classification-final-assignment

December 30, 2023

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
import numpy as np
import plotly.express as px
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
↪f1_score, confusion_matrix, roc_curve
from sklearn.metrics import roc_auc_score

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import RandomizedSearchCV
from xgboost import XGBClassifier as xgb

import shap
import lime
import lime.lime_tabular
```

[2]:
```
df = pd.read_csv("Bank Data.csv")
```

[3]:
```
pd.set_option('display.max_columns', None)
```

[4]:
```
df.head()
```

[4]:
```
       ID Customer_ID      Month             Name  Age          SSN  \
0  0x160a    CUS_0xd40  September    Aaron Maashoh   23  821-00-0265
1  0x160b    CUS_0xd40    October    Aaron Maashoh   24  821-00-0265
2  0x160c    CUS_0xd40   November    Aaron Maashoh   24  821-00-0265
3  0x160d    CUS_0xd40   December    Aaron Maashoh  24_  821-00-0265
4  0x1616   CUS_0x21b1  September  Rick Rothackerj   28  004-07-5839

  Occupation Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  \
```

```
0  Scientist      19114.12          1824.843333                  3
1  Scientist      19114.12          1824.843333                  3
2  Scientist      19114.12          1824.843333                  3
3  Scientist      19114.12                  NaN                  3
4    _______      34847.84          3037.986667                  2

   Num_Credit_Card  Interest_Rate Num_of_Loan  \
0                4              3           4
1                4              3           4
2                4              3           4
3                4              3           4
4                4              6           1

                                       Type_of_Loan  Delay_from_due_date  \
0  Auto Loan, Credit-Builder Loan, Personal Loan,…                    3
1  Auto Loan, Credit-Builder Loan, Personal Loan,…                    3
2  Auto Loan, Credit-Builder Loan, Personal Loan,…                   -1
3  Auto Loan, Credit-Builder Loan, Personal Loan,…                    4
4                              Credit-Builder Loan                    3

  Num_of_Delayed_Payment Changed_Credit_Limit  Num_Credit_Inquiries  \
0                      7                11.27                2022.0
1                      9                13.27                   4.0
2                      4                12.27                   4.0
3                      5                11.27                   4.0
4                      1                 5.42                   5.0

  Credit_Mix Outstanding_Debt  Credit_Utilization_Ratio  \
0       Good           809.98                 35.030402
1       Good           809.98                 33.053114
2       Good           809.98                 33.811894
3       Good           809.98                 32.430559
4       Good           605.03                 25.926822

       Credit_History_Age Payment_of_Min_Amount  Total_EMI_per_month  \
0   22 Years and 9 Months                    No            49.574949
1  22 Years and 10 Months                    No            49.574949
2                     NaN                    No            49.574949
3   23 Years and 0 Months                    No            49.574949
4   27 Years and 3 Months                    No            18.816215

  Amount_invested_monthly                 Payment_Behaviour  \
0      236.64268203272135    Low_spent_Small_value_payments
1      21.465380264657146  High_spent_Medium_value_payments
2      148.23393788500925   Low_spent_Medium_value_payments
3       39.08251089460281  High_spent_Medium_value_payments
4      39.684018417945296   High_spent_Large_value_payments
```

```
     Monthly_Balance
0  186.26670208571772
1  361.44400385378196
2  264.67544623342997
3  343.82687322383634
4   485.2984336755923
```

The targeted variable is Credit_Mix

[5]:
```python
df.rename(columns={'Credit_Mix': 'Credit_Score'}, inplace=True)
```

0.1 Data Exploration and Preprocessing:

[6]:
```python
df.head()
```

[6]:
```
       ID Customer_ID      Month             Name  Age          SSN  \
0  0x160a   CUS_0xd40  September    Aaron Maashoh   23  821-00-0265
1  0x160b   CUS_0xd40    October    Aaron Maashoh   24  821-00-0265
2  0x160c   CUS_0xd40   November    Aaron Maashoh   24  821-00-0265
3  0x160d   CUS_0xd40   December    Aaron Maashoh  24_  821-00-0265
4  0x1616  CUS_0x21b1  September  Rick Rothackerj   28  004-07-5839

  Occupation Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  \
0  Scientist      19114.12            1824.843333                  3
1  Scientist      19114.12            1824.843333                  3
2  Scientist      19114.12            1824.843333                  3
3  Scientist      19114.12                    NaN                  3
4    _______      34847.84            3037.986667                  2

   Num_Credit_Card  Interest_Rate Num_of_Loan  \
0                4              3           4
1                4              3           4
2                4              3           4
3                4              3           4
4                4              6           1

                                       Type_of_Loan  Delay_from_due_date  \
0  Auto Loan, Credit-Builder Loan, Personal Loan,…                     3
1  Auto Loan, Credit-Builder Loan, Personal Loan,…                     3
2  Auto Loan, Credit-Builder Loan, Personal Loan,…                    -1
3  Auto Loan, Credit-Builder Loan, Personal Loan,…                     4
4                               Credit-Builder Loan                    3

  Num_of_Delayed_Payment Changed_Credit_Limit  Num_Credit_Inquiries  \
0                      7                11.27                2022.0
1                      9                13.27                   4.0
```

```
2                       4               12.27                    4.0
3                       5               11.27                    4.0
4                       1                5.42                    5.0

  Credit_Score Outstanding_Debt  Credit_Utilization_Ratio  \
0         Good           809.98                 35.030402
1         Good           809.98                 33.053114
2         Good           809.98                 33.811894
3         Good           809.98                 32.430559
4         Good           605.03                 25.926822

       Credit_History_Age Payment_of_Min_Amount  Total_EMI_per_month  \
0   22 Years and 9 Months                    No            49.574949
1  22 Years and 10 Months                    No            49.574949
2                     NaN                    No            49.574949
3   23 Years and 0 Months                    No            49.574949
4   27 Years and 3 Months                    No            18.816215

  Amount_invested_monthly                 Payment_Behaviour  \
0       236.64268203272135   Low_spent_Small_value_payments
1       21.465380264657146  High_spent_Medium_value_payments
2       148.23393788500925   Low_spent_Medium_value_payments
3        39.08251089460281  High_spent_Medium_value_payments
4       39.684018417945296   High_spent_Large_value_payments

      Monthly_Balance
0  186.26670208571772
1  361.44400385378196
2  264.67544623342997
3  343.82687322383634
4   485.2984336755923
```

[7]:
```python
df.drop(columns=['Name', 'ID', 'SSN'], axis=1, inplace=True)
```

[8]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 24 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Customer_ID               50000 non-null  object
 1   Month                     50000 non-null  object
 2   Age                       50000 non-null  object
 3   Occupation                50000 non-null  object
 4   Annual_Income             50000 non-null  object
 5   Monthly_Inhand_Salary     42502 non-null  float64
```

```
 6   Num_Bank_Accounts         50000 non-null  int64
 7   Num_Credit_Card           50000 non-null  int64
 8   Interest_Rate             50000 non-null  int64
 9   Num_of_Loan               50000 non-null  object
 10  Type_of_Loan              44296 non-null  object
 11  Delay_from_due_date       50000 non-null  int64
 12  Num_of_Delayed_Payment    46502 non-null  object
 13  Changed_Credit_Limit      50000 non-null  object
 14  Num_Credit_Inquiries      48965 non-null  float64
 15  Credit_Score              50000 non-null  object
 16  Outstanding_Debt          50000 non-null  object
 17  Credit_Utilization_Ratio  50000 non-null  float64
 18  Credit_History_Age        45530 non-null  object
 19  Payment_of_Min_Amount     50000 non-null  object
 20  Total_EMI_per_month       50000 non-null  float64
 21  Amount_invested_monthly   47729 non-null  object
 22  Payment_Behaviour         50000 non-null  object
 23  Monthly_Balance           49438 non-null  object
dtypes: float64(4), int64(4), object(16)
memory usage: 9.2+ MB
```

[9]:
```python
df.describe()
```

[9]:
```
       Monthly_Inhand_Salary  Num_Bank_Accounts  Num_Credit_Card  \
count           42502.000000       50000.000000     50000.000000
mean             4182.004291          16.838260        22.921480
std              3174.109304         116.396848       129.314804
min               303.645417          -1.000000         0.000000
25%              1625.188333           3.000000         4.000000
50%              3086.305000           6.000000         5.000000
75%              5934.189094           7.000000         7.000000
max             15204.633333        1798.000000      1499.000000

       Interest_Rate  Delay_from_due_date  Num_Credit_Inquiries  \
count   50000.000000         50000.000000          48965.000000
mean       68.772640            21.052640             30.080200
std       451.602363            14.860397            196.984121
min         1.000000            -5.000000              0.000000
25%         8.000000            10.000000              4.000000
50%        13.000000            18.000000              7.000000
75%        20.000000            28.000000             10.000000
max      5799.000000            67.000000           2593.000000

       Credit_Utilization_Ratio  Total_EMI_per_month
count              50000.000000         50000.000000
mean                  32.279581          1491.304305
std                    5.106238          8595.647887
```

```
min                 20.509652          0.000000
25%                 28.061040         32.222388
50%                 32.280390         74.733349
75%                 36.468591        176.157491
max                 48.540663      82398.000000
```

Check missing values

[10]:
```
df.isnull().sum()
```

[10]:
```
Customer_ID                    0
Month                          0
Age                            0
Occupation                     0
Annual_Income                  0
Monthly_Inhand_Salary       7498
Num_Bank_Accounts              0
Num_Credit_Card                0
Interest_Rate                  0
Num_of_Loan                    0
Type_of_Loan                5704
Delay_from_due_date            0
Num_of_Delayed_Payment      3498
Changed_Credit_Limit           0
Num_Credit_Inquiries        1035
Credit_Score                   0
Outstanding_Debt               0
Credit_Utilization_Ratio       0
Credit_History_Age          4470
Payment_of_Min_Amount          0
Total_EMI_per_month            0
Amount_invested_monthly     2271
Payment_Behaviour              0
Monthly_Balance              562
dtype: int64
```

0.1.1 Changing datas into numeric

[11]:
```
df.dtypes
```

[11]:
```
Customer_ID                  object
Month                        object
Age                          object
Occupation                   object
Annual_Income                object
Monthly_Inhand_Salary       float64
Num_Bank_Accounts             int64
```

```
Num_Credit_Card              int64
Interest_Rate                int64
Num_of_Loan                 object
Type_of_Loan                object
Delay_from_due_date          int64
Num_of_Delayed_Payment      object
Changed_Credit_Limit        object
Num_Credit_Inquiries       float64
Credit_Score                object
Outstanding_Debt            object
Credit_Utilization_Ratio   float64
Credit_History_Age          object
Payment_of_Min_Amount       object
Total_EMI_per_month        float64
Amount_invested_monthly     object
Payment_Behaviour           object
Monthly_Balance             object
dtype: object
```

[12]:
```
df.head()
```

[12]:
```
  Customer_ID      Month  Age Occupation Annual_Income  Monthly_Inhand_Salary  \
0   CUS_0xd40  September   23  Scientist      19114.12            1824.843333
1   CUS_0xd40    October   24  Scientist      19114.12            1824.843333
2   CUS_0xd40   November   24  Scientist      19114.12            1824.843333
3   CUS_0xd40   December  24_  Scientist      19114.12                    NaN
4  CUS_0x21b1  September   28    _______      34847.84            3037.986667

   Num_Bank_Accounts  Num_Credit_Card  Interest_Rate Num_of_Loan  \
0                  3                4              3           4
1                  3                4              3           4
2                  3                4              3           4
3                  3                4              3           4
4                  2                4              6           1

                                     Type_of_Loan  Delay_from_due_date  \
0  Auto Loan, Credit-Builder Loan, Personal Loan,…                    3
1  Auto Loan, Credit-Builder Loan, Personal Loan,…                    3
2  Auto Loan, Credit-Builder Loan, Personal Loan,…                   -1
3  Auto Loan, Credit-Builder Loan, Personal Loan,…                    4
4                               Credit-Builder Loan                     3

  Num_of_Delayed_Payment Changed_Credit_Limit  Num_Credit_Inquiries  \
0                      7                11.27                2022.0
1                      9                13.27                   4.0
2                      4                12.27                   4.0
3                      5                11.27                   4.0
```

```
4                      1                 5.42                   5.0

  Credit_Score Outstanding_Debt  Credit_Utilization_Ratio  \
0         Good           809.98                 35.030402
1         Good           809.98                 33.053114
2         Good           809.98                 33.811894
3         Good           809.98                 32.430559
4         Good           605.03                 25.926822

       Credit_History_Age Payment_of_Min_Amount  Total_EMI_per_month  \
0   22 Years and 9 Months                    No            49.574949
1  22 Years and 10 Months                    No            49.574949
2                     NaN                    No            49.574949
3   23 Years and 0 Months                    No            49.574949
4   27 Years and 3 Months                    No            18.816215

  Amount_invested_monthly                 Payment_Behaviour  \
0      236.64268203272135    Low_spent_Small_value_payments
1       21.465380264657146  High_spent_Medium_value_payments
2      148.23393788500925   Low_spent_Medium_value_payments
3       39.08251089460281  High_spent_Medium_value_payments
4      39.684018417945296   High_spent_Large_value_payments

      Monthly_Balance
0  186.26670208571772
1  361.44400385378196
2  264.67544623342997
3  343.82687322383634
4   485.2984336755923
```

[13]:
```
wrong_cols = ['Age', 'Occupation', 'Annual_Income', 'Num_of_Loan',␣
↪'Num_of_Delayed_Payment', 'Changed_Credit_Limit', 'Outstanding_Debt',␣
↪'Amount_invested_monthly', 'Monthly_Balance']
```

[14]:
```
for col in wrong_cols:
    df[col] = df[col].str.replace('_', '')
    try:
        df[col] = df[col].astype('float')
    except:
        continue
```

[15]:
```
for i in wrong_cols:
    print(i, ':', df[i].dtypes)
```

```
Age : float64
Occupation : object
Annual_Income : float64
```

```
Num_of_Loan : float64
Num_of_Delayed_Payment : float64
Changed_Credit_Limit : object
Outstanding_Debt : float64
Amount_invested_monthly : float64
Monthly_Balance : float64
```

0.1.2 Data inconsistencies

[16]:
```python
for col in df.columns:
    print(f"---###*** {col} ---###***")
    print(df[col].value_counts())
```

```
---###*** Customer_ID ---###***
Customer_ID
CUS_0xd40     4
CUS_0x9bf4    4
CUS_0x5ae3    4
CUS_0xbe9a    4
CUS_0x4874    4
             ..
CUS_0x2eb4    4
CUS_0x7863    4
CUS_0x9d89    4
CUS_0xc045    4
CUS_0x942c    4
Name: count, Length: 12500, dtype: int64
---###*** Month ---###***
Month
September    12500
October      12500
November     12500
December     12500
Name: count, dtype: int64
---###*** Age ---###***
Age
39.0      1570
32.0      1529
44.0      1500
22.0      1493
35.0      1483
          …
1419.0       1
120.0        1
2552.0       1
2698.0       1
4975.0       1
Name: count, Length: 928, dtype: int64
```

```
---######*** Occupation ---######***
Occupation
                3438
Lawyer          3324
Engineer        3212
Architect       3195
Mechanic        3168
Developer       3146
Accountant      3133
MediaManager    3130
Scientist       3104
Teacher         3103
Entrepreneur    3103
Journalist      3037
Doctor          3027
Manager         3000
Musician        2947
Writer          2933
Name: count, dtype: int64
---######*** Annual_Income ---######***
Annual_Income
17273.83       8
36585.12       8
95596.35       8
40341.16       8
9141.63        8
              ..
5937799.00     1
19395184.00    1
7838666.00     1
24004088.00    1
3287738.00     1
Name: count, Length: 12989, dtype: int64
---######*** Monthly_Inhand_Salary ---######***
Monthly_Inhand_Salary
1315.560833     8
6639.560000     7
2295.058333     7
6082.187500     7
536.431250      7
               ..
12386.966240    1
5993.870000     1
6763.330000     1
7729.695181     1
2312.785000     1
Name: count, Length: 12793, dtype: int64
---######*** Num_Bank_Accounts ---######***
```

```
Num_Bank_Accounts
6       6504
7       6408
8       6387
4       6100
5       6068
        …
855        1
1262       1
908        1
603        1
1727       1
Name: count, Length: 540, dtype: int64
---######*** Num_Credit_Card ---######***
Num_Credit_Card
5       9210
7       8271
6       8243
4       7072
3       6539
        …
662        1
445        1
78         1
1488       1
955        1
Name: count, Length: 819, dtype: int64
---######*** Interest_Rate ---######***
Interest_Rate
8       2503
5       2500
6       2368
12      2288
10      2259
        …
1573       1
3279       1
1166       1
5613       1
4252       1
Name: count, Length: 945, dtype: int64
---######*** Num_of_Loan ---######***
Num_of_Loan
2.0       7515
3.0       7514
4.0       7368
0.0       5446
1.0       5295
```

```
          …
621.0        1
1040.0       1
1496.0       1
570.0        1
1296.0       1
Name: count, Length: 252, dtype: int64
---###*** Type_of_Loan ---###***
Type_of_Loan
Not Specified
704
Credit-Builder Loan
640
Personal Loan
636
Debt Consolidation Loan
632
Student Loan
620
                                                          …
Not Specified, Mortgage Loan, Auto Loan, and Payday Loan
4
Payday Loan, Mortgage Loan, Debt Consolidation Loan, and Student Loan
4
Debt Consolidation Loan, Auto Loan, Personal Loan, Debt Consolidation Loan,
Student Loan, and Credit-Builder Loan                      4
Student Loan, Auto Loan, Student Loan, Credit-Builder Loan, Home Equity Loan,
Debt Consolidation Loan, and Debt Consolidation Loan      4
Personal Loan, Auto Loan, Mortgage Loan, Student Loan, and Student Loan
4
Name: count, Length: 6260, dtype: int64
---###*** Delay_from_due_date ---###***
Delay_from_due_date
 13    1761
 15    1759
 8     1680
 9     1656
 10    1645
       …
 65      30
 63      21
-5       18
 66      12
 67       7
Name: count, Length: 73, dtype: int64
---###*** Num_of_Delayed_Payment ---###***
Num_of_Delayed_Payment
19.0      2707
```

```
15.0      2674
16.0      2637
17.0      2636
18.0      2631
          …
1146.0       1
288.0        1
3556.0       1
3393.0       1
2034.0       1
Name: count, Length: 411, dtype: int64
---######*** Changed_Credit_Limit ---######***
Changed_Credit_Limit
_                       1059
11.5                      70
11.32                     63
7.01                      60
7.35                      60
                        …
-0.6099999999999999        1
21.61                      1
12.010000000000002         1
0.43000000000000016        1
29.17                      1
Name: count, Length: 3927, dtype: int64
---######*** Num_Credit_Inquiries ---######***
Num_Credit_Inquiries
5.0       4709
4.0       4402
6.0       4375
7.0       4295
8.0       3922
          …
1471.0       1
307.0        1
1326.0       1
904.0        1
352.0        1
Name: count, Length: 750, dtype: int64
---######*** Credit_Score ---######***
Credit_Score
Standard    18379
Good        12260
_            9805
Bad          9556
Name: count, dtype: int64
---######*** Outstanding_Debt ---######***
Outstanding_Debt
```

```
1109.03    12
1151.70    12
1360.45    12
460.46     12
1058.13     8
           ..
4230.04     4
641.99      4
98.61       4
2614.48     4
502.38      4
Name: count, Length: 12203, dtype: int64
---###*** Credit_Utilization_Ratio ---###***
Credit_Utilization_Ratio
35.030402    1
24.962925    1
32.546656    1
35.641022    1
27.277364    1
            ..
40.725304    1
33.004488    1
26.441658    1
24.342582    1
34.108530    1
Name: count, Length: 50000, dtype: int64
---###*** Credit_History_Age ---###***
Credit_History_Age
20 Years and 1 Months     254
16 Years and 1 Months     254
18 Years and 7 Months     252
19 Years and 7 Months     252
18 Years and 6 Months     250
                         …
4 Years and 5 Months       21
0 Years and 11 Months      16
33 Years and 11 Months     15
34 Years and 0 Months      14
0 Years and 10 Months      13
Name: count, Length: 399, dtype: int64
---###*** Payment_of_Min_Amount ---###***
Payment_of_Min_Amount
Yes    26158
No     17849
NM      5993
Name: count, dtype: int64
---###*** Total_EMI_per_month ---###***
Total_EMI_per_month
```

```
0.000000        5002
49.574949          4
16.941903          4
420.199367         4
550.679394         4
                ...
65628.000000       1
92.396923          1
191.296729         1
61274.000000       1
50090.000000       1
Name: count, Length: 13144, dtype: int64
---###*** Amount_invested_monthly ---###***
Amount_invested_monthly
10000.000000    2175
0.000000         106
236.642682         1
160.097717         1
320.456645         1
                ...
197.217131         1
366.231484         1
34.899406          1
256.908305         1
220.457878         1
Name: count, Length: 45450, dtype: int64
---###*** Payment_Behaviour ---###***
Payment_Behaviour
Low_spent_Small_value_payments      12694
High_spent_Medium_value_payments     8922
High_spent_Large_value_payments      6844
Low_spent_Medium_value_payments      6837
High_spent_Small_value_payments      5651
Low_spent_Large_value_payments       5252
!@9#%8                               3800
Name: count, dtype: int64
---###*** Monthly_Balance ---###***
Monthly_Balance
-3.333333e+26    6
 1.862667e+02    1
 2.234078e+02    1
 3.054379e+02    1
 3.895375e+02    1
                ..
 4.212569e+02    1
 1.944403e+02    1
 2.999578e+02    1
 3.758979e+02    1
```

```
 3.603797e+02    1
Name: count, Length: 49433, dtype: int64
```

