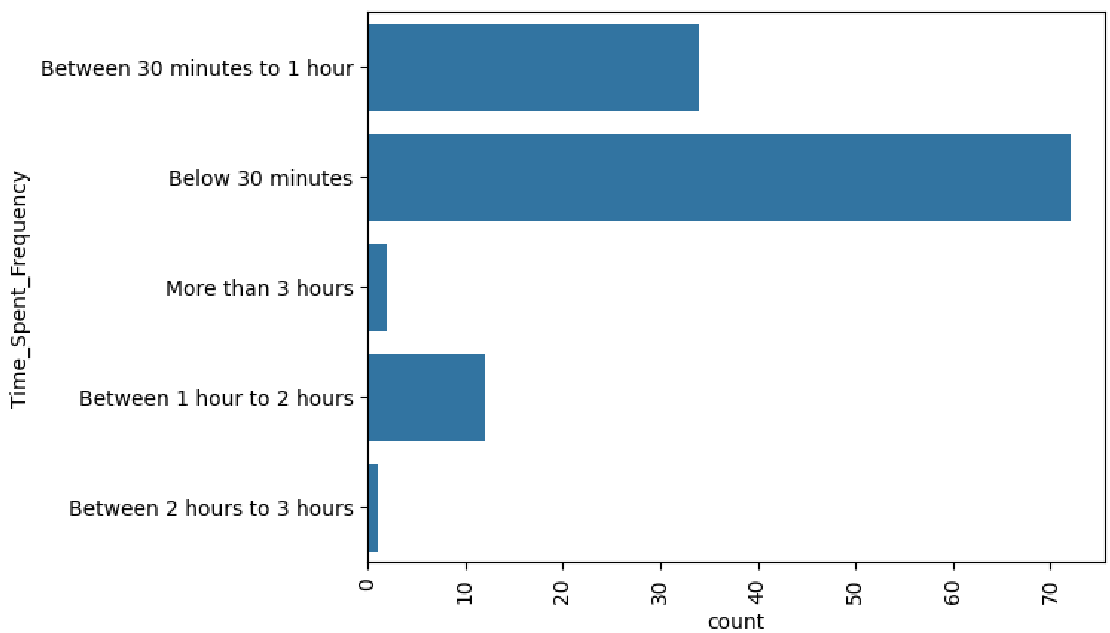
CountPlot for the column: Visit_Frequency



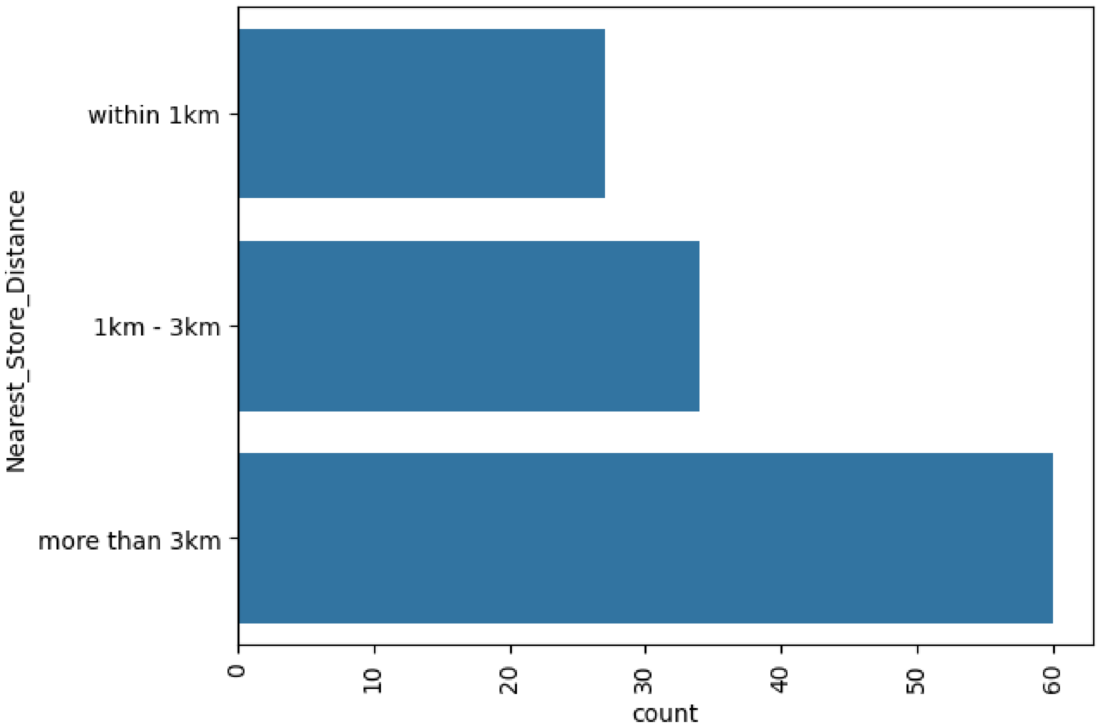
CountPlot for the column: Service_preferred



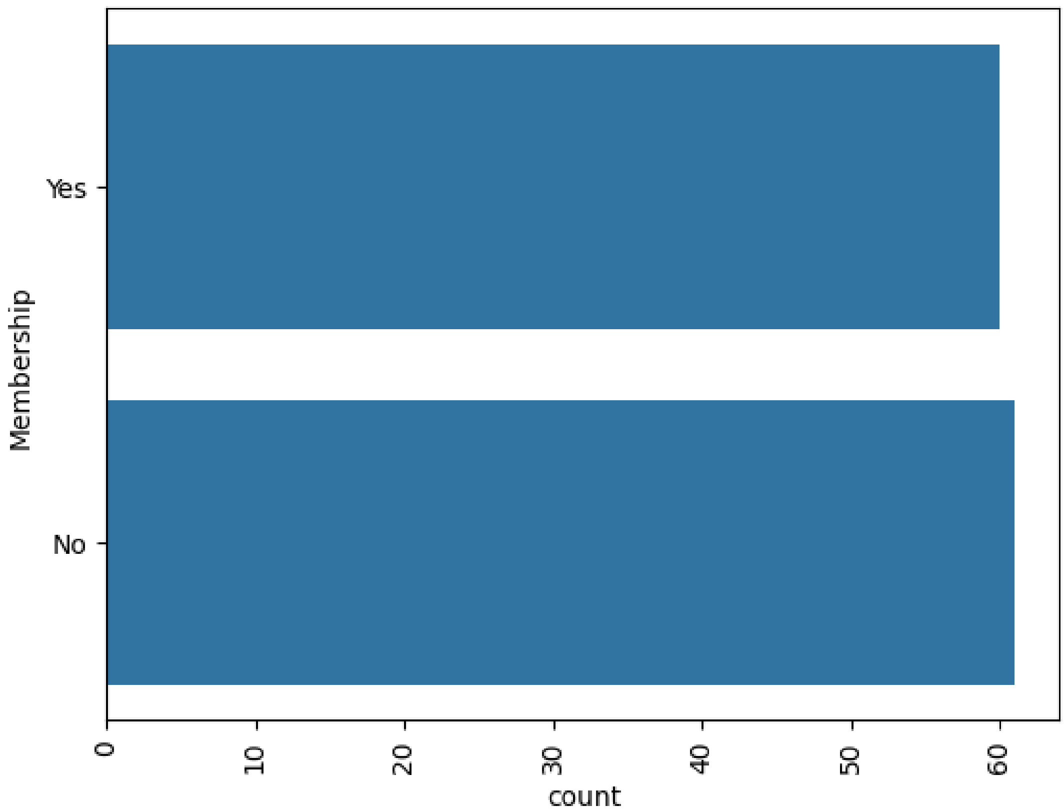
CountPlot for the column: Time_Spent_Frequency



