Bank Churn Analysis

March 15, 2023

1 Bank Churners

This project will represent the analysis made on a data set which represents the customer churn in a Banking Industry and means to classify customers based on certain info about them (demographic, banking habits, etc) using clustering and multiple classification algorithms and thus expressing which method is superior for the application and the use case along with observing the difference in their performance.

The methodologies used for classification are k-NN (k Nearest Neighbors) and Logistic Regression and for clustering, k-Means.

1.0.1 What is Attrition ?

The process of reudcing something's strength or effectiveness through sustained attack or pressure. Attrition occurs when the workforce dwindles at a company as people leave and are replaced Attrition is often called a hiring freeze and is seen as **a less disruptive way to trim the workfoce and reduce payroll than layoffs**

Since we are trying to predict attrition, the best model will be one that covers the most attrited customers from our test dataset (we wont care mucn for the false positives)

[ ]:
```
# Importing all the libraries necessary

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import sklearn
import scipy.stats
from IPython.display import display

from matplotlib import figure
from IPython.display import Image
```

```
from sklearn.model_selection import train_test_split, StratifiedKFold,␣
↪GridSearchCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import LabelEncoder


from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.cluster import KMeans

from sklearn.metrics import roc_curve
from sklearn.metrics import auc
from sklearn.metrics import accuracy_score, recall_score, f1_score,␣
↪precision_score, make_scorer, silhouette_score
from sklearn.metrics import confusion_matrix , classification_report
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import cross_val_score, KFold


from scipy.stats import chi2

import warnings
warnings.filterwarnings("ignore")
```

[ ]:
```
# Exploring The Data

df_explore = pd.read_csv('BankChurners.csv')
df_explore.head(20)
```

[ ]:
```
     CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
0   768805383  Existing Customer            45      M                3
1   818770008  Existing Customer            49      F                5
2   713982108  Existing Customer            51      M                3
3   769911858  Existing Customer            40      F                4
4   709106358  Existing Customer            40      M                3
5   713061558  Existing Customer            44      M                2
6   810347208  Existing Customer            51      M                4
7   818906208  Existing Customer            32      M                0
8   710930508  Existing Customer            37      M                3
9   719661558  Existing Customer            48      M                2
10  708790833  Existing Customer            42      M                5
11  710821833  Existing Customer            65      M                1
12  710599683  Existing Customer            56      M                1
13  816082233  Existing Customer            35      M                3
```

```
14  712396908  Existing Customer          57      F              2
15  714885258  Existing Customer          44      M              4
16  709967358  Existing Customer          48      M              4
17  753327333  Existing Customer          41      M              3
18  806160108  Existing Customer          61      M              1
19  709327383  Existing Customer          45      F              2

   Education_Level Marital_Status Income_Category Card_Category  \
0      High School        Married     $60K - $80K          Blue
1         Graduate         Single  Less than $40K          Blue
2         Graduate        Married    $80K - $120K          Blue
3      High School        Unknown  Less than $40K          Blue
4       Uneducated        Married     $60K - $80K          Blue
5         Graduate        Married     $40K - $60K          Blue
6          Unknown        Married         $120K +          Gold
7      High School        Unknown     $60K - $80K        Silver
8       Uneducated         Single     $60K - $80K          Blue
9         Graduate         Single    $80K - $120K          Blue
10      Uneducated        Unknown         $120K +          Blue
11         Unknown        Married     $40K - $60K          Blue
12         College         Single    $80K - $120K          Blue
13        Graduate        Unknown     $60K - $80K          Blue
14        Graduate        Married  Less than $40K          Blue
15         Unknown        Unknown    $80K - $120K          Blue
16   Post-Graduate         Single    $80K - $120K          Blue
17         Unknown        Married    $80K - $120K          Blue
18     High School        Married     $40K - $60K          Blue
19        Graduate        Married         Unknown          Blue

    Months_on_book  …  Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  \
0               39  …       12691.0                  777          11914.0
1               44  …        8256.0                  864           7392.0
2               36  …        3418.0                    0           3418.0
3               34  …        3313.0                 2517            796.0
4               21  …        4716.0                    0           4716.0
5               36  …        4010.0                 1247           2763.0
6               46  …       34516.0                 2264          32252.0
7               27  …       29081.0                 1396          27685.0
8               36  …       22352.0                 2517          19835.0
9               36  …       11656.0                 1677           9979.0
10              31  …        6748.0                 1467           5281.0
11              54  …        9095.0                 1587           7508.0
12              36  …       11751.0                    0          11751.0
13              30  …        8547.0                 1666           6881.0
14              48  …        2436.0                  680           1756.0
15              37  …        4234.0                  972           3262.0
16              36  …       30367.0                 2362          28005.0
```

```
17        34  …     13535.0                  1291          12244.0
18        56  …      3193.0                  2517            676.0
19        37  …     14470.0                  1157          13313.0

    Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  \
0                  1.335             1144              42
1                  1.541             1291              33
2                  2.594             1887              20
3                  1.405             1171              20
4                  2.175              816              28
5                  1.376             1088              24
6                  1.975             1330              31
7                  2.204             1538              36
8                  3.355             1350              24
9                  1.524             1441              32
10                 0.831             1201              42
11                 1.433             1314              26
12                 3.397             1539              17
13                 1.163             1311              33
14                 1.190             1570              29
15                 1.707             1348              27
16                 1.708             1671              27
17                 0.653             1028              21
18                 1.831             1336              30
19                 0.966             1207              21

    Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio  \
0                 1.625                  0.061
1                 3.714                  0.105
2                 2.333                  0.000
3                 2.333                  0.760
4                 2.500                  0.000
5                 0.846                  0.311
6                 0.722                  0.066
7                 0.714                  0.048
8                 1.182                  0.113
9                 0.882                  0.144
10                0.680                  0.217
11                1.364                  0.174
12                3.250                  0.000
13                2.000                  0.195
14                0.611                  0.279
15                1.700                  0.230
16                0.929                  0.078
17                1.625                  0.095
18                1.143                  0.788
19                0.909                  0.080
```

```
    Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_De
pendent_count_Education_Level_Months_Inactive_12_mon_1  \
0                                            0.000093
1                                            0.000057
2                                            0.000021
3                                            0.000134
4                                            0.000022
5                                            0.000055
6                                            0.000123
7                                            0.000086
8                                            0.000045
9                                            0.000303
10                                           0.000191
11                                           0.000198
12                                           0.000048
13                                           0.000096
14                                           0.000114
15                                           0.000063
16                                           0.000236
17                                           0.000150
18                                           0.000175
19                                           0.000055

    Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_De
pendent_count_Education_Level_Months_Inactive_12_mon_2
0                                             0.99991
1                                             0.99994
2                                             0.99998
3                                             0.99987
4                                             0.99998
5                                             0.99994
6                                             0.99988
7                                             0.99991
8                                             0.99996
9                                             0.99970
10                                            0.99981
11                                            0.99980
12                                            0.99995
13                                            0.99990
14                                            0.99989
15                                            0.99994
16                                            0.99976
17                                            0.99985
18                                            0.99983
19                                            0.99994
```

[20 rows x 23 columns]

[ ]:
```
# Printing out all the columns

print('-------------Columns-------------')
print(list(df_explore.columns))
```

```
-------------Columns-------------
['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender', 'Dependent_count',
'Education_Level', 'Marital_Status', 'Income_Category', 'Card_Category',
'Months_on_book', 'Total_Relationship_Count', 'Months_Inactive_12_mon',
'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt', 'Total_Trans_Ct',
'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio', 'Naive_Bayes_Classifier_Attritio
n_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Month
s_Inactive_12_mon_1', 'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Conta
cts_Count_12_mon_Dependent_count_Education_Level_Months_Inactive_12_mon_2']
```

[ ]:
```
# Check if any column has null values.

df_explore.isnull().sum()
```

[ ]:
```
CLIENTNUM
0
Attrition_Flag
0
Customer_Age
0
Gender
0
Dependent_count
0
Education_Level
0
Marital_Status
0
Income_Category
0
Card_Category
0
Months_on_book
0
Total_Relationship_Count
0
Months_Inactive_12_mon
0
Contacts_Count_12_mon
```

```
0
Credit_Limit
0
Total_Revolving_Bal
0
Avg_Open_To_Buy
0
Total_Amt_Chng_Q4_Q1
0
Total_Trans_Amt
0
Total_Trans_Ct
0
Total_Ct_Chng_Q4_Q1
0
Avg_Utilization_Ratio
0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depend
ent_count_Education_Level_Months_Inactive_12_mon_1    0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depend
ent_count_Education_Level_Months_Inactive_12_mon_2    0
dtype: int64
```

[ ]:
```
#Check for duplicate
print('---Check for duplicate---')
print(df_explore[df_explore.duplicated(keep=False)])
print('\n')

print('---Check for duplicate IDS---')
print(df_explore.duplicated(subset=['CLIENTNUM']).unique())
print('\n')

#check for Unknown Values
print('---Check for Unknowns---')
print(df_explore[df_explore == 'Unknown'].count())
print('\n')

#Check for Zero Values
print('---Check for 0 Values---')
df_explore[df_explore == 0].count()
```

```
---Check for duplicate---
Empty DataFrame
Columns: [CLIENTNUM, Attrition_Flag, Customer_Age, Gender, Dependent_count,
Education_Level, Marital_Status, Income_Category, Card_Category, Months_on_book,
Total_Relationship_Count, Months_Inactive_12_mon, Contacts_Count_12_mon,
Credit_Limit, Total_Revolving_Bal, Avg_Open_To_Buy, Total_Amt_Chng_Q4_Q1,
```

```
Total_Trans_Amt, Total_Trans_Ct, Total_Ct_Chng_Q4_Q1, Avg_Utilization_Ratio, Nai
ve_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent
_count_Education_Level_Months_Inactive_12_mon_1, Naive_Bayes_Classifier_Attritio
n_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_Education_Level_Month
s_Inactive_12_mon_2]
Index: []

[0 rows x 23 columns]


---Check for duplicate IDS---
[False]


---Check for Unknowns---
CLIENTNUM
0
Attrition_Flag
0
Customer_Age
0
Gender
0
Dependent_count
0
Education_Level
1519
Marital_Status
749
Income_Category
1112
Card_Category
0
Months_on_book
0
Total_Relationship_Count
0
Months_Inactive_12_mon
0
Contacts_Count_12_mon
0
Credit_Limit
0
Total_Revolving_Bal
0
Avg_Open_To_Buy
0
Total_Amt_Chng_Q4_Q1
```

```
0
Total_Trans_Amt
0
Total_Trans_Ct
0
Total_Ct_Chng_Q4_Q1
0
Avg_Utilization_Ratio
0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depend
ent_count_Education_Level_Months_Inactive_12_mon_1       0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depend
ent_count_Education_Level_Months_Inactive_12_mon_2       0
dtype: int64


---Check for 0 Values---
```

[ ]:

```
CLIENTNUM
0
Attrition_Flag
0
Customer_Age
0
Gender
0
Dependent_count
904
Education_Level
0
Marital_Status
0
Income_Category
0
Card_Category
0
Months_on_book
0
Total_Relationship_Count
0
Months_Inactive_12_mon
29
Contacts_Count_12_mon
399
Credit_Limit
0
Total_Revolving_Bal
```

```
2470
Avg_Open_To_Buy
0
Total_Amt_Chng_Q4_Q1
5
Total_Trans_Amt
0
Total_Trans_Ct
0
Total_Ct_Chng_Q4_Q1
7
Avg_Utilization_Ratio
2470
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depend
ent_count_Education_Level_Months_Inactive_12_mon_1       0
Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Depend
ent_count_Education_Level_Months_Inactive_12_mon_2       0
dtype: int64
```

1.0.2 We have unknown values in

-
-
-

Income_Category

1.0.3 We can later map them or filter them before building the models

[ ]:
```
# Counting the Attrition Flags

aggregate_data = df_explore.groupby(['Attrition_Flag'])['CLIENTNUM'].count().
 ↪to_frame().reset_index()
aggregate_data
```

[ ]:
```
      Attrition_Flag  CLIENTNUM
0  Attrited Customer       1627
1  Existing Customer       8500
```

2 Exploratory Data Analysis

We have some categorical data, lets see if the groups are statistically significant or insignificant

2.1 Gender

[ ]:
```
# Gender

cross_tab = pd.crosstab(index=df_explore['Gender'],
                        columns=df_explore['Attrition_Flag'])


cross_tab_prop = pd.crosstab(index=df_explore['Gender'],
                        columns=df_explore['Attrition_Flag'],
                             normalize="index")

cross_tab_prop = cross_tab_prop.reset_index()
cross_tab_prop
```

[ ]:

Attrition_Flag	Gender	Attrited Customer	Existing Customer
0	F	0.173572	0.826428
1	M	0.146152	0.853848

[ ]:
```
### Check if the difference between the two groups is statistically significant

chi2_stat, p, dof, expected = scipy.stats.chi2_contingency(cross_tab)

print(f"chi2 statistic:     {chi2_stat:.5g}")
print(f"p-value:            {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:\n",expected)
```

```
chi2 statistic:     13.866
p-value:            0.00019636
degrees of freedom: 1
expected frequencies:
 [[ 860.81425891 4497.18574109]
 [ 766.18574109 4002.81425891]]
```

The difference here is significant as p-value < 0.05

2.1.1 Income

[ ]:
```
cross_tab = pd.crosstab(index=df_explore['Income_Category'],
                        columns=df_explore['Attrition_Flag'])

cross_tab_prop = pd.crosstab(index=df_explore['Income_Category'],
                        columns=df_explore['Attrition_Flag'],
                             normalize="index")
cross_tab_prop
```

[ ]:
```
Attrition_Flag   Attrited Customer  Existing Customer
Income_Category
$120K +                   0.173315           0.826685
$40K - $60K               0.151397           0.848603
$60K - $80K               0.134807           0.865193
$80K - $120K              0.157655           0.842345
Less than $40K            0.171862           0.828138
Unknown                   0.168165           0.831835
```

[ ]:
```
### Check if the difference between the income groups is statistically␣
↪significant


chi2_stat, p, dof, expected = scipy.stats.chi2_contingency(cross_tab)

print(f"chi2 statistic:     {chi2_stat:.5g}")
print(f"p-value:            {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:\n",expected)
```

```
chi2 statistic:     12.832
p-value:            0.025002
degrees of freedom: 5
expected frequencies:
 [[ 116.79954577  610.20045423]
 [ 287.5807248  1502.4192752 ]
 [ 225.24479115 1176.75520885]
 [ 246.61252098 1288.38747902]
 [ 572.10891676 2988.89108324]
 [ 178.65350054  933.34649946]]
```

Here the p-value is 0.025002 which is less than (<) 0.05. Hence we can say that the difference between the groups here is significant.

2.1.2 Marital_Status

[ ]:
```
cross_tab = pd.crosstab(index=df_explore['Marital_Status'],
                        columns=df_explore['Attrition_Flag'])

cross_tab_prop = pd.crosstab(index=df_explore['Marital_Status'],
                        columns=df_explore['Attrition_Flag'],
                             normalize="index")
cross_tab_prop
```

[ ]:
```
Attrition_Flag  Attrited Customer  Existing Customer
Marital_Status
Divorced                 0.161765           0.838235
```

```
Married                  0.151269          0.848731
Single                   0.169414          0.830586
Unknown                  0.172230          0.827770
```

[ ]:
```python
### Check if the difference between the marital status groups is statistically␣
↪significant

chi2_stat, p, dof, expected = scipy.stats.chi2_contingency(cross_tab)

print(f"chi2 statistic:     {chi2_stat:.5g}")
print(f"p-value:            {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:\n",expected)
```

```
chi2 statistic:     6.0561
p-value:            0.10891
degrees of freedom: 3
expected frequencies:
 [[ 120.17339785  627.82660215]
 [ 753.01165202 3933.98834798]
 [ 633.48089266 3309.51910734]
 [ 120.33405747  628.66594253]]
```

The difference is not significant

2.1.3 Card_Category

[ ]:
```python
cross_tab = pd.crosstab(index=df_explore['Card_Category'],
                        columns=df_explore['Attrition_Flag'])

cross_tab_prop = pd.crosstab(index=df_explore['Card_Category'],
                        columns=df_explore['Attrition_Flag'],
                             normalize="index")
cross_tab_prop
```

[ ]:
```
Attrition_Flag  Attrited Customer  Existing Customer
Card_Category
Blue                     0.160979           0.839021
Gold                     0.181034           0.818966
Platinum                 0.250000           0.750000
Silver                   0.147748           0.852252
```

[ ]:
```python
### Check if the difference between the card_category groups is statistically␣
↪significant

chi2_stat, p, dof, expected = scipy.stats.chi2_contingency(cross_tab)
```

```
print(f"chi2 statistic:     {chi2_stat:.5g}")
print(f"p-value:            {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:\n",expected)
```

```
chi2 statistic:     2.2342
p-value:            0.52524
degrees of freedom: 3
expected frequencies:
 [[1.51598420e+03 7.92001580e+03]
 [1.86365162e+01 9.73634838e+01]
 [3.21319246e+00 1.67868075e+01]
 [8.91660906e+01 4.65833909e+02]]
```

The difference is not significant

2.1.4 Education level

[ ]:
```
cross_tab = pd.crosstab(index=df_explore['Education_Level'],
                        columns=df_explore['Attrition_Flag'])

cross_tab_prop = pd.crosstab(index=df_explore['Education_Level'],
                        columns=df_explore['Attrition_Flag'],
                             normalize="index")
cross_tab_prop
```

[ ]:

Attrition_Flag Education_Level	Attrited Customer	Existing Customer
College	0.152024	0.847976
Doctorate	0.210643	0.789357
Graduate	0.155691	0.844309
High School	0.152012	0.847988
Post-Graduate	0.178295	0.821705
Uneducated	0.159381	0.840619
Unknown	0.168532	0.831468

[ ]:
```
### Check if the difference between the education_level groups is statistically␣
↪significant

chi2_stat, p, dof, expected = scipy.stats.chi2_contingency(cross_tab)

print(f"chi2 statistic:     {chi2_stat:.5g}")
print(f"p-value:            {p:.5g}")
print(f"degrees of freedom: {dof}")
print("expected frequencies:\n",expected)
```

```
chi2 statistic:     12.511
p-value:            0.051489
```

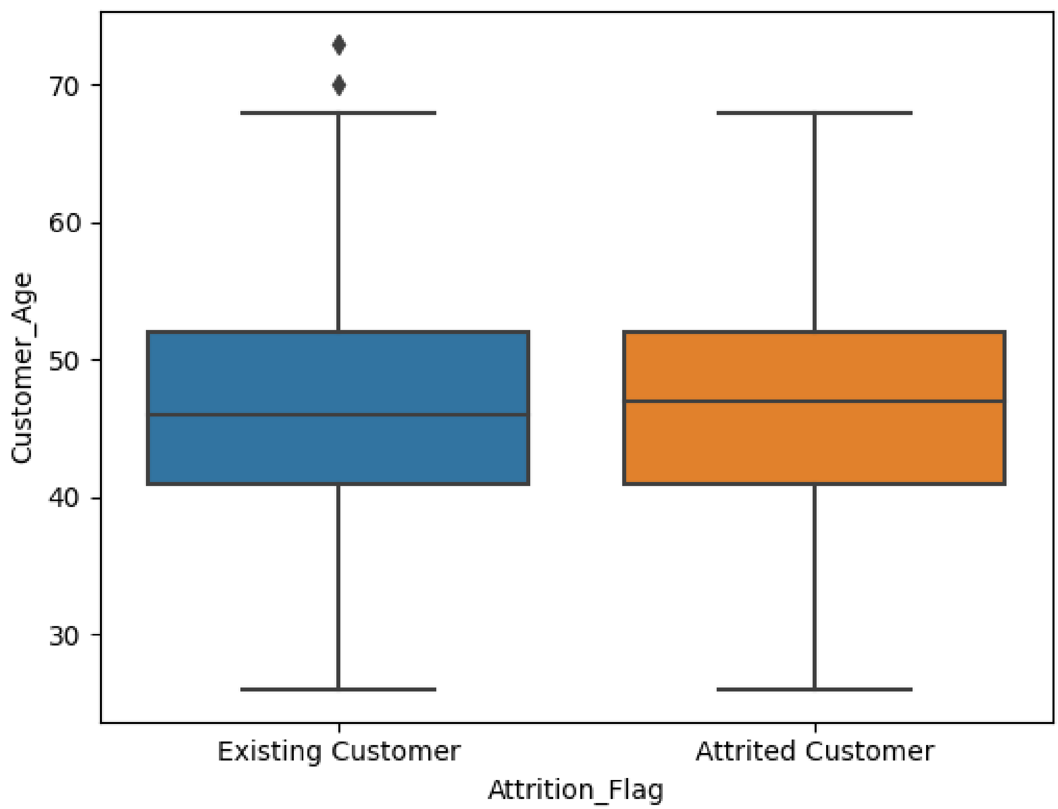
```
degrees of freedom: 6
expected frequencies:
 [[ 162.74819789  850.25180211]
 [  72.45748988  378.54251012]
 [ 502.54330009 2625.45669991]
 [ 323.40782068 1689.59217932]
 [  82.90036536  433.09963464]
 [ 238.90085909 1248.09914091]
 [ 244.04196702 1274.95803298]]
```

The difference is not significant

2.1.5 Numeric Values

[ ]:
```python
sns.boxplot(x='Attrition_Flag', y='Customer_Age', data=df_explore)
```