[17]:
```
df['Age'] = np.where((df.Age > 100) | (df.Age < 1), np.nan, df['Age'])
df['Occupation'] = df['Occupation'].replace('', np.nan)
df['Num_Bank_Accounts'] = np.where((df['Num_Bank_Accounts'] < 0) |␣
↪(df['Num_Bank_Accounts'] > 12), np.nan, df['Num_Bank_Accounts'])
df['Num_Credit_Card'] = np.where((df['Num_Credit_Card'] < 0) |␣
↪(df['Num_Credit_Card'] > 200), np.nan, df['Num_Credit_Card'])
df['Interest_Rate'] = np.where((df['Interest_Rate'] < 0) | (df['Interest_Rate']␣
↪> 100), np.nan, df['Interest_Rate'])
df['Num_of_Loan'] = np.where((df['Num_of_Loan'] < 0) | (df['Num_of_Loan'] >␣
↪100), np.nan, df['Num_of_Loan'])
df['Num_of_Delayed_Payment'] = np.where((df['Num_of_Delayed_Payment'] < 0) |␣
↪(df['Num_of_Delayed_Payment'] > 100), np.nan, df['Num_of_Delayed_Payment'])
df['Changed_Credit_Limit'] = pd.to_numeric(df['Changed_Credit_Limit'])
df['Changed_Credit_Limit'] = np.where(df['Changed_Credit_Limit'] < 1, np.nan,␣
↪df['Changed_Credit_Limit'])
```

[18]:
```
df['Num_Credit_Inquiries'] = np.where((df['Num_Credit_Inquiries'] < 0) |␣
↪(df['Num_Credit_Inquiries'] > 100), np.nan, df['Num_Credit_Inquiries'])
df['Credit_Score'] = df['Credit_Score'].replace('_', np.nan)
```

[19]:
```
def add_months(string):
    try:
        split_txt = string.split(" ")
        year_to_month = int(split_txt[0]) * 12
        month = int(split_txt[3])
        return month + year_to_month
    except:
        return string
```

[20]:
```
df['Credit_History_Age'] = df['Credit_History_Age'].apply(lambda string:␣
↪add_months(string)).astype(float)
```

[21]:
```
df['Payment_Behaviour'] = df['Payment_Behaviour'].replace('!@9#%8', np.nan)
```

[22]:
```
df['Payment_of_Min_Amount'] = df['Payment_of_Min_Amount'].replace('NM', np.nan)
```

Making customer id to to numerical value

[23]:
```
df['Customer_ID'] = df['Customer_ID'].apply(lambda id: int(id[4:], base=16))
```

Converting Type of loan into dummy variable. Because some users take multiple types of loan.

[24]:
```
loans = list(df['Type_of_Loan'].value_counts().index[:11])
```

[25]:
```
for string in loans:
    loan = df['Type_of_Loan'].str.contains(string, na=False)
    df.loc[loan, 'Type_of_Loan'] = string
```

[26]:
```
dum = pd.get_dummies(df['Type_of_Loan']).astype(int)
```

[27]:
```
df.drop('Type_of_Loan', axis=1, inplace=True)
df = pd.concat([df, dum], axis=1)
```

0.1.3 Handling NaN values

[28]:
```
Num_cols = []
cat_cols = []
for col in df.columns:
    if df[col].dtypes == 'object':
        cat_cols.append(col)
    else:
        Num_cols.append(col)
```

[29]:
```
#handle null values of numerical columns

for col in Num_cols:
    if df[col].isna:
        df[col].fillna(df[col].median(), inplace=True)
```

[30]:
```
#handle null values of categorical columns

for col in cat_cols:
    if df[col].isnull:
        df[col].fillna(df[col].mode()[0], inplace=True)
```

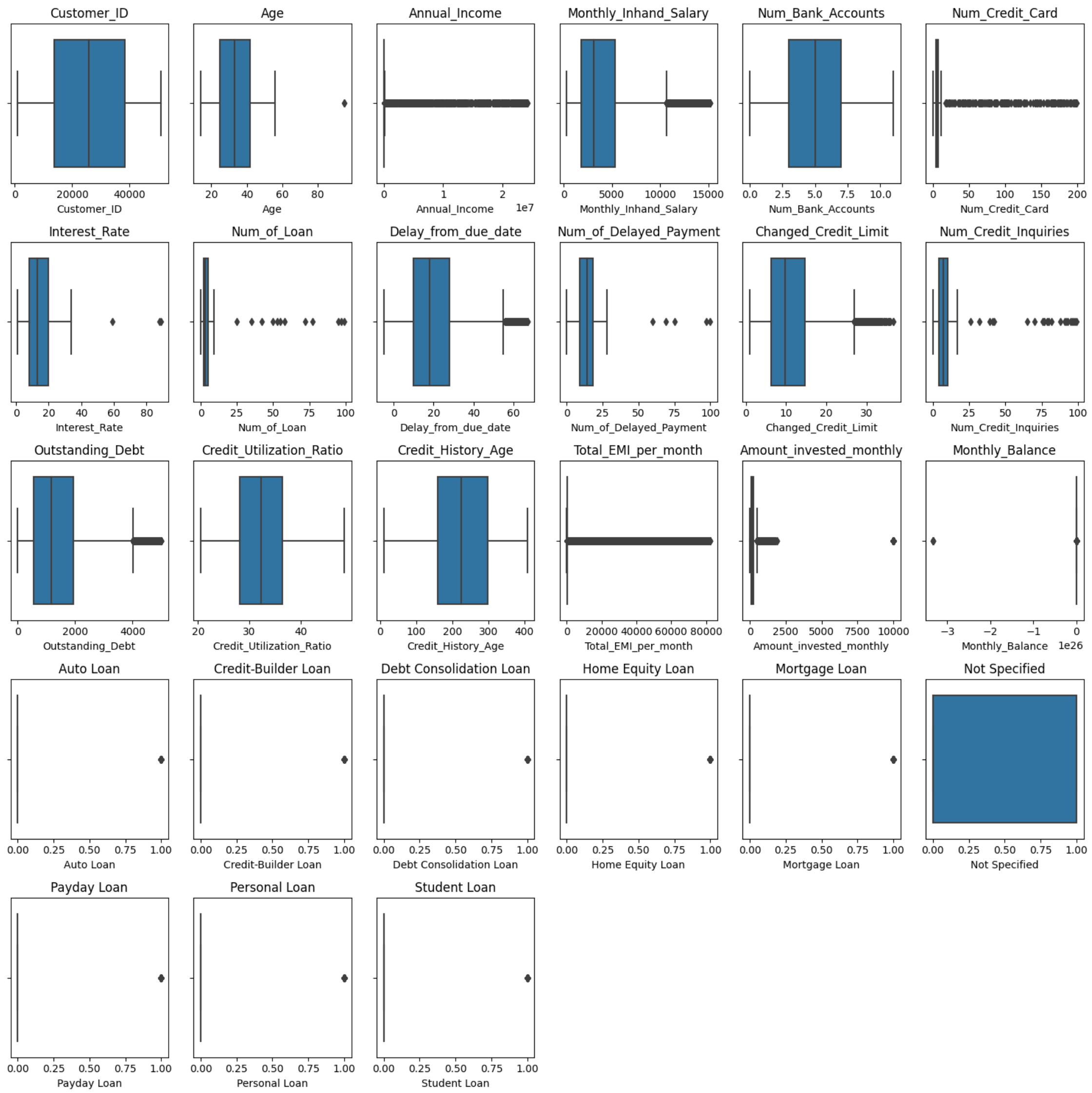
[31]:
```
df.isnull().sum()
```

[31]:
```
Customer_ID                 0
Month                       0
Age                         0
Occupation                  0
Annual_Income               0
Monthly_Inhand_Salary       0
Num_Bank_Accounts           0
Num_Credit_Card             0
Interest_Rate               0
Num_of_Loan                 0
Delay_from_due_date         0
Num_of_Delayed_Payment      0
Changed_Credit_Limit        0
Num_Credit_Inquiries        0
```

```
Credit_Score                0
Outstanding_Debt            0
Credit_Utilization_Ratio    0
Credit_History_Age          0
Payment_of_Min_Amount       0
Total_EMI_per_month         0
Amount_invested_monthly     0
Payment_Behaviour           0
Monthly_Balance             0
Auto Loan                   0
Credit-Builder Loan         0
Debt Consolidation Loan     0
Home Equity Loan            0
Mortgage Loan               0
Not Specified               0
Payday Loan                 0
Personal Loan               0
Student Loan                0
dtype: int64
```

0.1.4 Outliers

[32]:
```
plt.figure(figsize=(15,15))

for ax, col in enumerate(Num_cols):
    plt.subplot(5, 6, int(ax+1))
    plt.title(col)
    sns.boxplot(x=df[col], hue=df['Credit_Score'])

plt.tight_layout()
plt.show()
```



[33]:

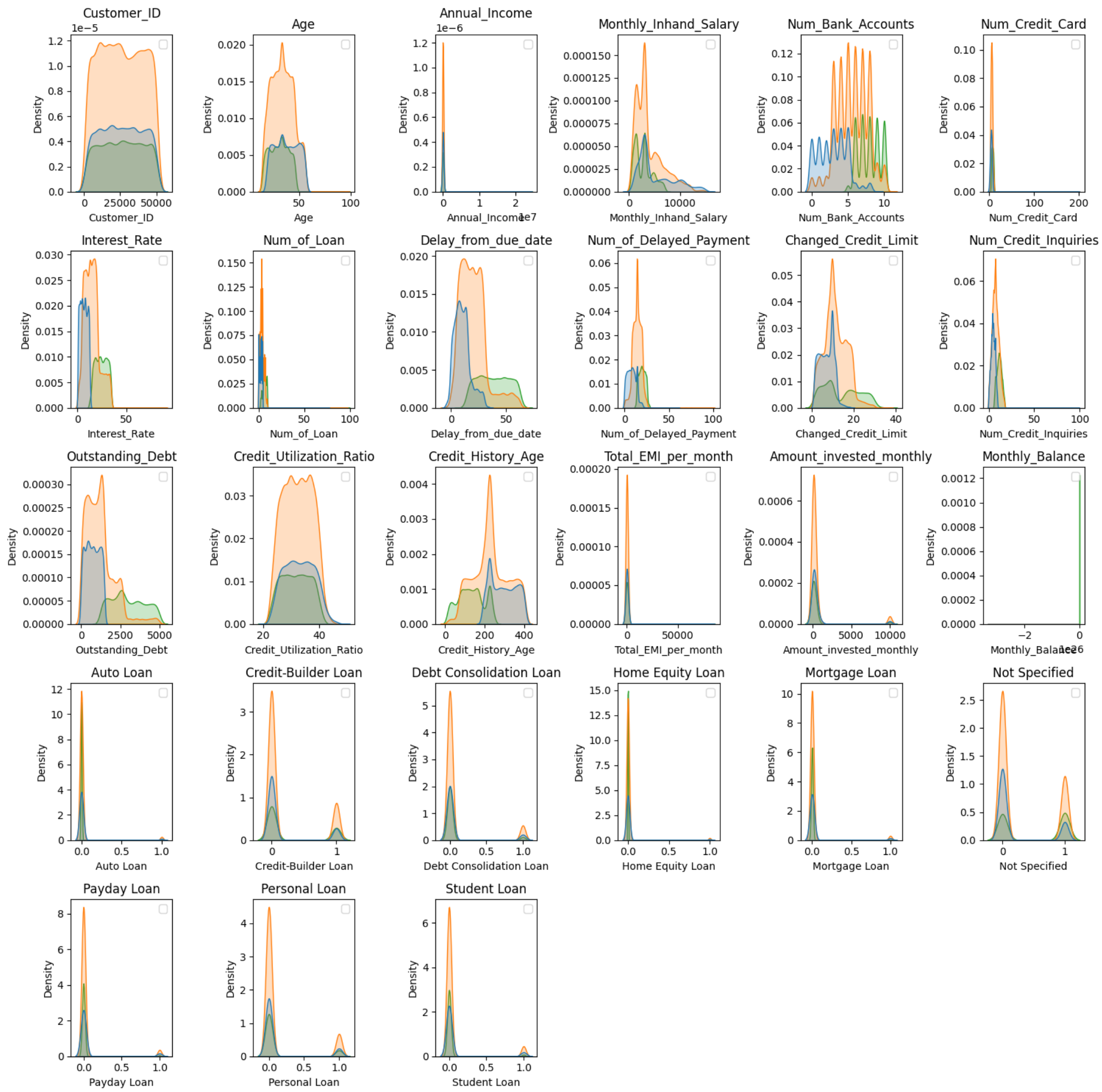
```python
plt.figure(figsize=(15,15))
for ax, col in enumerate(Num_cols):

    plt.subplot(5, 6, int(ax+1))
    plt.title(col)
    sns.kdeplot(x=df[col],fill=True, hue=df['Credit_Score'])
    plt.legend()
plt.tight_layout()
plt.show()
```

```
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
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.
```



[34]:
```python
def remove_outlier(dataframe, column_name):
    Q1 = df[column_name].quantile(.25)
    Q3 = df[column_name].quantile(.75)
```

```
    IQR = Q3 - Q1
    lower_fence = Q1-(1.5*IQR)
    upper_fence = Q3 + (1.5*IQR)
    df.loc[(df[column_name] <= lower_fence), column_name] = df[column_name].
 ↪median()
    df.loc[(df[column_name] >= upper_fence), column_name] = df[column_name].
 ↪median()

def check_outliers(dataframe, column_name):
    Q1 = df[column_name].quantile(.25)
    Q3 = df[column_name].quantile(.75)
    IQR = Q3 - Q1
    lower_fence = Q1-(1.5*IQR)
    upper_fence = Q3 + (1.5*IQR)
    if dataframe[(dataframe[column_name] < lower_fence) |␣
 ↪(dataframe[column_name] > upper_fence)].any(axis=None):
        return True
    else:
        return False
```

[35]:
```
for col in Num_cols:
    if col not in df.columns[-9:]:
        if check_outliers(df,col):
            remove_outlier(df, col)
```

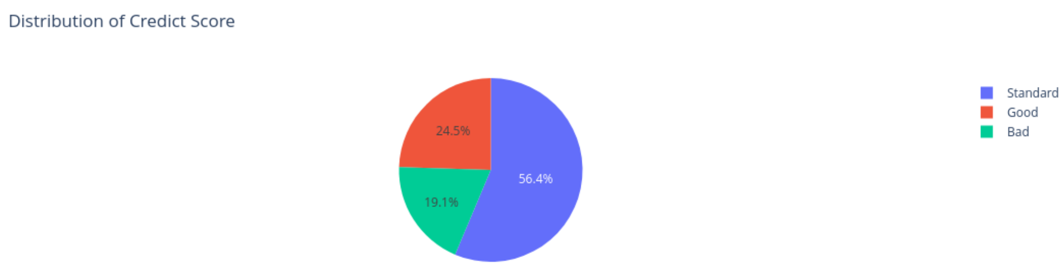
[36]:
```
for col in Num_cols:
    print(col,"---", check_outliers(df, col))
```

```
Customer_ID --- False
Age --- False
Annual_Income --- True
Monthly_Inhand_Salary --- True
Num_Bank_Accounts --- False
Num_Credit_Card --- False
Interest_Rate --- False
Num_of_Loan --- False
Delay_from_due_date --- True
Num_of_Delayed_Payment --- False
Changed_Credit_Limit --- True
Num_Credit_Inquiries --- False
Outstanding_Debt --- True
Credit_Utilization_Ratio --- False
Credit_History_Age --- False
Total_EMI_per_month --- True
Amount_invested_monthly --- True
Monthly_Balance --- True
Auto Loan --- True
```

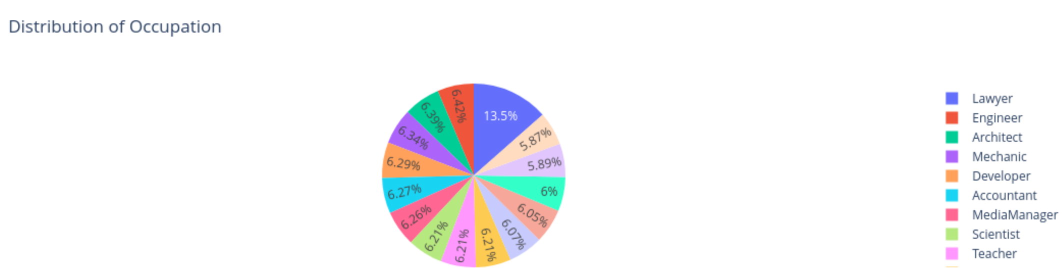
```
Credit-Builder Loan --- True
Debt Consolidation Loan --- True
Home Equity Loan --- True
Mortgage Loan --- True
Not Specified --- False
Payday Loan --- True
Personal Loan --- True
Student Loan --- True
```

0.1.5 EDA and Exploring the distribution of the target variable and features.

[37]:
```python
figure=px.pie(df["Credit_Score"].value_counts().reset_index(), values="count",␣
↪names="Credit_Score", title="Distribution of Credict Score")
figure.show()
```



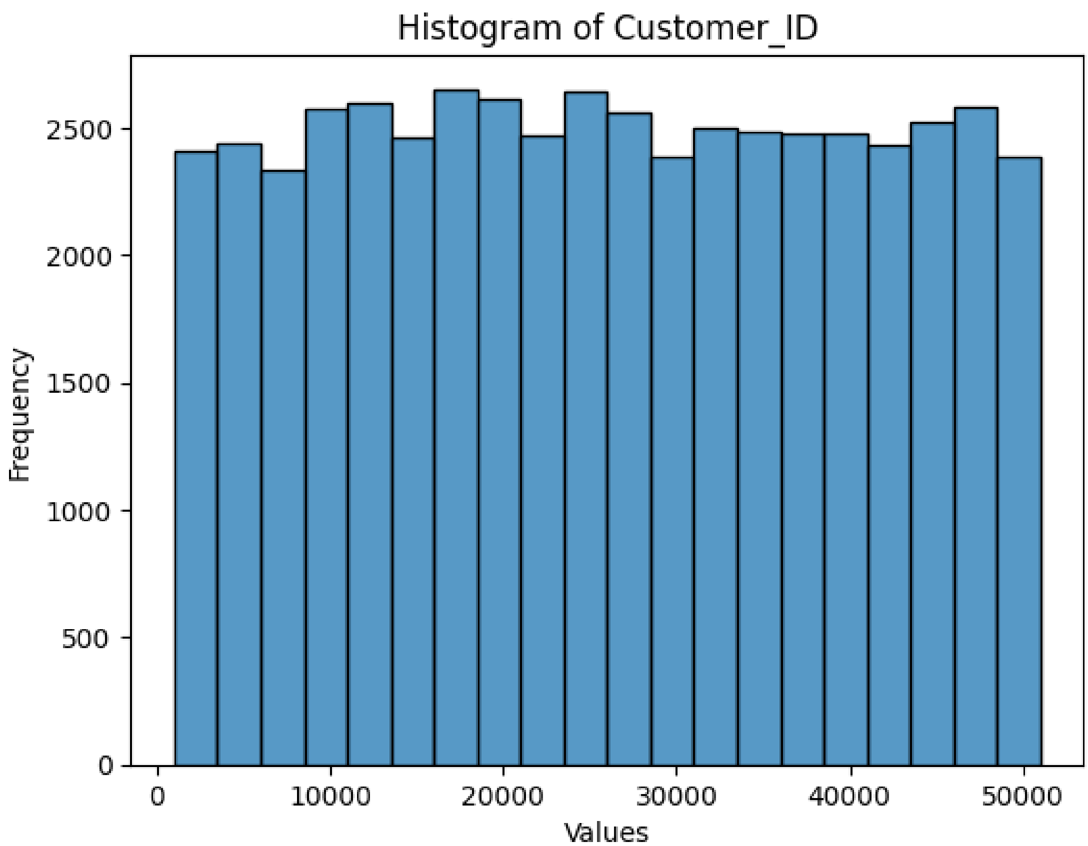
[38]:
```python
figure=px.pie(df["Occupation"].value_counts().reset_index(), values="count",␣
↪names="Occupation", title="Distribution of Occupation")
figure.show()
```