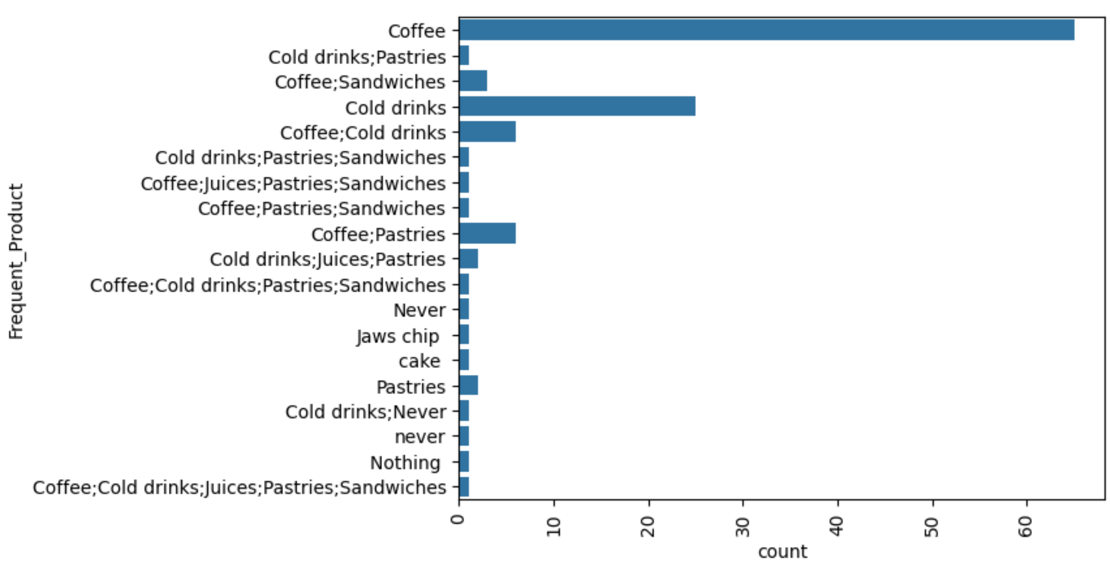
CountPlot for the column: Nearest_Store_Distance