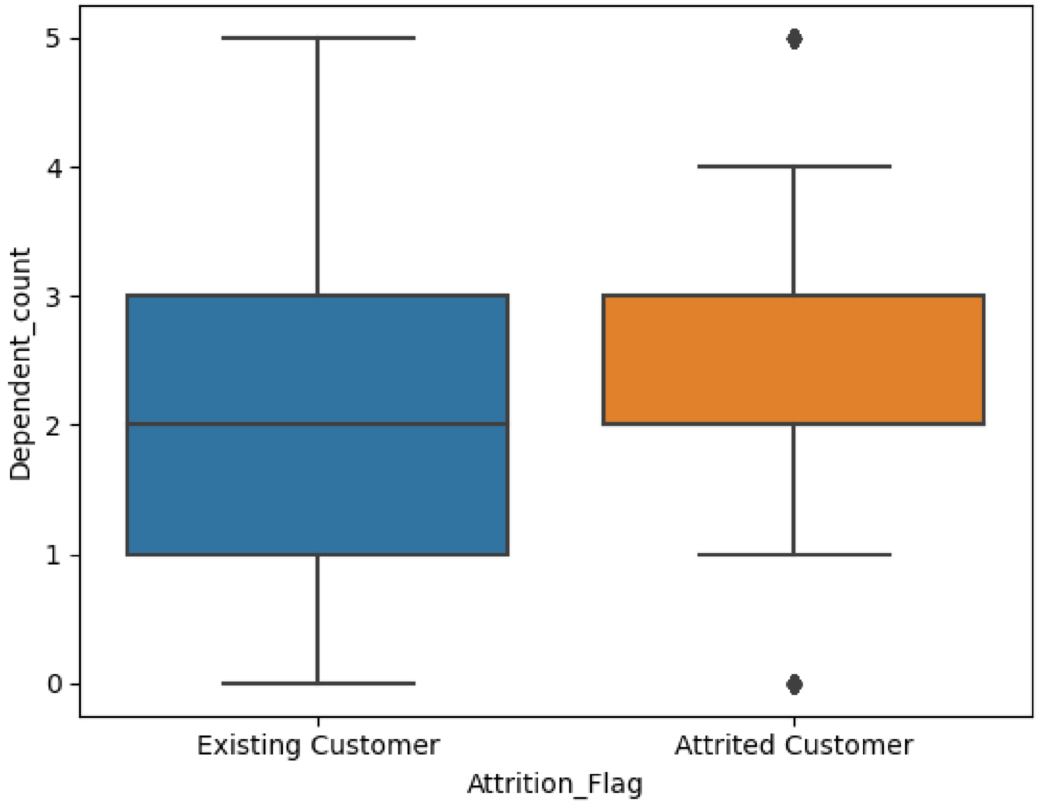
[ ]:
```
<Axes: xlabel='Attrition_Flag', ylabel='Customer_Age'>
```



The difference is not significant

[ ]:
```python
sns.boxplot(x='Attrition_Flag', y='Dependent_count', data=df_explore)
```

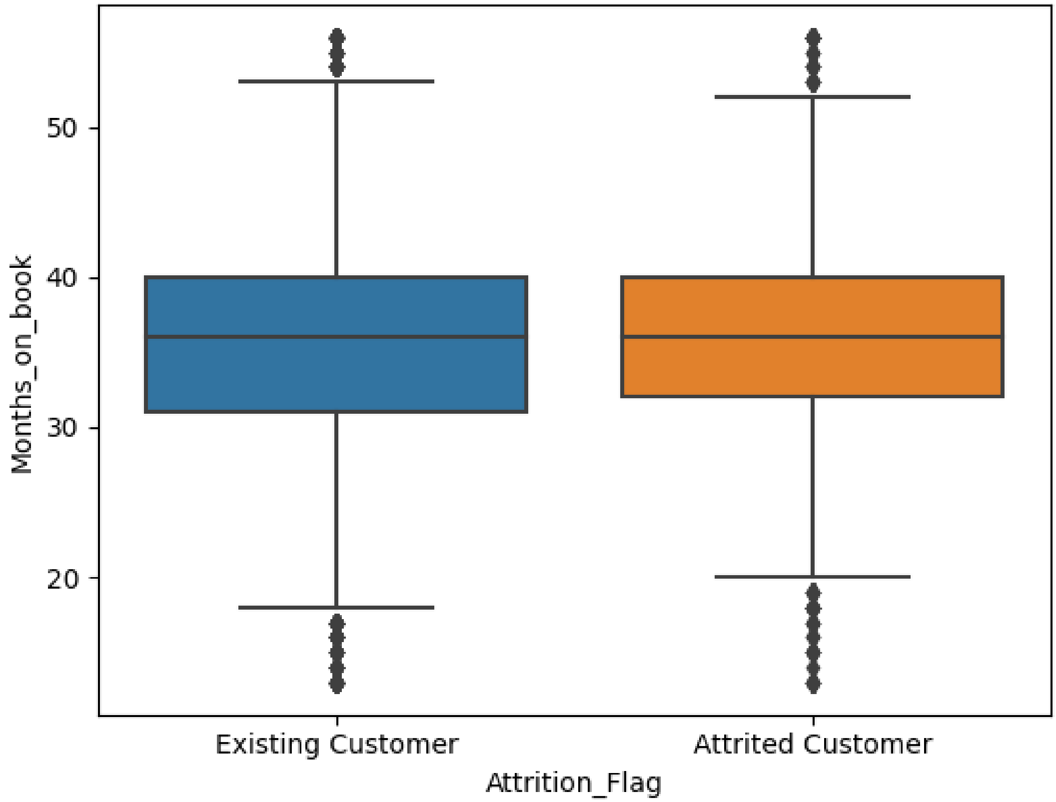
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Dependent_count'>



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Months_on_book', data=df_explore)
```

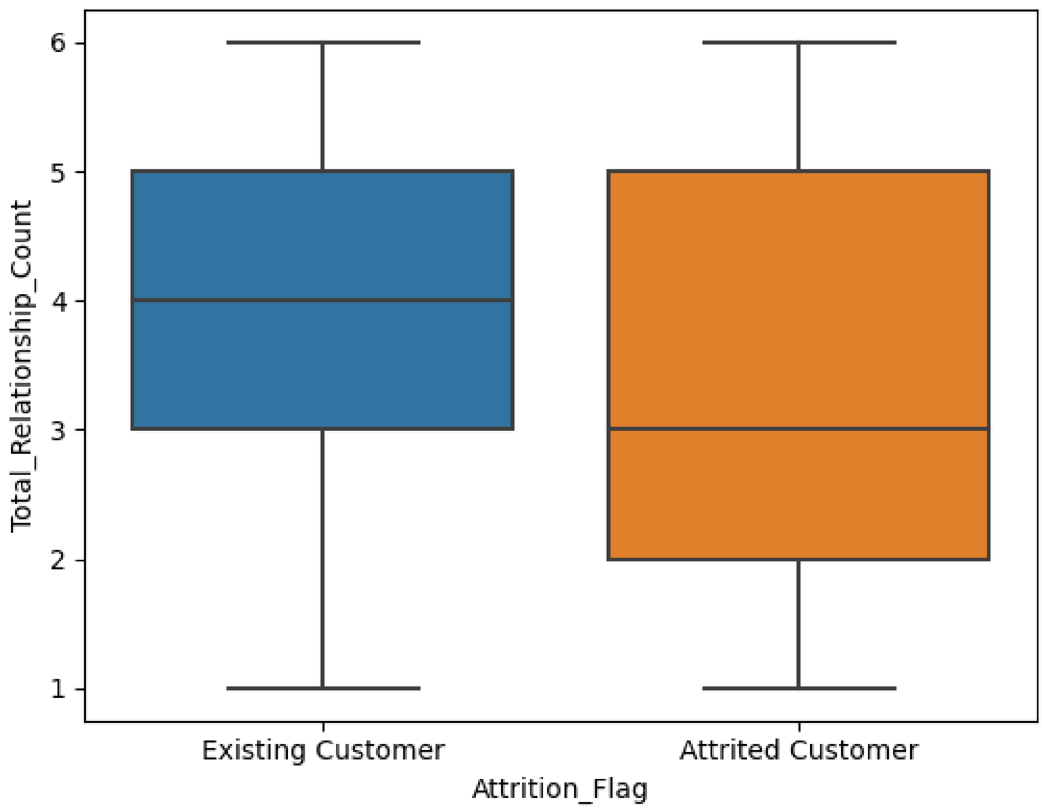
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Months_on_book'>



The difference is not significant

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Total_Relationship_Count', data=df_explore)
```

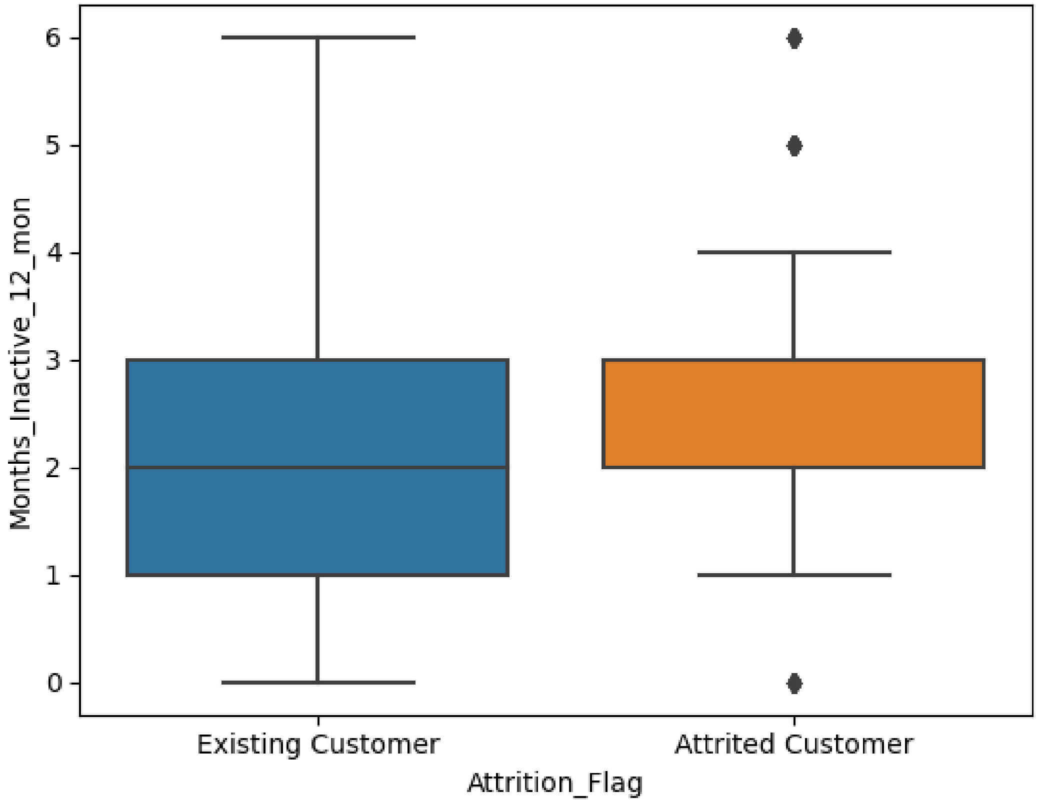
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Total_Relationship_Count'>



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Months_Inactive_12_mon', data=df_explore)
```

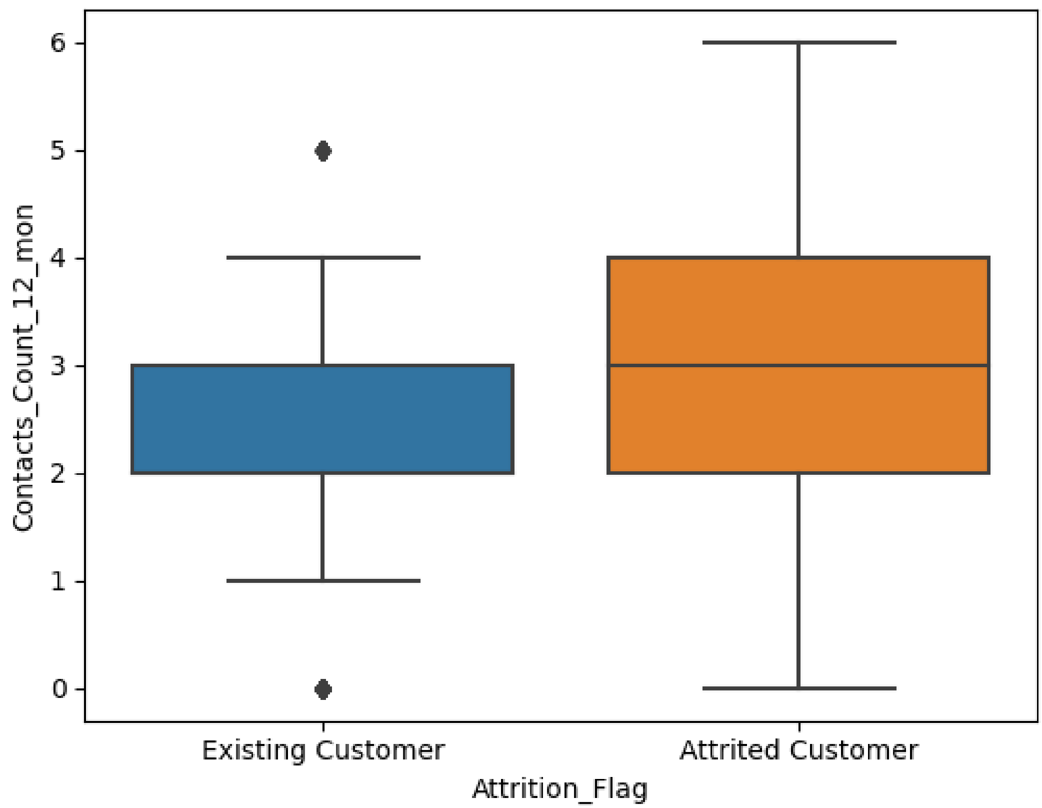
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Months_Inactive_12_mon'>



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Contacts_Count_12_mon', data=df_explore)
```

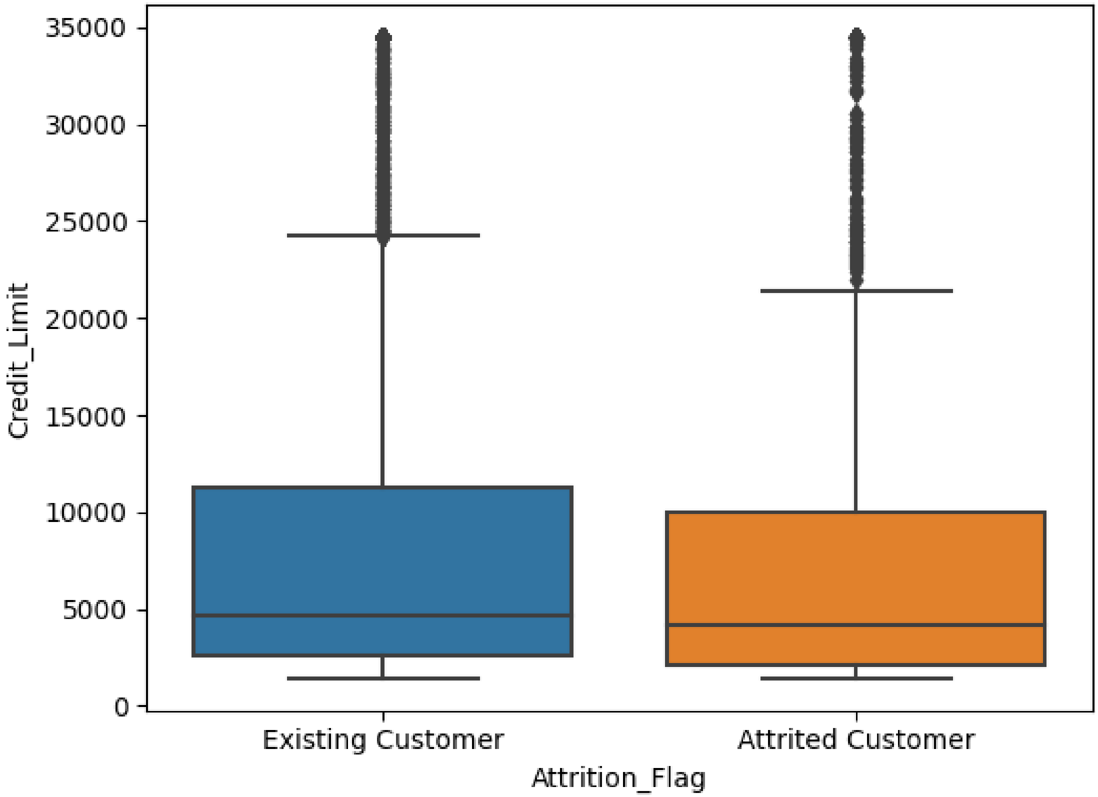
[ ]:
```
<Axes: xlabel='Attrition_Flag', ylabel='Contacts_Count_12_mon'>
```



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Credit_Limit', data=df_explore)
```

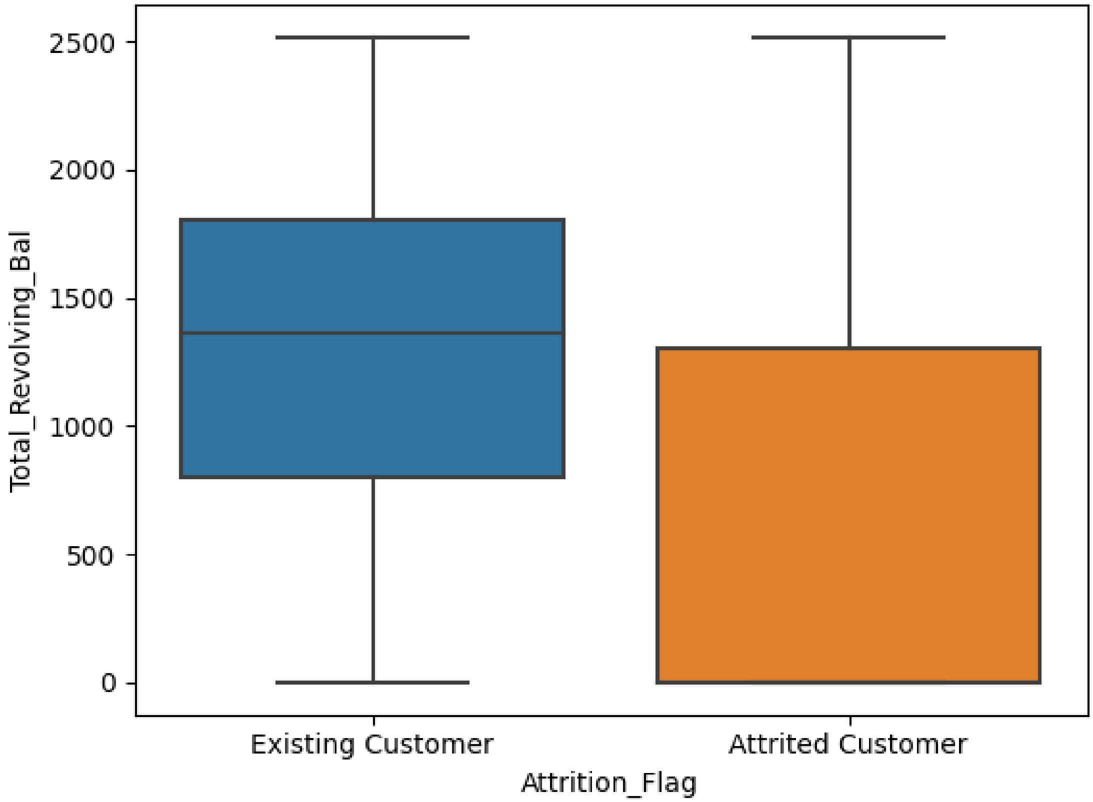
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Credit_Limit'>



The difference here is very small

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Total_Revolving_Bal', data=df_explore)
```

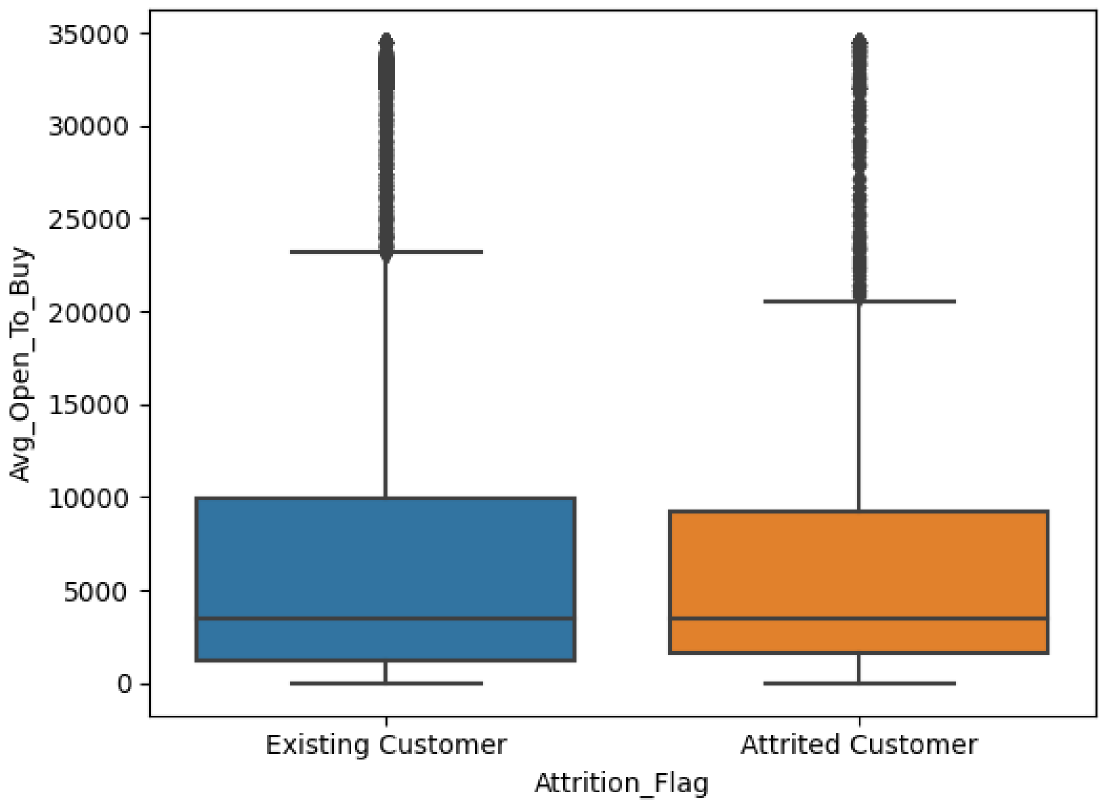
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Total_Revolving_Bal'>