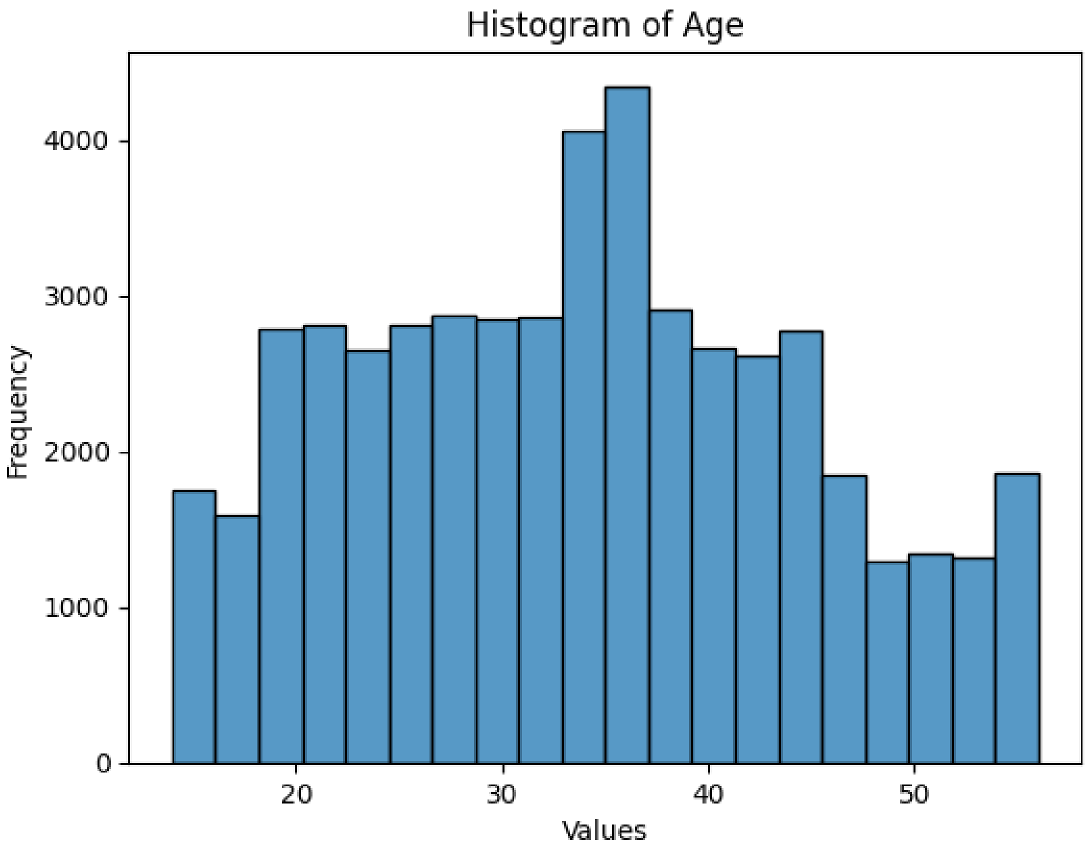
[39]:

```
for col in df.select_dtypes(['int', 'float']):
    sns.histplot(x=col, data=df, bins=20)
    plt.title(f"Histogram of {col}")
    plt.xlabel('Values')
    plt.ylabel('Frequency')
    plt.show()
```