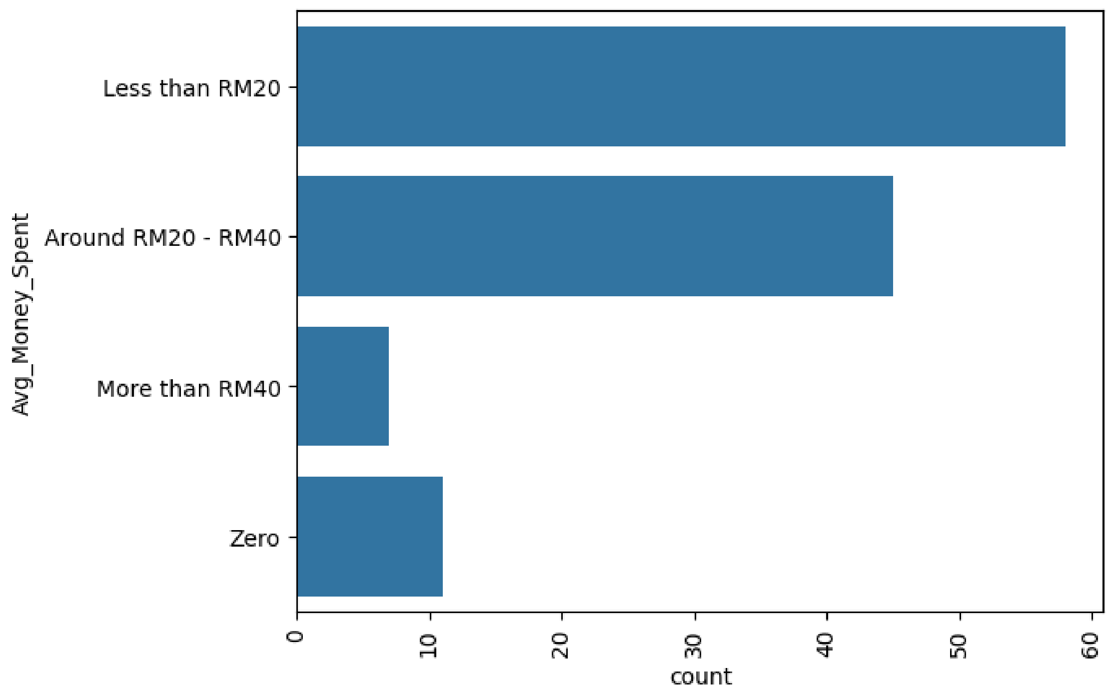
`CountPlot for the column: Membership`



CountPlot for the column: Frequent_Product

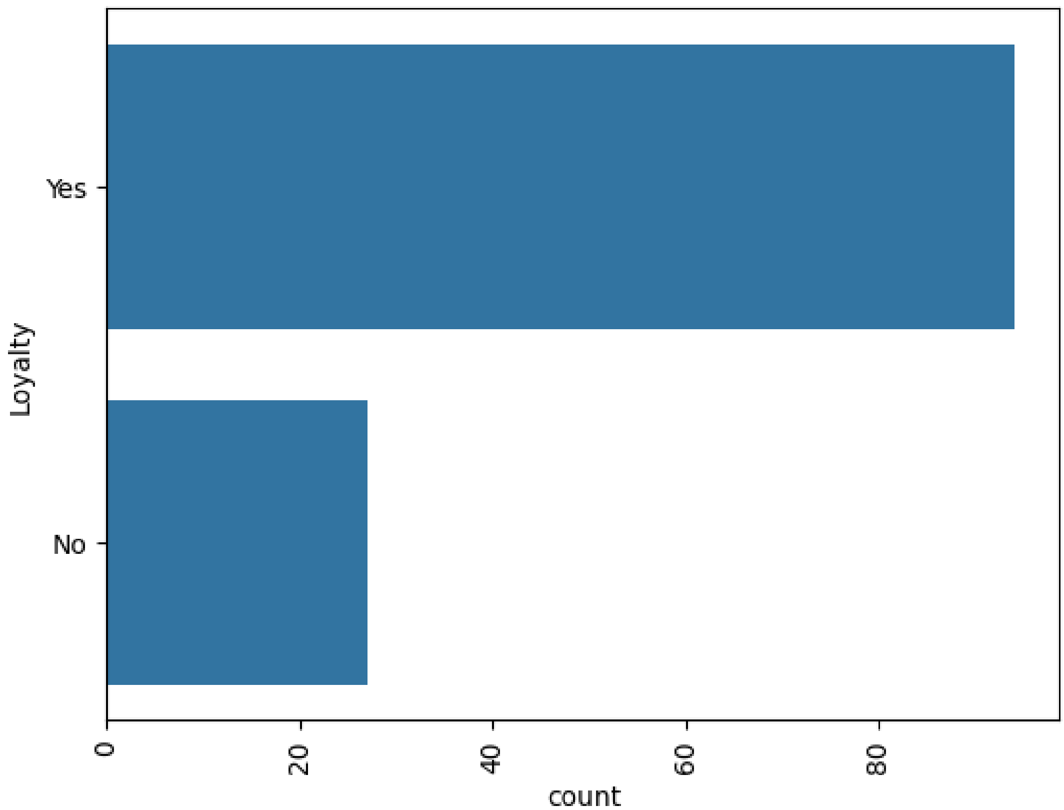


CountPlot for the column: Avg_Money_Spent

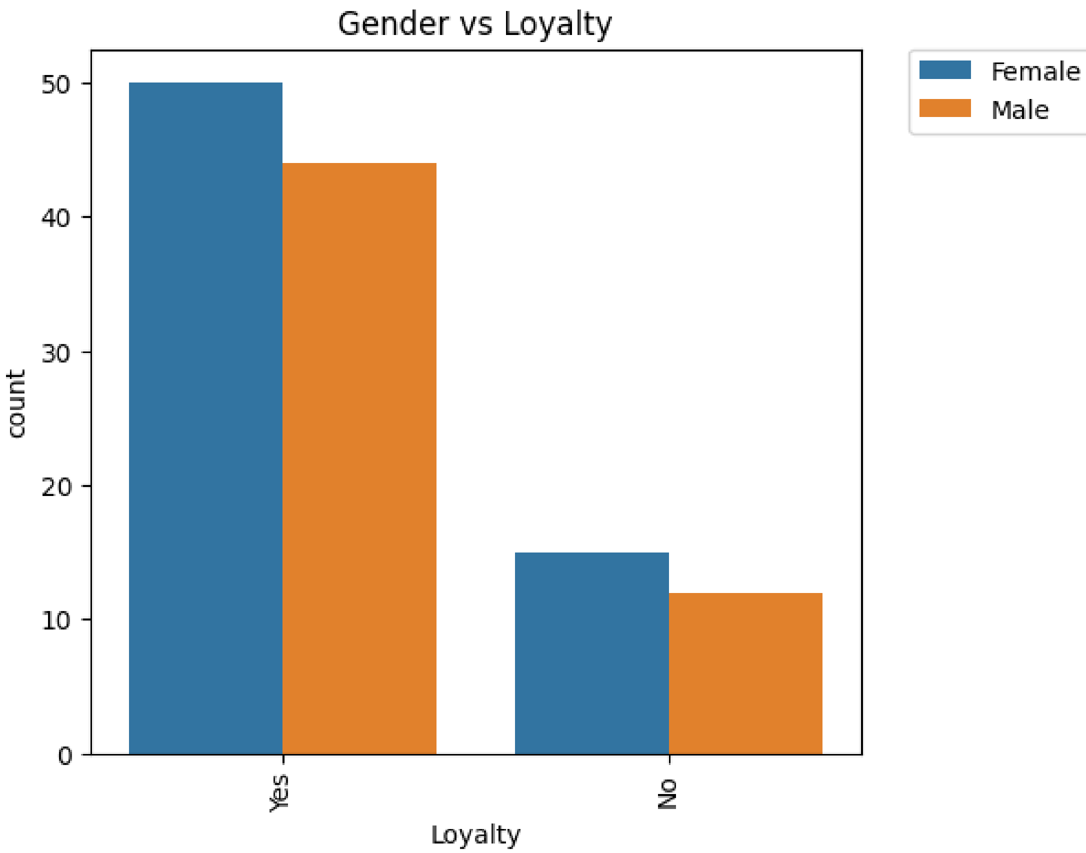


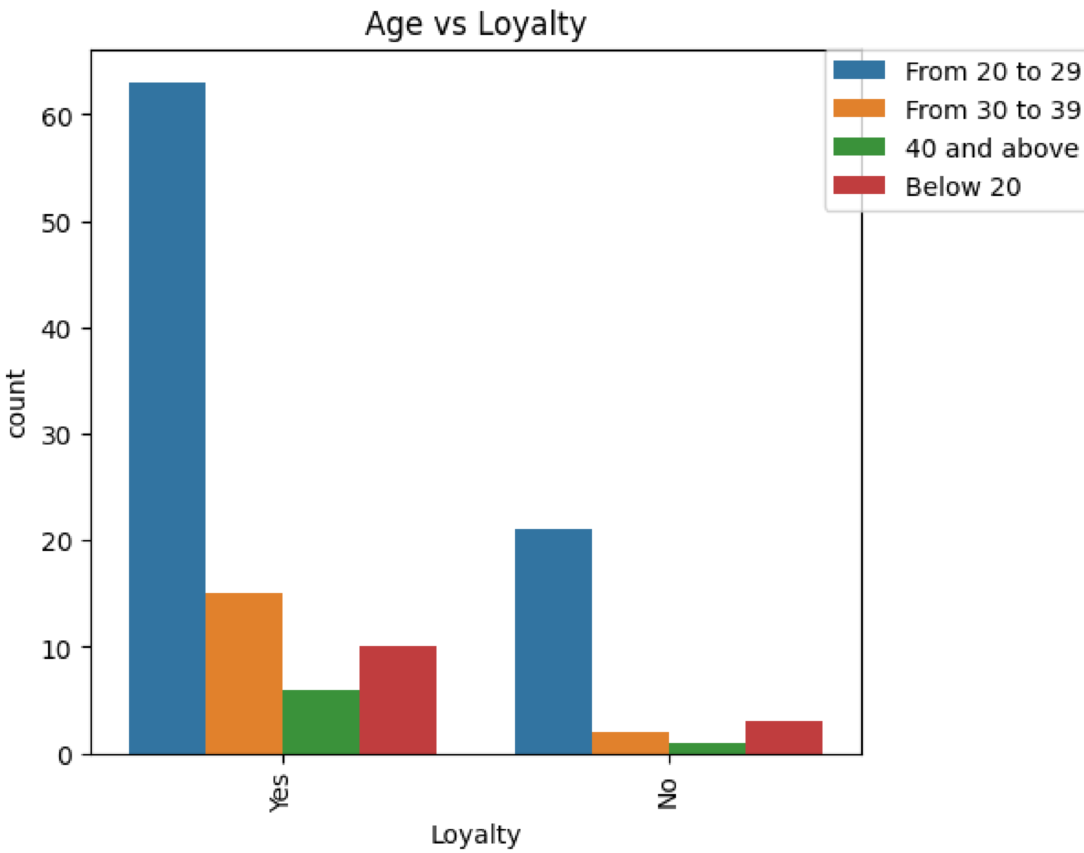
CountPlot for the column: Promotion_Source