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Avg_Open_To_Buy', data=df_explore)
```

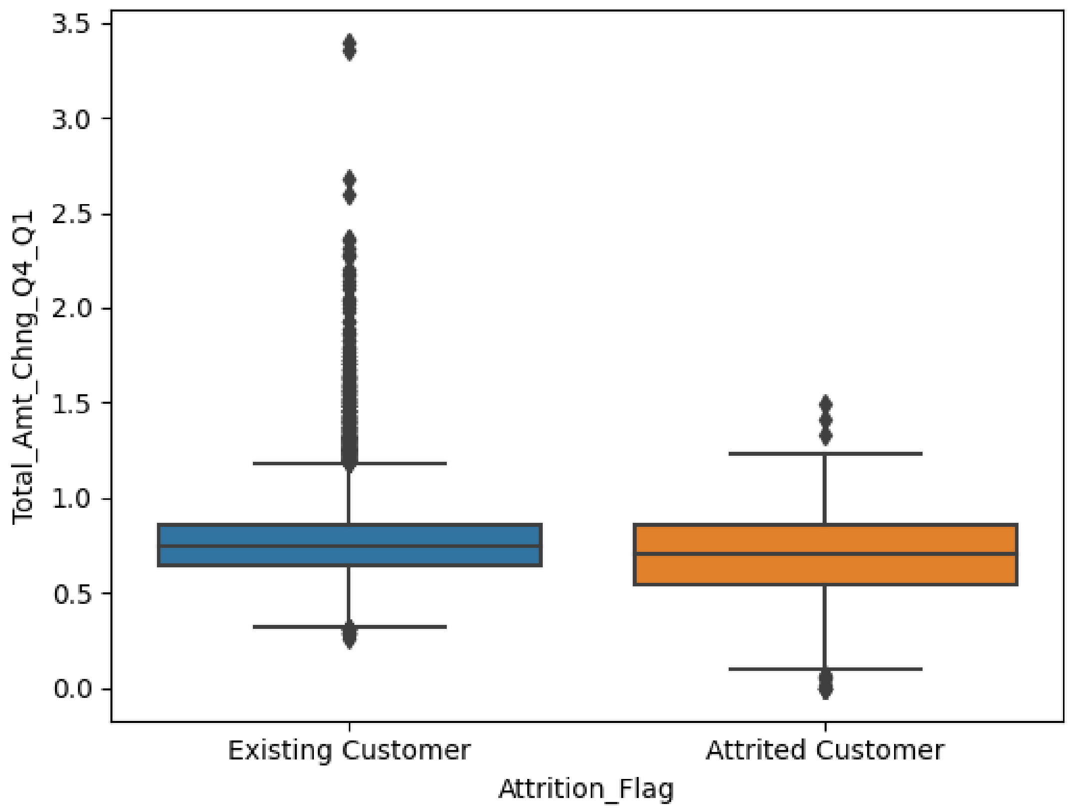
[ ]:
```
<Axes: xlabel='Attrition_Flag', ylabel='Avg_Open_To_Buy'>
```



The difference is not significant

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Total_Amt_Chng_Q4_Q1', data=df_explore)
```

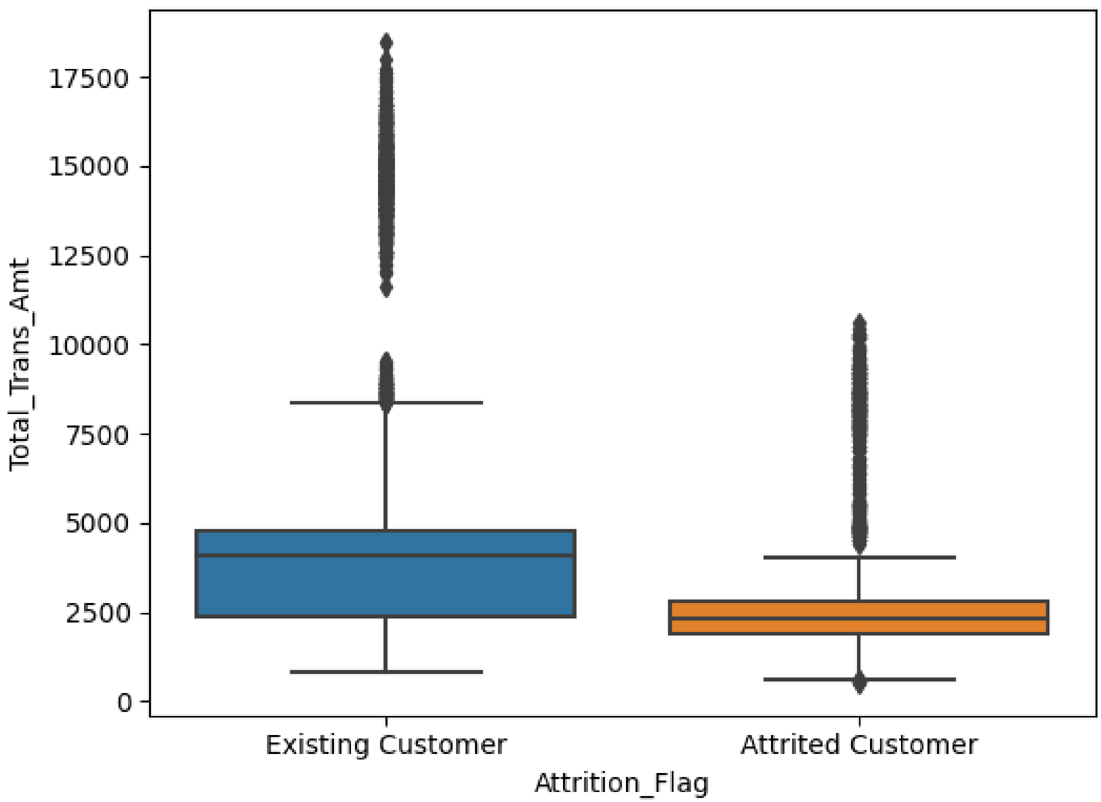
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Total_Amt_Chng_Q4_Q1'>



Theere is a slight difference

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Total_Trans_Amt', data=df_explore)
```

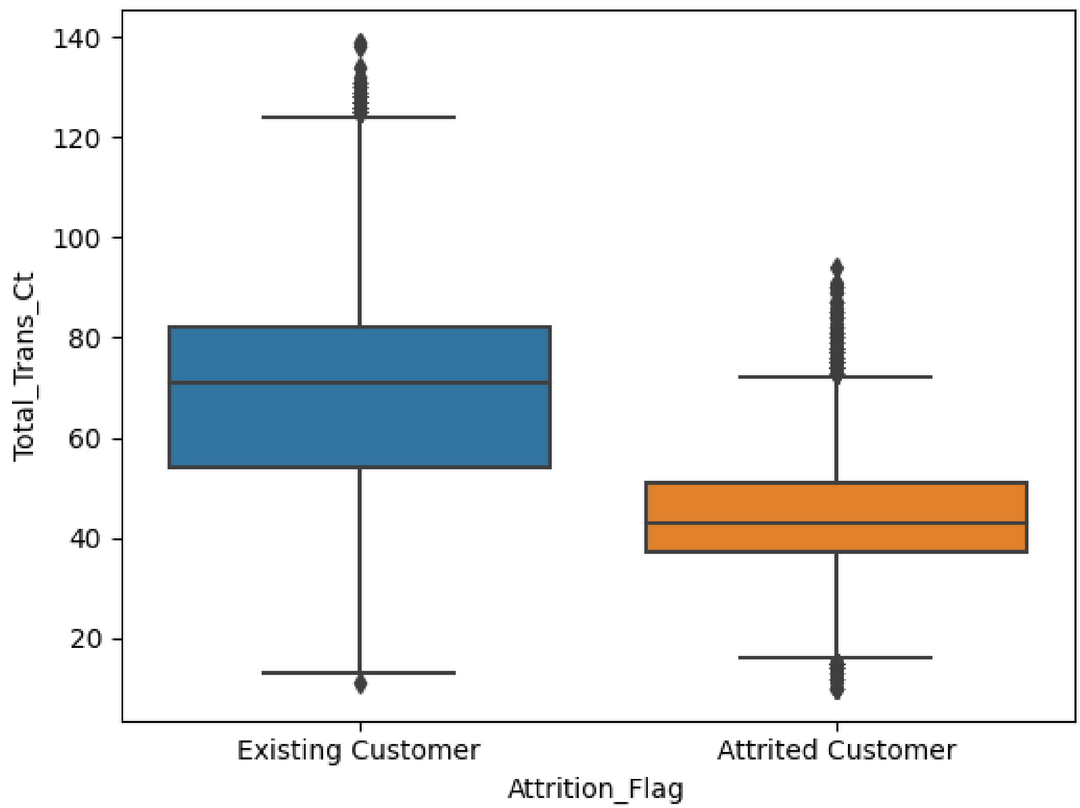
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Total_Trans_Amt'>



The difference is significant, hence it can be used in the model as predictor

```
[ ]: sns.boxplot(x='Attrition_Flag', y='Total_Trans_Ct', data=df_explore)
```

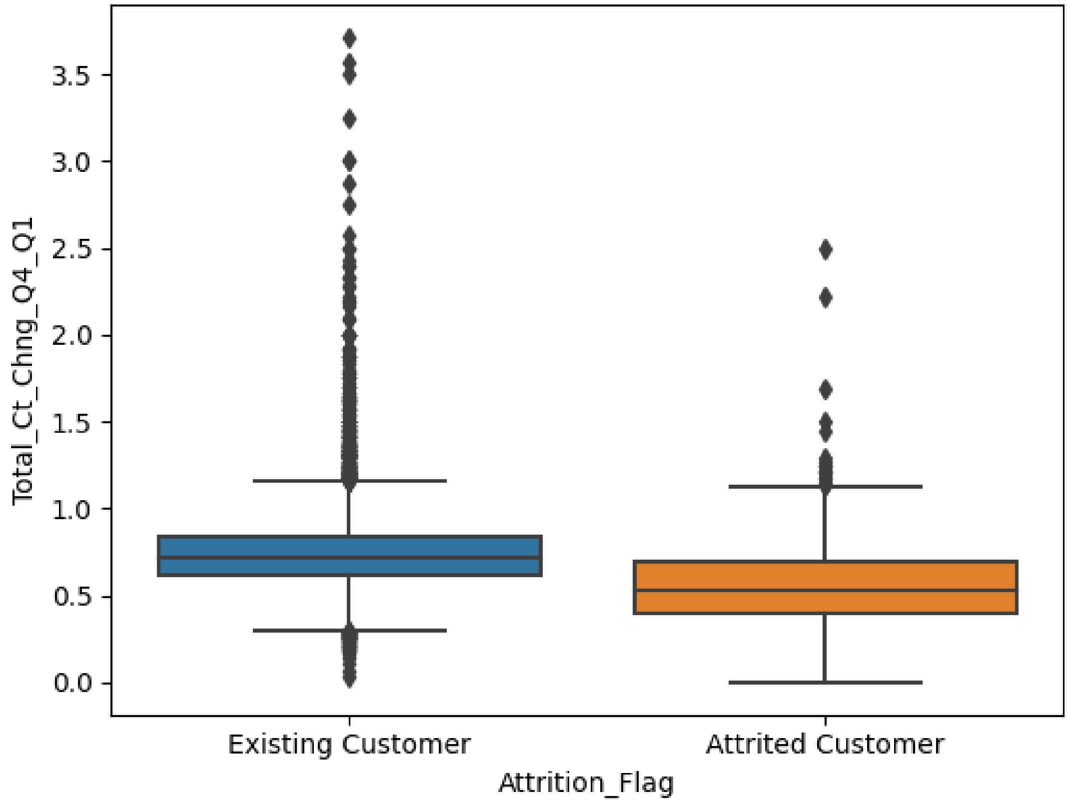
```
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Total_Trans_Ct'>
```



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Total_Ct_Chng_Q4_Q1', data=df_explore)
```

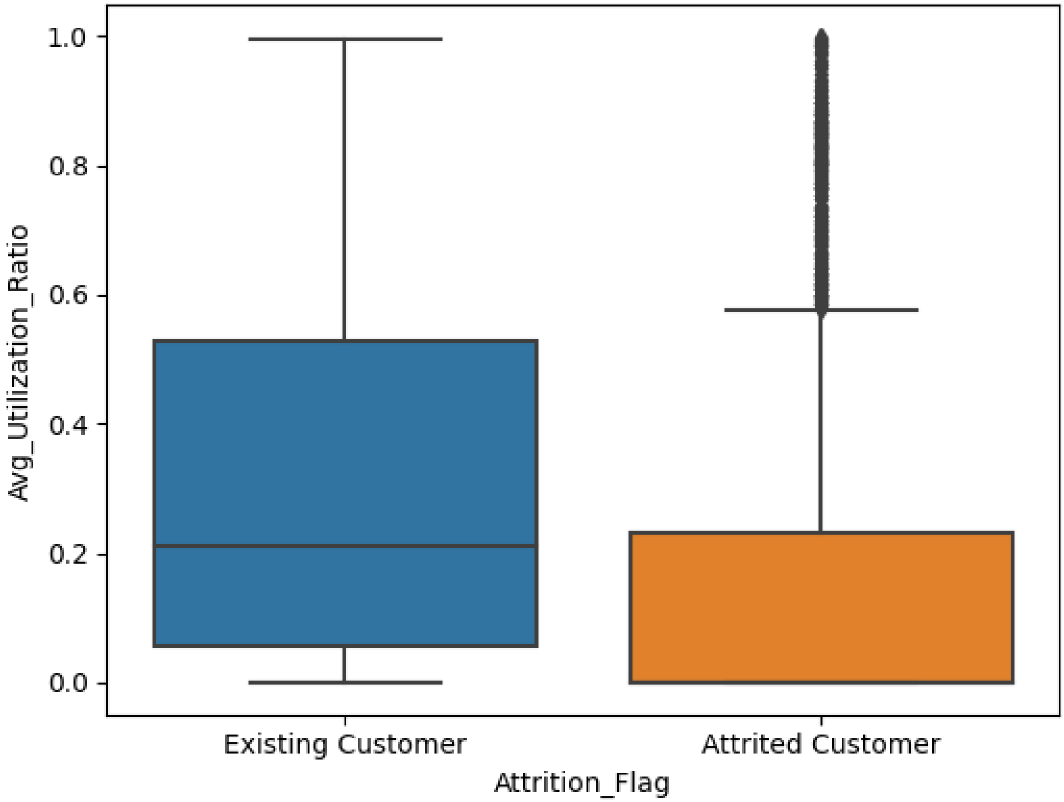
[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Total_Ct_Chng_Q4_Q1'>



The difference is slightly different

[ ]:
```
sns.boxplot(x='Attrition_Flag', y='Avg_Utilization_Ratio', data=df_explore)
```

[ ]: <Axes: xlabel='Attrition_Flag', ylabel='Avg_Utilization_Ratio'>



The difference is significant, hence it can be used in the model as predictor

[ ]:
```
# A pie plot to better represent the data

labels = df_explore.Attrition_Flag
values = df_explore.CLIENTNUM

fig = go.Figure(data=[go.Pie(labels=labels, values=values, pull=[0, 0.2])])
fig.update_layout(legend=dict(
    yanchor="top",
    y=0.90,
    xanchor="left",
    x=0.20
))
```

2.2 Data Preprocessing and cleaning

[ ]:
```
df_explore.head(20)
```

[ ]:
```
    CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
0   768805383  Existing Customer            45      M                3
```

```
1   818770008  Existing Customer           49      F               5
2   713982108  Existing Customer           51      M               3
3   769911858  Existing Customer           40      F               4
4   709106358  Existing Customer           40      M               3
5   713061558  Existing Customer           44      M               2
6   810347208  Existing Customer           51      M               4
7   818906208  Existing Customer           32      M               0
8   710930508  Existing Customer           37      M               3
9   719661558  Existing Customer           48      M               2
10  708790833  Existing Customer           42      M               5
11  710821833  Existing Customer           65      M               1
12  710599683  Existing Customer           56      M               1
13  816082233  Existing Customer           35      M               3
14  712396908  Existing Customer           57      F               2
15  714885258  Existing Customer           44      M               4
16  709967358  Existing Customer           48      M               4
17  753327333  Existing Customer           41      M               3
18  806160108  Existing Customer           61      M               1
19  709327383  Existing Customer           45      F               2

   Education_Level Marital_Status Income_Category Card_Category  \
0      High School        Married     $60K - $80K          Blue
1         Graduate         Single  Less than $40K          Blue
2         Graduate        Married    $80K - $120K          Blue
3      High School        Unknown  Less than $40K          Blue
4       Uneducated        Married     $60K - $80K          Blue
5         Graduate        Married     $40K - $60K          Blue
6          Unknown        Married         $120K +          Gold
7      High School        Unknown     $60K - $80K        Silver
8       Uneducated         Single     $60K - $80K          Blue
9         Graduate         Single    $80K - $120K          Blue
10      Uneducated        Unknown         $120K +          Blue
11         Unknown        Married     $40K - $60K          Blue
12         College         Single    $80K - $120K          Blue
13        Graduate        Unknown     $60K - $80K          Blue
14        Graduate        Married  Less than $40K          Blue
15         Unknown        Unknown    $80K - $120K          Blue
16   Post-Graduate         Single    $80K - $120K          Blue
17         Unknown        Married    $80K - $120K          Blue
18     High School        Married     $40K - $60K          Blue
19        Graduate        Married         Unknown          Blue

    Months_on_book  …  Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  \
0               39  …       12691.0                  777          11914.0
1               44  …        8256.0                  864           7392.0
2               36  …        3418.0                    0           3418.0
3               34  …        3313.0                 2517            796.0
```

```
4               21  …     4716.0                   0        4716.0
5               36  …     4010.0                1247        2763.0
6               46  …    34516.0                2264       32252.0
7               27  …    29081.0                1396       27685.0
8               36  …    22352.0                2517       19835.0
9               36  …    11656.0                1677        9979.0
10              31  …     6748.0                1467        5281.0
11              54  …     9095.0                1587        7508.0
12              36  …    11751.0                   0       11751.0
13              30  …     8547.0                1666        6881.0
14              48  …     2436.0                 680        1756.0
15              37  …     4234.0                 972        3262.0
16              36  …    30367.0                2362       28005.0
17              34  …    13535.0                1291       12244.0
18              56  …     3193.0                2517         676.0
19              37  …    14470.0                1157       13313.0

    Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  \
0                  1.335             1144              42
1                  1.541             1291              33
2                  2.594             1887              20
3                  1.405             1171              20
4                  2.175              816              28
5                  1.376             1088              24
6                  1.975             1330              31
7                  2.204             1538              36
8                  3.355             1350              24
9                  1.524             1441              32
10                 0.831             1201              42
11                 1.433             1314              26
12                 3.397             1539              17
13                 1.163             1311              33
14                 1.190             1570              29
15                 1.707             1348              27
16                 1.708             1671              27
17                 0.653             1028              21
18                 1.831             1336              30
19                 0.966             1207              21

    Total_Ct_Chng_Q4_Q1  Avg_Utilization_Ratio  \
0                 1.625                  0.061
1                 3.714                  0.105
2                 2.333                  0.000
3                 2.333                  0.760
4                 2.500                  0.000
5                 0.846                  0.311
6                 0.722                  0.066
```

```
7                 0.714                   0.048
8                 1.182                   0.113
9                 0.882                   0.144
10                0.680                   0.217
11                1.364                   0.174
12                3.250                   0.000
13                2.000                   0.195
14                0.611                   0.279
15                1.700                   0.230
16                0.929                   0.078
17                1.625                   0.095
18                1.143                   0.788
19                0.909                   0.080

    Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_De
pendent_count_Education_Level_Months_Inactive_12_mon_1  \
0                                            0.000093
1                                            0.000057
2                                            0.000021
3                                            0.000134
4                                            0.000022
5                                            0.000055
6                                            0.000123
7                                            0.000086
8                                            0.000045
9                                            0.000303
10                                           0.000191
11                                           0.000198
12                                           0.000048
13                                           0.000096
14                                           0.000114
15                                           0.000063
16                                           0.000236
17                                           0.000150
18                                           0.000175
19                                           0.000055

    Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_De
pendent_count_Education_Level_Months_Inactive_12_mon_2
0                                             0.99991
1                                             0.99994
2                                             0.99998
3                                             0.99987
4                                             0.99998
5                                             0.99994
6                                             0.99988
7                                             0.99991
```

```
8                                                    0.99996
9                                                    0.99970
10                                                   0.99981
11                                                   0.99980
12                                                   0.99995
13                                                   0.99990
14                                                   0.99989
15                                                   0.99994
16                                                   0.99976
17                                                   0.99985
18                                                   0.99983
19                                                   0.99994

[20 rows x 23 columns]
```

As we see from above, there are two columns that represent classification for the Attrition flag using Naive Baues Classifier. We cannot have this in our data set as this will skew the results entrirely and would lead to a 100% accuracy score which would not be a true representation for the accuracy of the model we apply and thus the classification would become invalid.