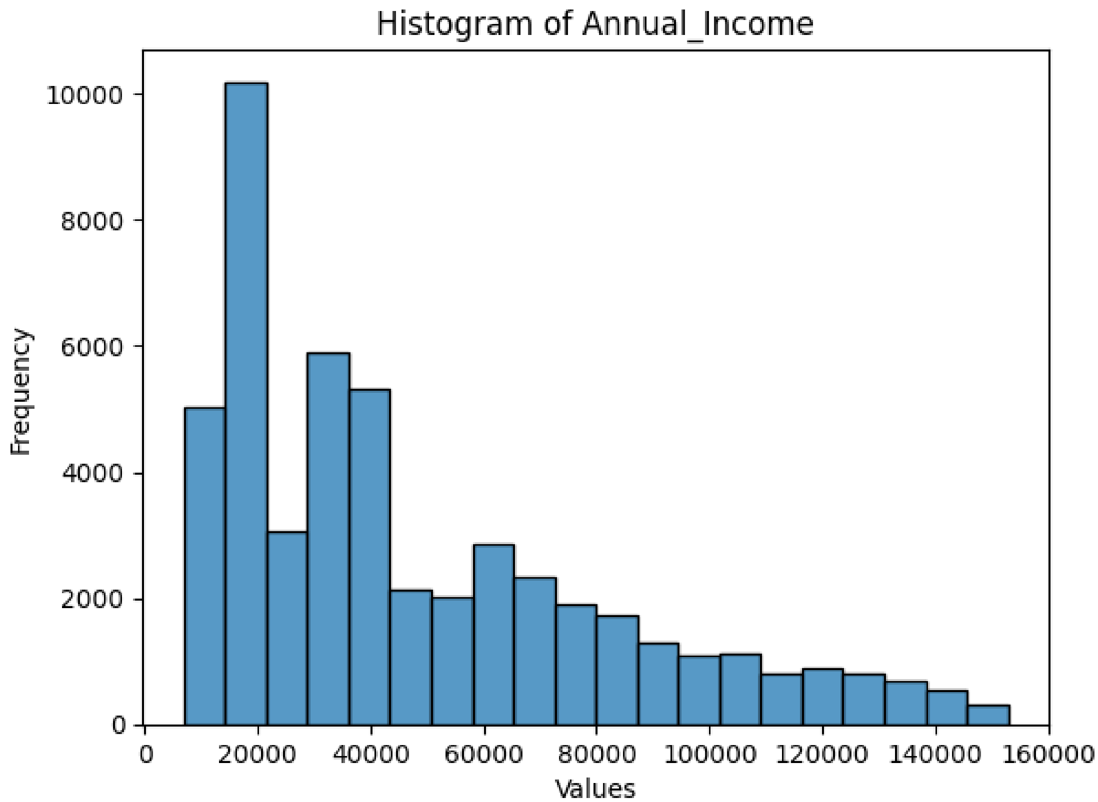
```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



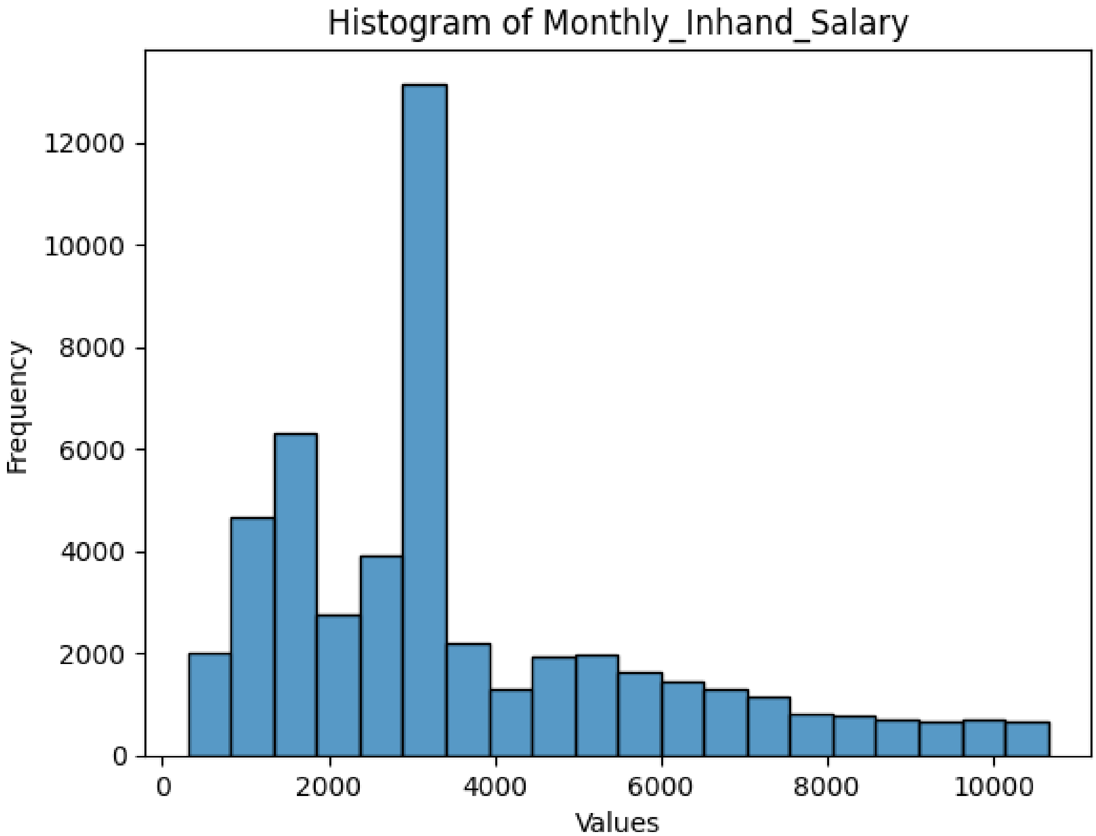

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



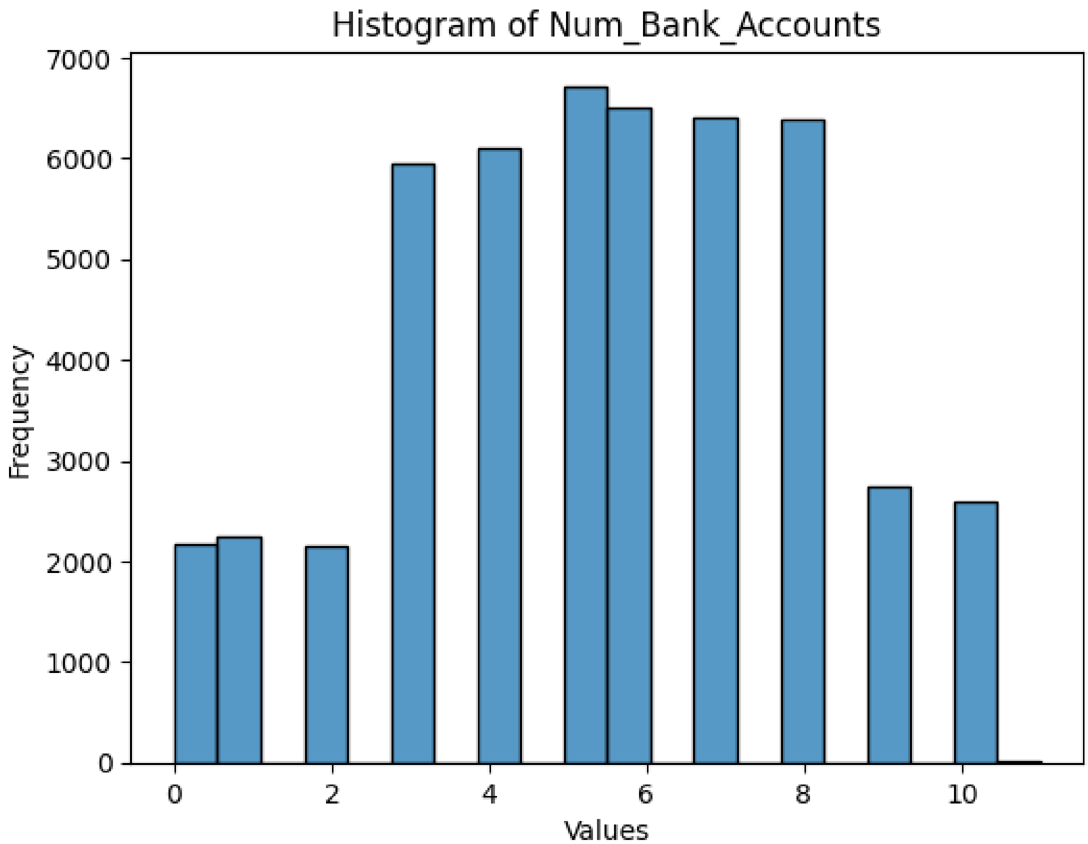

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



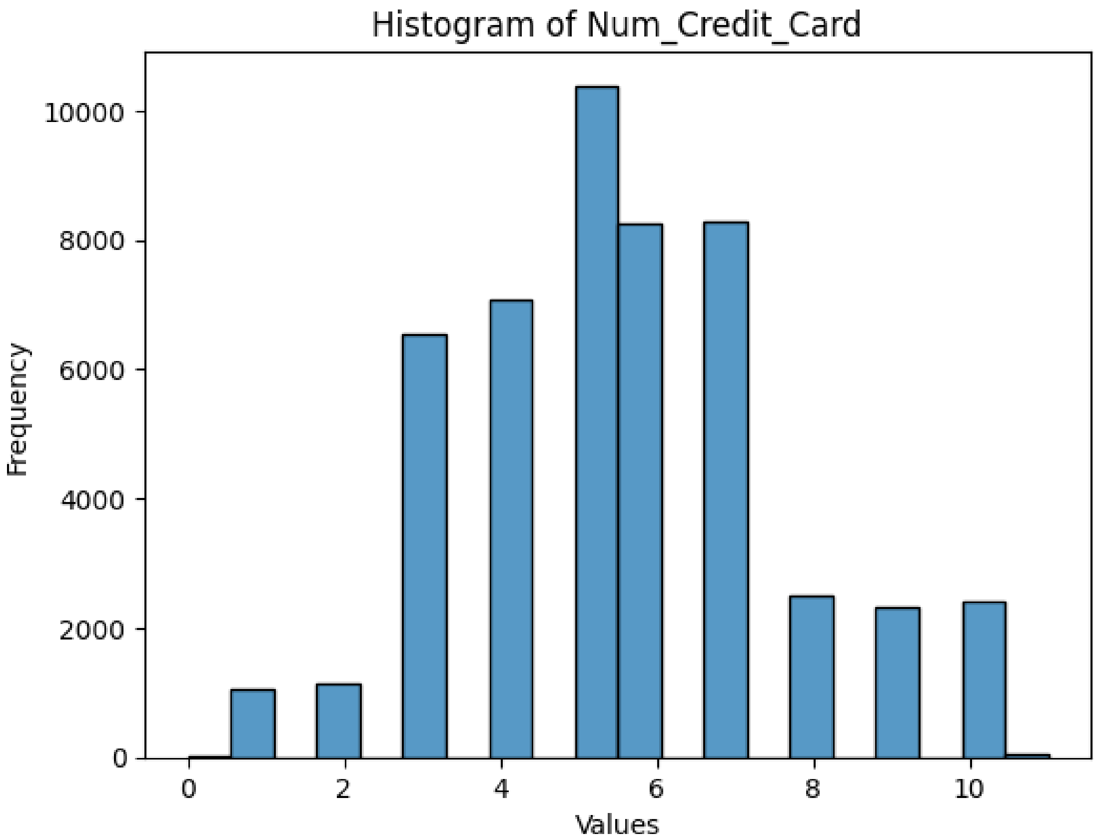

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



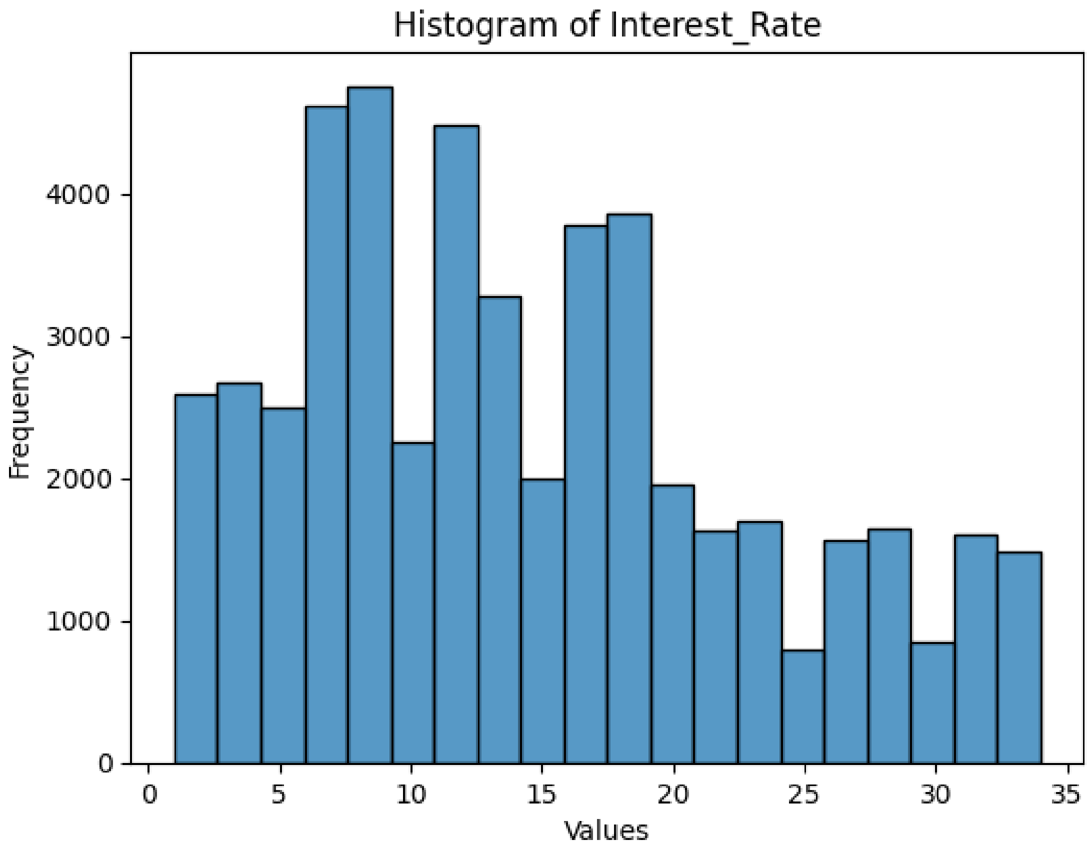

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



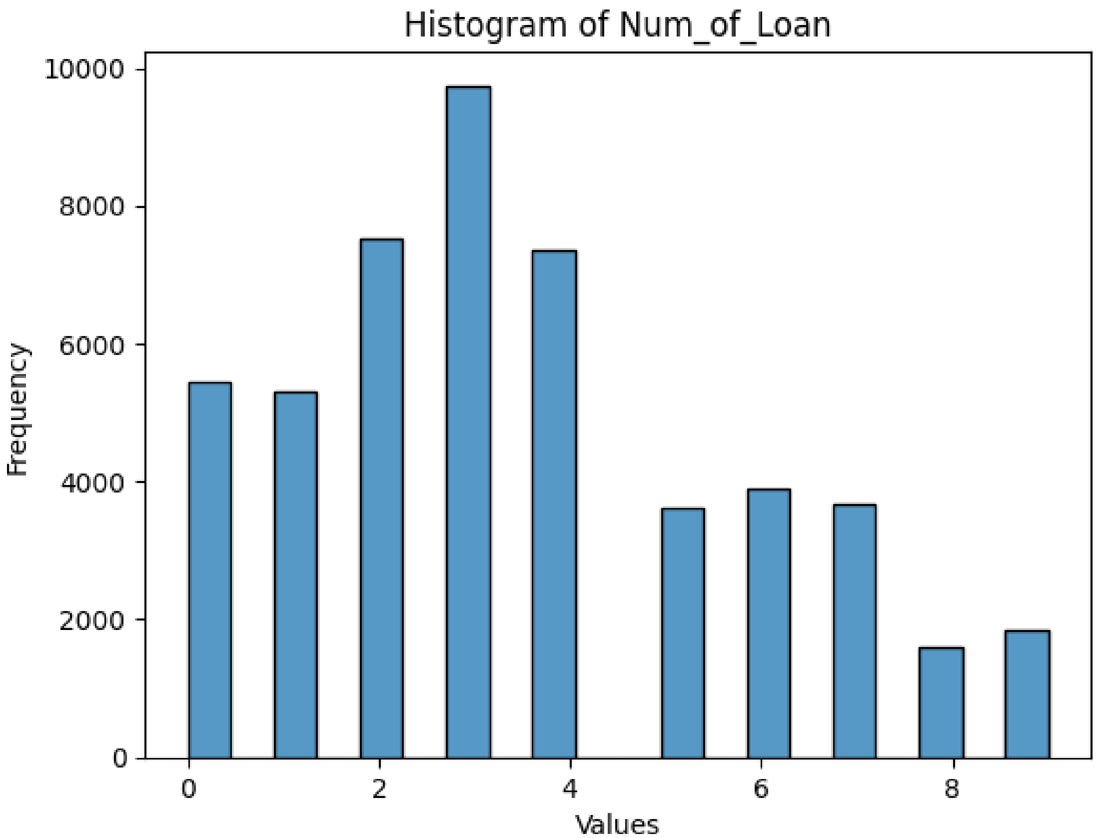

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



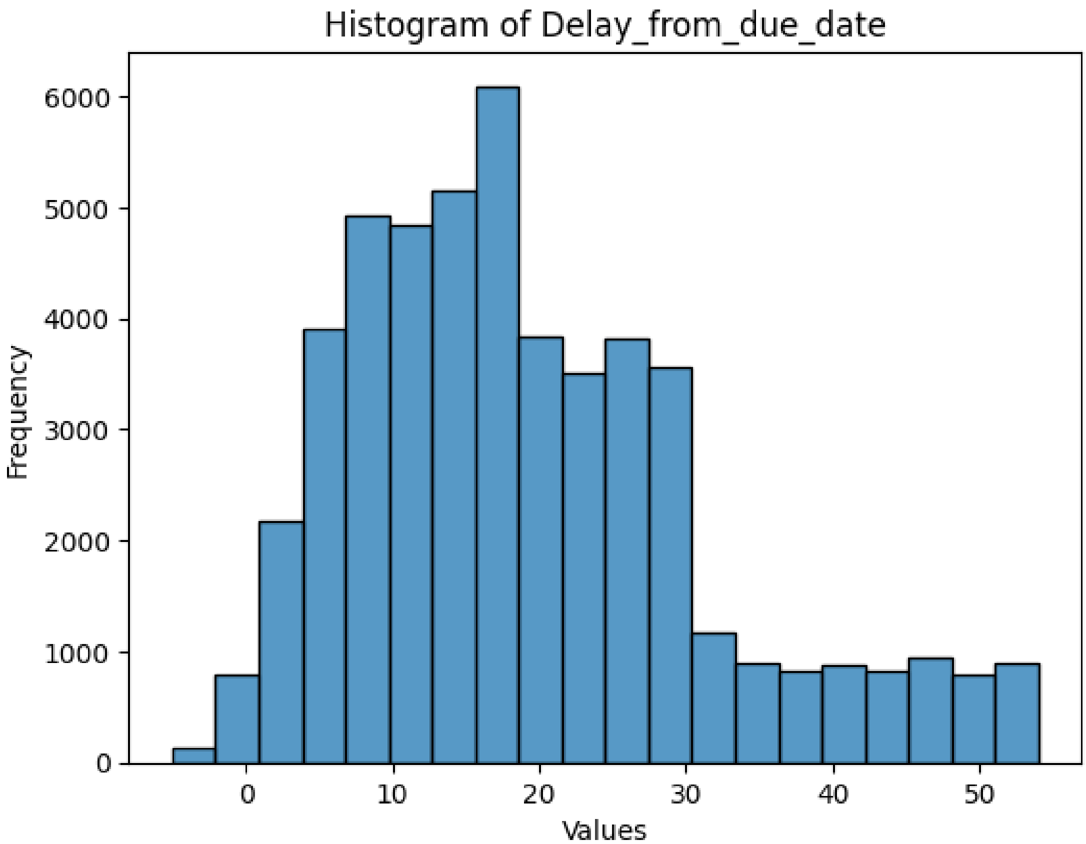

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



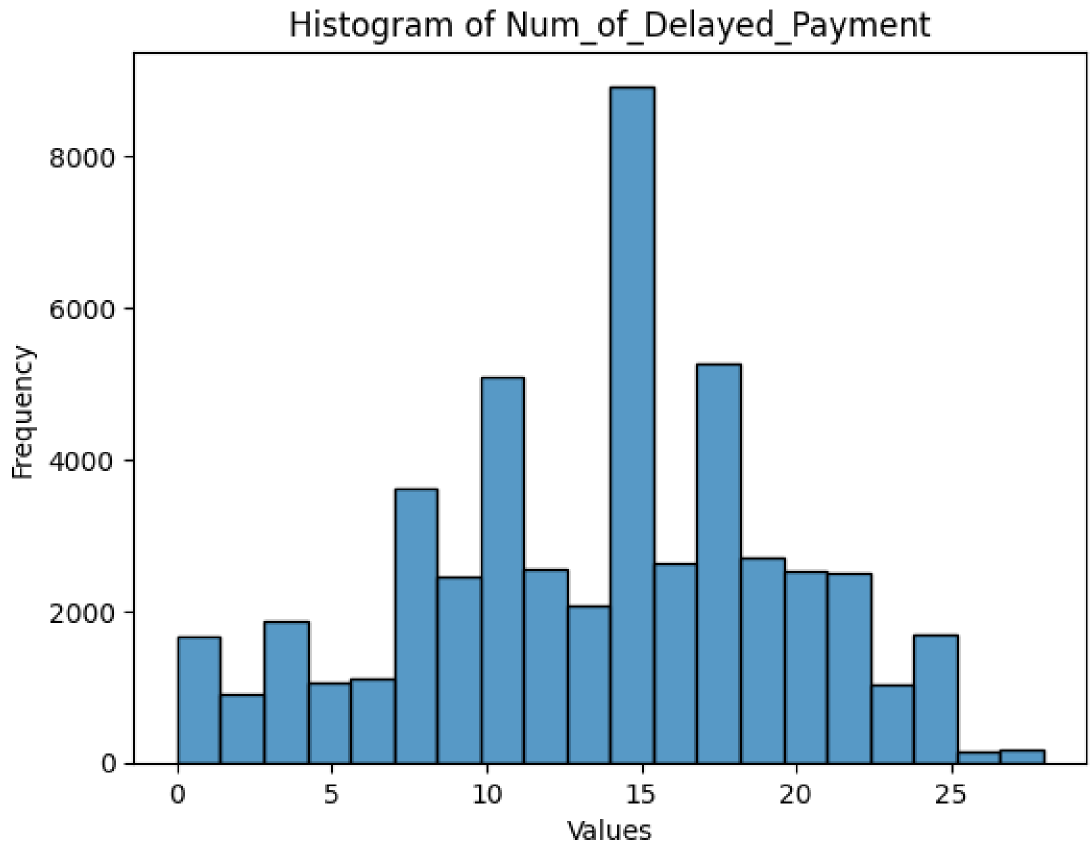

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



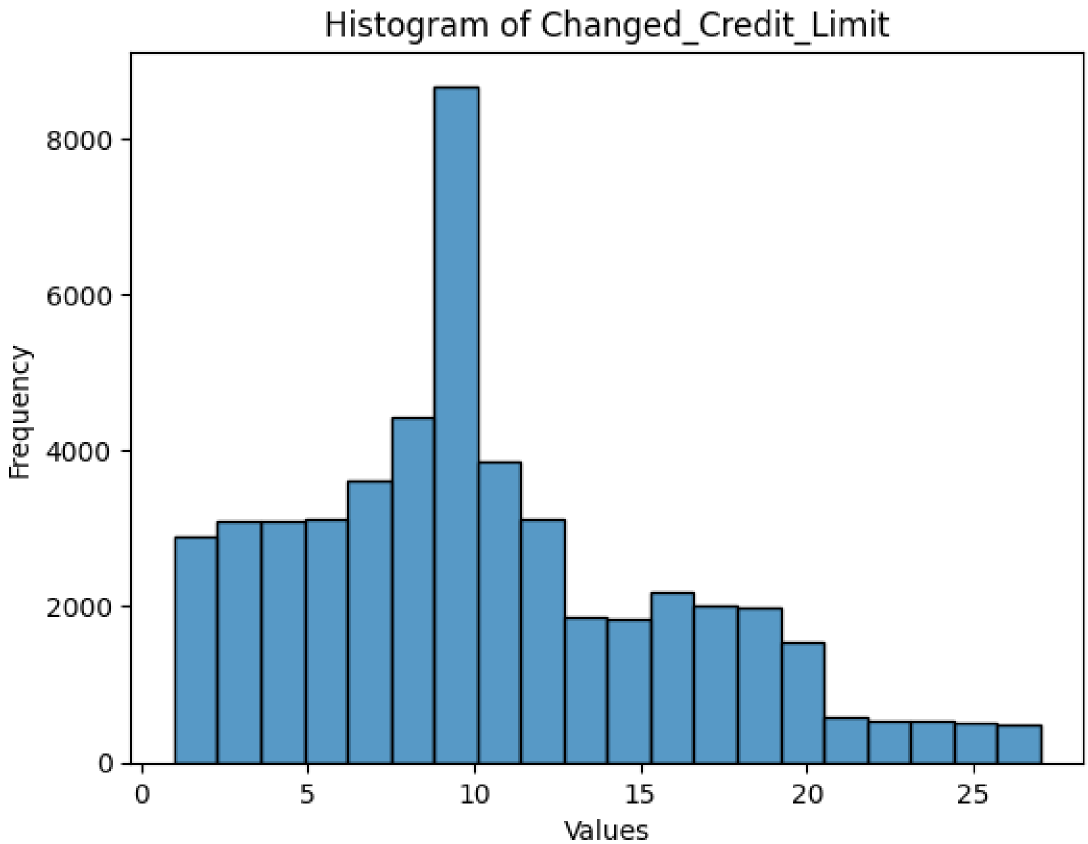