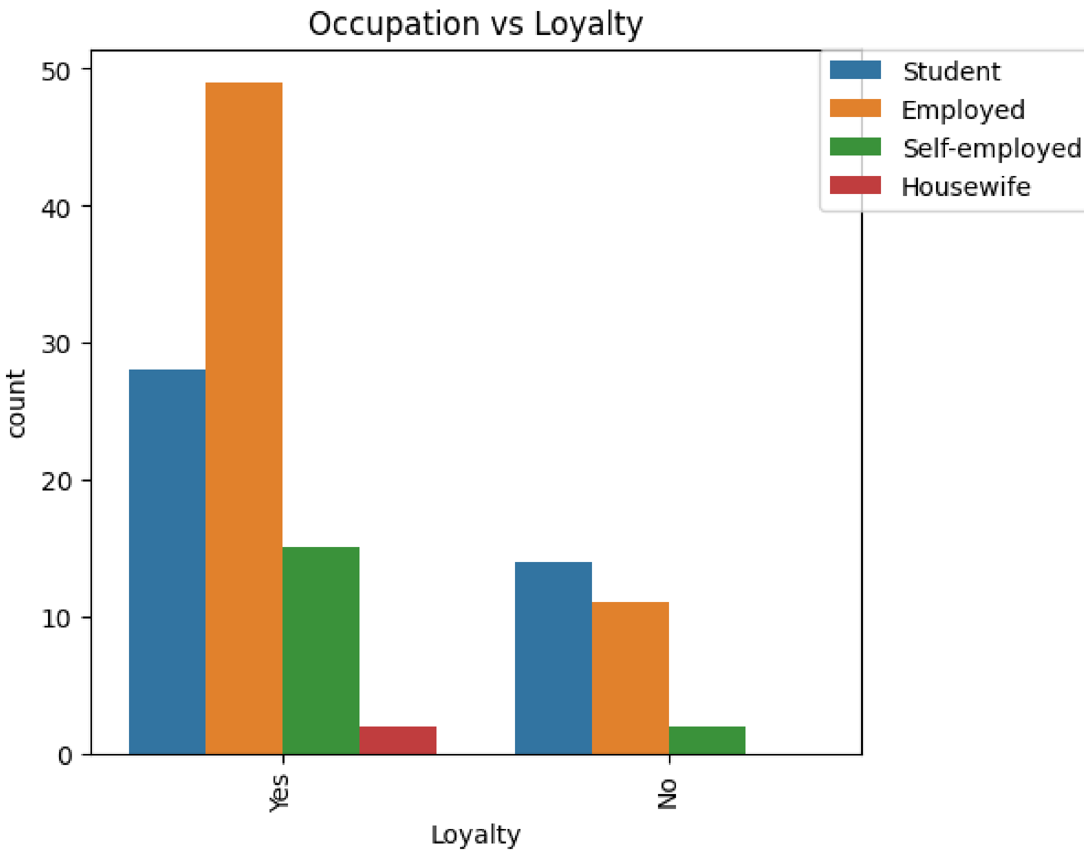
CountPlot for the column: Loyalty

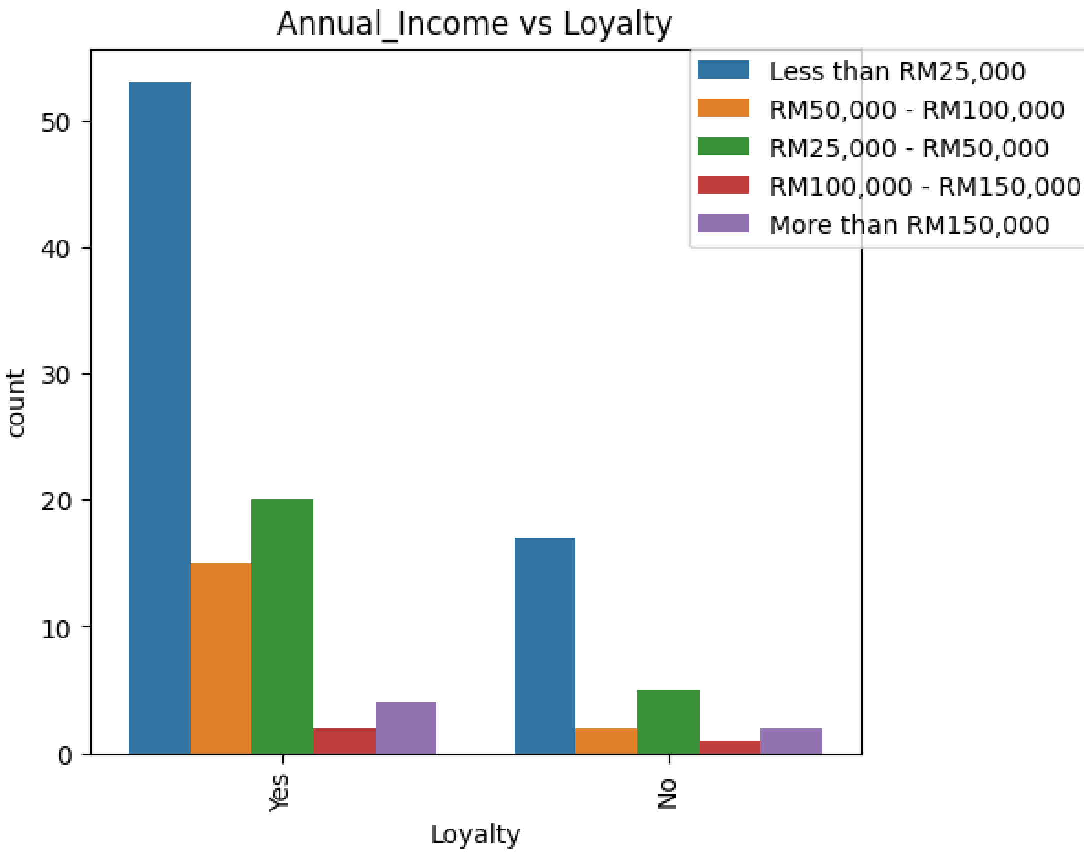


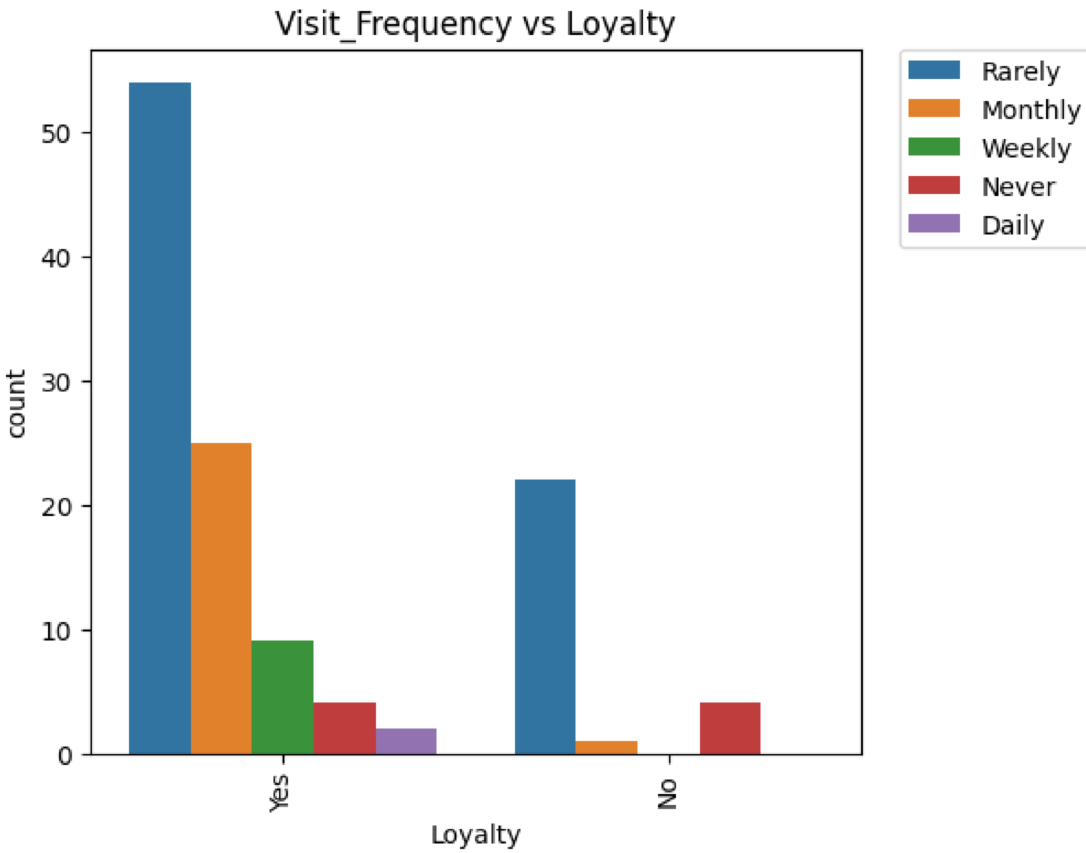
[124]:
```
for i in cat_cols[1:]:
    plt.figure(figsize=(25,5))
    plt.subplot(1,4,1)
    sns.countplot(x=data.Loyalty, hue=data[i])
    plt.title(i+" vs Loyalty")
    plt.xticks(rotation=90)
    plt.legend(bbox_to_anchor=(1.3,1), borderaxespad=0)
    plt.show()
```