[ ]:
```python
# Listing all the columns in the dataset and lookinng some info about the data
print('\n')
print('-------------Dataset-------------')
df_explore.head(10)
```

```
-------------Dataset-------------
```

[ ]:
```
    CLIENTNUM     Attrition_Flag  Customer_Age Gender  Dependent_count  \
0  768805383  Existing Customer            45      M                3
1  818770008  Existing Customer            49      F                5
2  713982108  Existing Customer            51      M                3
3  769911858  Existing Customer            40      F                4
4  709106358  Existing Customer            40      M                3
5  713061558  Existing Customer            44      M                2
6  810347208  Existing Customer            51      M                4
7  818906208  Existing Customer            32      M                0
8  710930508  Existing Customer            37      M                3
9  719661558  Existing Customer            48      M                2

  Education_Level Marital_Status Income_Category Card_Category  \
0     High School        Married     $60K - $80K          Blue
1        Graduate         Single  Less than $40K          Blue
2        Graduate        Married    $80K - $120K          Blue
3     High School        Unknown  Less than $40K          Blue
4      Uneducated        Married     $60K - $80K          Blue
```

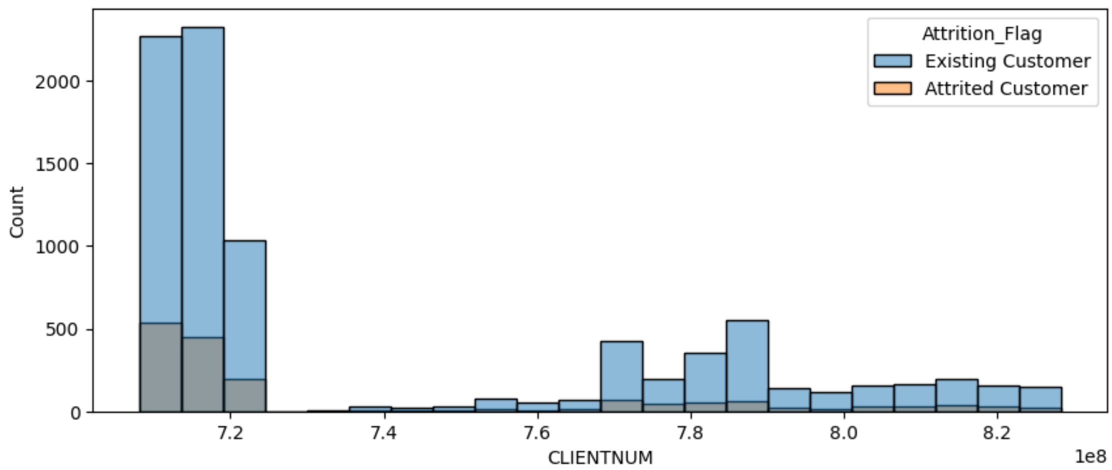
```
5       Graduate        Married     $40K - $60K          Blue
6        Unknown        Married         $120K +          Gold
7    High School        Unknown     $60K - $80K        Silver
8     Uneducated         Single     $60K - $80K          Blue
9       Graduate         Single    $80K - $120K          Blue

   Months_on_book  …  Credit_Limit  Total_Revolving_Bal  Avg_Open_To_Buy  \
0              39  …       12691.0                  777          11914.0
1              44  …        8256.0                  864           7392.0
2              36  …        3418.0                    0           3418.0
3              34  …        3313.0                 2517            796.0
4              21  …        4716.0                    0           4716.0
5              36  …        4010.0                 1247           2763.0
6              46  …       34516.0                 2264          32252.0
7              27  …       29081.0                 1396          27685.0
8              36  …       22352.0                 2517          19835.0
9              36  …       11656.0                 1677           9979.0

   Total_Amt_Chng_Q4_Q1  Total_Trans_Amt  Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  \
0                 1.335             1144              42                1.625
1                 1.541             1291              33                3.714
2                 2.594             1887              20                2.333
3                 1.405             1171              20                2.333
4                 2.175              816              28                2.500
5                 1.376             1088              24                0.846
6                 1.975             1330              31                0.722
7                 2.204             1538              36                0.714
8                 3.355             1350              24                1.182
9                 1.524             1441              32                0.882

   Avg_Utilization_Ratio  \
0                  0.061
1                  0.105
2                  0.000
3                  0.760
4                  0.000
5                  0.311
6                  0.066
7                  0.048
8                  0.113
9                  0.144

   Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dep
endent_count_Education_Level_Months_Inactive_12_mon_1  \
0                                           0.000093
1                                           0.000057
2                                           0.000021
```

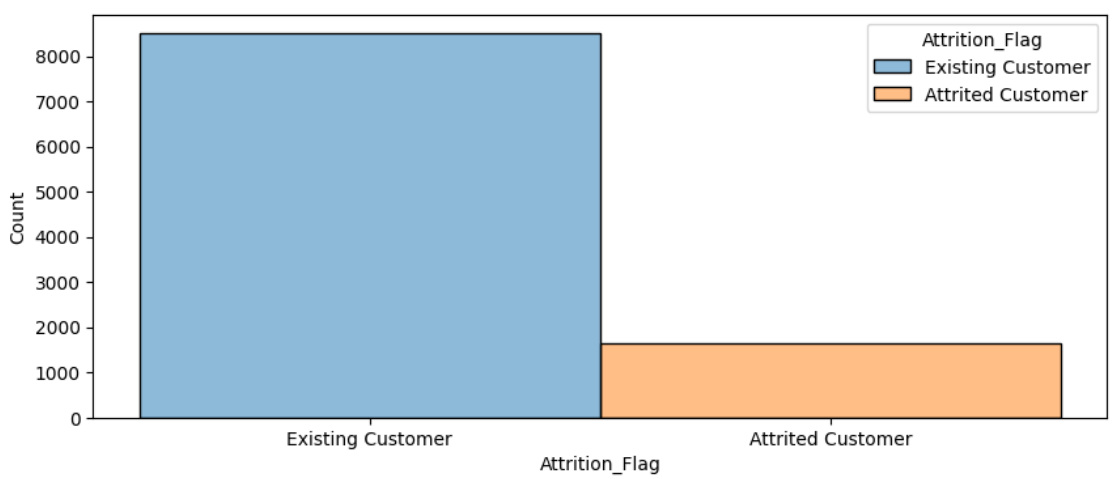
```
3                                            0.000134
4                                            0.000022
5                                            0.000055
6                                            0.000123
7                                            0.000086
8                                            0.000045
9                                            0.000303

   Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dep
endent_count_Education_Level_Months_Inactive_12_mon_2
0                                             0.99991
1                                             0.99994
2                                             0.99998
3                                             0.99987
4                                             0.99998
5                                             0.99994
6                                             0.99988
7                                             0.99991
8                                             0.99996
9                                             0.99970

[10 rows x 23 columns]
```

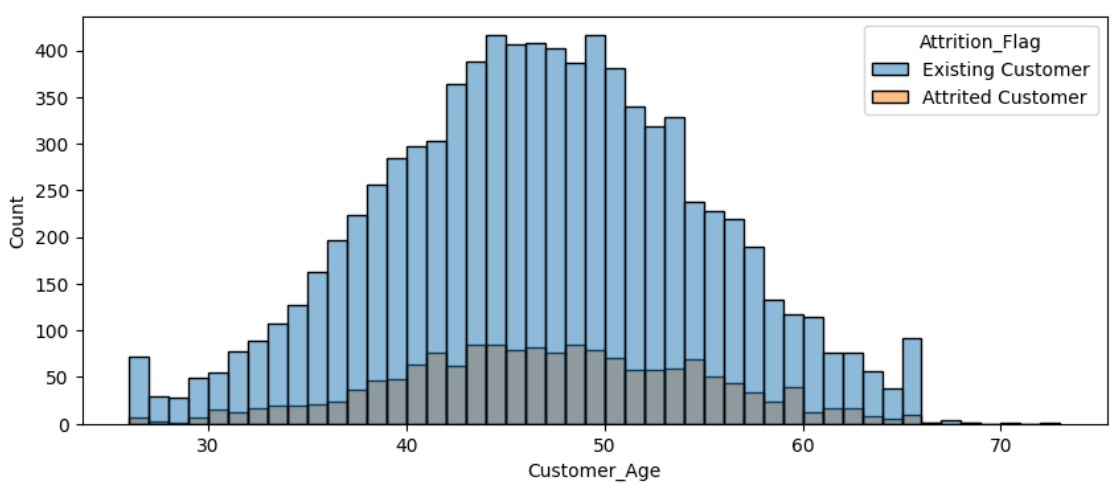
[ ]:
```python
# Dropping the Naive_Bayes_Classifier columns (There are 2 in the dataset)

df = df_explore.
 ↪drop(columns=['Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_De
 ↪'Naive_Bayes_Classifier_Attrition_Flag_Card_Category_Contacts_Count_12_mon_Dependent_count_
df.columns
```

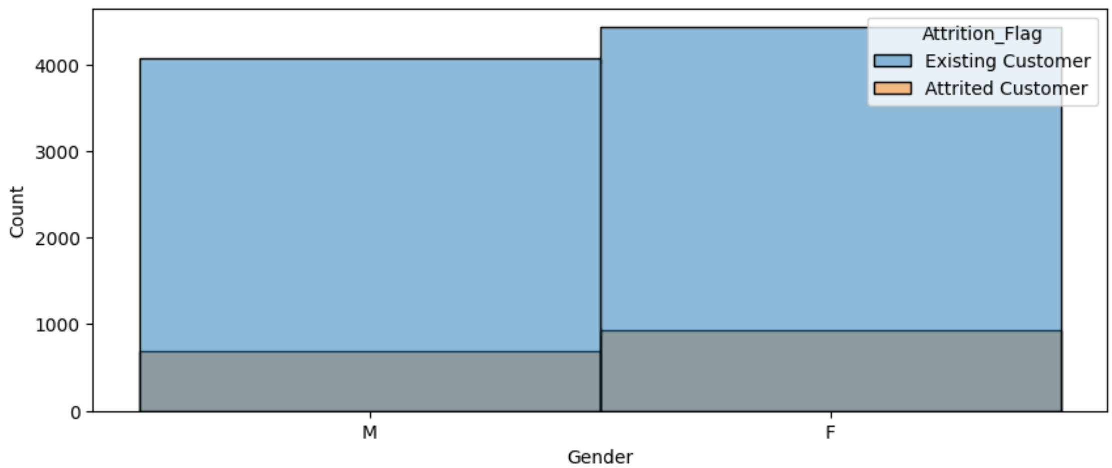
[ ]:
```
Index(['CLIENTNUM', 'Attrition_Flag', 'Customer_Age', 'Gender',
       'Dependent_count', 'Education_Level', 'Marital_Status',
       'Income_Category', 'Card_Category', 'Months_on_book',
       'Total_Relationship_Count', 'Months_Inactive_12_mon',
       'Contacts_Count_12_mon', 'Credit_Limit', 'Total_Revolving_Bal',
       'Avg_Open_To_Buy', 'Total_Amt_Chng_Q4_Q1', 'Total_Trans_Amt',
       'Total_Trans_Ct', 'Total_Ct_Chng_Q4_Q1', 'Avg_Utilization_Ratio'],
      dtype='object')
```

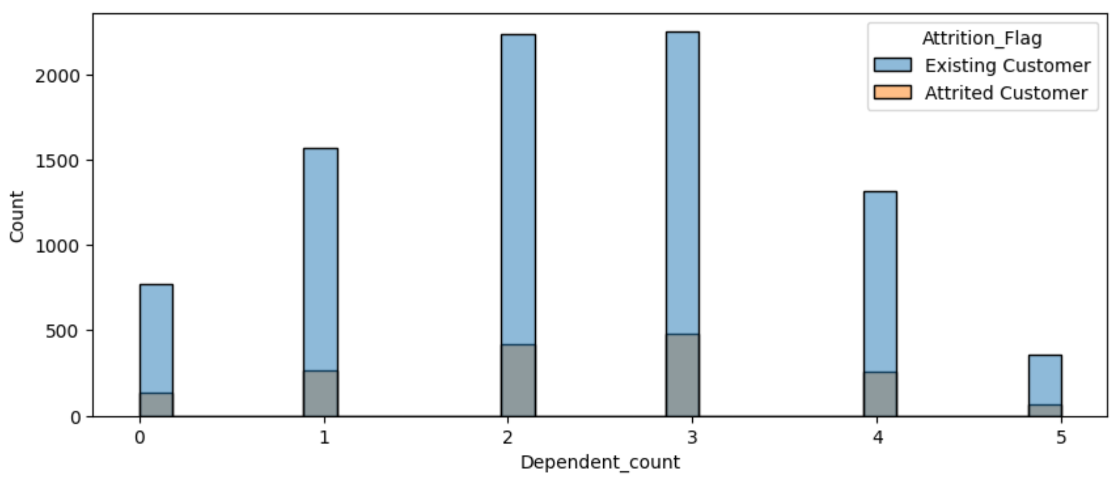
[ ]:
```python
for i in df.columns:
    if i not in ['NBC1','NBC2']:
        plt.figure(figsize=(10,4))
        sns.histplot(data=df,x=i,hue='Attrition_Flag')
        plt.show()
```

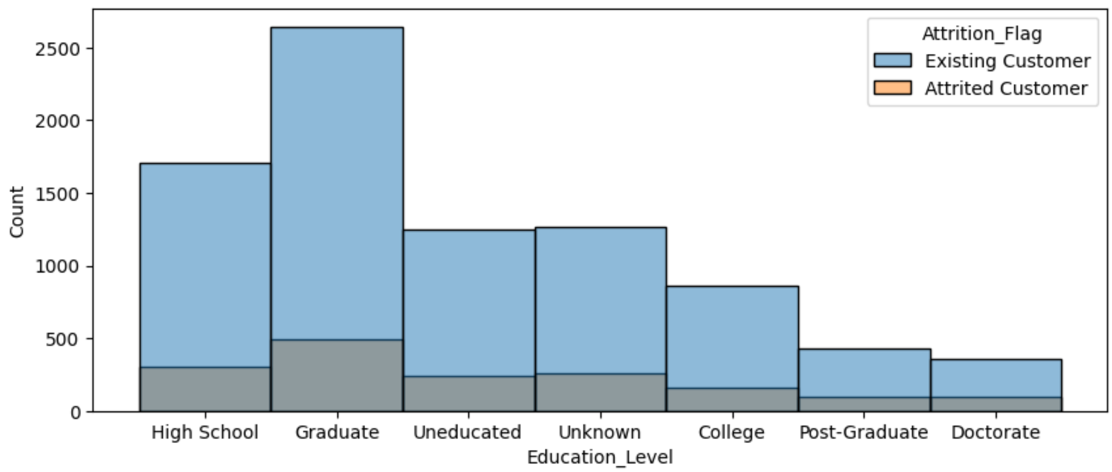

Attrition_Flag
Existing Customer
Attrited Customer
Count
0
500
1000
1500
2000
7.2
7.4
7.6
7.8
8.0
8.2
CLIENTNUM
1e8

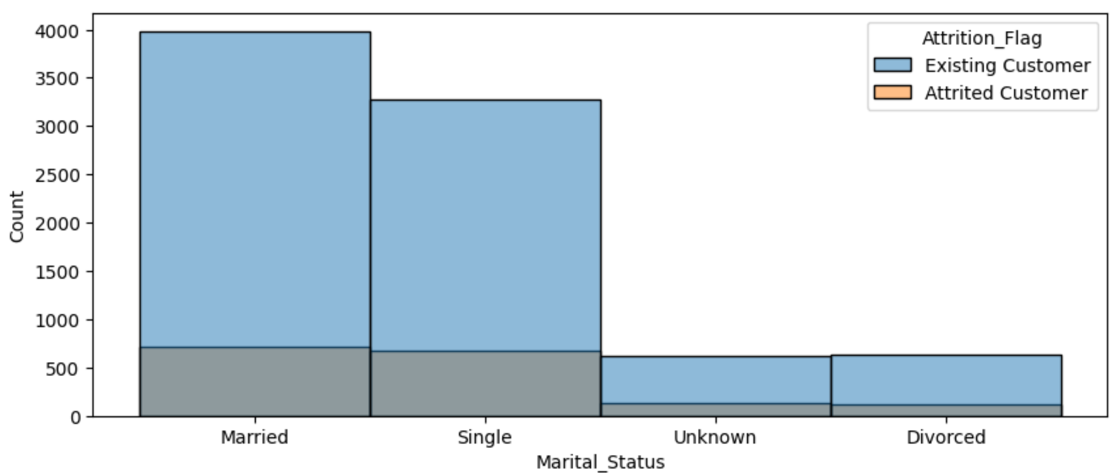

Attrition_Flag
Existing Customer
Attrited Customer
Count
0
1000
2000
3000
4000
5000
6000
7000
8000
Existing Customer
Attrited Customer
Attrition_Flag

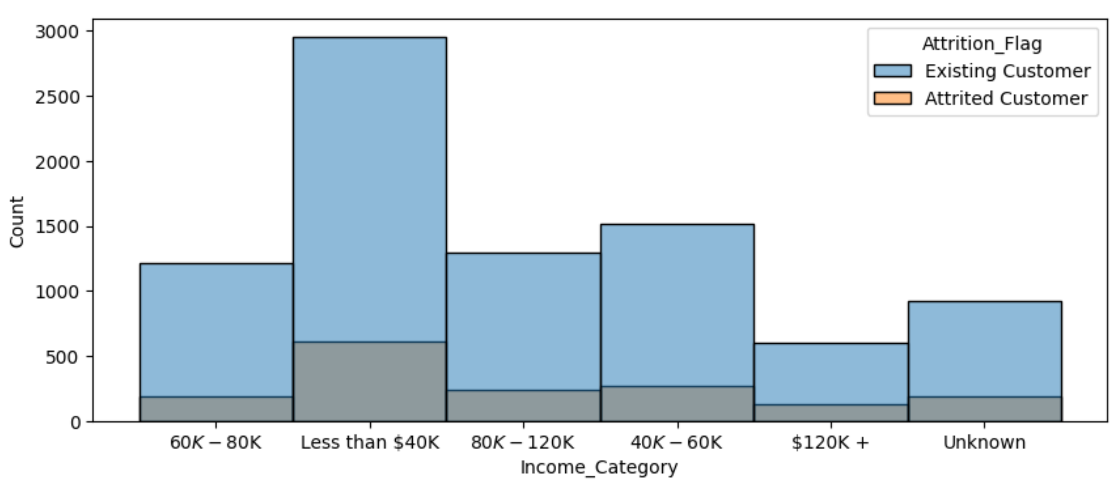

Attrition_Flag
Existing Customer
Attrited Customer
Count
0
50
100
150
200
250
300
350
400
30
40
50
60
70
Customer_Age

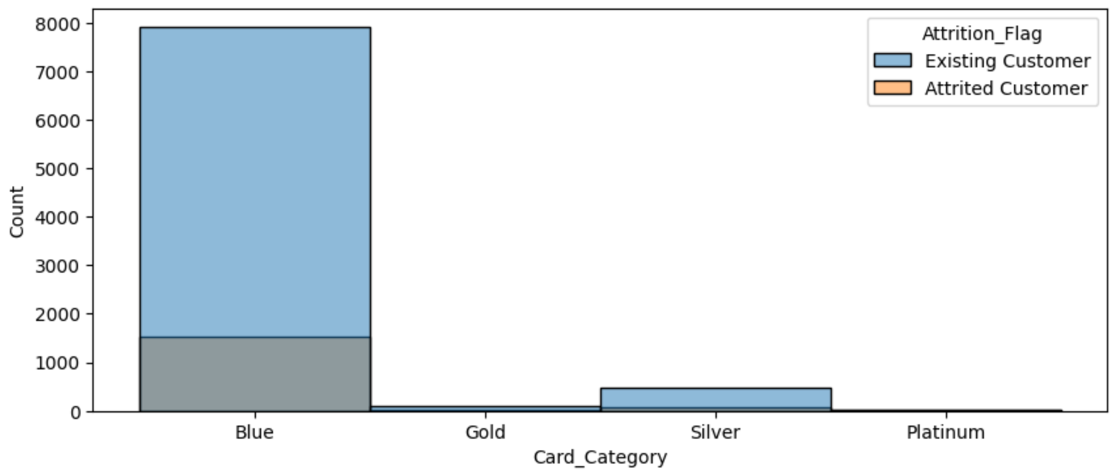

Attrition_Flag
Existing Customer
Attrited Customer
Count
Gender
M
F
0
1000
2000
3000
4000

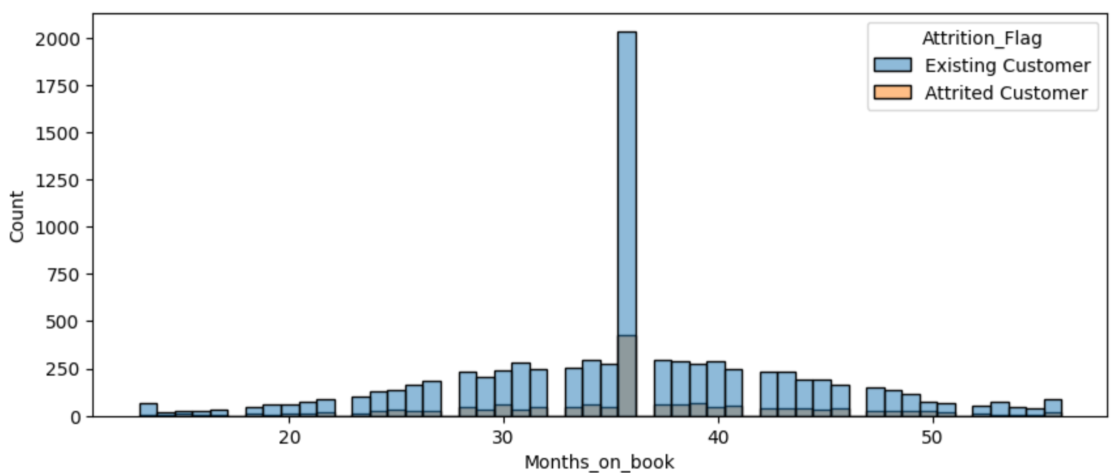

Attrition_Flag
Existing Customer
Attrited Customer
Count
Dependent_count
0
500
1000
1500
2000
0
1
2
3
4
5

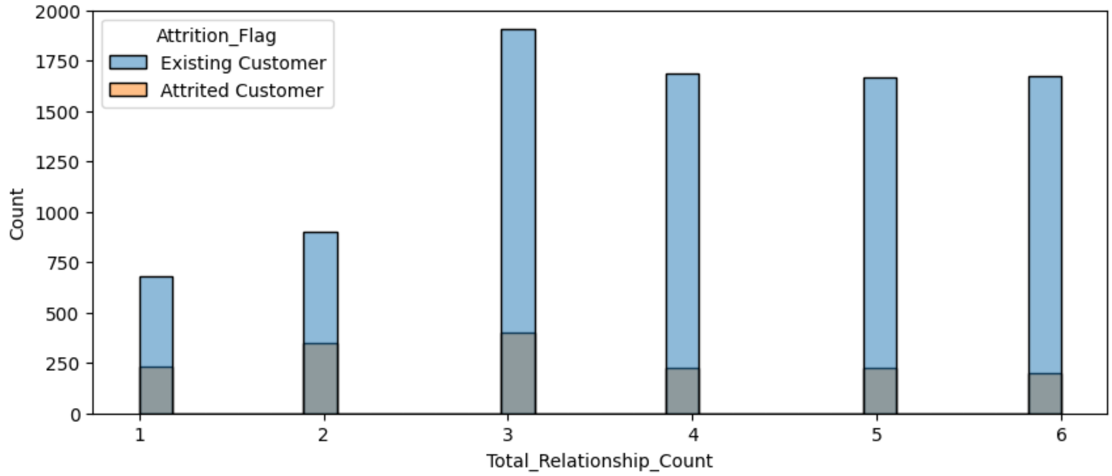

Attrition_Flag
Existing Customer
Attrited Customer
Count
Education_Level
High School
Graduate
Uneducated
Unknown
College
Post-Graduate
Doctorate