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



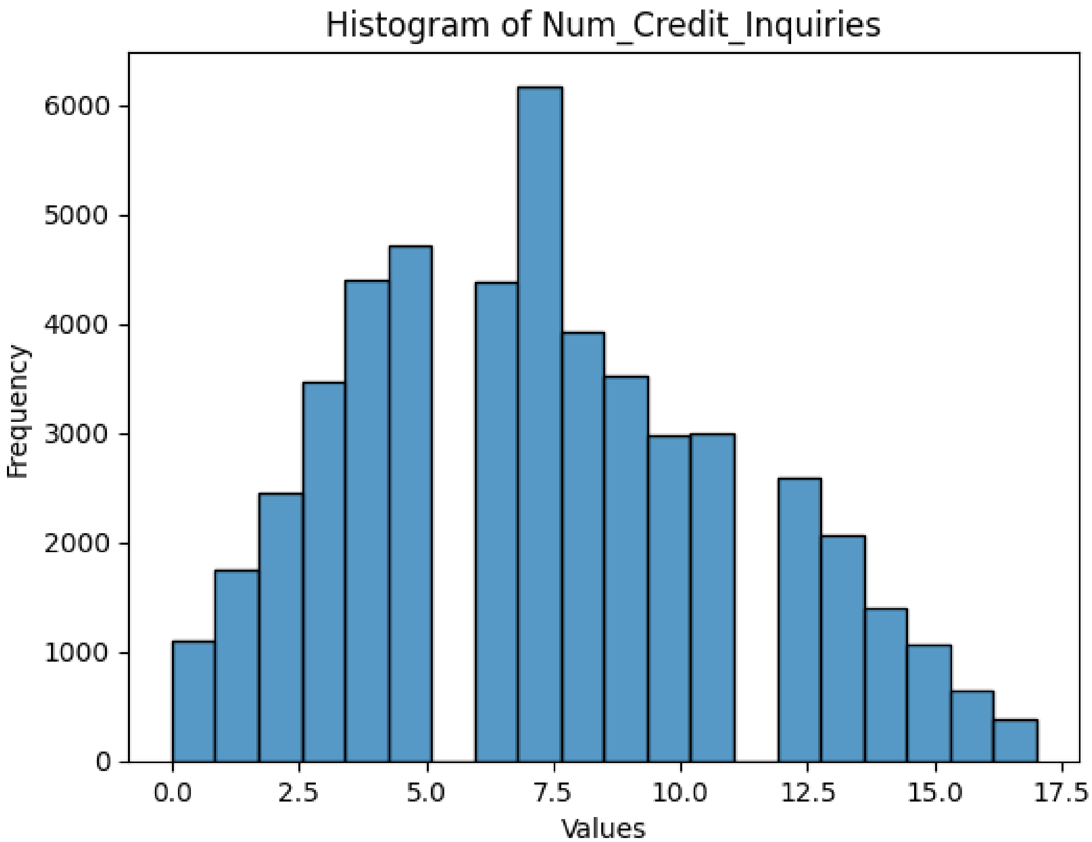

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



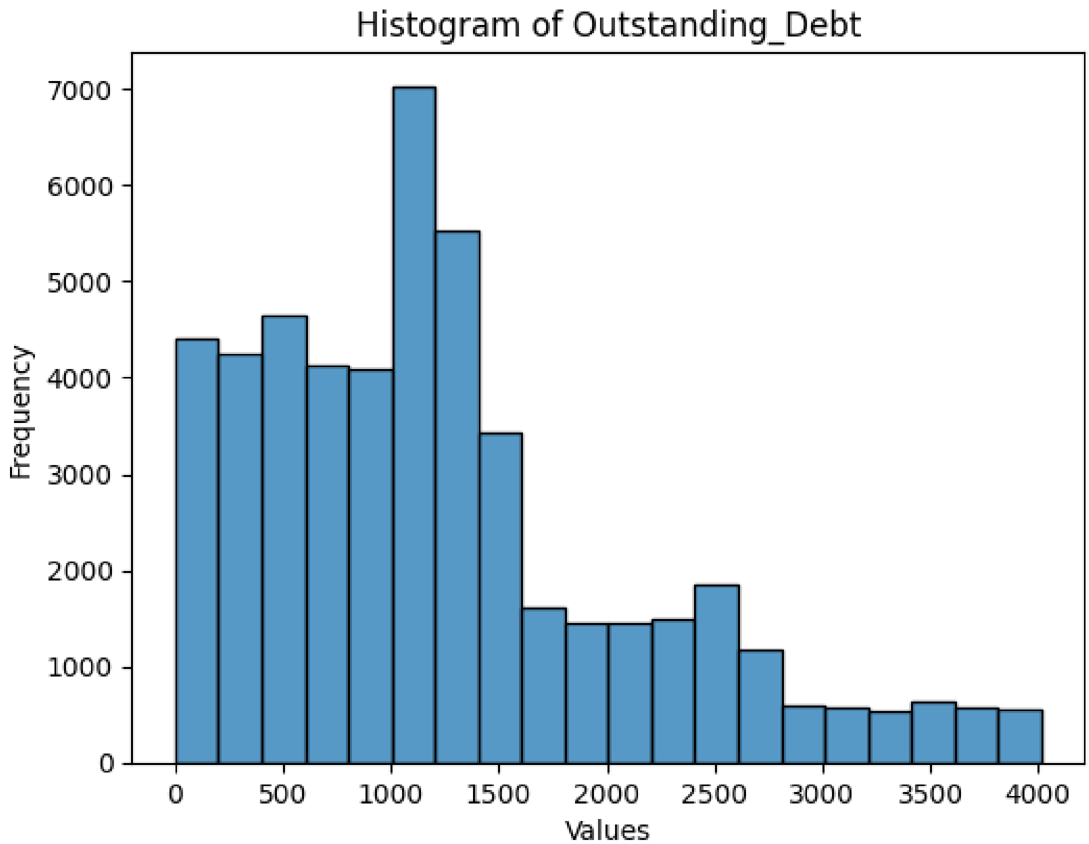

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



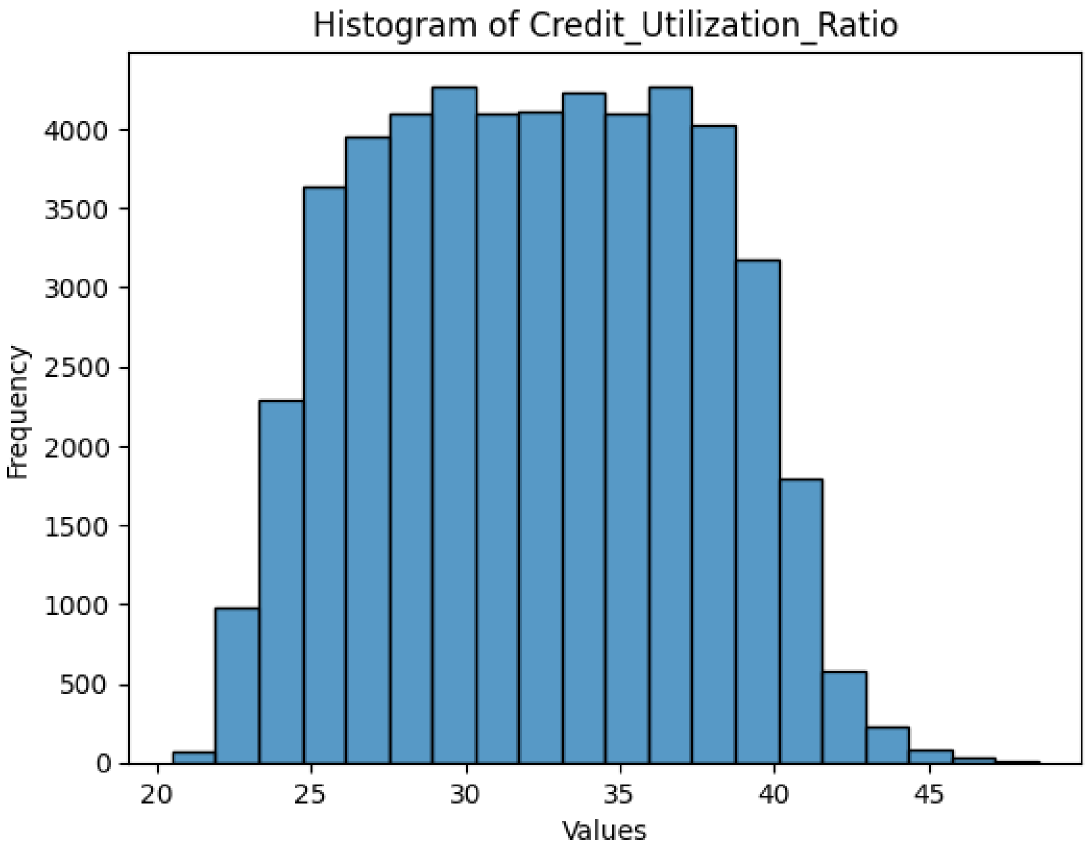

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



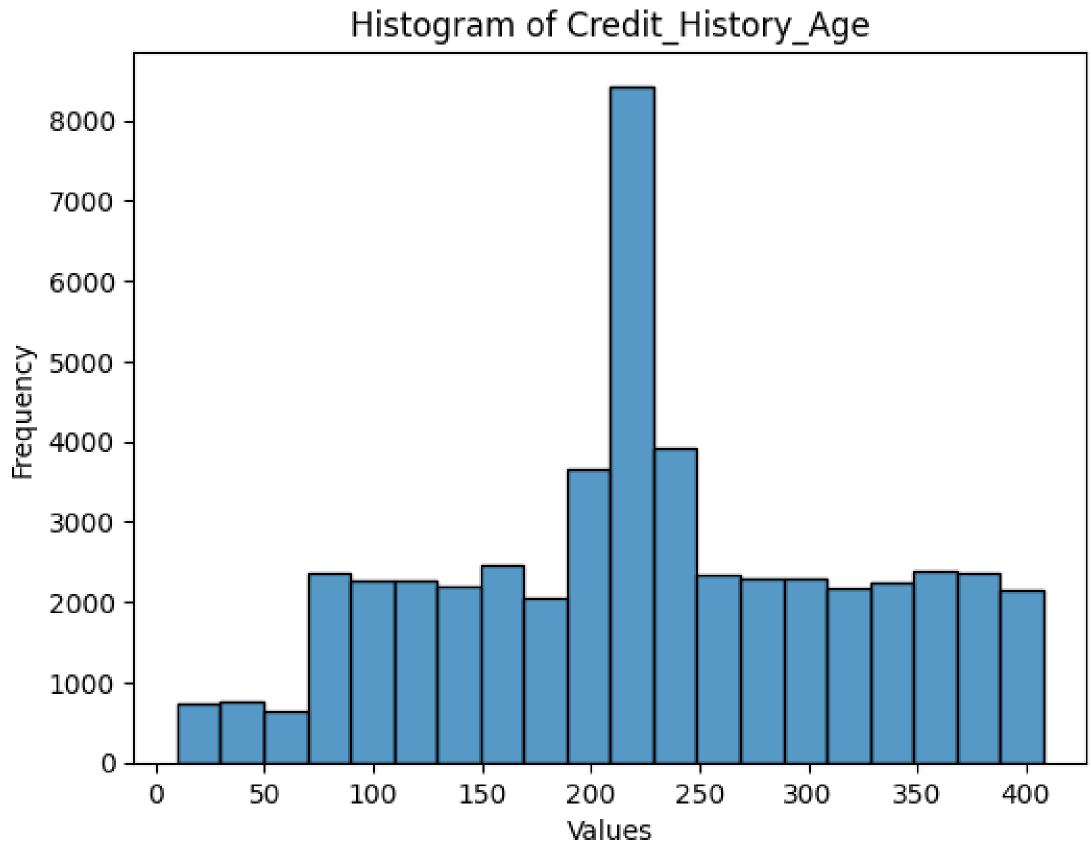

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



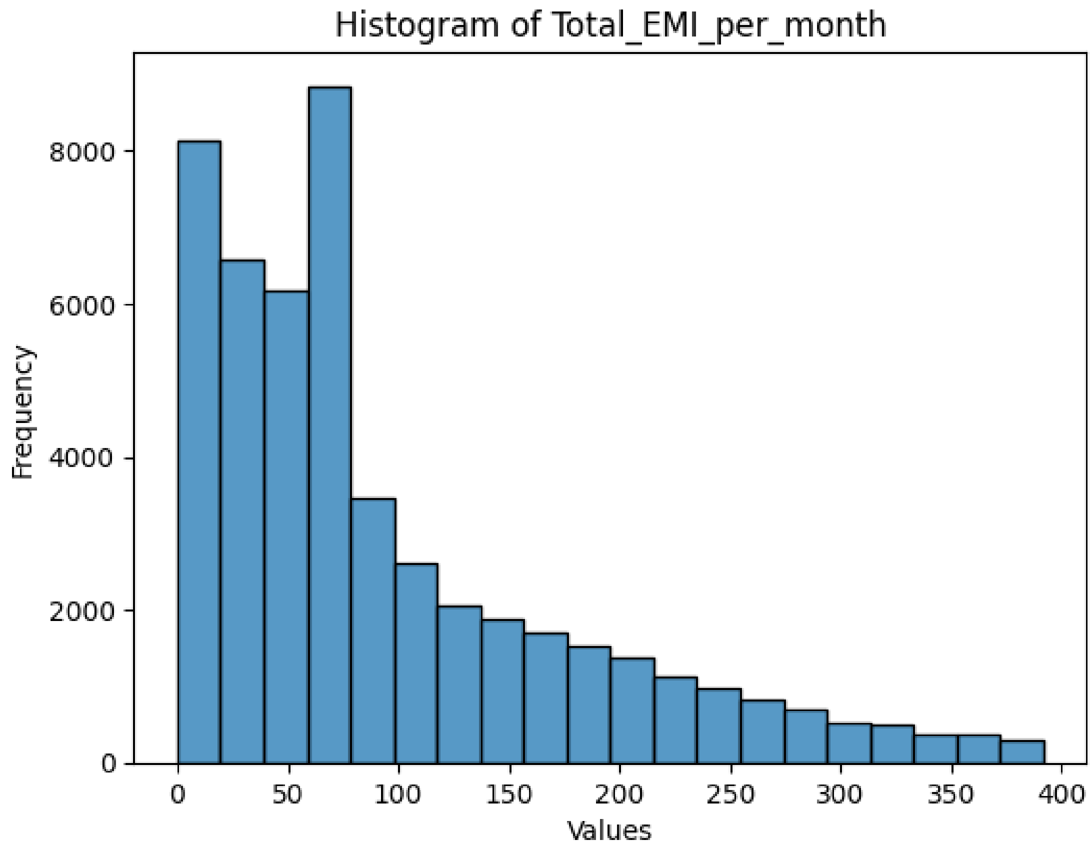

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



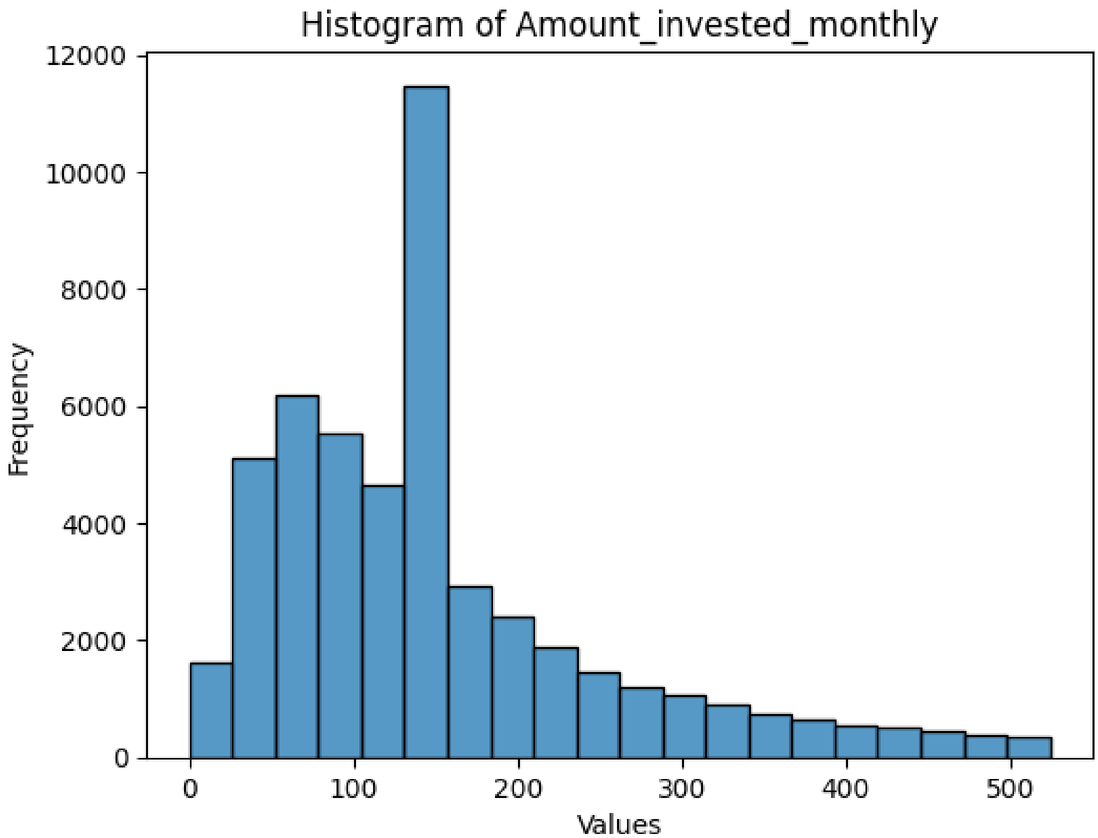

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



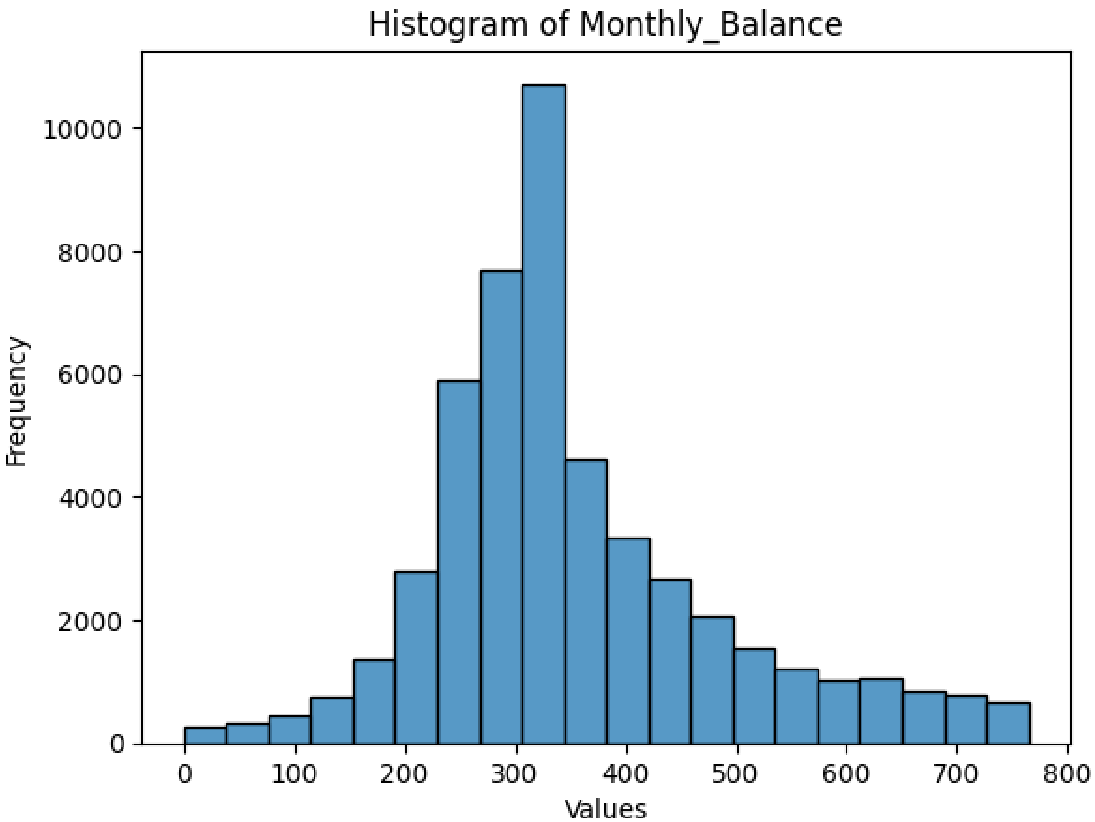

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



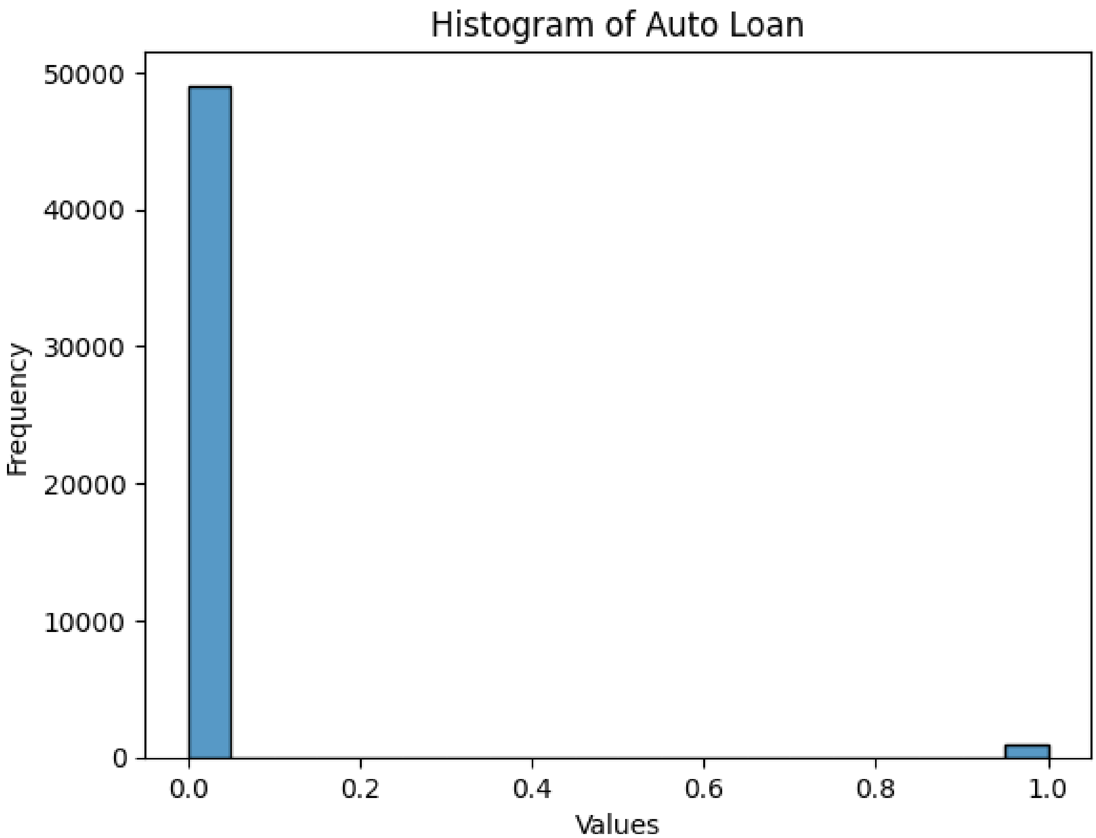