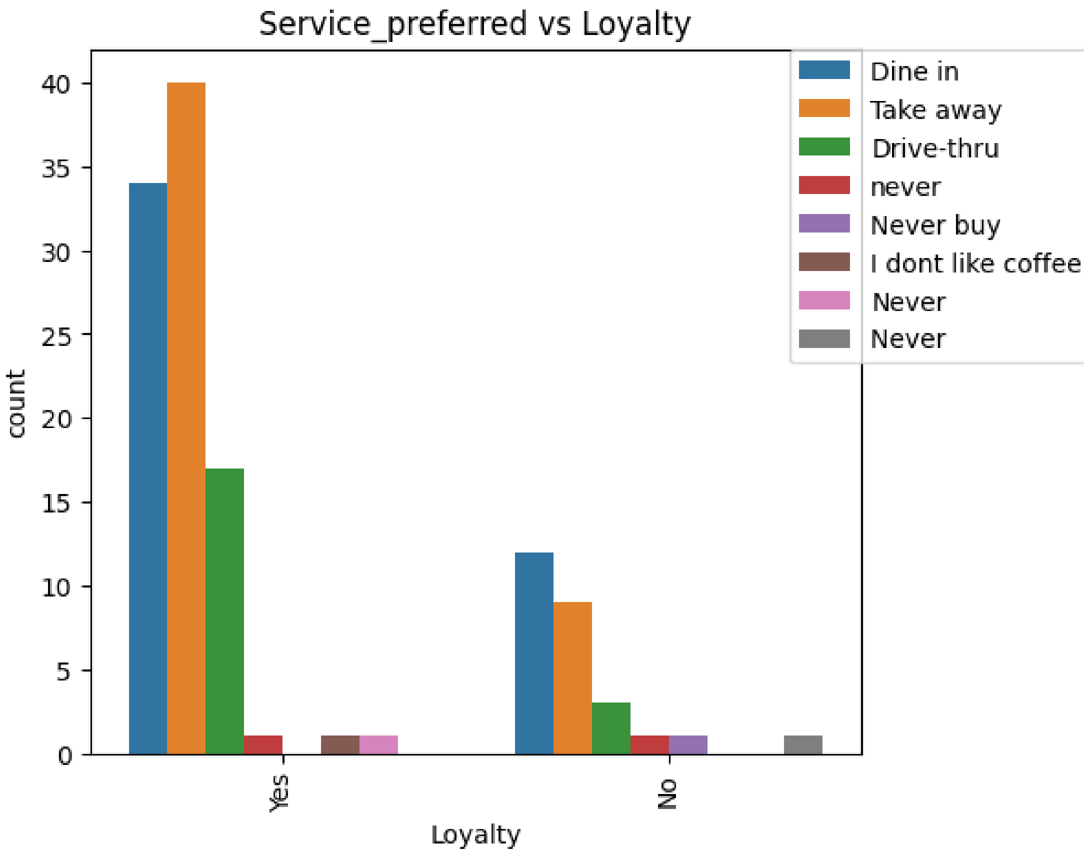

Gender vs Loyalty
Female
Male
count
50
40
30
20
10
0
Yes
No
Loyalty

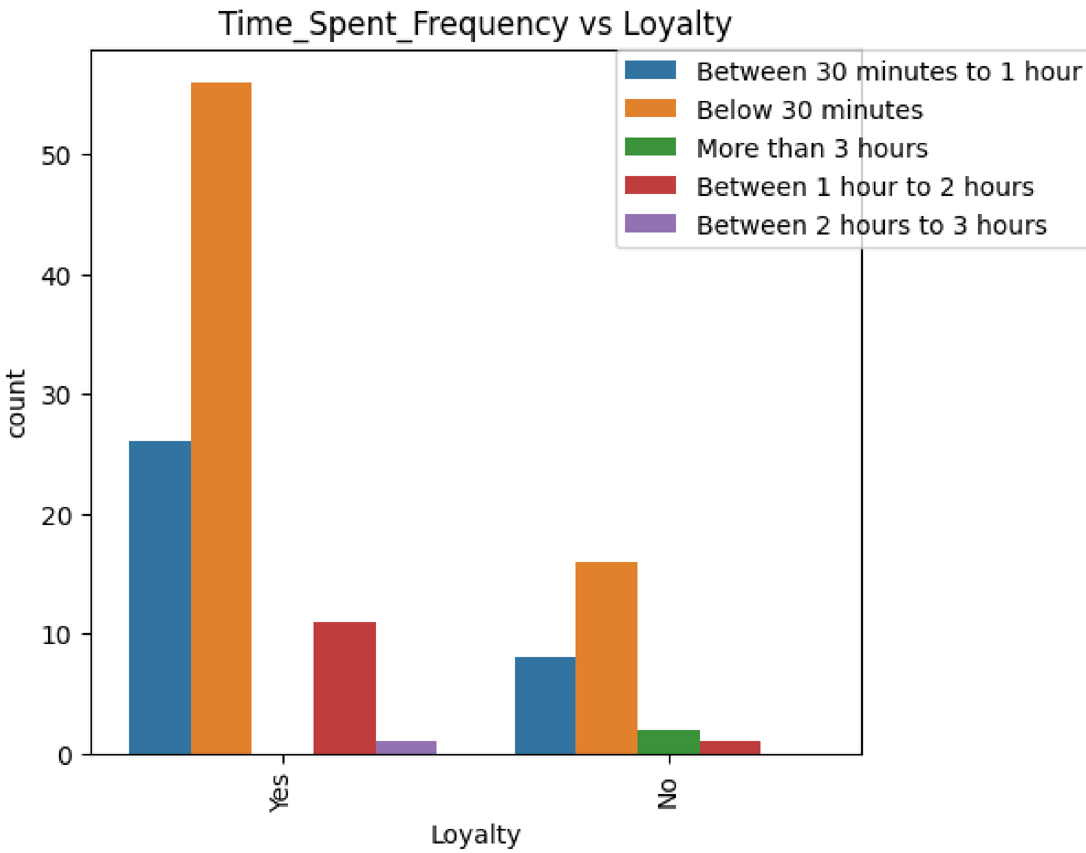

Age vs Loyalty
From 20 to 29
From 30 to 39
40 and above
Below 20
count
0
10
20
30
40
50
60
Yes
No
Loyalty

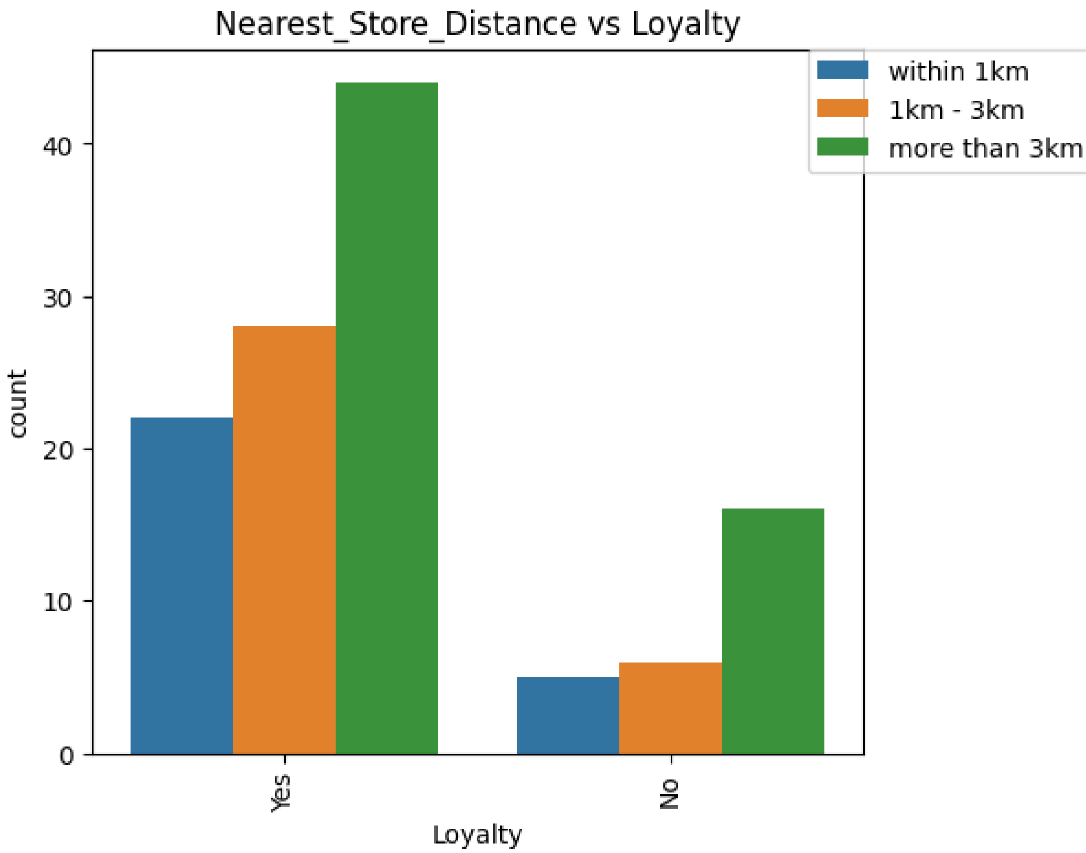

Occupation vs Loyalty
50
40
30
20
10
0
count
Student
Employed
Self-employed
Housewife
Yes
No
Loyalty