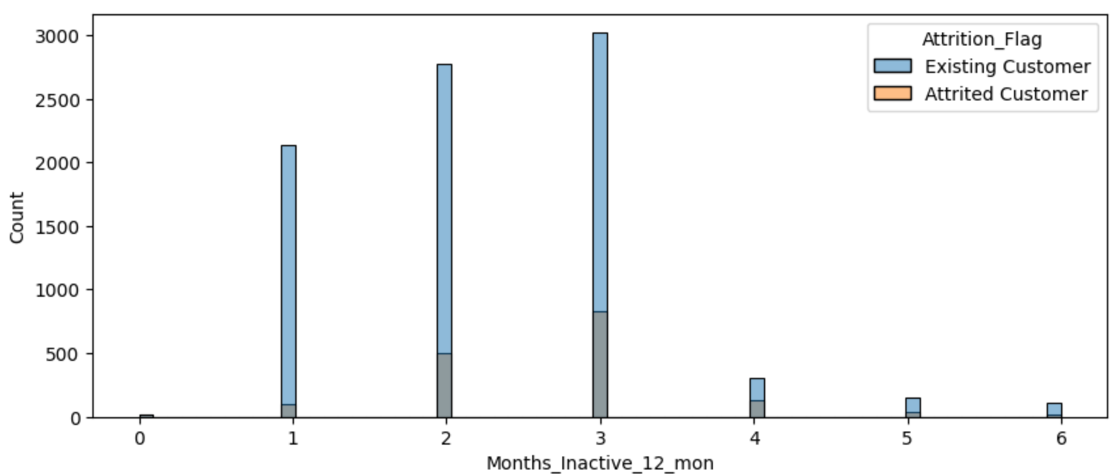

Attrition_Flag
Existing Customer
Attrited Customer
Count
Marital_Status
Married
Single
Unknown
Divorced

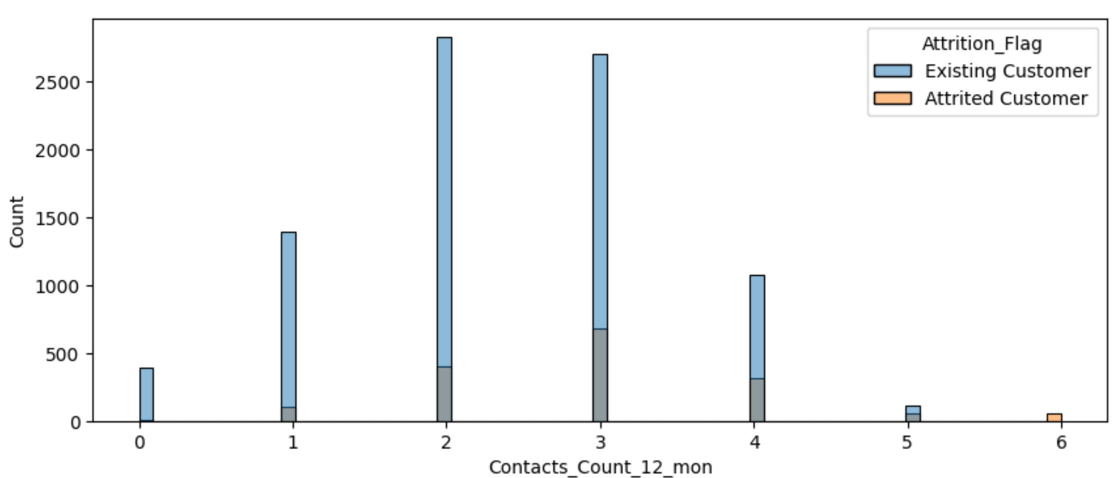

Attrition_Flag
Existing Customer
Attrited Customer
Count
Income_Category
60K − 80K
Less than $40K
80K − 120K
40K − 60K
$120K +
Unknown

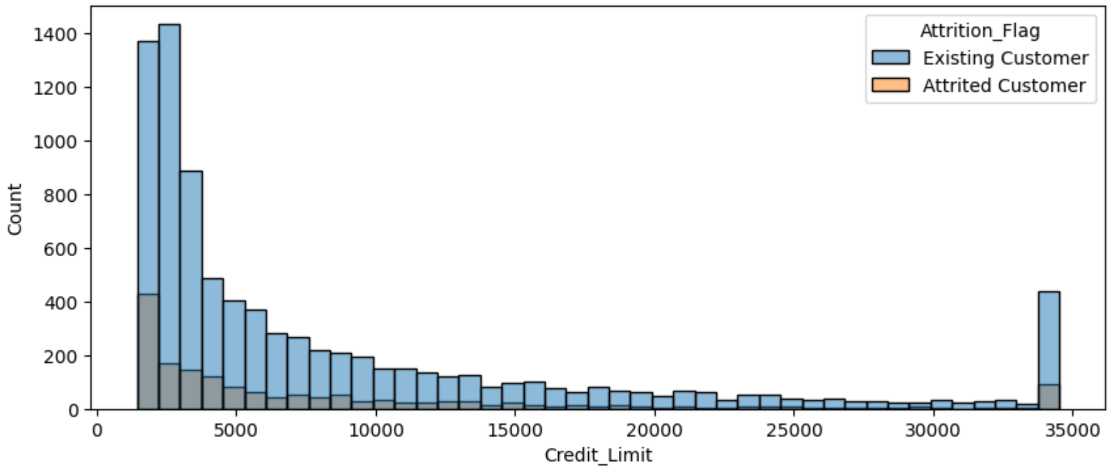

Attrition_Flag
Existing Customer
Attrited Customer
Count
8000
7000
6000
5000
4000
3000
2000
1000
0
Blue
Gold
Silver
Platinum
Card_Category


Attrition_Flag
Existing Customer
Attrited Customer
Count
2000
1750
1500
1250
1000
750
500
250
0
20
30
40
50
Months_on_book

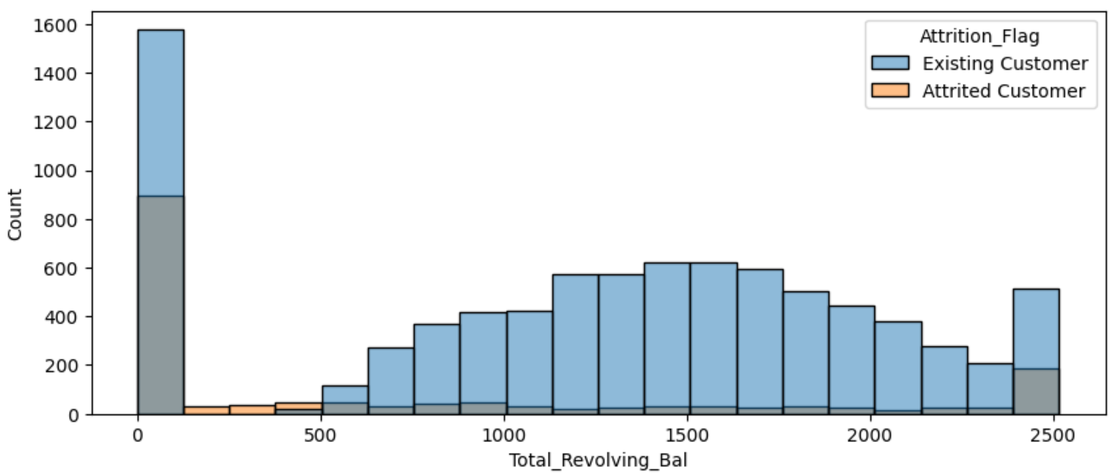

Attrition_Flag
Existing Customer
Attrited Customer
Count
Total_Relationship_Count


Attrition_Flag
Existing Customer
Attrited Customer
Count
Months_Inactive_12_mon

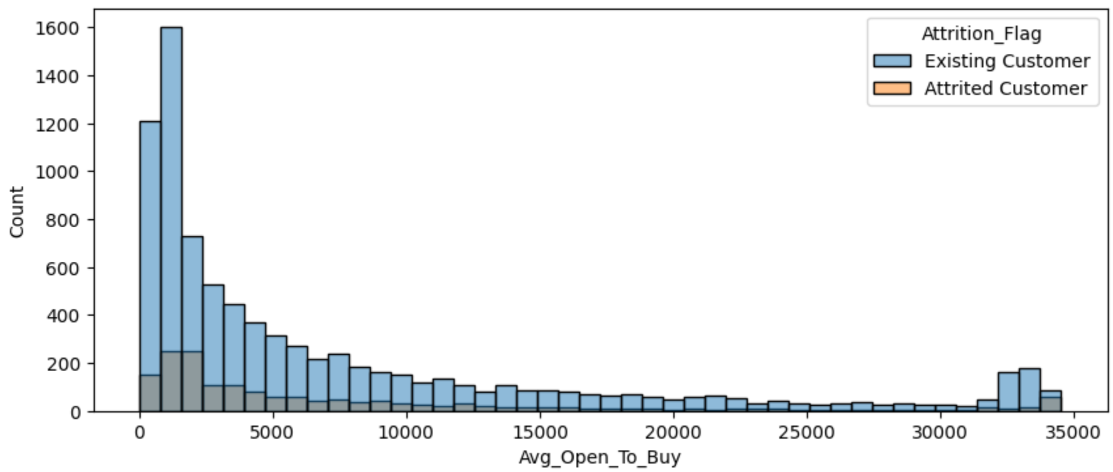

Attrition_Flag
Existing Customer
Attrited Customer
Count
Contacts_Count_12_mon

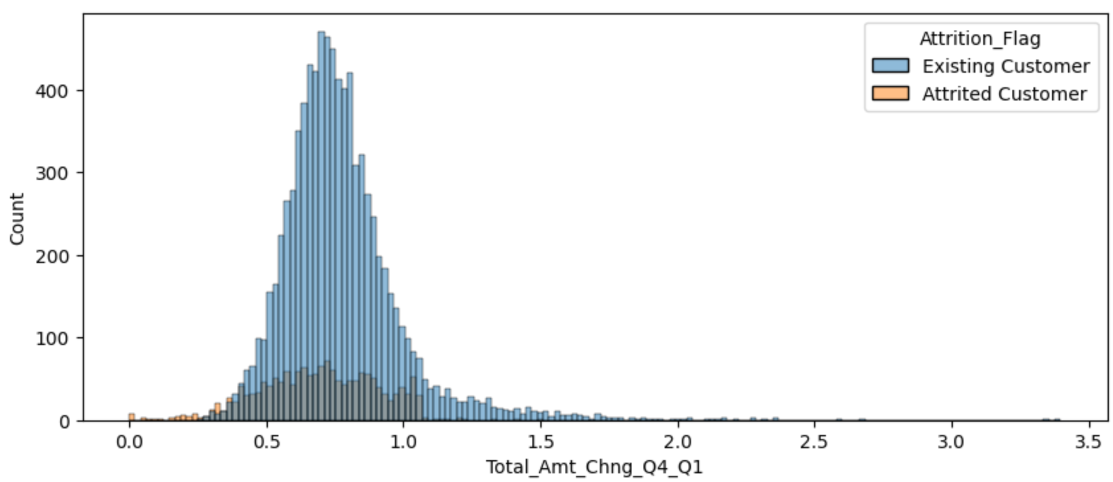


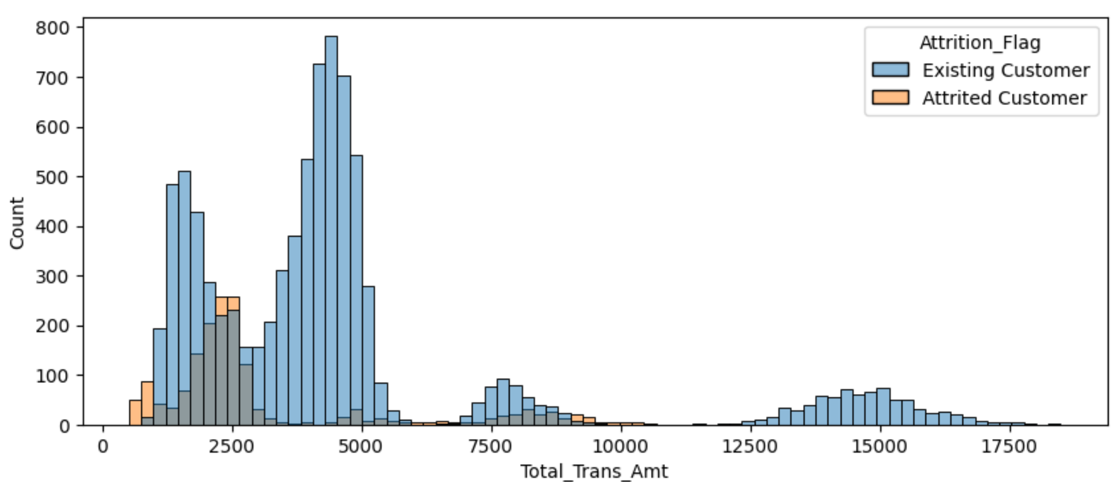
Attrition_Flag
Existing Customer
Attrited Customer
Count
Credit_Limit
0
5000
10000
15000
20000
25000
30000
35000
0
200
400
600
800
1000
1200
1400

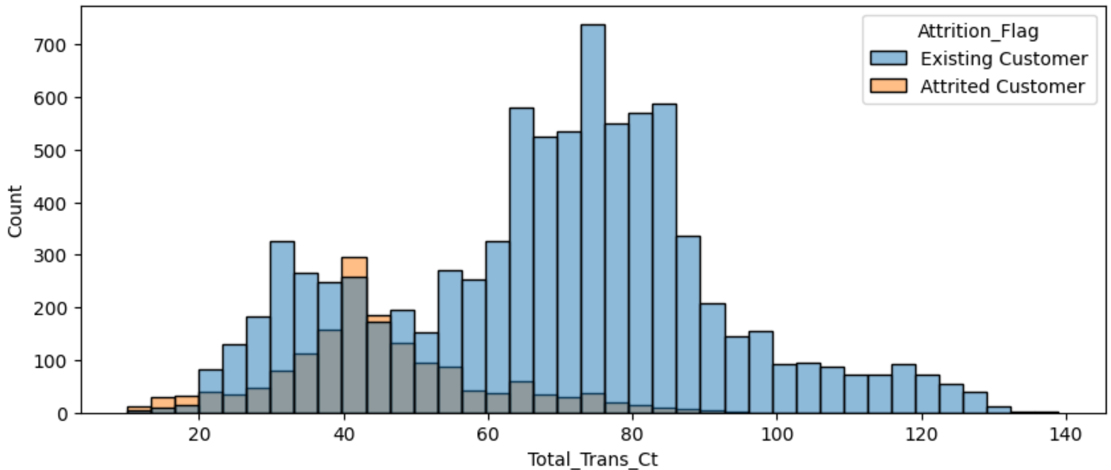


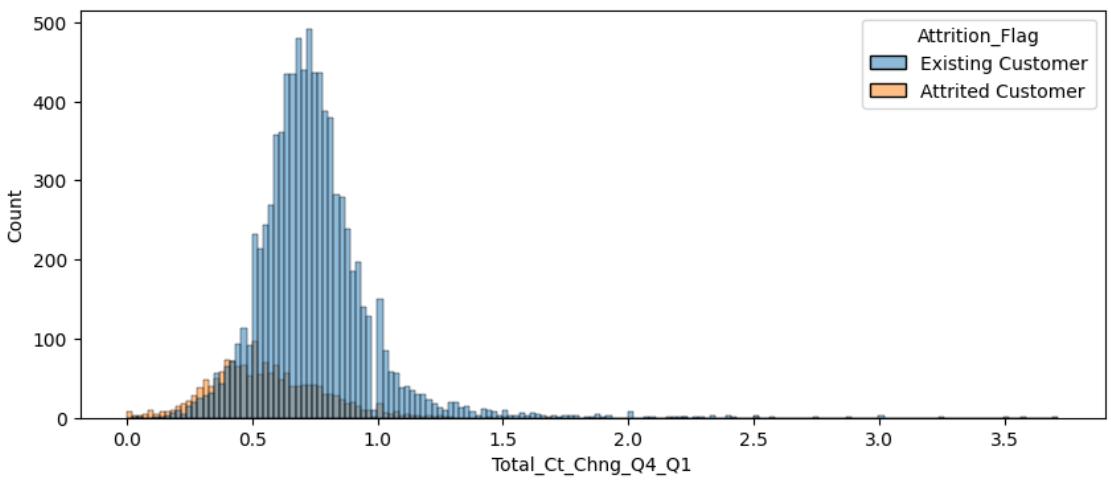
Attrition_Flag
Existing Customer
Attrited Customer
Count
Total_Revolving_Bal
0
500
1000
1500
2000
2500
0
200
400
600
800
1000
1200
1400
1600

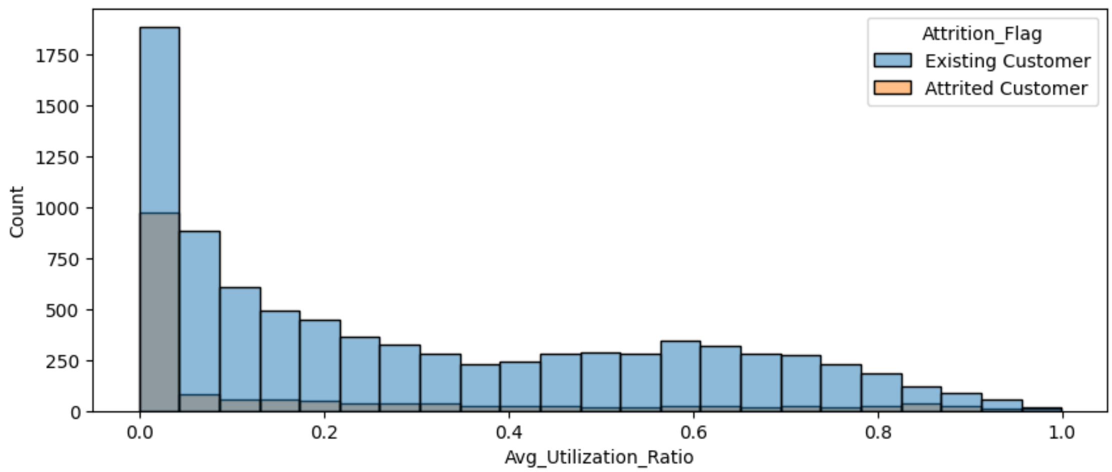

Attrition_Flag
Existing Customer
Attrited Customer
Count
Avg_Open_To_Buy


Attrition_Flag
Existing Customer
Attrited Customer
Count
Total_Amt_Chng_Q4_Q1


Attrition_Flag
Existing Customer
Attrited Customer
Count
Total_Trans_Amt


Attrition_Flag
Existing Customer
Attrited Customer
Count
Total_Trans_Ct


Attrition_Flag
Existing Customer
Attrited Customer
Count
Total_Ct_Chng_Q4_Q1



2.3 Converting Categorical Data into Numerical Data

We need to convert categorical data into numerical data as this would massively help in making sure that all data is accounted for its relation to the attriotion flag and helps us fully understand the relationship with the attrtition.