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



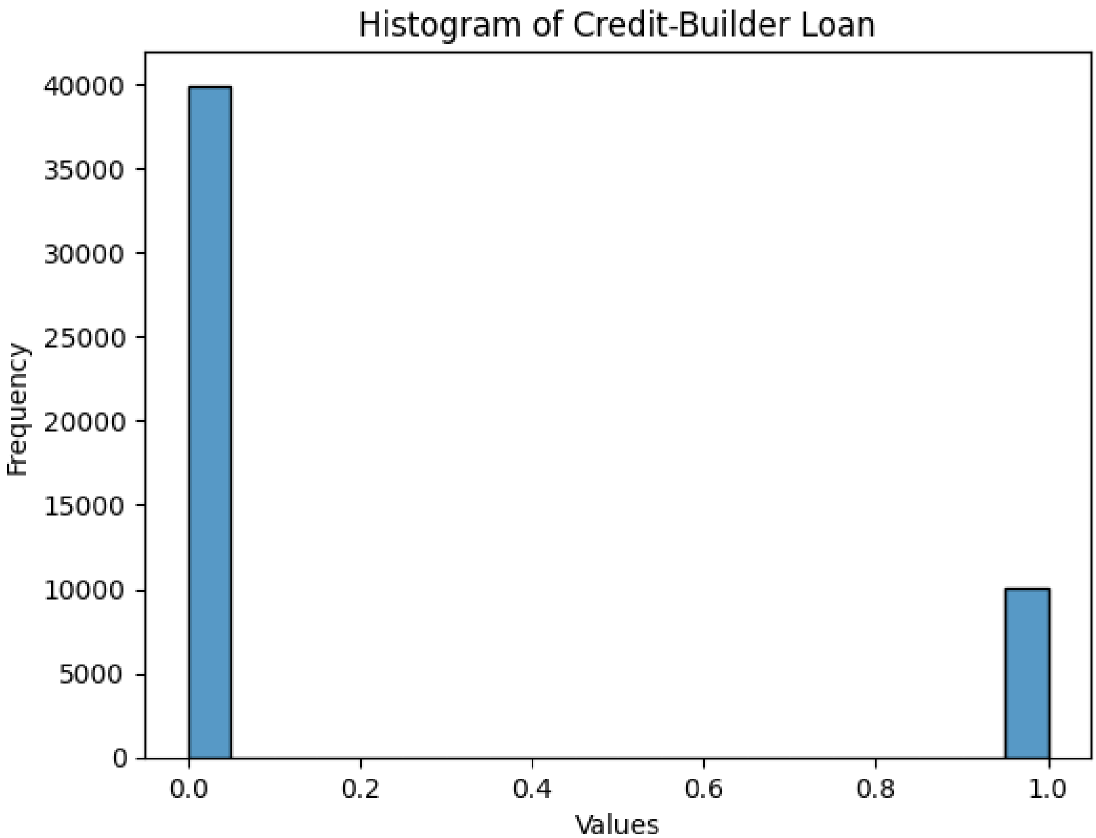

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



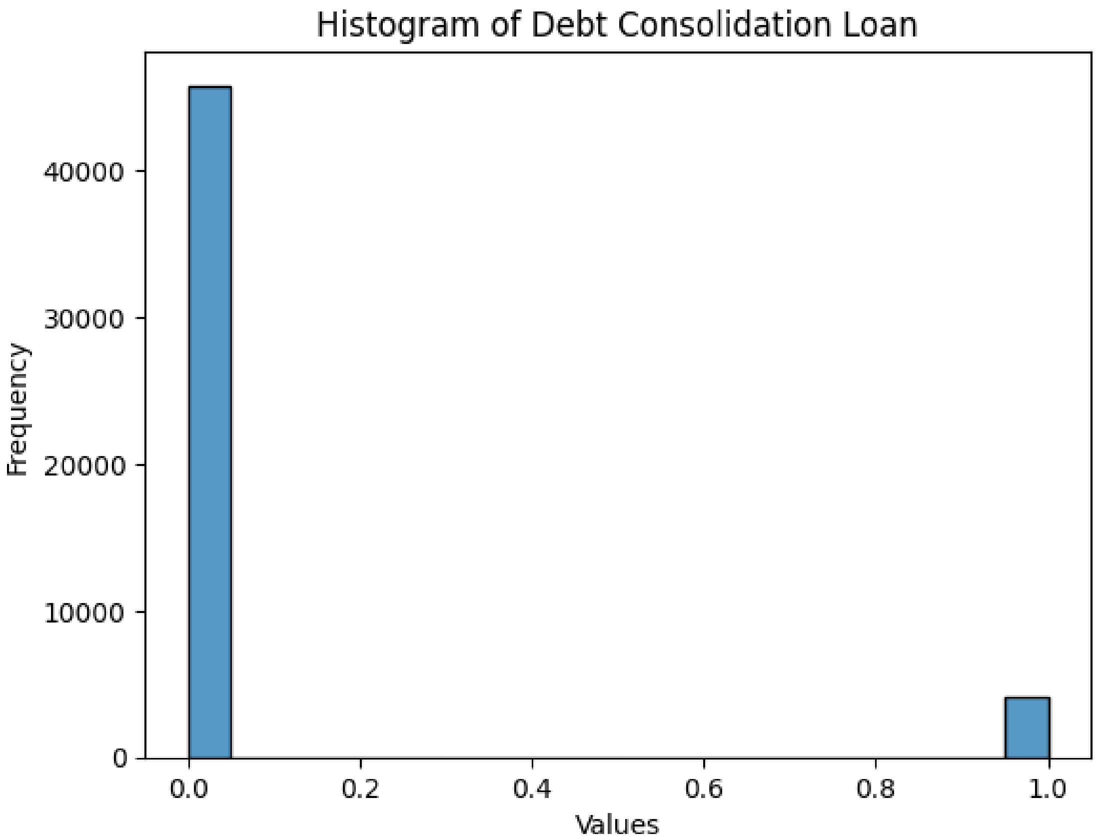

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



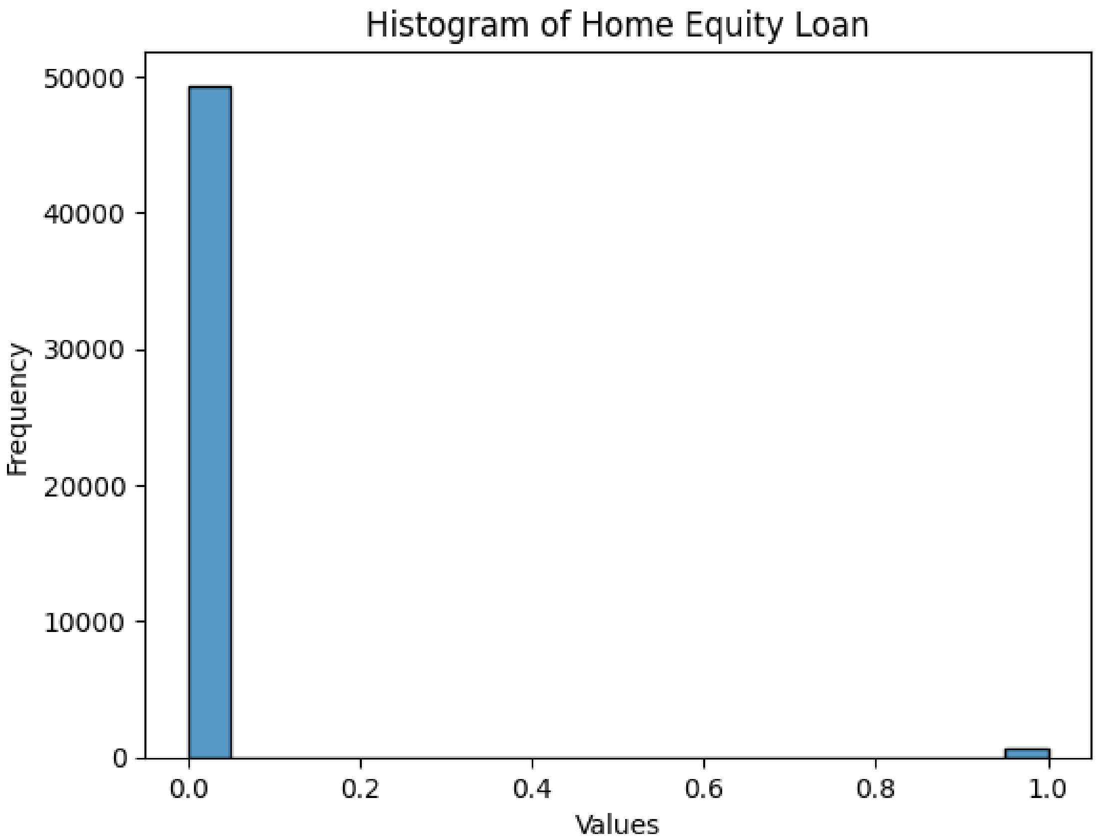

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



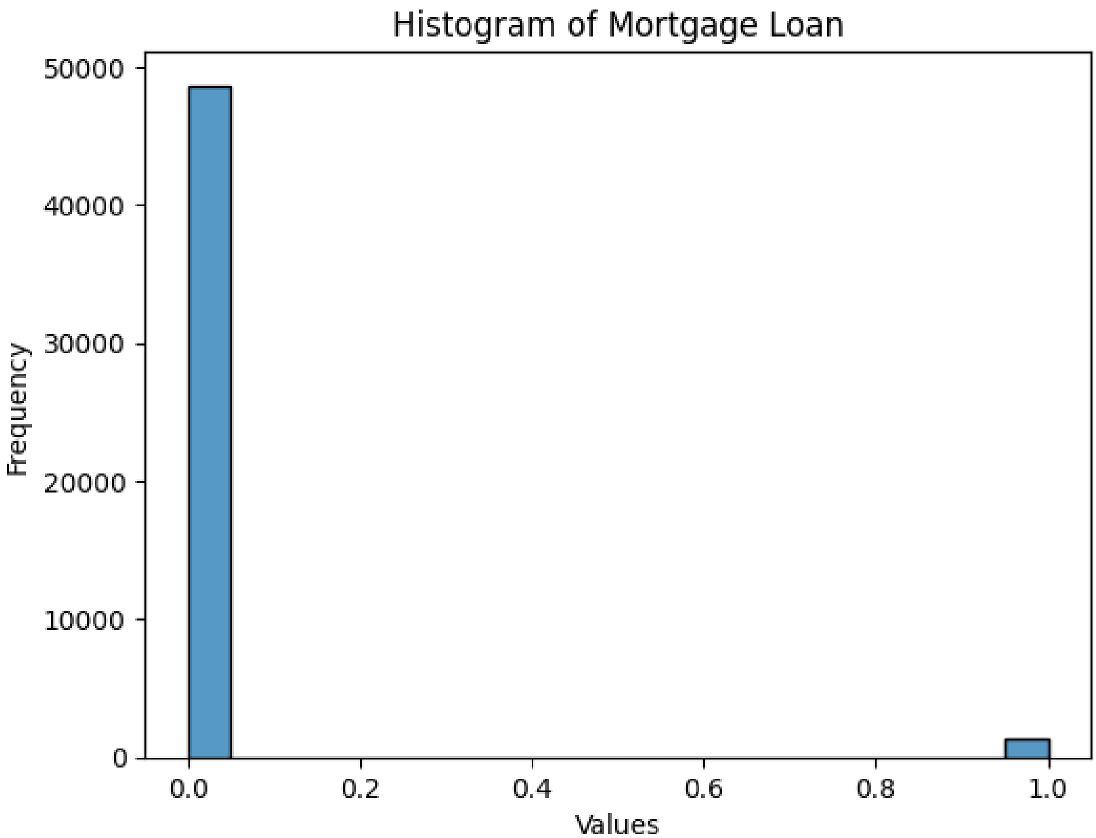

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



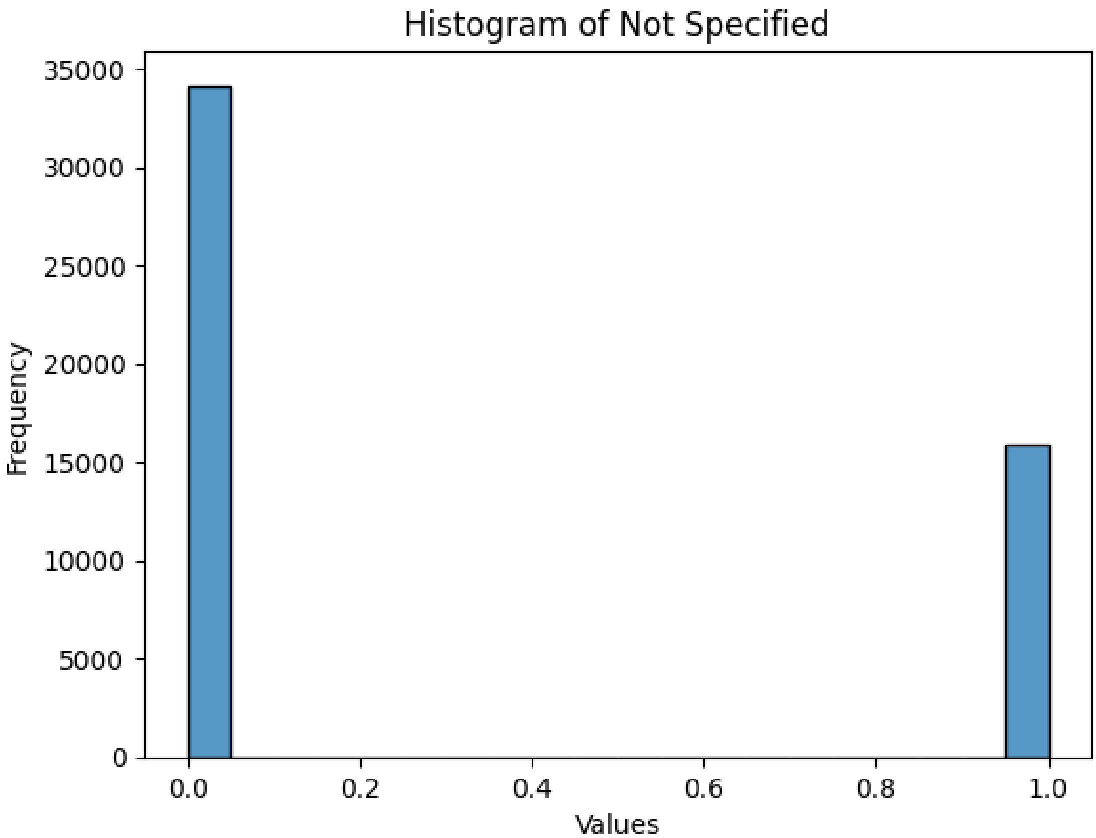

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



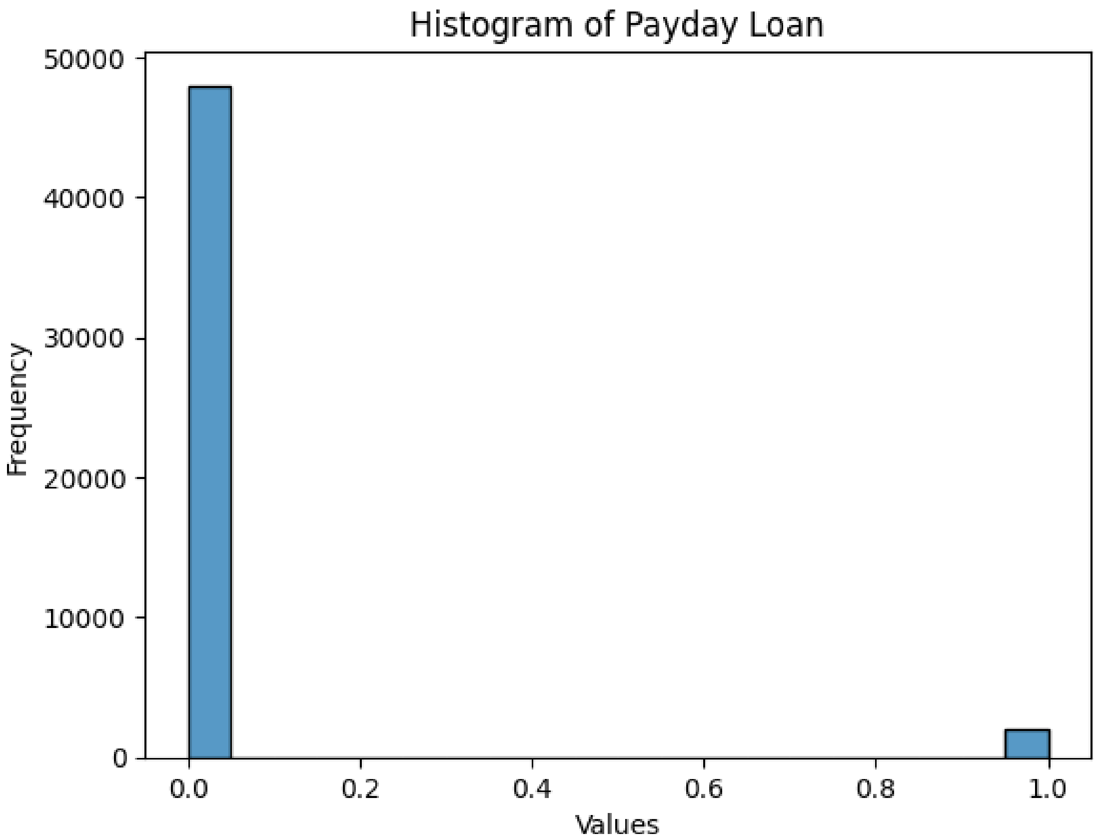

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



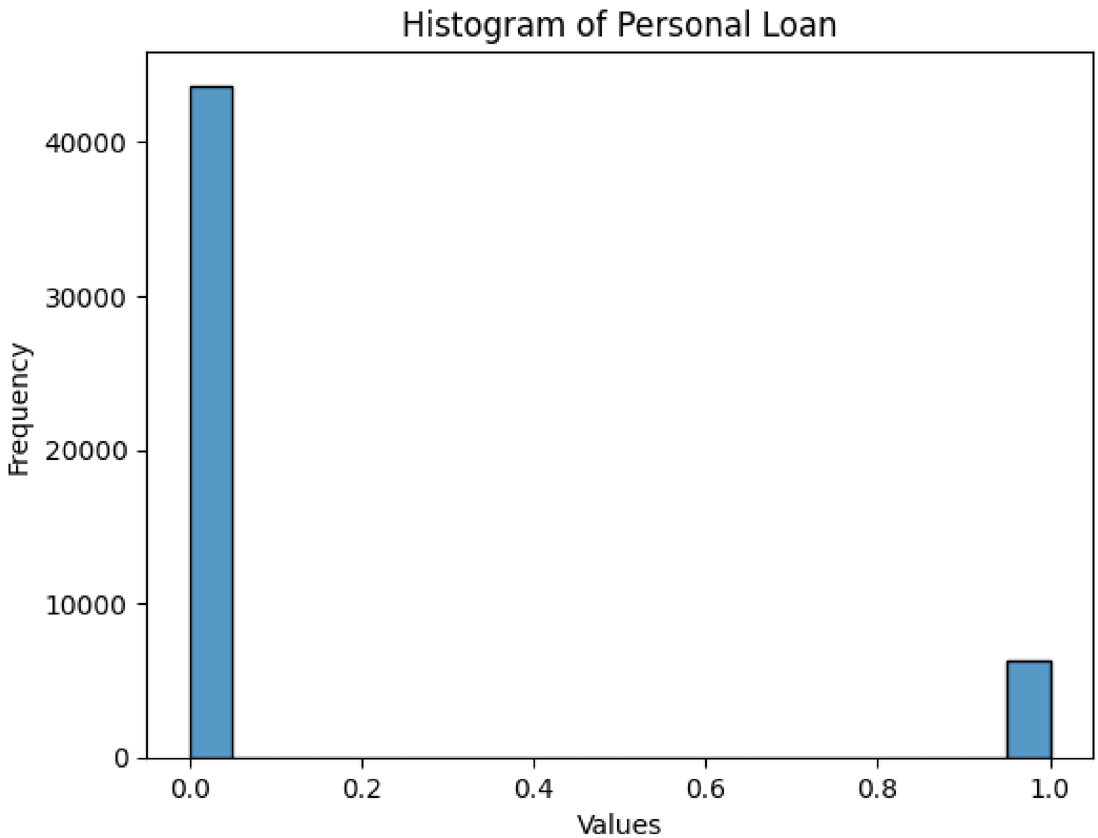

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```



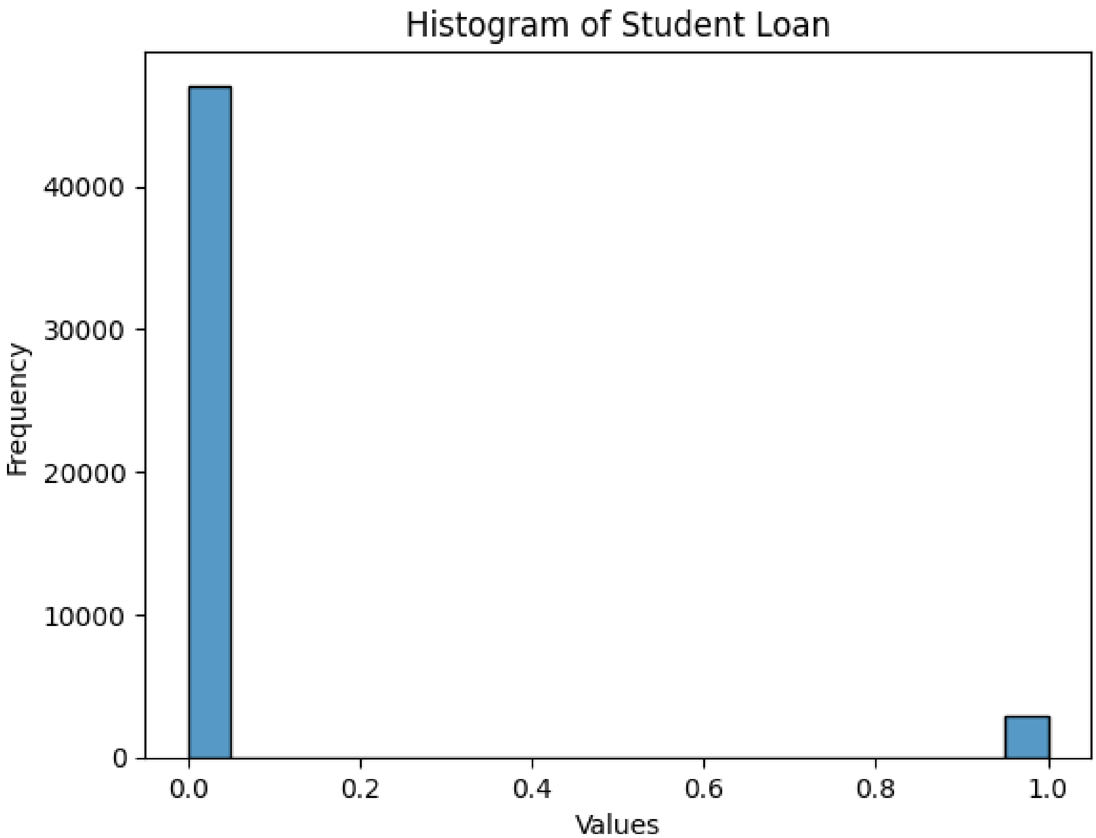

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```