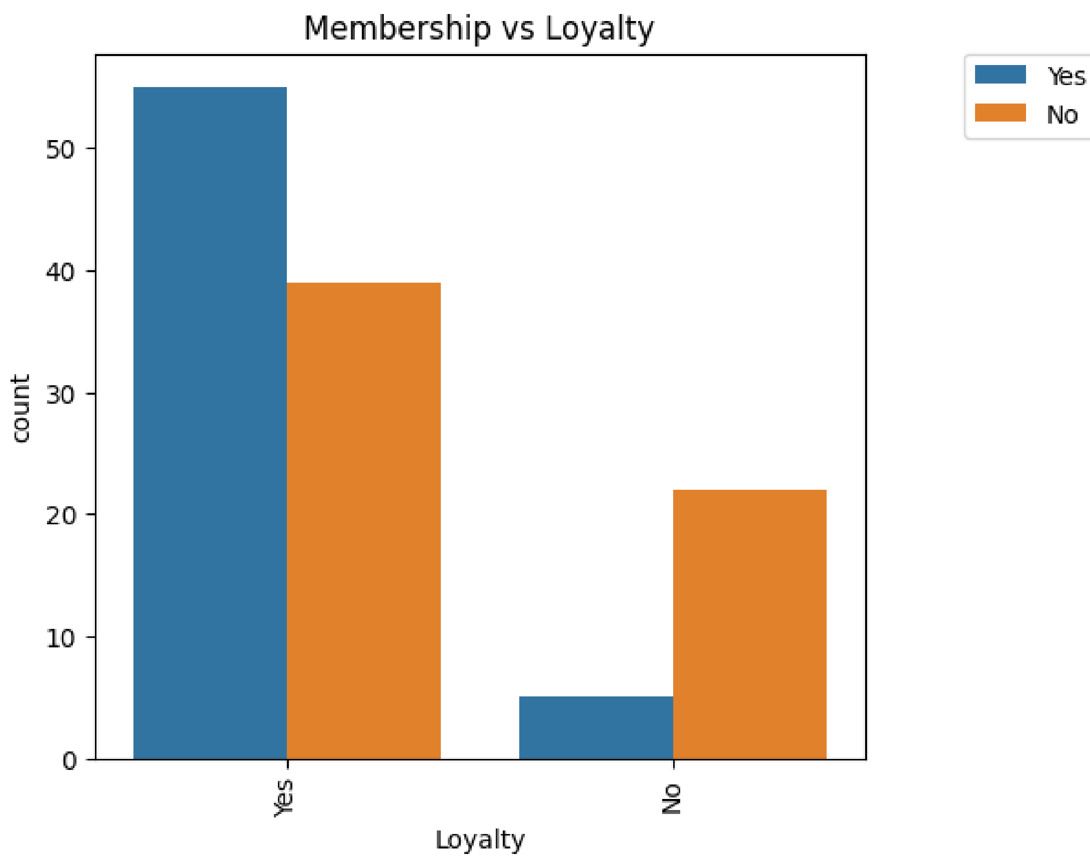

Annual_Income vs Loyalty
Less than RM25,000
RM50,000 - RM100,000
RM25,000 - RM50,000
RM100,000 - RM150,000
More than RM150,000
count
0
10
20
30
40
50
Yes
No
Loyalty

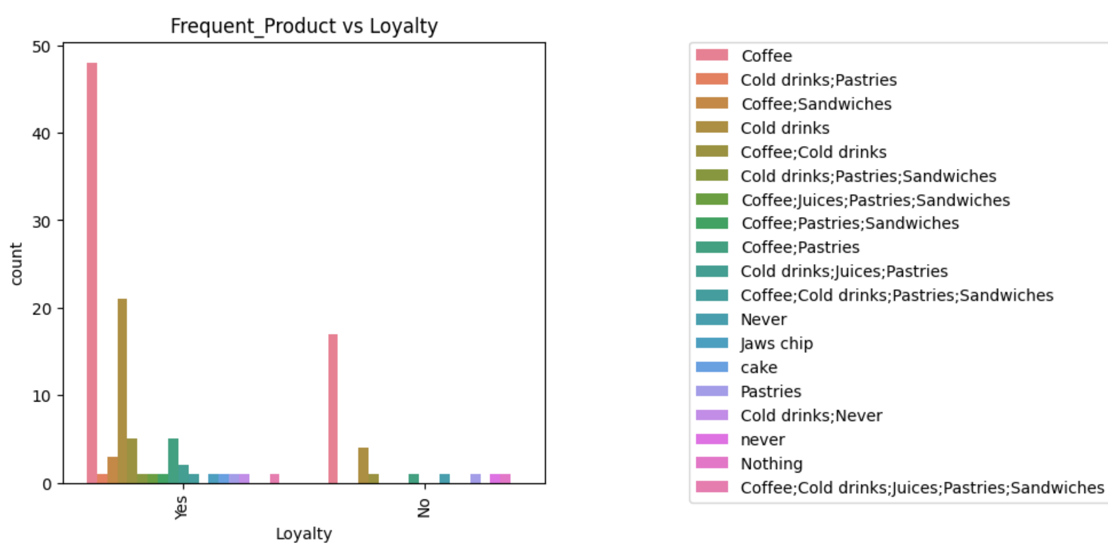

Visit_Frequency vs Loyalty
count
0
10
20
30
40
50
Yes
No
Loyalty
Rarely
Monthly
Weekly
Never
Daily

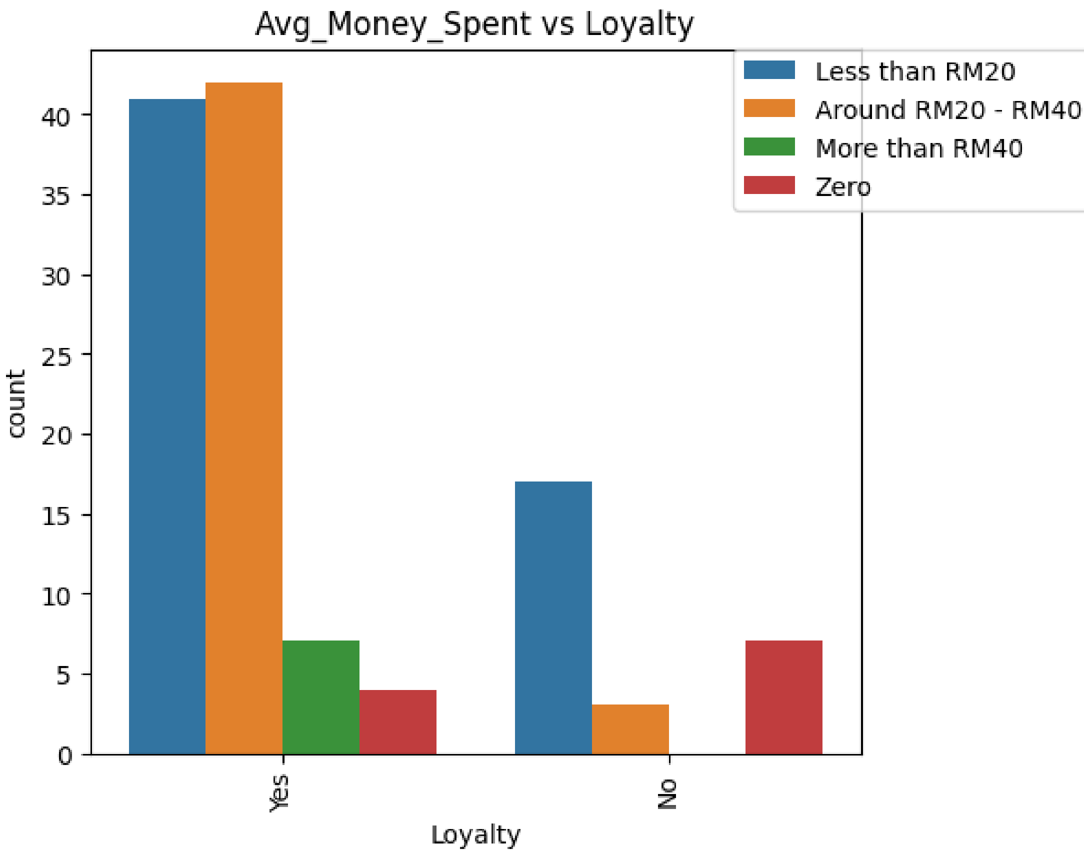

Service_preferred vs Loyalty
count
0
5
10
15
20
25
30
35
40
Yes
No
Loyalty
Dine in
Take away
Drive-thru
never
Never buy
I dont like coffee
Never
Never


Time_Spent_Frequency vs Loyalty
Between 30 minutes to 1 hour
Below 30 minutes
More than 3 hours
Between 1 hour to 2 hours
Between 2 hours to 3 hours
count
0
10
20
30
40
50
Yes
No
Loyalty