[ ]:
```
cat_to_num = [
    "Attrition_Flag",
    "Gender",
    "Education_Level",
    "Marital_Status",
    "Income_Category",
    "Card_Category"
]

cat_mapping = dict()
lenc = LabelEncoder()
for c in cat_to_num:
    cat_mapping[c] = lenc.fit_transform(df.loc[:, c])
cat_df = pd.DataFrame(cat_mapping)
```

[ ]:
```
numerical_features = [f for f in df.columns if f not in cat_to_num]
df_numerical = df.loc[:, numerical_features]
```

[ ]:
```
df_main = pd.concat([cat_df, df_numerical], axis=1)
display(df_main.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10127 entries, 0 to 10126
Data columns (total 21 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
```

```
 0   Attrition_Flag            10127 non-null  int64
 1   Gender                    10127 non-null  int64
 2   Education_Level           10127 non-null  int64
 3   Marital_Status            10127 non-null  int64
 4   Income_Category           10127 non-null  int64
 5   Card_Category             10127 non-null  int64
 6   CLIENTNUM                 10127 non-null  int64
 7   Customer_Age              10127 non-null  int64
 8   Dependent_count           10127 non-null  int64
 9   Months_on_book            10127 non-null  int64
 10  Total_Relationship_Count  10127 non-null  int64
 11  Months_Inactive_12_mon    10127 non-null  int64
 12  Contacts_Count_12_mon     10127 non-null  int64
 13  Credit_Limit              10127 non-null  float64
 14  Total_Revolving_Bal       10127 non-null  int64
 15  Avg_Open_To_Buy           10127 non-null  float64
 16  Total_Amt_Chng_Q4_Q1      10127 non-null  float64
 17  Total_Trans_Amt           10127 non-null  int64
 18  Total_Trans_Ct            10127 non-null  int64
 19  Total_Ct_Chng_Q4_Q1       10127 non-null  float64
 20  Avg_Utilization_Ratio     10127 non-null  float64
dtypes: float64(5), int64(16)
memory usage: 1.6 MB

None
```

[ ]:
```
df_main.head(20)
```

[ ]:
```
    Attrition_Flag  Gender  Education_Level  Marital_Status  Income_Category  \
0                1       1                3               1                2
1                1       0                2               2                4
2                1       1                2               1                3
3                1       0                3               3                4
4                1       1                5               1                2
5                1       1                2               1                1
6                1       1                6               1                0
7                1       1                3               3                2
8                1       1                5               2                2
9                1       1                2               2                3
10               1       1                5               3                0
11               1       1                6               1                1
12               1       1                0               2                3
13               1       1                2               3                2
14               1       0                2               1                4
15               1       1                6               3                3
16               1       1                4               2                3
17               1       1                6               1                3
18               1       1                3               1                1
```

19	1	0	2	1	5

	Card_Category	CLIENTNUM	Customer_Age	Dependent_count	Months_on_book \
0	0	768805383	45	3	39
1	0	818770008	49	5	44
2	0	713982108	51	3	36
3	0	769911858	40	4	34
4	0	709106358	40	3	21
5	0	713061558	44	2	36
6	1	810347208	51	4	46
7	3	818906208	32	0	27
8	0	710930508	37	3	36
9	0	719661558	48	2	36
10	0	708790833	42	5	31
11	0	710821833	65	1	54
12	0	710599683	56	1	36
13	0	816082233	35	3	30
14	0	712396908	57	2	48
15	0	714885258	44	4	37
16	0	709967358	48	4	36
17	0	753327333	41	3	34
18	0	806160108	61	1	56
19	0	709327383	45	2	37

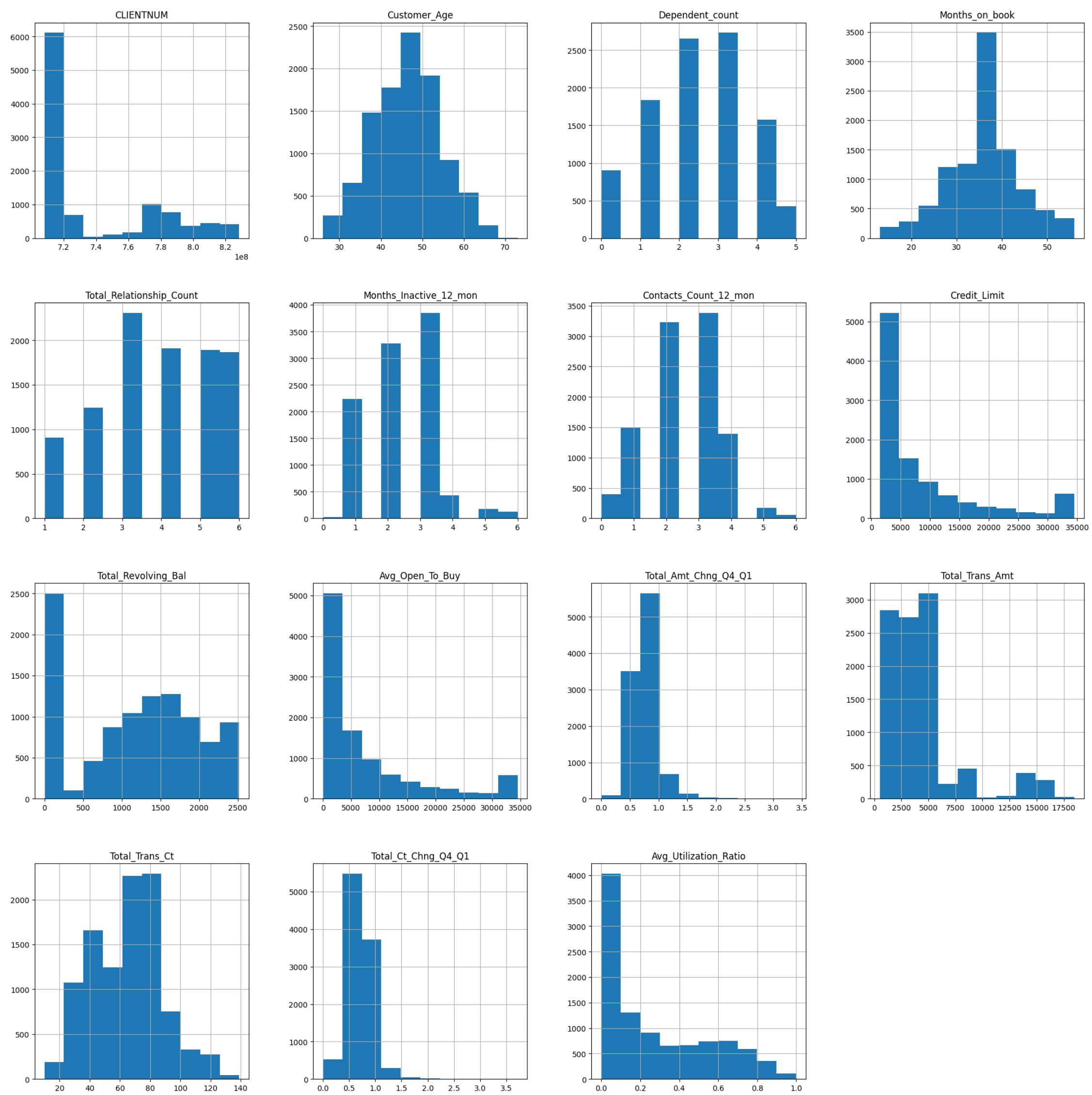
	…	Months_Inactive_12_mon	Contacts_Count_12_mon	Credit_Limit \
0	…	1	3	12691.0
1	…	1	2	8256.0
2	…	1	0	3418.0
3	…	4	1	3313.0
4	…	1	0	4716.0
5	…	1	2	4010.0
6	…	1	3	34516.0
7	…	2	2	29081.0
8	…	2	0	22352.0
9	…	3	3	11656.0
10	…	3	2	6748.0
11	…	2	3	9095.0
12	…	6	0	11751.0
13	…	1	3	8547.0
14	…	2	2	2436.0
15	…	1	2	4234.0
16	…	2	3	30367.0
17	…	4	1	13535.0
18	…	2	3	3193.0
19	…	1	2	14470.0

	Total_Revolving_Bal	Avg_Open_To_Buy	Total_Amt_Chng_Q4_Q1 \

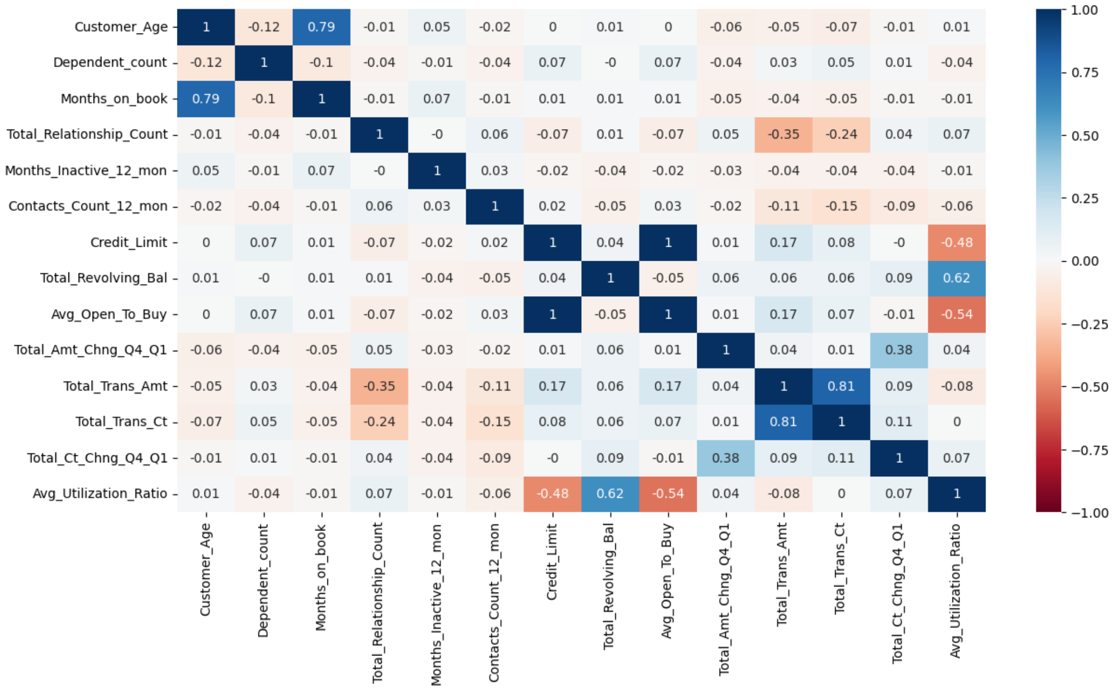
```
0                777          11914.0                  1.335
1                864           7392.0                  1.541
2                  0           3418.0                  2.594
3               2517            796.0                  1.405
4                  0           4716.0                  2.175
5               1247           2763.0                  1.376
6               2264          32252.0                  1.975
7               1396          27685.0                  2.204
8               2517          19835.0                  3.355
9               1677           9979.0                  1.524
10              1467           5281.0                  0.831
11              1587           7508.0                  1.433
12                 0          11751.0                  3.397
13              1666           6881.0                  1.163
14               680           1756.0                  1.190
15               972           3262.0                  1.707
16              2362          28005.0                  1.708
17              1291          12244.0                  0.653
18              2517            676.0                  1.831
19              1157          13313.0                  0.966

    Total_Trans_Amt  Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  \
0              1144              42                1.625
1              1291              33                3.714
2              1887              20                2.333
3              1171              20                2.333
4               816              28                2.500
5              1088              24                0.846
6              1330              31                0.722
7              1538              36                0.714
8              1350              24                1.182
9              1441              32                0.882
10             1201              42                0.680
11             1314              26                1.364
12             1539              17                3.250
13             1311              33                2.000
14             1570              29                0.611
15             1348              27                1.700
16             1671              27                0.929
17             1028              21                1.625
18             1336              30                1.143
19             1207              21                0.909

    Avg_Utilization_Ratio
0                   0.061
1                   0.105
2                   0.000
```

```
3                    0.760
4                    0.000
5                    0.311
6                    0.066
7                    0.048
8                    0.113
9                    0.144
10                   0.217
11                   0.174
12                   0.000
13                   0.195
14                   0.279
15                   0.230
16                   0.078
17                   0.095
18                   0.788
19                   0.080

[20 rows x 21 columns]
```

[ ]:
```
# Plot for Numerical Data

fig = plt.figure(figsize = (20,20))
ax = fig.gca()
plt.tight_layout()
df.loc[:, ~df_main.columns.isin(['CLIENTNUM', "Attrition_Flag_Int"])].hist(ax =
 ↪ax)
fig.show()
```



[ ]:
```
correlation = df.loc[:, ~df.columns.isin(['CLIENTNUM'])].corr().round(2)
plt.figure(figsize = (14,7))
sns.heatmap(correlation, annot = True, cmap = 'RdBu', vmin=-1, vmax=1)
plt.show()
```



2.3.1 Observing the heatmap above, we can state the following.

-
-
-

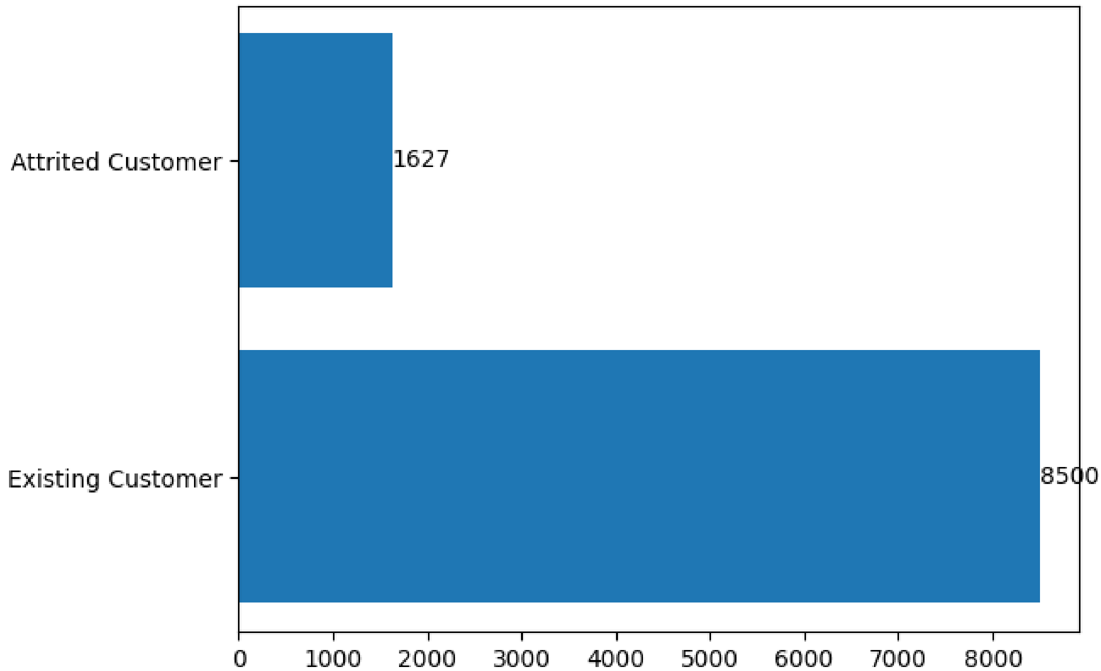
[ ]:
```
# Counting the Attrition Flags
# Attrited Customer = 0, Existing Customer = 1.

aggregate_data = df_main.groupby(['Attrition_Flag'])['CLIENTNUM'].count().
 ↪to_frame().reset_index()
aggregate_data
```

[ ]:
```
   Attrition_Flag  CLIENTNUM
0               0       1627
1               1       8500
```

[ ]:
```
x = ['Existing Customer','Attrited Customer']
y = [df['Attrition_Flag'].value_counts()[0],df['Attrition_Flag'].
 ↪value_counts()[1]]
fig, ax = plt.subplots()
bars = ax.barh(x, y)
```

```
ax.bar_label(bars)
```

[ ]: [Text(0, 0, '8500'), Text(0, 0, '1627')]



[ ]:
```
# Put all the columns except Attrition_Flag in X_data and only the
↪Attrition_Flag in y_data

X_data = df_main[[i for i in df_main.columns if i != "Attrition_Flag"]]
y_data = df_main["Attrition_Flag"]
```

[ ]:
```
params = {
    "max_depth": range(1, 9),
    "min_samples_split": [5, 7, 9, 12, 15],
    "min_samples_leaf": [5, 7, 9, 10, 12]
}

s_kfold = StratifiedKFold(n_splits=7, shuffle=True, random_state=42)

estimator = RandomForestClassifier(random_state=42)
scoring = make_scorer(f1_score)

grid_search = GridSearchCV(
    estimator=estimator,
```

```
    param_grid=params,
    scoring=scoring,
    cv=s_kfold,
    n_jobs=-1
)

search_results = grid_search.fit(X_data, y_data)

best_params = search_results.best_params_
best_score = search_results.best_score_

print(f"Best parameters: {best_params}")
print(f"Best score: {best_score}")
```

```
Best parameters: {'max_depth': 8, 'min_samples_leaf': 5, 'min_samples_split':
15}
Best score: 0.9672994684011581
```

[ ]:
```
search_results.best_params_
```

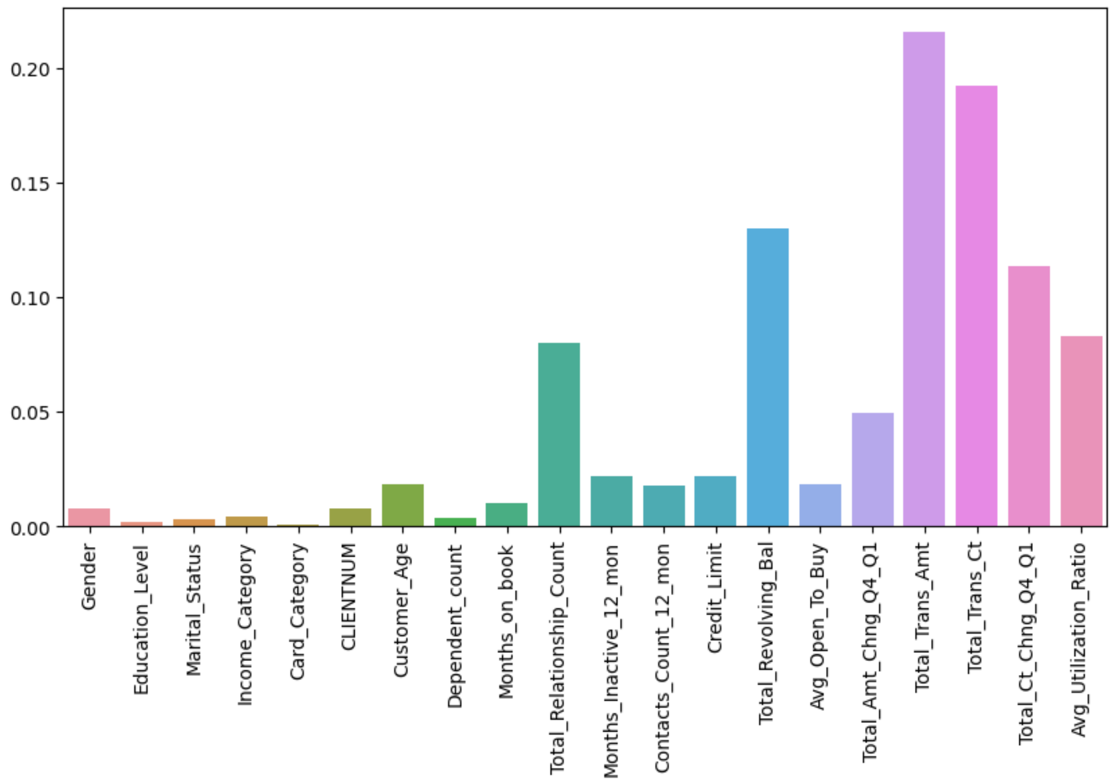
[ ]:
```
{'max_depth': 8, 'min_samples_leaf': 5, 'min_samples_split': 15}
```

[ ]:
```
fi = pd.Series(search_results.best_estimator_.feature_importances_, index =
 ↪X_data.columns)
fi
```

[ ]:
```
Gender                      0.007973
Education_Level             0.001810
Marital_Status              0.002889
Income_Category             0.004264
Card_Category               0.000735
CLIENTNUM                   0.007766
Customer_Age                0.018100
Dependent_count             0.003545
Months_on_book              0.010033
Total_Relationship_Count    0.079702
Months_Inactive_12_mon      0.021845
Contacts_Count_12_mon       0.017897
Credit_Limit                0.021925
Total_Revolving_Bal         0.130040
Avg_Open_To_Buy             0.018592
Total_Amt_Chng_Q4_Q1        0.049282
Total_Trans_Amt             0.215404
Total_Trans_Ct              0.191910
Total_Ct_Chng_Q4_Q1         0.113417
Avg_Utilization_Ratio       0.082870
```

```
dtype: float64
```

[ ]:
```
plt.figure(figsize=(10,5))
ax = plt.gca()

sns.barplot(x=fi.index, y=fi, ax=ax)
ax.tick_params(rotation=90, axis='x');
```



Thus we can say that the most important features to classify attrition are as follows -

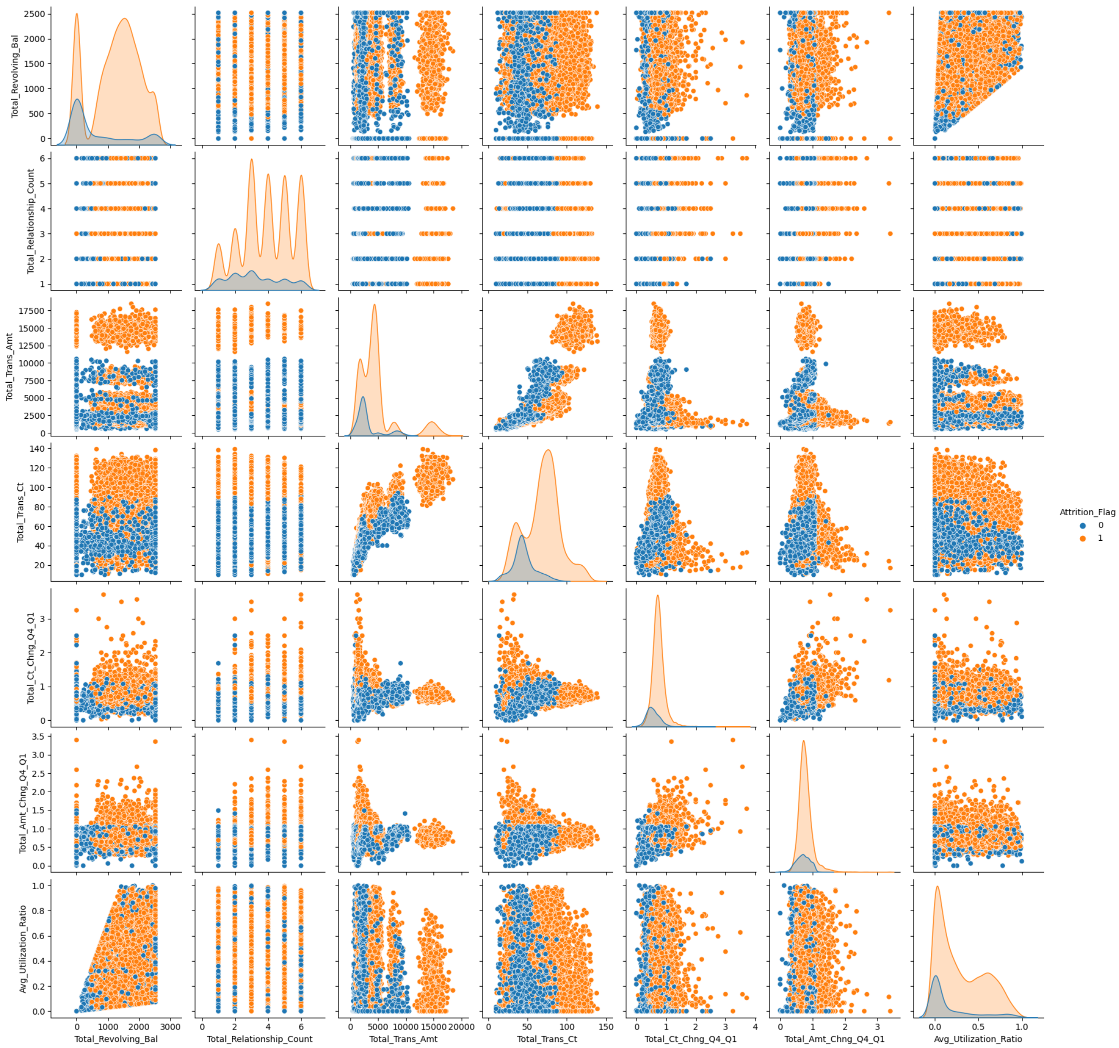
- ***Total_Relationship_Count***
- ***Total_Revolving_Bal***
- ***Total_Amt_Chng_Q4_Q1***
- ***Total_Trans_Amt***
- ***Total_Trans_Ct***
- ***Total_Ct_Chng_Q4_Q1***
- ***Avg_Utilization_Ratio***

[ ]:
```
important_features = fi[fi >= 0.04].index.to_list()
important_features
```

[ ]:
```
['Total_Relationship_Count',
 'Total_Revolving_Bal',
 'Total_Amt_Chng_Q4_Q1',
 'Total_Trans_Amt',
 'Total_Trans_Ct',
 'Total_Ct_Chng_Q4_Q1',
 'Avg_Utilization_Ratio']
```

[ ]:
```
reduced_data = df_main.loc[:,['Total_Revolving_Bal',
 'Total_Relationship_Count',
 'Total_Trans_Amt',
 'Total_Trans_Ct',
 'Total_Ct_Chng_Q4_Q1',
 'Total_Amt_Chng_Q4_Q1',
 'Avg_Utilization_Ratio',
'Attrition_Flag']]

df_reduced = reduced_data.copy()
```

[ ]:
```
sns.pairplot(reduced_data, hue='Attrition_Flag', diag_kind='kde');
```



[ ]:
```
std_scaler = StandardScaler()
reduced_data = std_scaler.fit_transform(reduced_data)

std_scaler2 = StandardScaler()
full_data = std_scaler2.fit_transform(df_main)
```

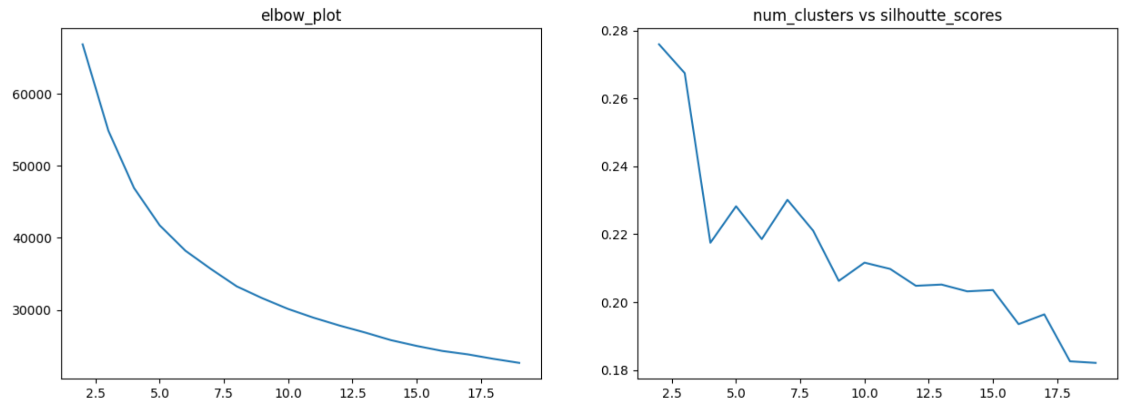
[ ]:
```
'''lis_main = []
sil_score_main = []
for k in range(2, 20):
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(full_data)
    predict = model.fit_predict(full_data)

    lis_main.append(model.inertia_)
    sil_score_main.append(silhouette_score(full_data,predict))
```

```
    '''
```

```
[ ]: 'lis_main = []\nsil_score_main = []\nfor k in range(2, 20):\n    model =
KMeans(n_clusters=k, random_state=42)\n    model.fit(full_data)\n    predict =
model.fit_predict(full_data)\n\n    lis_main.append(model.inertia_)\n
sil_score_main.append(silhouette_score(full_data,predict))\n    '
```

```
[ ]: lis = []
sil_score_reduced = []
for k in range(2, 20):
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(reduced_data)
    predict = model.fit_predict(reduced_data)

    lis.append(model.inertia_)
    sil_score_reduced.append(silhouette_score(reduced_data,predict))
```

```
[ ]: f, ax = plt.subplots(nrows=1, ncols=2, figsize=(15,5))
sns.lineplot(x=range(2,20), y=lis, ax=ax[0])
ax[0].set_title("elbow_plot");

sns.lineplot(x=range(2,20), y=sil_score_reduced, ax=ax[1])
ax[1].set_title("num_clusters vs silhoutte_scores");
```



silhoutte score at K = 6. This is the best one we could get. We see elbow at 6 so it is the best value as per the observation

```
[ ]: kmeans_model = KMeans(n_clusters=6, random_state=42)
kmeans_model.fit(reduced_data)
kmeans_prediction = kmeans_model.fit_predict(reduced_data)

silhouette_score(reduced_data,kmeans_prediction)
```

[ ]: 0.2185546719412025

```
[ ]: # inverse transform and assign to kmeans_predicted
     kmeans_predicted = pd.DataFrame(reduced_data)
     kmeans_predicted["pred_clusters"] = kmeans_prediction
```

```
[ ]: kmeans_predicted.groupby("pred_clusters").count().iloc[:, 1]
```

```
[ ]: pred_clusters
     0    2334
     1    1883
     2    2498
     3    1494
     4    1124
     5     794
     Name: 1, dtype: int64
```

```
[ ]: print(kmeans_predicted.columns)
```

```
Index([0, 1, 2, 3, 4, 5, 6, 7, 'pred_clusters'], dtype='object')
```

```
[ ]: kmeans_predicted.head()
```

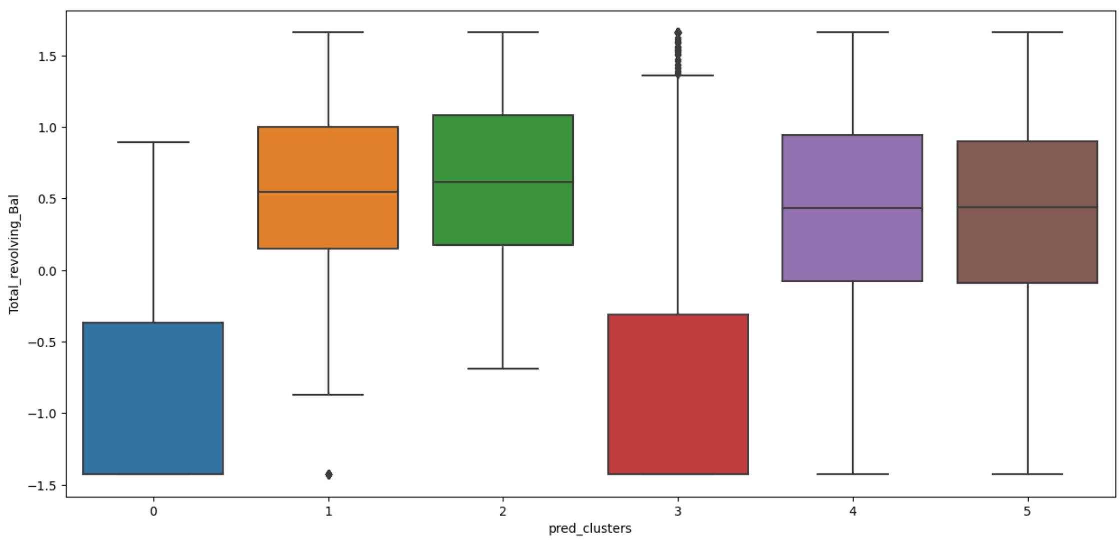
```
[ ]:          0         1         2         3          4         5         6  \
     0 -0.473422  0.763943 -0.959707 -0.973895   3.834003  2.623494 -0.775882
     1 -0.366667  1.407306 -0.916433 -1.357340  12.608573  3.563293 -0.616276
     2 -1.426858  0.120579 -0.740982 -1.911206   6.807864  8.367214 -0.997155
     3  1.661686 -0.522785 -0.951758 -1.911206   6.807864  2.942843  1.759686
     4 -1.426858  0.763943 -1.056263 -1.570365   7.509325  6.455682 -0.997155

               7  pred_clusters
     0  0.437506              5
     1  0.437506              5
     2  0.437506              5
     3  0.437506              5
     4  0.437506              5
```

```
[ ]: kmeans_predicted.rename(columns = {0:'Total_revolving_Bal'}, inplace = True)
     kmeans_predicted.rename(columns={1:'Total_Relationship_Count'}, inplace=True)
     kmeans_predicted.rename(columns = {2:'Total_Trans_Amt'}, inplace = True)
     kmeans_predicted.rename(columns = {3:'Total_Trans_Ct'}, inplace = True)
     kmeans_predicted.rename(columns = {4:'Total_Ct_Chng_Q4_Q1'}, inplace = True)
     kmeans_predicted.rename(columns = {5:'Total_Amt_Chng_Q4_Q1'}, inplace = True)
     kmeans_predicted.rename(columns = {6:'Avg_Utilization_Ratio'}, inplace = True)
     kmeans_predicted.rename(columns = {7:'Attrition_Flag'}, inplace = True)
```

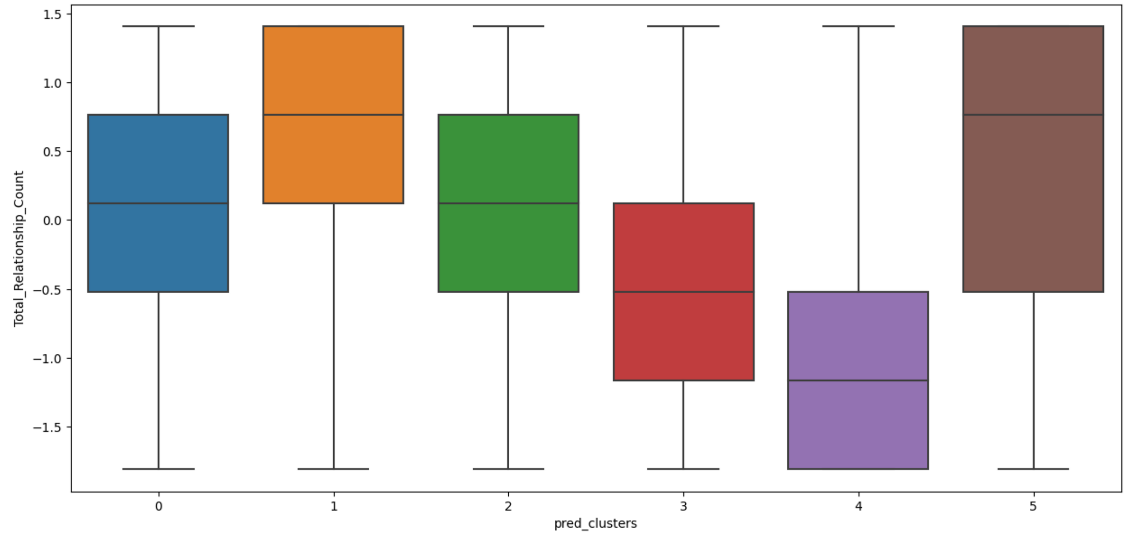
```
[ ]: kmeans_predicted.head()
```

[ ]:
```
   Total_revolving_Bal  Total_Relationship_Count  Total_Trans_Amt  \
0            -0.473422                  0.763943        -0.959707
1            -0.366667                  1.407306        -0.916433
2            -1.426858                  0.120579        -0.740982
3             1.661686                 -0.522785        -0.951758
4            -1.426858                  0.763943        -1.056263

   Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  Total_Amt_Chng_Q4_Q1  \
0       -0.973895             3.834003              2.623494
1       -1.357340            12.608573              3.563293
2       -1.911206             6.807864              8.367214
3       -1.911206             6.807864              2.942843
4       -1.570365             7.509325              6.455682

   Avg_Utilization_Ratio  Attrition_Flag  pred_clusters
0              -0.775882        0.437506              5
1              -0.616276        0.437506              5
2              -0.997155        0.437506              5
3               1.759686        0.437506              5
4              -0.997155        0.437506              5
```

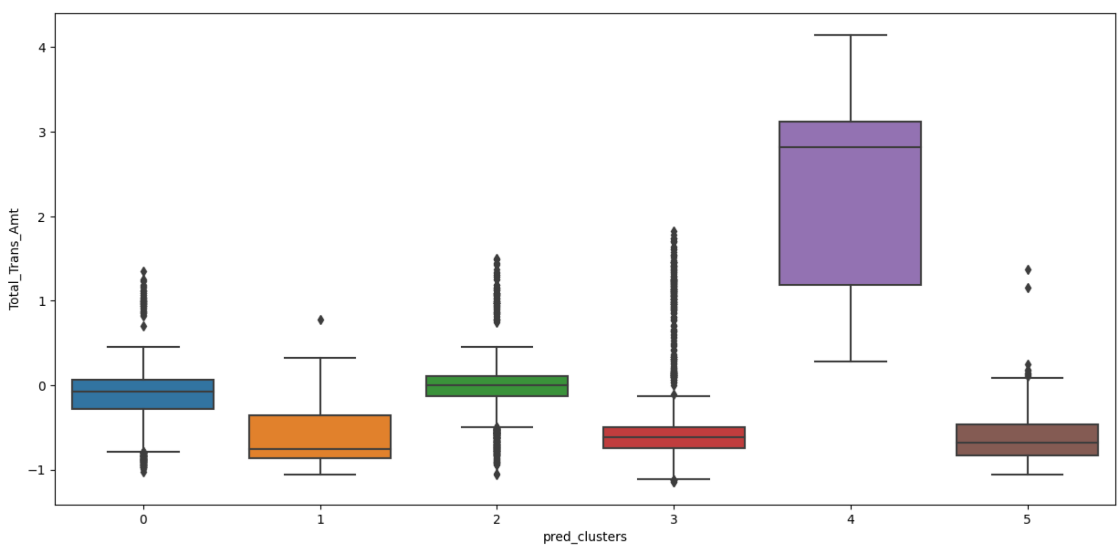
[ ]:
```
plt.figure(figsize=(15,7))
sns.boxplot(y=kmeans_predicted['Total_revolving_Bal'],␣
↪x=kmeans_predicted["pred_clusters"]);
```



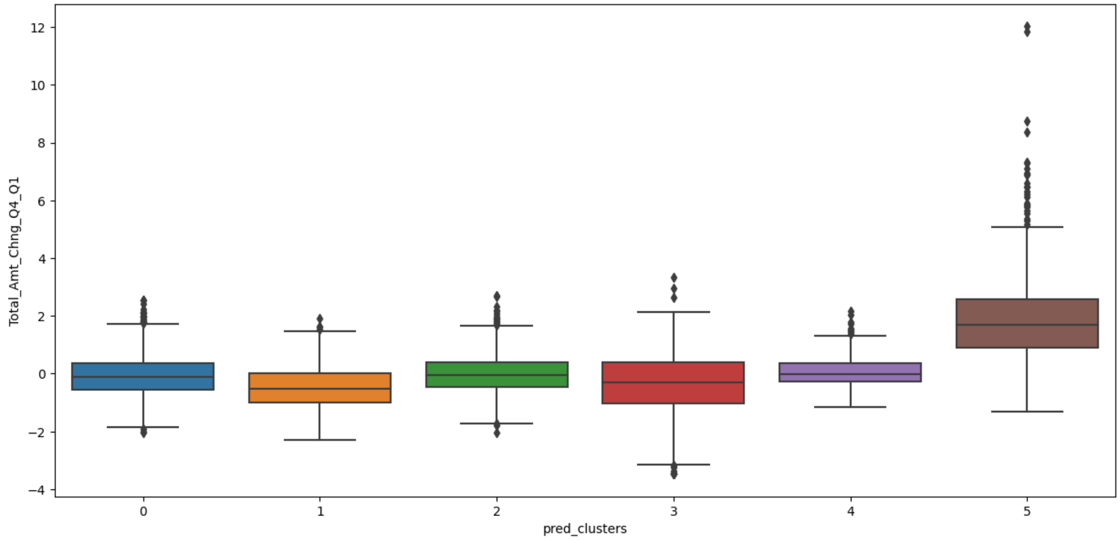
[ ]:
```
plt.figure(figsize=(15,7))
sns.boxplot(y=kmeans_predicted["Total_Relationship_Count"],␣
↪x=kmeans_predicted["pred_clusters"]);
```



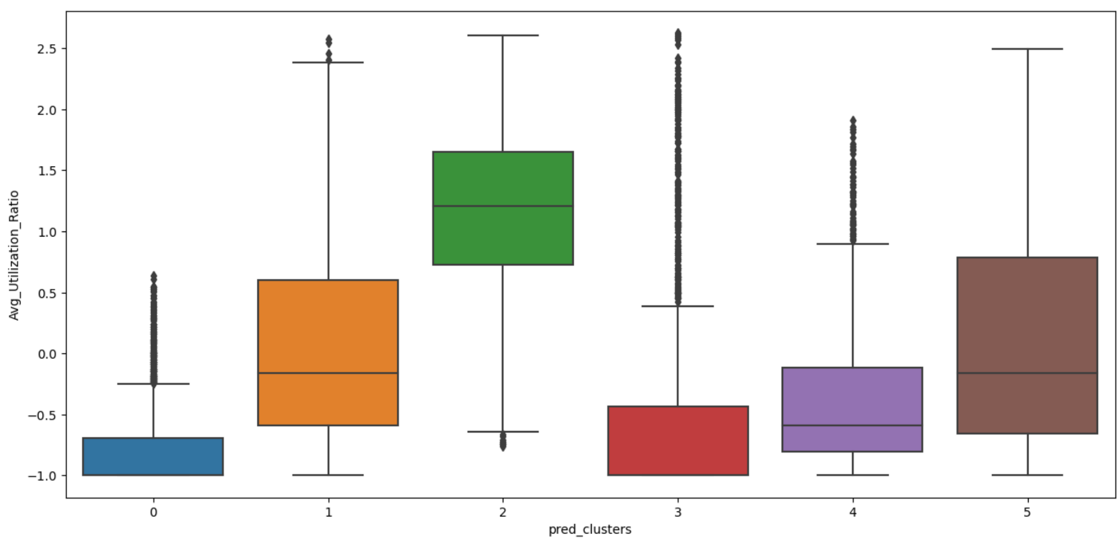

```
[ ]: plt.figure(figsize=(15,7))
     sns.boxplot(y=kmeans_predicted["Total_Trans_Amt"],␣
      ↪x=kmeans_predicted["pred_clusters"]);
```




```
[ ]: plt.figure(figsize=(15,7))
     sns.boxplot(y=kmeans_predicted["Total_Amt_Chng_Q4_Q1"],␣
      ↪x=kmeans_predicted["pred_clusters"]);
```




```
[ ]: plt.figure(figsize=(15,7))
     sns.boxplot(y=kmeans_predicted['Avg_Utilization_Ratio'],␣
      ↪x=kmeans_predicted["pred_clusters"]);
```



- *Visually at least, there appears to be a very small amount overlap of clusters*
- *This means the clusters are essentially different and thus we can differentiate the custeomers based on the attrition*
- *silhoutte score obtained is not good as the range lies between 1 and -1, and we see the score to be arount 0.21 which is decent but makes it clear the KMeans is not one of the better option.*

The silhoutte score is quite low so it does not seem to be a good method for this data set as the silhouette score defines how close are the data points to the said clusters and can help determine if the classification is valid. Due to this, we can not rely on this method due to the ambiguity being present in the data set and thus resulting in a low score.