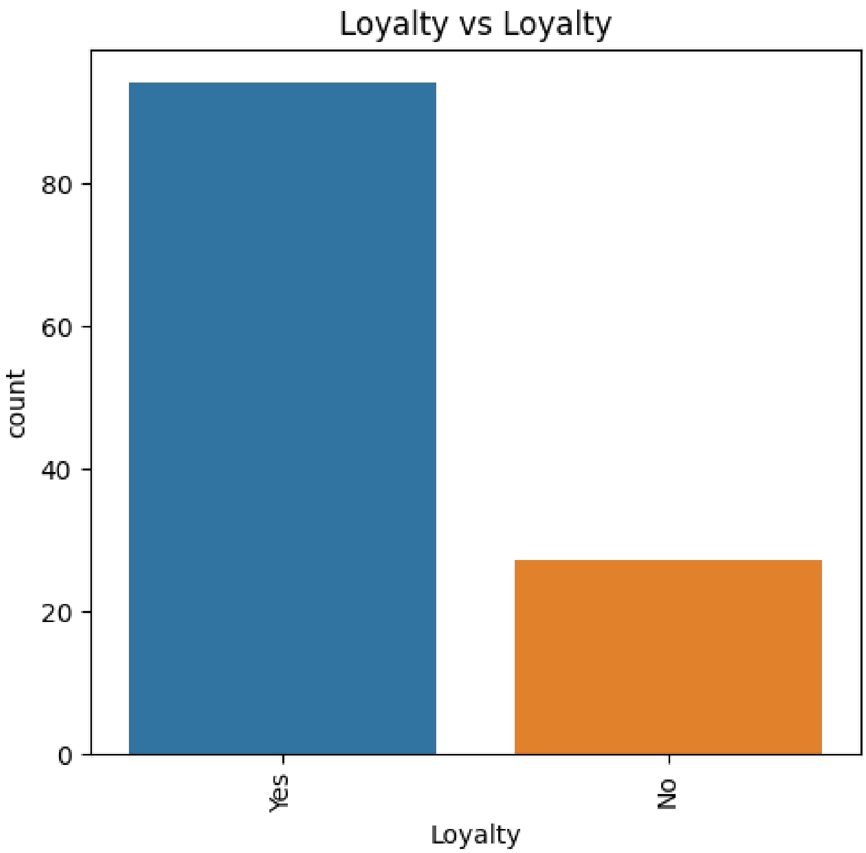

Nearest_Store_Distance vs Loyalty
within 1km
1km - 3km
more than 3km
count
0
10
20
30
40
Yes
No
Loyalty


Membership vs Loyalty
count
Loyalty
Yes
No
50
40
30
20
10
0
Yes
No


Frequent_Product vs Loyalty
count
Loyalty
Yes
No
50
40
30
20
10
0
Coffee
Cold drinks;Pastries
Coffee;Sandwiches
Cold drinks
Coffee;Cold drinks
Cold drinks;Pastries;Sandwiches
Coffee;Juices;Pastries;Sandwiches
Coffee;Pastries;Sandwiches
Coffee;Pastries
Cold drinks;Juices;Pastries
Coffee;Cold drinks;Pastries;Sandwiches
Never
Jaws chip
cake
Pastries
Cold drinks;Never
never
Nothing
Coffee;Cold drinks;Juices;Pastries;Sandwiches




```
No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
```



[126]:
```
data=data.drop(columns='Timestamp',axis=1)
#unwanted column
```

Data Encoding:

Label encoding is applied to convert categorical columns into a numerical format. This transformation is necessary for machine learning models that require numerical input.

[128]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

[140]:
```
data['Gender']=le.fit_transform(data['Gender'])
data['Occupation']=le.fit_transform(data['Occupation'])
data['Visit_Frequency']=le.fit_transform(data['Visit_Frequency'])
data['Service_preferred']=le.fit_transform(data['Service_preferred'])
data['Membership']=le.fit_transform(data['Membership'])
data['Loyalty']=le.fit_transform(data['Loyalty'])
data['Annual_Income']=le.fit_transform(data['Annual_Income'])
data['Time_Spent_Frequency']=le.fit_transform(data['Time_Spent_Frequency'])
data['Age']=le.fit_transform(data['Age'])
```

```
data['Frequent_Product']=le.fit_transform(data['Frequent_Product'])
data['Avg_Money_Spent']=le.fit_transform(data['Avg_Money_Spent'])
data['Promotion_Source']=le.fit_transform(data['Promotion_Source'])
data['Nearest_Store_Distance']=le.fit_transform(data['Nearest_Store_Distance'])
```

[141]:

```
data.tail()
```

[141]:

```
     Gender  Age  Occupation  Annual_Income  Visit_Frequency  \
117       1    0           2              3                1
118       1    2           0              0                1
119       1    2           3              0                3
120       0    2           0              0                3
121       1    2           0              4                3

     Service_preferred  Time_Spent_Frequency  Nearest_Store_Distance  \
117                  0                     1                       0
118                  0                     1                       0
119                  0                     3                       0
120                  6                     0                       2
121                  0                     3                       0

     Membership  Frequent_Product  Avg_Money_Spent  \
117           1                 0                0
118           1                 2                2
119           0                 1                1
120           0                 0                1
121           0                 0                1

     Quality_Rating_vs_Other_Brands  Price_Rating  Sales_Promotion_Importance  \
117                               3             3                           5
118                               5             5                           5
119                               3             2                           4
120                               4             4                           4
121                               1             1                           5

     Ambiance_Rating  WiFi_Rating  Service_Rating  \
117                3            2               4
118                5            5               5
119                3            3               3
120                4            4               4
121                4            3               3

     Meetings_hangouts_preference  Promotion_Source  Loyalty
117                             4                21        1
118                             5                25        1
119                             4                16        0
120                             4                15        1
```

```
121                                2                  3         0
```

SPLITING THE DATASET INTO TARGET & DESCRIPTIVE FEATURES

[152]:
```
x=data.drop('Loyalty',axis=1)
y=data.Loyalty
```

Data Augmentation:

The code employs the Synthetic Minority Over-sampling Technique (SMOTE) to balance the class distribution of the target variable 'Loyalty.' Imbalanced datasets can lead to biased models, so SMOTE generates synthetic samples to create a more balanced dataset.

[153]:
```
from imblearn.over_sampling import SMOTE
#dataargumentation using sysnthetic minority oversampling technique
x_arg,y_arg=SMOTE().fit_resample(x,y)
```

[154]:
```
y_arg.value_counts()
```

[154]:
```
Loyalty
1    94
0    94
Name: count, dtype: int64
```

Data Splitting:

The dataset is split into training and testing sets using train_test_split. An 80-20 split ratio is used, with 80% of the data for training and 20% for testing.

[156]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_arg,y_arg,test_size=0.
 ↪2,random_state=42)
#spliting the dataset
```

Machine Learning Model:

A Random Forest Classifier is chosen as the classification algorithm. Random Forest is an ensemble learning method known for its performance in classification tasks. The model is trained on the training data using rfc.fit(x_train, y_train).

[157]:
```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
#implementing randomforestclassifer
```

[157]:
```
RandomForestClassifier()
```

PREDICTING VALUES

[158]:
```
y_pred=rfc.predict(x_test)
```

[159]:
```
y_pred
```

[159]:
```
array([0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0])
```

Model Evaluation:

The trained Random Forest Classifier is used to make predictions on the test data with y_pred = rfc.predict(x_test). The metrics.classification_report function is used to compute classification metrics, including precision, recall, F1-score, and support for each class. A confusion matrix is displayed to visualize the model's performance, and the accuracy score is calculated using accuracy_score.

[160]:
```
from sklearn import metrics
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.86      0.88        21
           1       0.83      0.88      0.86        17