2.4 Applying KNN

[ ]:
```
df_reduced.head(20)
```

[ ]:
```
    Total_Revolving_Bal  Total_Relationship_Count  Total_Trans_Amt  \
0                   777                         5             1144
1                   864                         6             1291
2                     0                         4             1887
3                  2517                         3             1171
4                     0                         5              816
5                  1247                         3             1088
6                  2264                         6             1330
7                  1396                         2             1538
8                  2517                         5             1350
9                  1677                         6             1441
10                 1467                         5             1201
11                 1587                         6             1314
12                    0                         3             1539
13                 1666                         5             1311
14                  680                         5             1570
15                  972                         5             1348
16                 2362                         6             1671
17                 1291                         4             1028
18                 2517                         2             1336
19                 1157                         6             1207

    Total_Trans_Ct  Total_Ct_Chng_Q4_Q1  Total_Amt_Chng_Q4_Q1  \
0               42                1.625                 1.335
1               33                3.714                 1.541
2               20                2.333                 2.594
3               20                2.333                 1.405
4               28                2.500                 2.175
5               24                0.846                 1.376
6               31                0.722                 1.975
7               36                0.714                 2.204
8               24                1.182                 3.355
9               32                0.882                 1.524
10              42                0.680                 0.831
11              26                1.364                 1.433
12              17                3.250                 3.397
13              33                2.000                 1.163
```

```
14          29          0.611          1.190
15          27          1.700          1.707
16          27          0.929          1.708
17          21          1.625          0.653
18          30          1.143          1.831
19          21          0.909          0.966

    Avg_Utilization_Ratio  Attrition_Flag
0                   0.061               1
1                   0.105               1
2                   0.000               1
3                   0.760               1
4                   0.000               1
5                   0.311               1
6                   0.066               1
7                   0.048               1
8                   0.113               1
9                   0.144               1
10                  0.217               1
11                  0.174               1
12                  0.000               1
13                  0.195               1
14                  0.279               1
15                  0.230               1
16                  0.078               1
17                  0.095               1
18                  0.788               1
19                  0.080               1
```

[ ]:
```
x = df_reduced.drop(columns=['Attrition_Flag'])
y = df_reduced['Attrition_Flag']
x.shape, y.shape
```

[ ]:
```
((10127, 7), (10127,))
```

[ ]:
```
x.head(20)
```

[ ]:
```
    Total_Revolving_Bal  Total_Relationship_Count  Total_Trans_Amt  \
0                   777                         5             1144
1                   864                         6             1291
2                     0                         4             1887
3                  2517                         3             1171
4                     0                         5              816
5                  1247                         3             1088
6                  2264                         6             1330
7                  1396                         2             1538
8                  2517                         5             1350
```

9	1677	6	1441
10	1467	5	1201
11	1587	6	1314
12	0	3	1539
13	1666	5	1311
14	680	5	1570
15	972	5	1348
16	2362	6	1671
17	1291	4	1028
18	2517	2	1336
19	1157	6	1207

	Total_Trans_Ct	Total_Ct_Chng_Q4_Q1	Total_Amt_Chng_Q4_Q1 \
0	42	1.625	1.335
1	33	3.714	1.541
2	20	2.333	2.594
3	20	2.333	1.405
4	28	2.500	2.175
5	24	0.846	1.376
6	31	0.722	1.975
7	36	0.714	2.204
8	24	1.182	3.355
9	32	0.882	1.524
10	42	0.680	0.831
11	26	1.364	1.433
12	17	3.250	3.397
13	33	2.000	1.163
14	29	0.611	1.190
15	27	1.700	1.707
16	27	0.929	1.708
17	21	1.625	0.653
18	30	1.143	1.831
19	21	0.909	0.966

	Avg_Utilization_Ratio
0	0.061
1	0.105
2	0.000
3	0.760
4	0.000
5	0.311
6	0.066
7	0.048
8	0.113
9	0.144
10	0.217
11	0.174

```
12                    0.000
13                    0.195
14                    0.279
15                    0.230
16                    0.078
17                    0.095
18                    0.788
19                    0.080
```

[ ]:
```
# StandardScaler standardizes a feature by subtracting the mean and then␣
↪scaling to unit variance

X = StandardScaler().fit_transform(x)
X[:3]
```

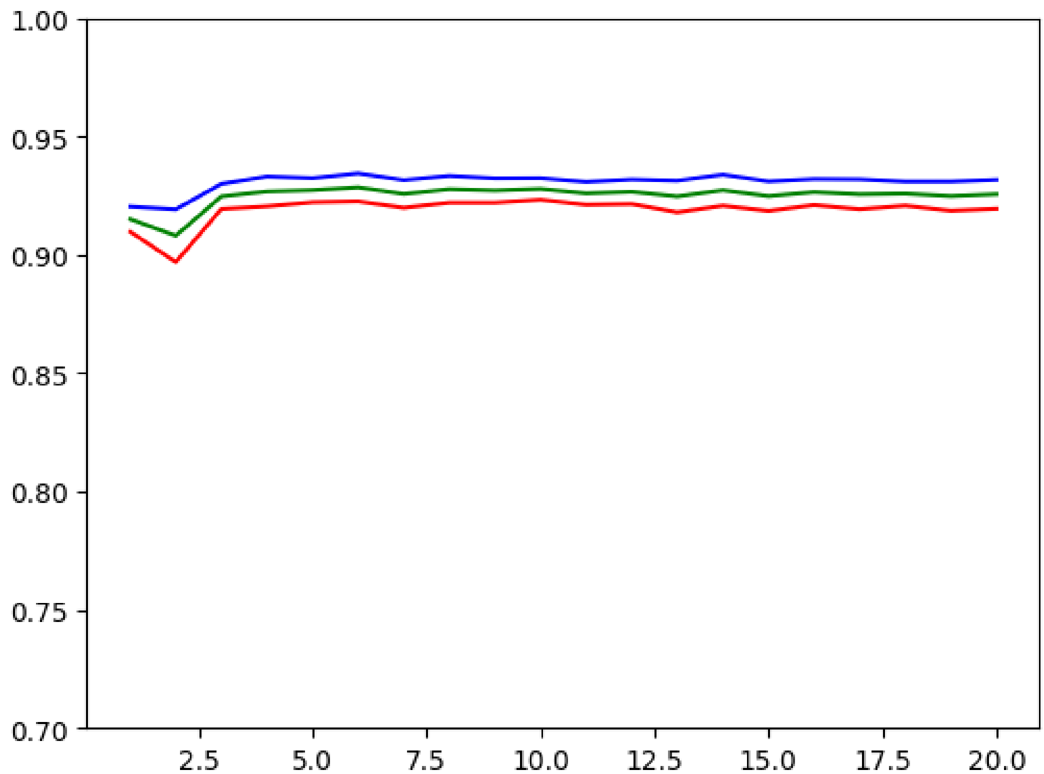
[ ]:
```
array([[-0.47342222,  0.76394261, -0.95970657, -0.97389518,  3.8340026 ,
         2.62349444, -0.77588223],
       [-0.36666682,  1.40730617, -0.91643261, -1.35734038, 12.60857291,
         3.56329284, -0.61627565],
       [-1.42685834,  0.12057905, -0.74098169, -1.91120566,  6.80786367,
         8.36721381, -0.99715499]])
```

[ ]:
```
# Creating Test and Train split for KNN.

x_train, x_test, y_train, y_test = train_test_split(X,y,stratify=y)
```

[ ]:
```
# Fucntion to check for different K - Nearest neighbors

def knn_scores(X, y, start, stop, step):
    list_scores = []
    for i in range(start,stop,step):
        model = KNeighborsClassifier(n_neighbors=i)
        kfold = KFold(n_splits=10, shuffle=True, random_state=44)
        cross_val_scores_KNN = cross_val_score(model, X,y,cv=kfold)
        dict_row_score = {'mean_score': np.mean(cross_val_scores_KNN),␣
↪'std_score': np.std(cross_val_scores_KNN), 'n_neighbours' :i}
        list_scores.append(dict_row_score)

    df_knn_scores = pd.DataFrame(list_scores)

    df_knn_scores['Lower_Bound'] = df_knn_scores['mean_score] -␣
↪df_knn_scores['std_score']
    df_knn_scores['Upper_Bound'] = df_knn_scores['mean_score'] +␣
↪df_knn_scores['std_score']

    return df_knn_scores
```

[ ]:
```
# Try 1 to 20 neigbours
df_knn_scores = knn_scores(x_train, y_train, 1, 21, 1)
```

[ ]:
```
# Plotting best scorses.

plt.plot(df_knn_scores['n_neighbours'], df_knn_scores['Lower_Bound'], color␣
↪='r')
plt.plot(df_knn_scores['n_neighbours'], df_knn_scores['mean_score'], color ='g')
plt.plot(df_knn_scores['n_neighbours'], df_knn_scores['Upper_Bound'], color␣
↪='b')
plt.ylim(0.7,1);
```



[ ]:
```
# Check the hyperparameter with the higher score
df_knn_scores.loc[df_knn_scores.mean_score == df_knn_scores.mean_score.max()]
```

[ ]:

	mean_score	std_score	n_neighbours	Lower_Bound	Upper_Bound
5	0.928508	0.00592	6	0.922588	0.934428

[ ]:
```
best_k = df_knn_scores.loc[df_knn_scores.mean_score == df_knn_scores.mean_score.
↪max(), 'n_neighbours'].values[0]
print('The best K is ' + str(best_k))
```

```
model = KNeighborsClassifier(n_neighbors=best_k)
model.fit(x_train, y_train)
```

```
The best K is 6
```

[ ]:
```
KNeighborsClassifier(n_neighbors=6)
```

[ ]:
```
print('The accuracy for the train ' + str(model.score(x_train, y_train) * 100)␣
↪+ '%')
```

```
The accuracy for the train 94.98354180381831%
```

[ ]:
```
# Applying the model

y_knn_predict = model.predict(x_test)
```

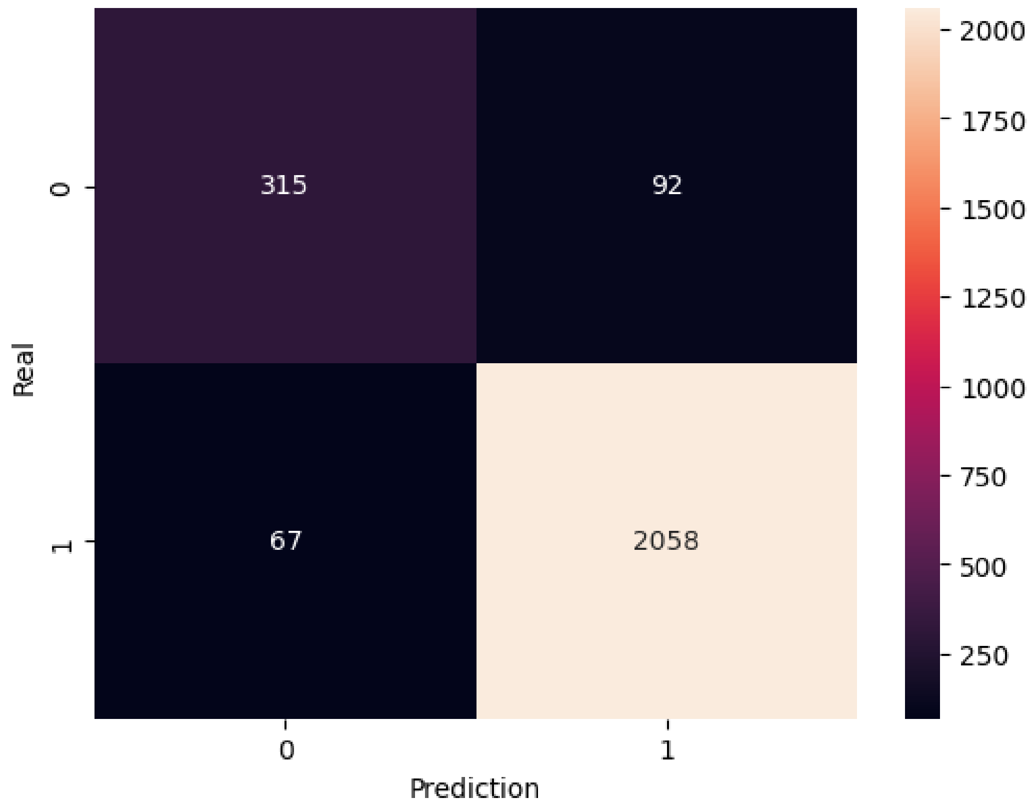
[ ]:
```
# Calculating the Specificity score.

confusion_knn=confusion_matrix(y_test, y_knn_predict)
TP = confusion_knn[1, 1]
TN = confusion_knn[0, 0]
FP = confusion_knn[0, 1]
FN = confusion_knn[1, 0]

specificity_knn = TN / (TN + FP)
print('The Specificity Score is ' + str(specificity_knn))
```

```
The Specificity Score is 0.773955773955774
```

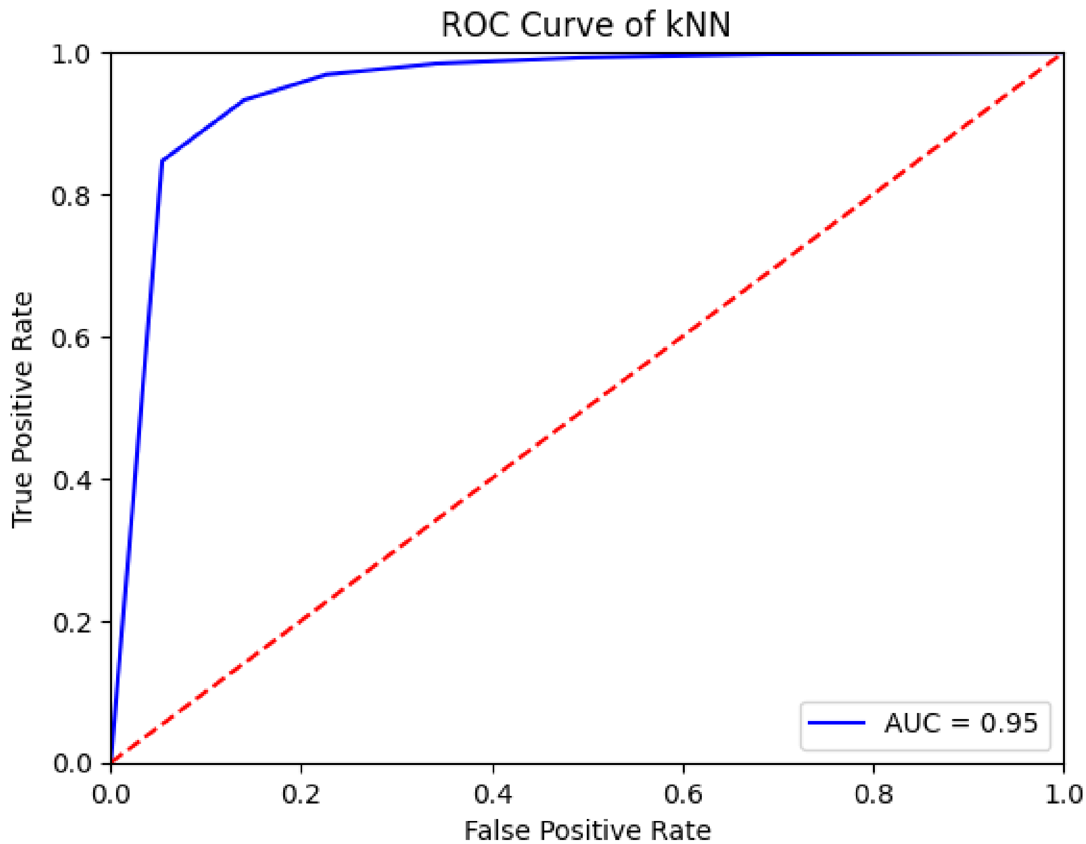
[ ]:
```
# Accuracy, Precision, Recall, F1 score

df_KNN = pd.DataFrame(classification_report(y_knn_predict,y_test,␣
↪digits=2,output_dict=True)).T

df_KNN['support'] = df_KNN.support.apply(int)

df_KNN.style.background_gradient(cmap='viridis',
                                 subset=pd.IndexSlice['0':'9', :'f1-score'])
```

[ ]:
```
<pandas.io.formats.style.Styler at 0x7f047569a2d0>
```

[ ]:
```
# Heat Map for KNN

sns.heatmap(confusion_matrix(y_test, y_knn_predict), annot=True, fmt='.0f')
plt.ylabel('Real')
plt.xlabel('Prediction');
```



[ ]:
```
# ROC Curve of KNN

y_scores = model.predict_proba(x_test)
fpr, tpr, threshold = roc_curve(y_test, y_scores[:, 1])
roc_auc = auc(fpr, tpr)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of kNN')
plt.show()
```



2.4.1 Logistic Regression

[ ]:
```
# Creating test and train splits for Logistic Regression.

x_train, x_test, y_train, y_test = train_test_split(X, y,stratify=y,␣
    ↪random_state =44 )
lr = LogisticRegression(penalty=None)
lr.fit(x_train, y_train);
y_predict_lr = lr.predict(x_test)
y_lr_pred_prob = lr.predict_proba(x_test)
print(y_predict_lr)
print('\n')
print(y_lr_pred_prob)
```

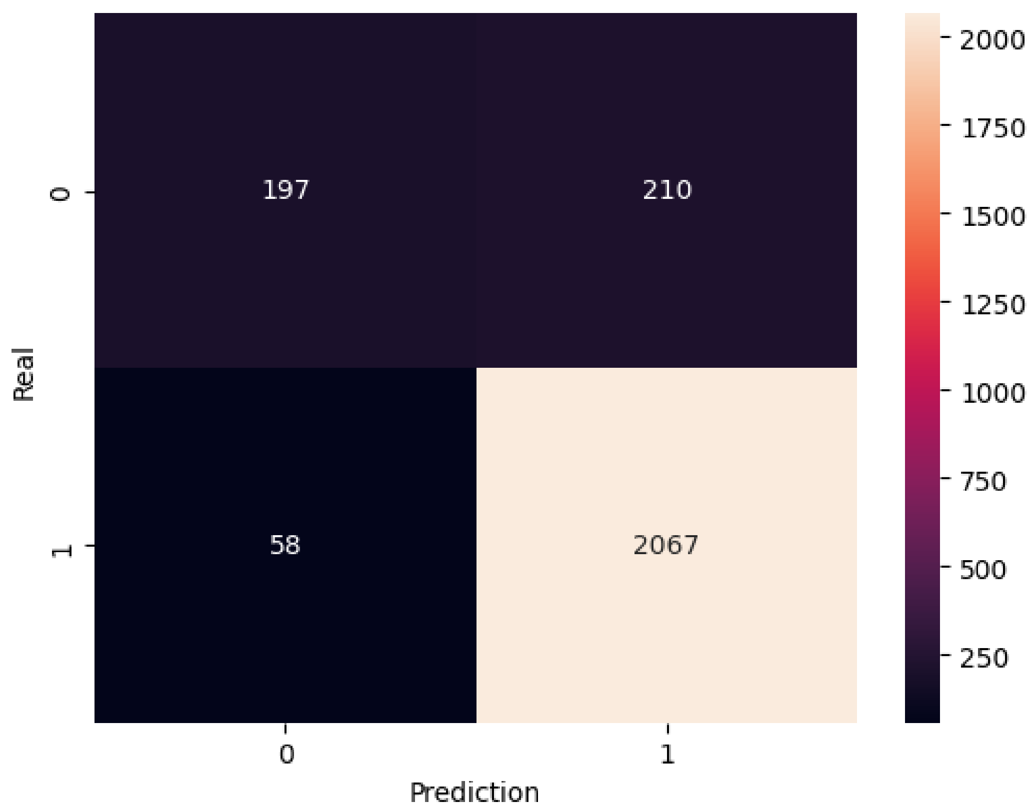
```
[1 1 1 … 1 1 1]


[[0.01166497 0.98833503]
 [0.00961253 0.99038747]
 [0.08751191 0.91248809]
```

…
 [0.25987518 0.74012482]
 [0.05248239 0.94751761]
 [0.00899795 0.99100205]]

[ ]:
```
print('The accuracy for the train ' + str(lr.score(x_train, y_train) * 100) +
 ↪'%')
```

The accuracy for the train 89.07175773535221%

[ ]:
```
# Heat Map

sns.heatmap(confusion_matrix(y_test, y_predict_lr), annot=True, fmt='.0f')
plt.ylabel('Real')
plt.xlabel('Prediction');
```



[ ]:
```
# Accuracy, Precision, Recall, F1 score

df_logR = pd.DataFrame(classification_report(y_predict_lr,y_test,
 ↪digits=2,output_dict=True)).T
```

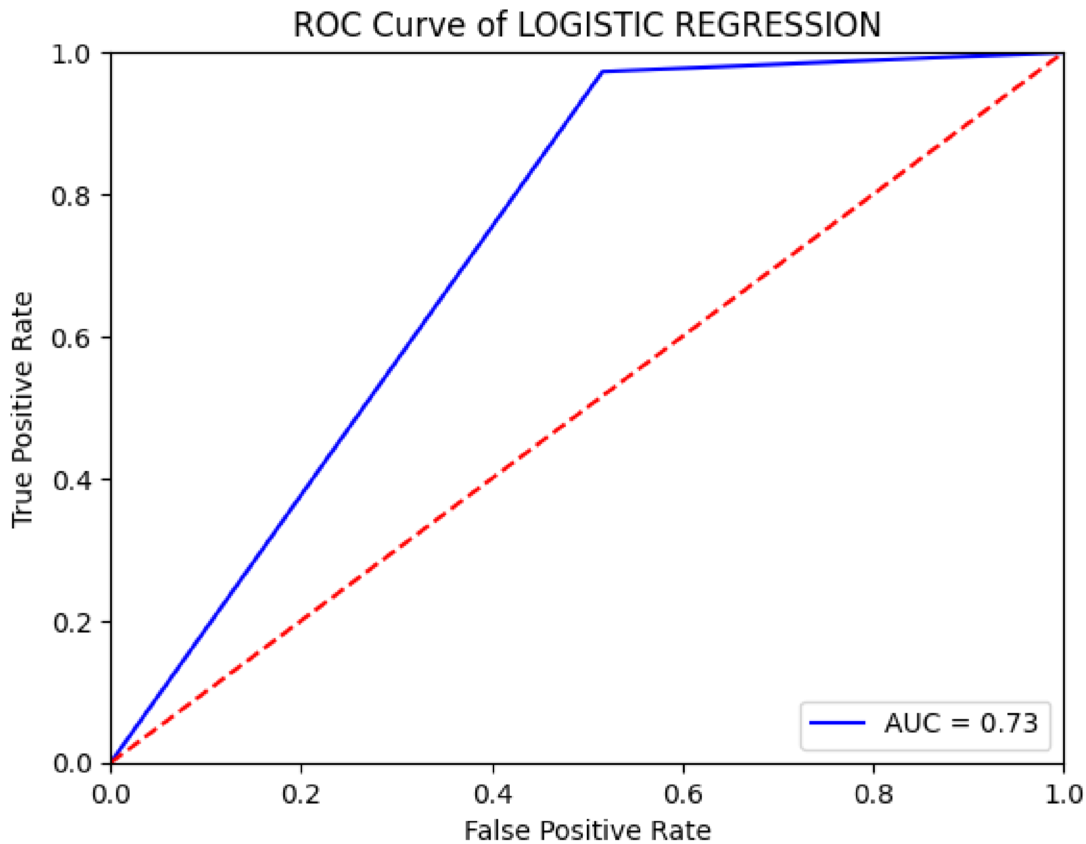
```
df_logR['support'] = df_logR.support.apply(int)

df_logR.style.background_gradient(cmap='viridis',
                              subset=pd.IndexSlice['0':'9', :'f1-score'])
```

[ ]: <pandas.io.formats.style.Styler at 0x7f04754dc410>

[ ]:
```
confusion_knn=confusion_matrix(y_test, y_predict_lr)
TP = confusion_knn[1, 1]
TN = confusion_knn[0, 0]
FP = confusion_knn[0, 1]
FN = confusion_knn[1, 0]

specificity_lr = TN / (TN + FP)
print('The Specificity Score is ' + str(specificity_lr))
```

The Specificity Score is 0.48402948402948404

[ ]:
```
# ROC Curve of Logistic Regression

Y_scores = lr.predict_proba(x_test)
fpr, tpr, threshold = roc_curve(y_test, y_predict_lr)
roc_auc = auc(fpr, tpr)

plt.title('Logistic RegressionClassifier')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of LOGISTIC REGRESSION')
plt.show()
```



2.5 Conclusion

2.5.1 1. Based on the results obtained, it can be concluded that KNN outperforms logistic regression in predicting customer churn in the banking industry. KNN achieved higher accuracy scores on both the training and test sets, as well as higher recall and specificity scores, indicating that it is better at identifying customers who are likely to churn and customers who are not likely to churn, respectively.

2.5.2 2. However, it is important to note that logistic regression achieved a lower specificity score compared to KNN, indicating that it has a higher false positive rate. This means that logistic regression may incorrectly predict more non-churned customers as churned, resulting in potentially unnecessary retention measures

2.5.3 3. If the bank wants to prioritize identifying as many customers who are likely to churn as possible while also minimizing the false positive rate, then KNN may be a better choice.

2.5.4 4. Using the K Means method, the bank can also look for patterns in the behavior of the customers and find additional ways to help with retention.