    accuracy                           0.87        38
   macro avg       0.87      0.87      0.87        38
weighted avg       0.87      0.87      0.87        38
```

[170]:
```
from sklearn.metrics import accuracy_score,confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
accuracy_score(y_test,y_pred)*100
```

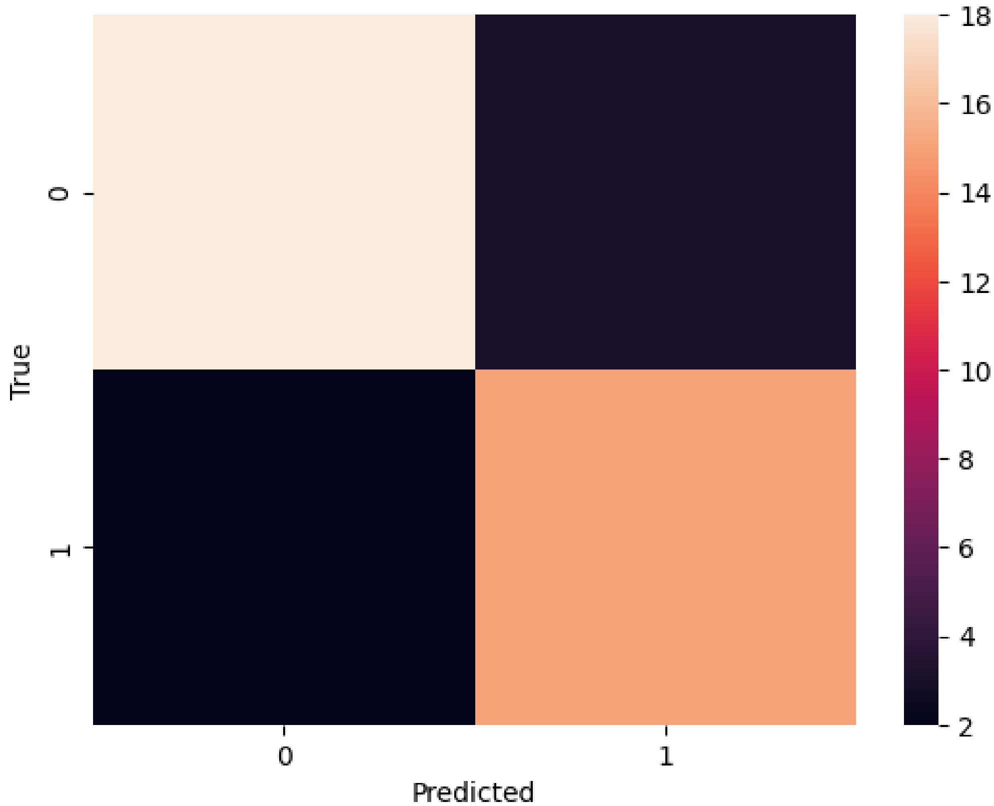
```
[[18  3]
 [ 2 15]]
```

[170]:
```
86.8421052631579
```

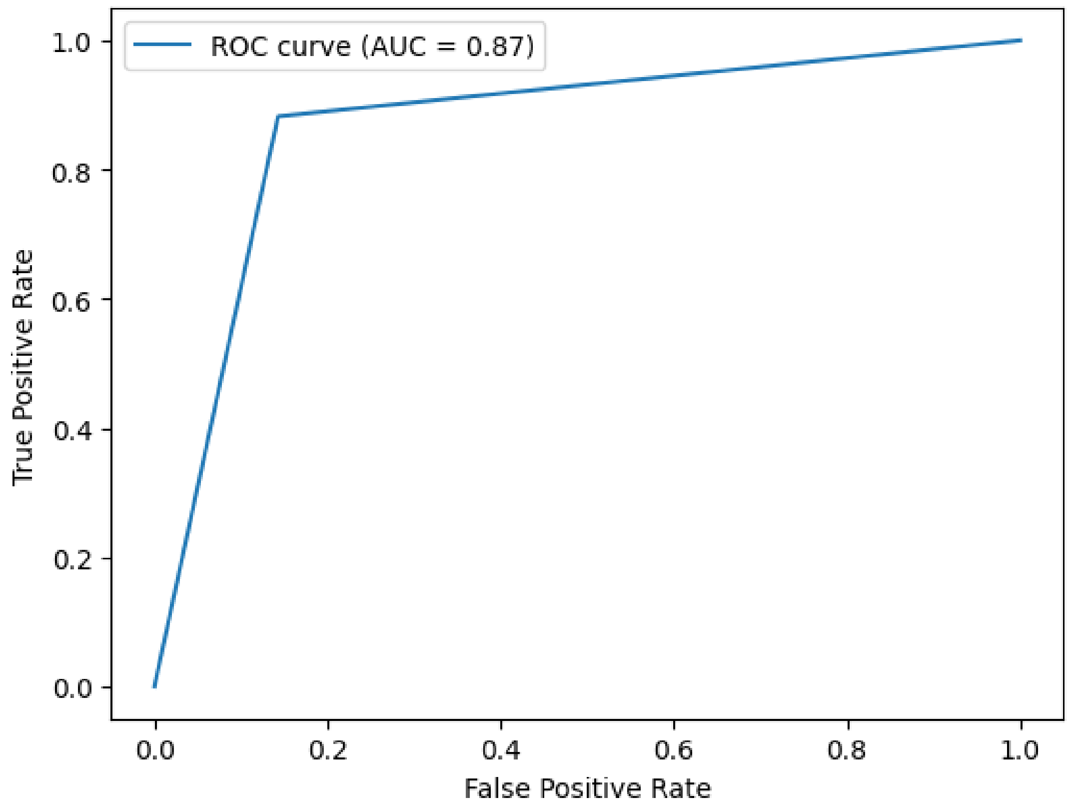
Model Performance Visualization:

A heatmap is generated using Seaborn to visualize the confusion matrix. The heatmap helps understand how well the model predicts each class. A Receiver Operating Characteristic (ROC) curve is plotted to assess the model's performance in a binary classification context. The area under the ROC curve (AUC) is also calculated.

[164]:
```
sns.heatmap(cm)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.show()
```




```
[166]: from sklearn.metrics import roc_curve, roc_auc_score
fpr, tpr, _ = roc_curve(y_test, y_pred)
auc = roc_auc_score(y_test, y_pred)
plt.plot(fpr, tpr, label="ROC curve (AUC = {:.2f})".format(auc))
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
#roc curve is used to check the accuracy of predicted binary-classifer
```



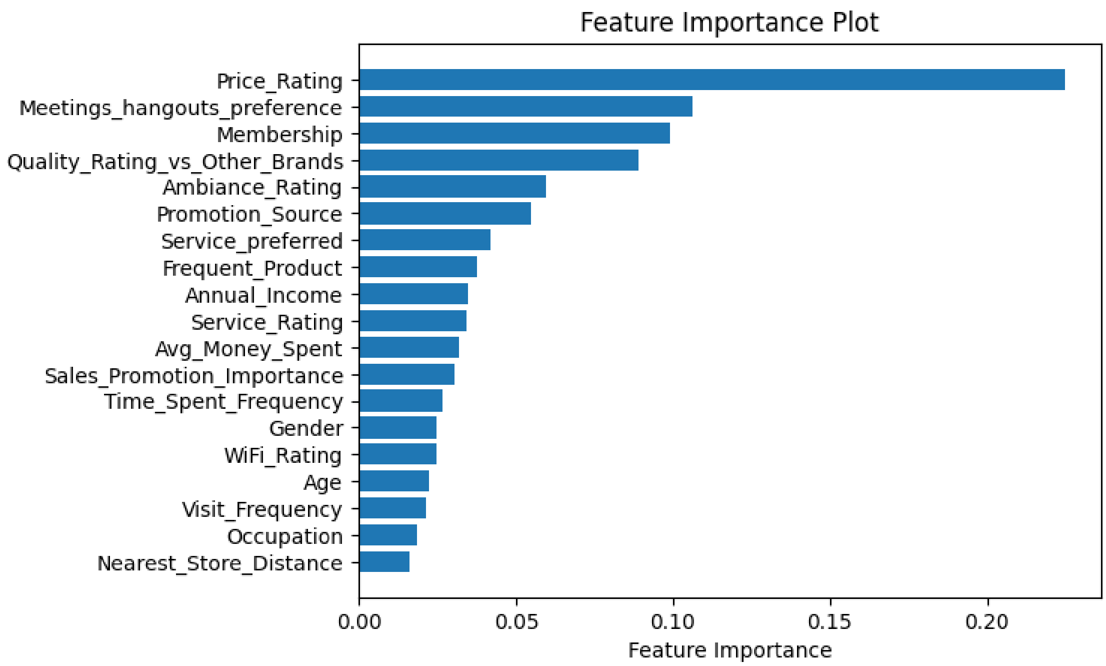
Feature Importance:

The code employs a Random Forest Classifier to evaluate feature importance in predicting customer loyalty. A bar plot is used to display the relative importance of each feature.

[167]:
```
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

forest = RandomForestClassifier(n_estimators=100)
forest.fit(x_train, y_train)
feature_importance = forest.feature_importances_
feature_names = x_train.columns
sorted_idx = np.argsort(feature_importance)
pos = np.arange(sorted_idx.shape[0]) + 0.5

plt.barh(pos, feature_importance[sorted_idx], align="center")
plt.yticks(pos, feature_names[sorted_idx])
plt.xlabel("Feature Importance")
plt.title("Feature Importance Plot")
plt.show()
```



Feature Importance:

The code trains another Random Forest Classifier to evaluate feature importance in predicting customer loyalty. The feature_importances_ attribute is used to obtain the importance of each feature. A horizontal bar plot is created to visualize and compare the relative importance of each feature.

Overall, this code demonstrates a comprehensive process for analyzing and modeling customer loyalty prediction for Starbucks. It preprocesses the data, employs a robust classification algorithm, and evaluates the model's performance while considering the importance of different features. The findings are based on the specific execution of the code and can vary with different datasets and settings. Further optimization and fine-tuning may be necessary for practical applications.