```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```




```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```

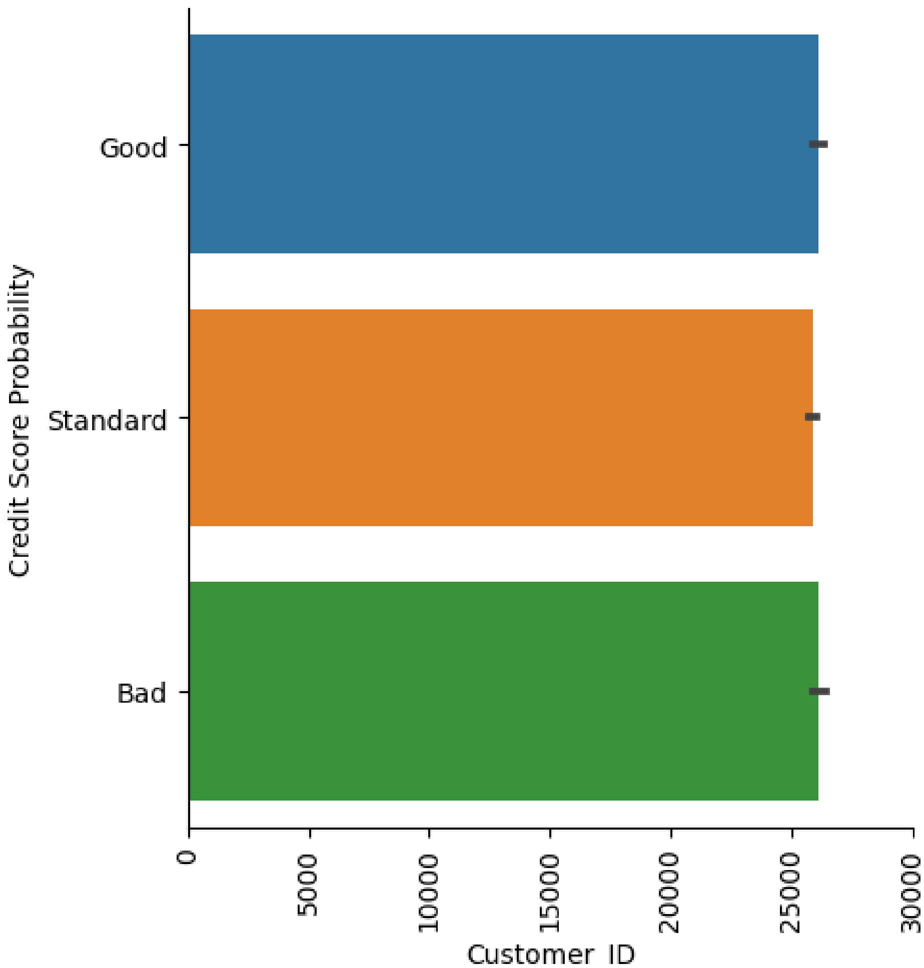


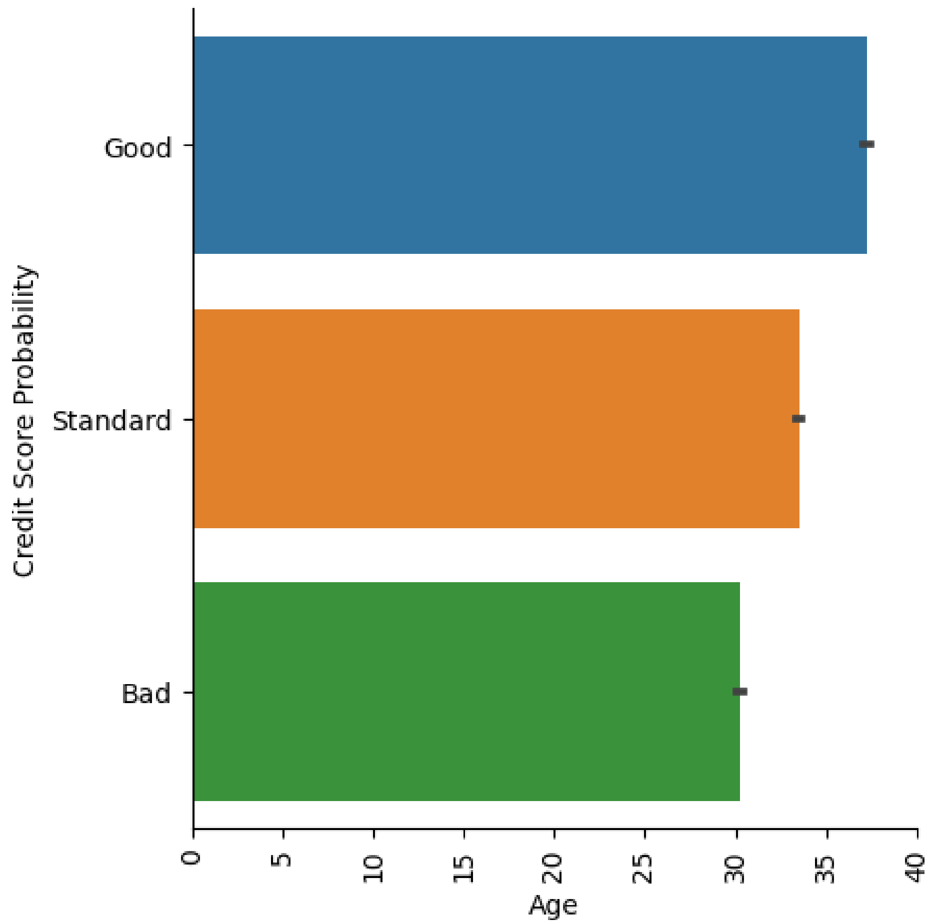
[40]:
```
x = df.drop(['Credit_Score'], axis=1)
```

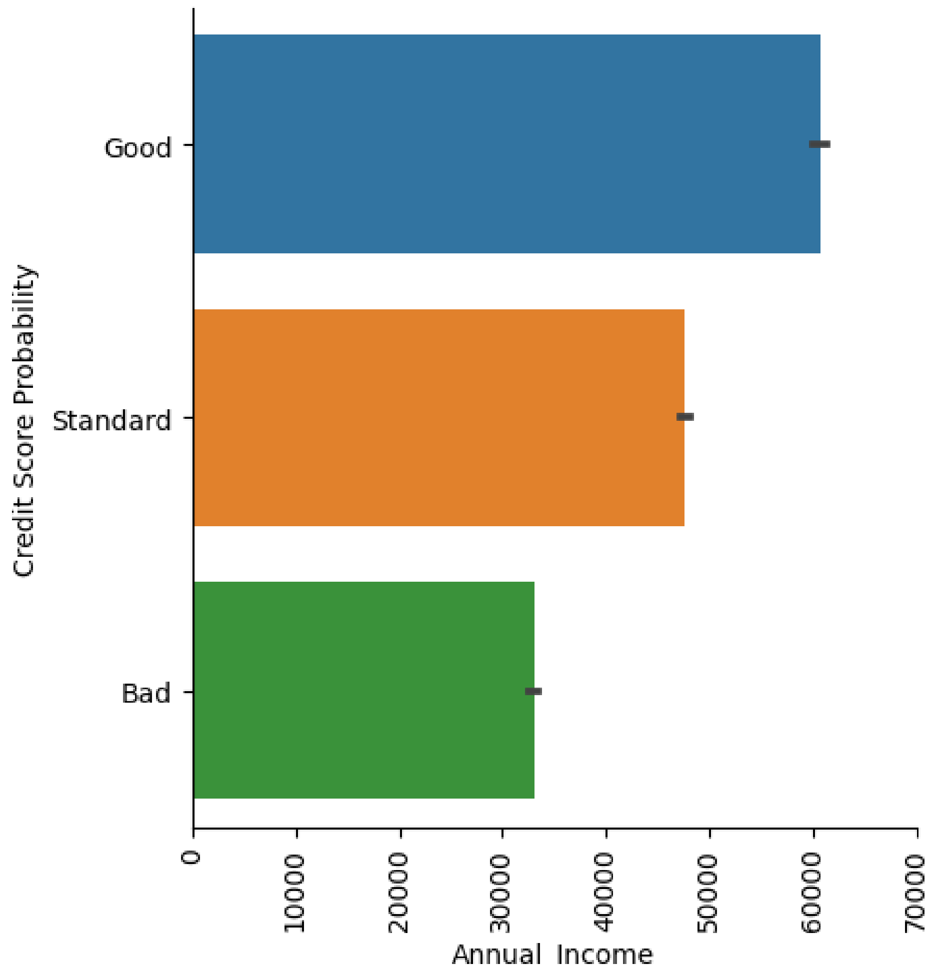
[41]:
```
y = df['Credit_Score'].map({"Bad": 0, "Standard": 1, "Good": 2})
```

[70]:
```
for col in x.select_dtypes(['int', 'float']):
    plot = sns.catplot(x=col, y='Credit_Score', data=df, kind="bar")
    plot.set_axis_labels(f"{col}", "Credit Score Probability")
    plot.set_xticklabels(rotation=90)
```

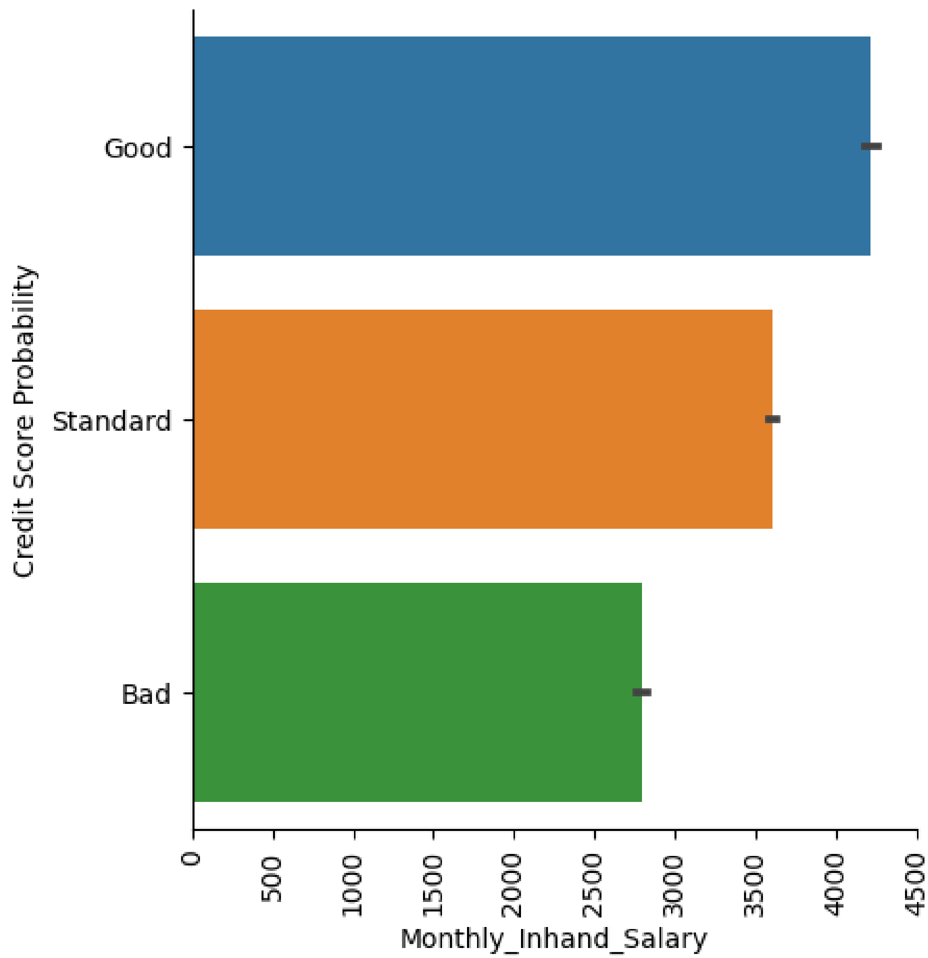
```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/axisgrid.py:447:
RuntimeWarning:

More than 20 figures have been opened. Figures created through the pyplot
interface (`matplotlib.pyplot.figure`) are retained until explicitly closed and
may consume too much memory. (To control this warning, see the rcParam
`figure.max_open_warning`). Consider using `matplotlib.pyplot.close()`.
```

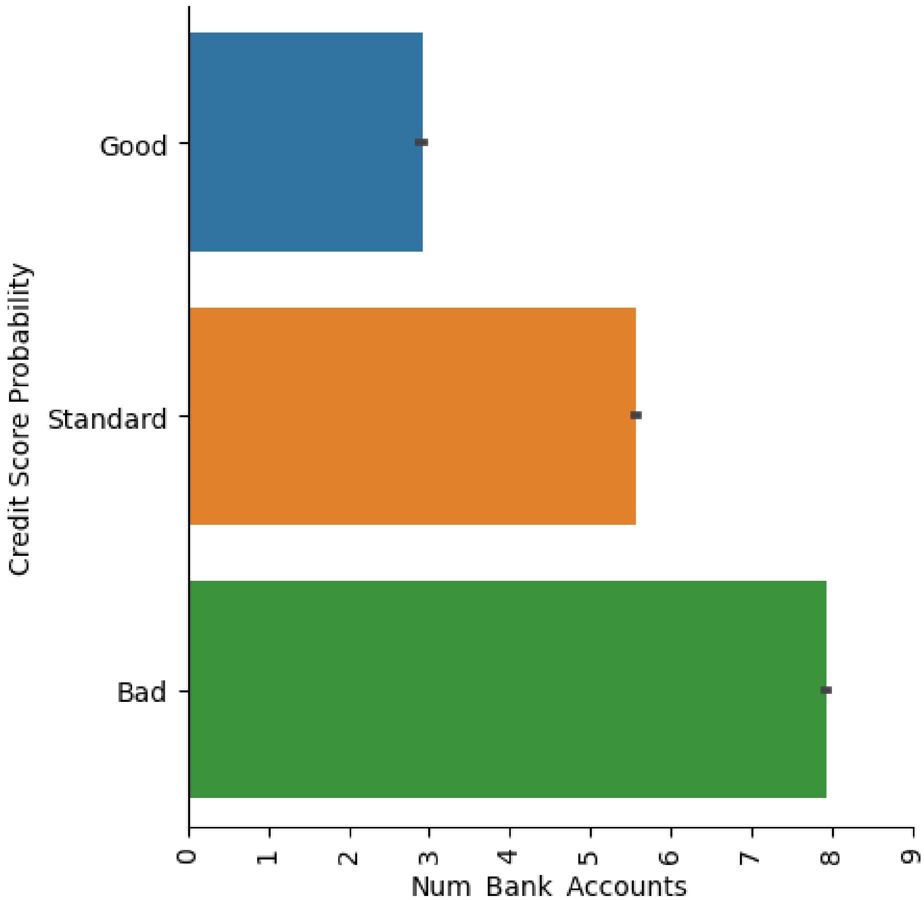

Good
Standard
Bad
Credit Score Probability
0
5000
10000
15000
20000
25000
30000
Customer_ID


Credit Score Probability
Good
Standard
Bad
0
5
10
15
20
25
30
35
40
Age

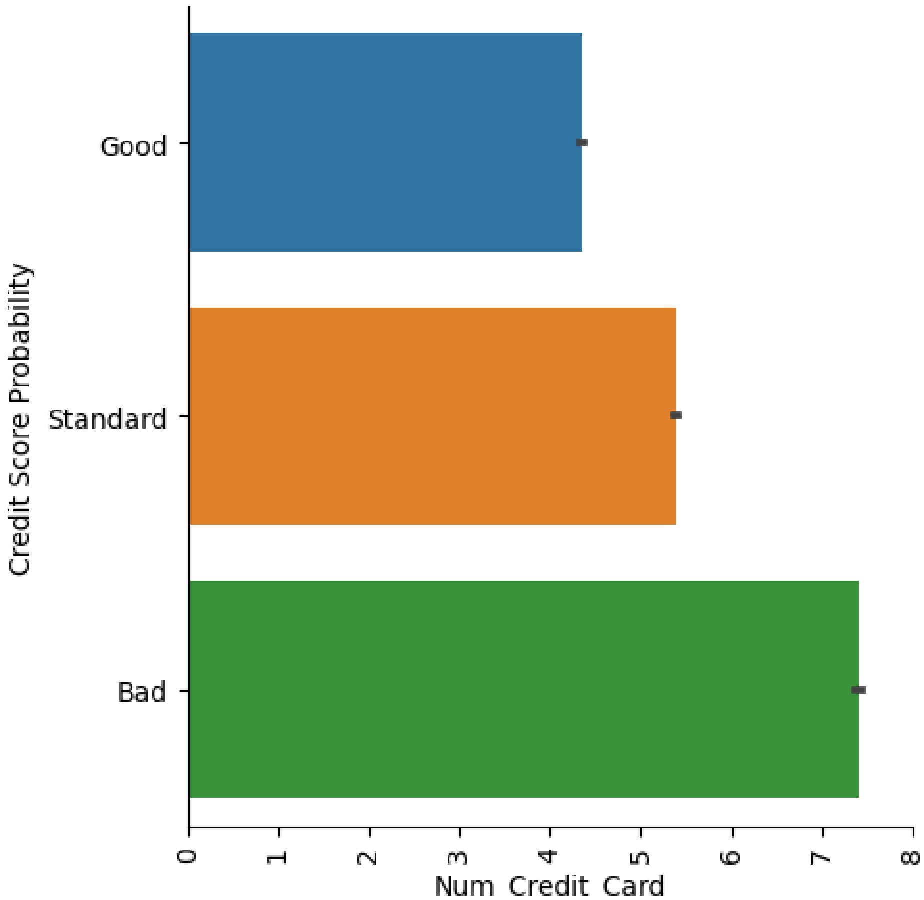


Good
Standard
Bad
Credit Score Probability
0
10000
20000
30000
40000
50000
60000
70000
Annual_Income

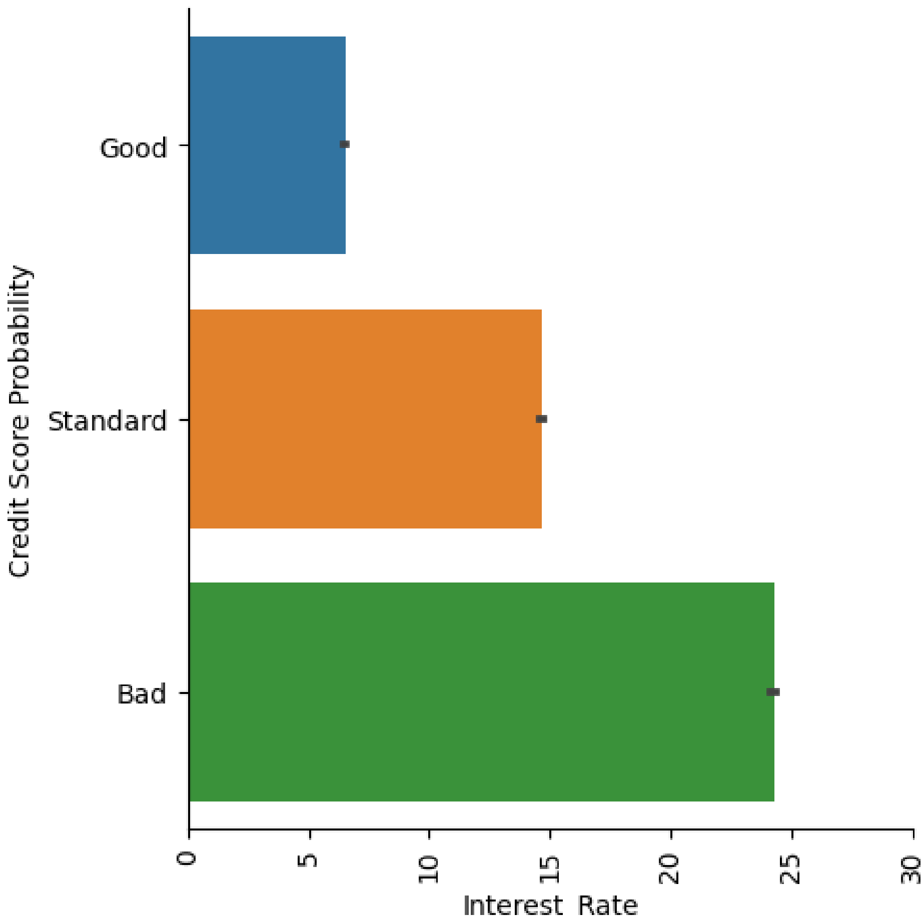

Good
Standard
Bad
Credit Score Probability
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Monthly_Inhand_Salary

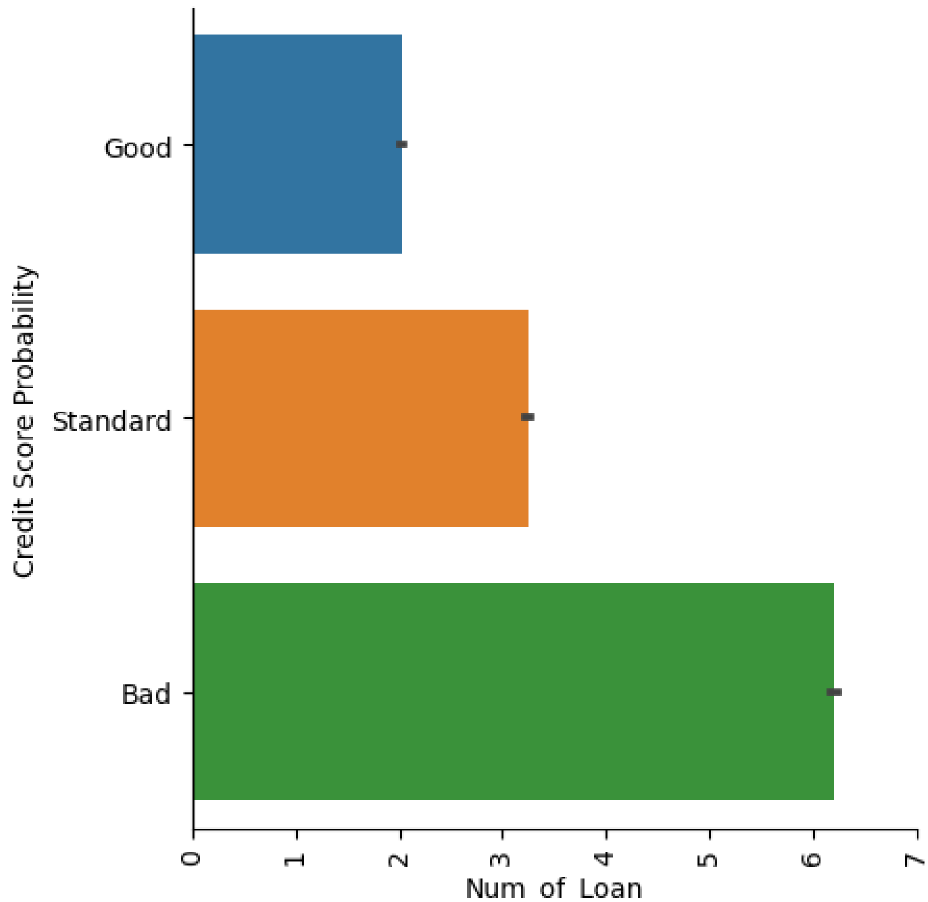


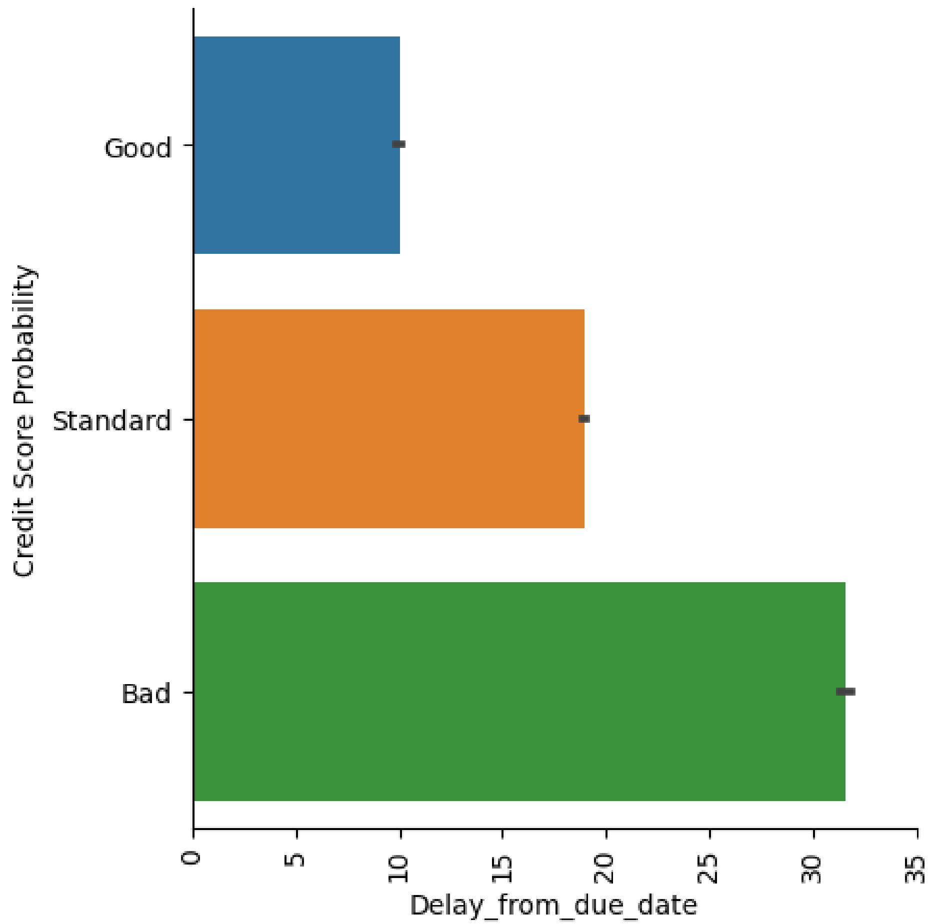
Good
Standard
Bad
Credit Score Probability
0
1
2
3
4
5
6
7
8
9
Num_Bank_Accounts

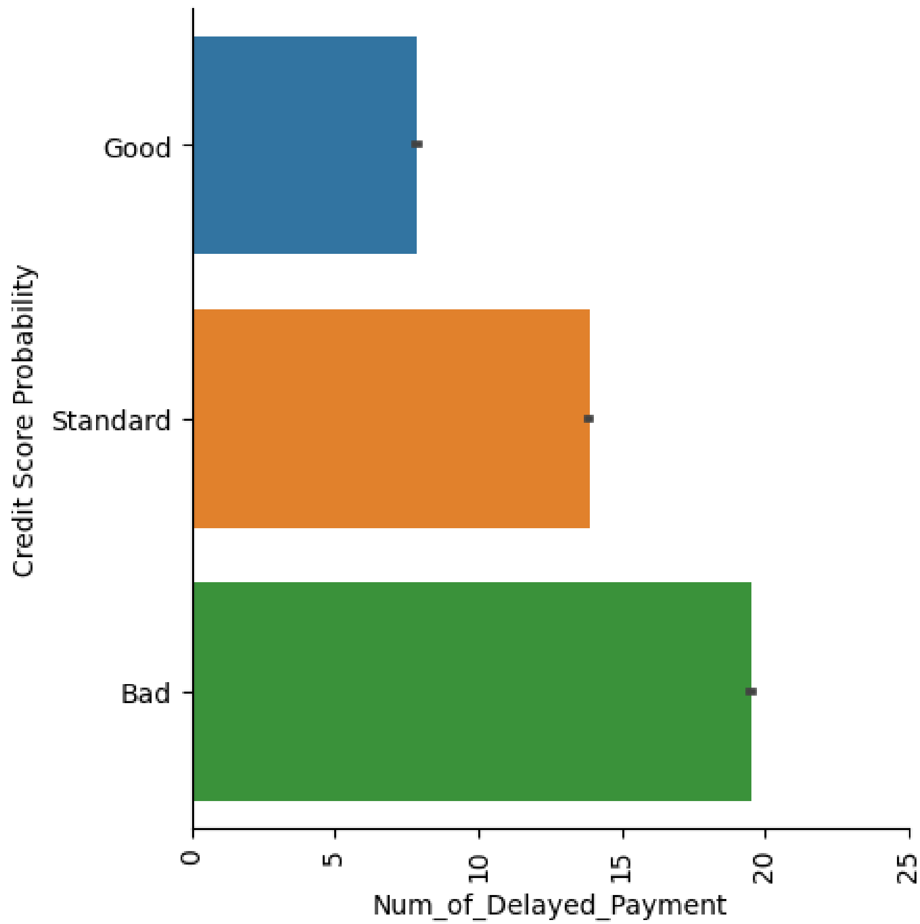


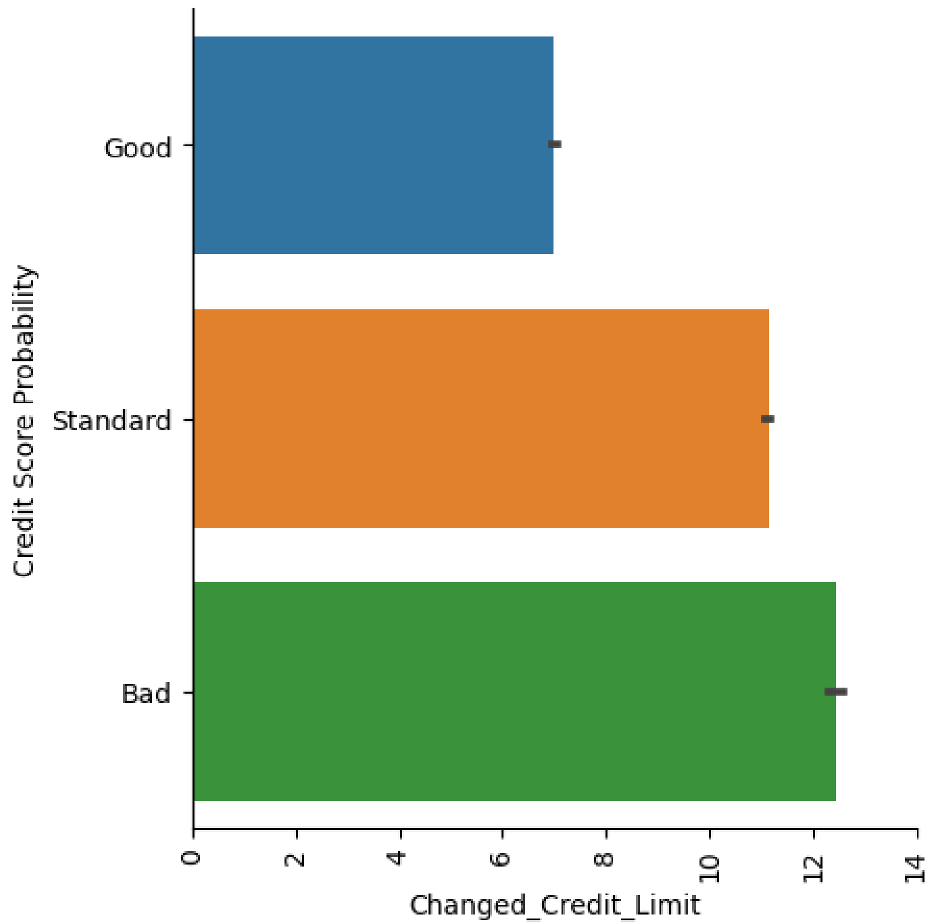
Good
Standard
Bad
Credit Score Probability
0
1
2
3
4
5
6
7
8
Num_Credit_Card


Good
Standard
Bad
Credit Score Probability
0
5
10
15
20
25
30
Interest_Rate

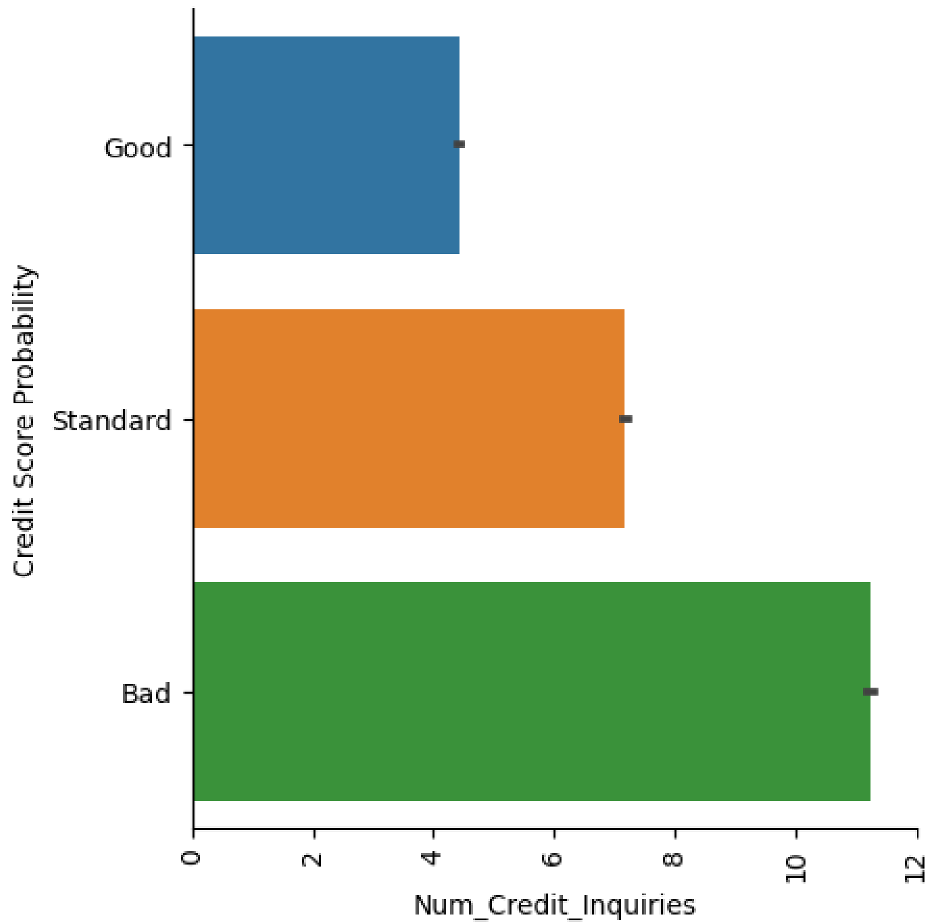

Good
Standard
Bad
Credit Score Probability
0
1
2
3
4
5
6
7
Num_of_Loan

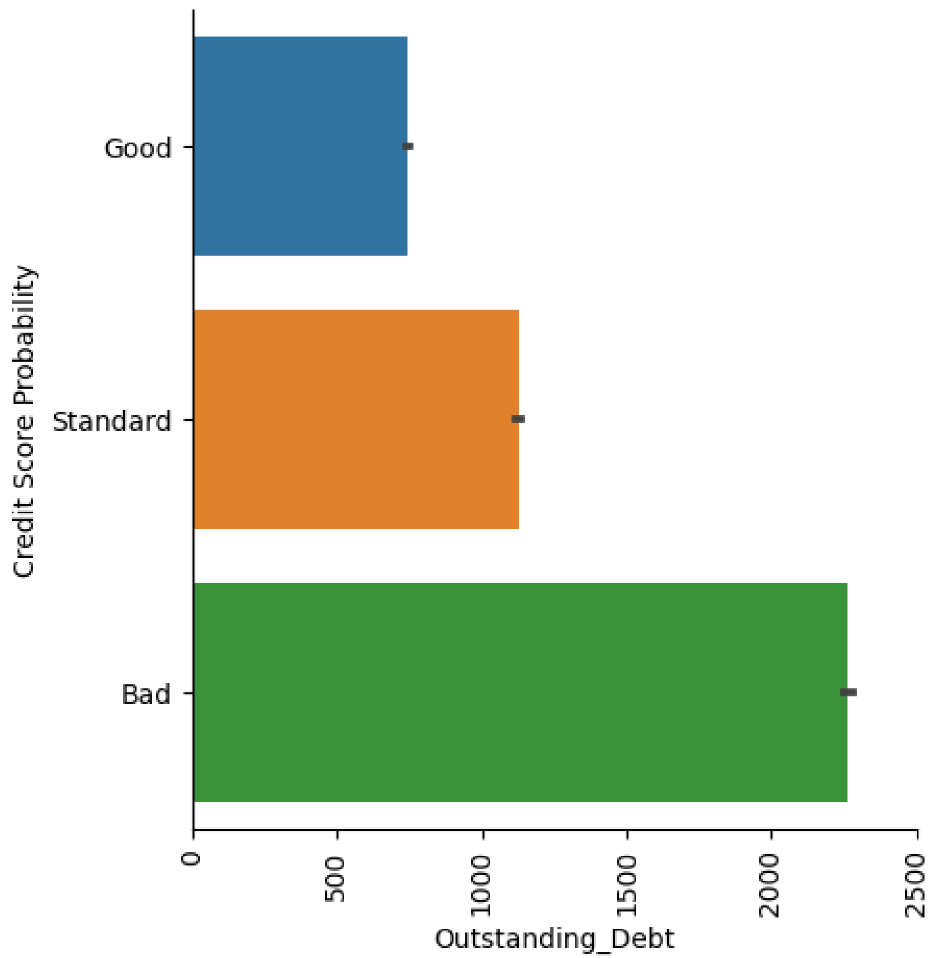

Good
Standard
Bad
Credit Score Probability
0
5
10
15
20
25
30
35
Delay_from_due_date

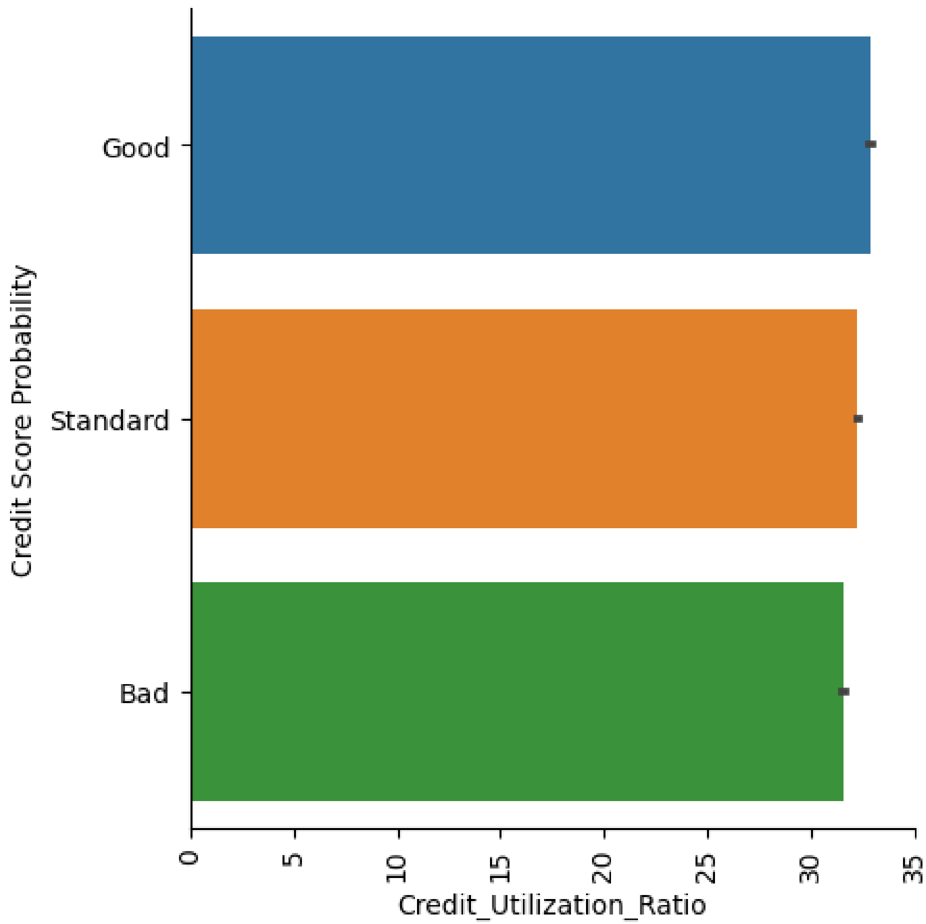

Credit Score Probability
Good
Standard
Bad
0
5
10
15
20
25
Num_of_Delayed_Payment

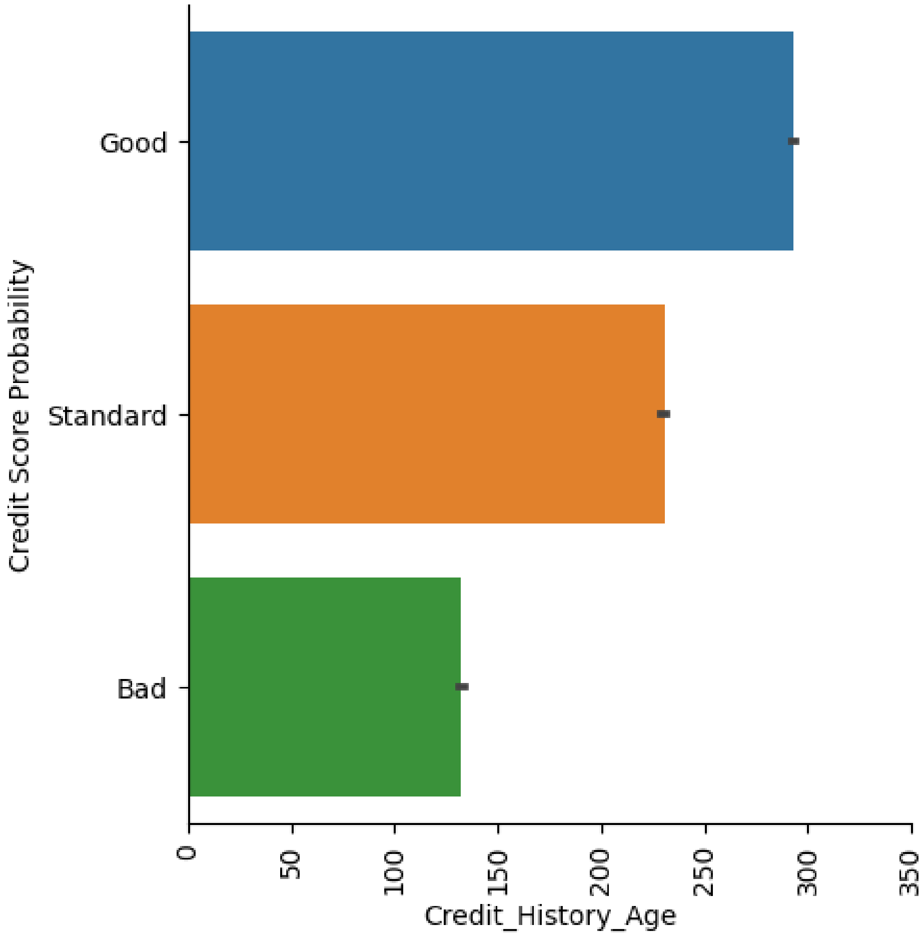


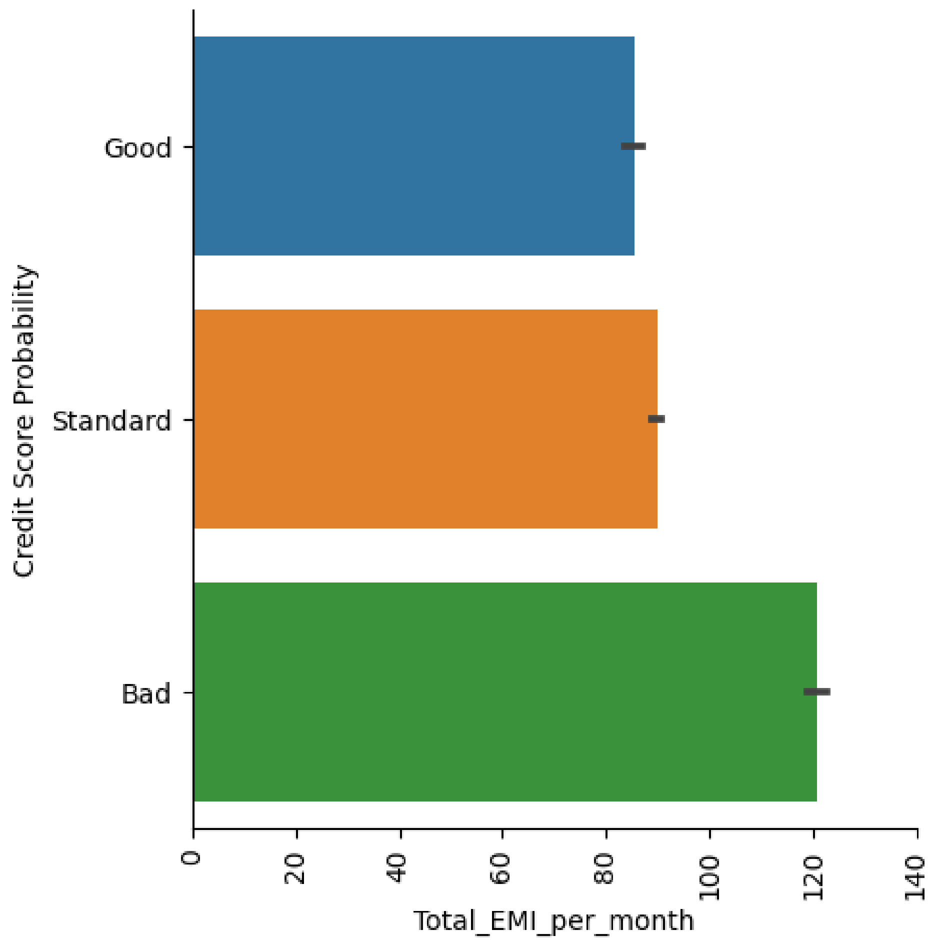
Credit Score Probability
Good
Standard
Bad
0
2
4
6
8
10
12
14
Changed_Credit_Limit


Good
Standard
Bad
Credit Score Probability
0
2
4
6
8
10
12
Num_Credit_Inquiries

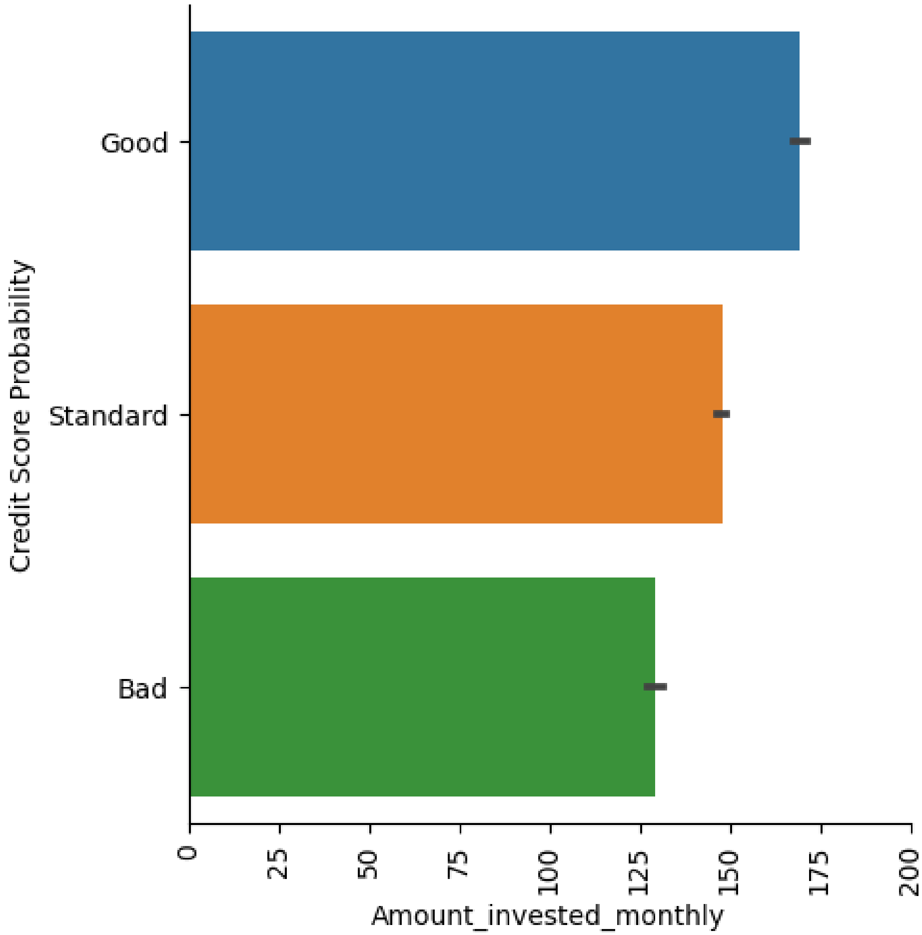

Good
Standard
Bad
Credit Score Probability
0
500
1000
1500
2000
2500
Outstanding_Debt


Credit Score Probability
Good
Standard
Bad
0
5
10
15
20
25
30
35
Credit_Utilization_Ratio

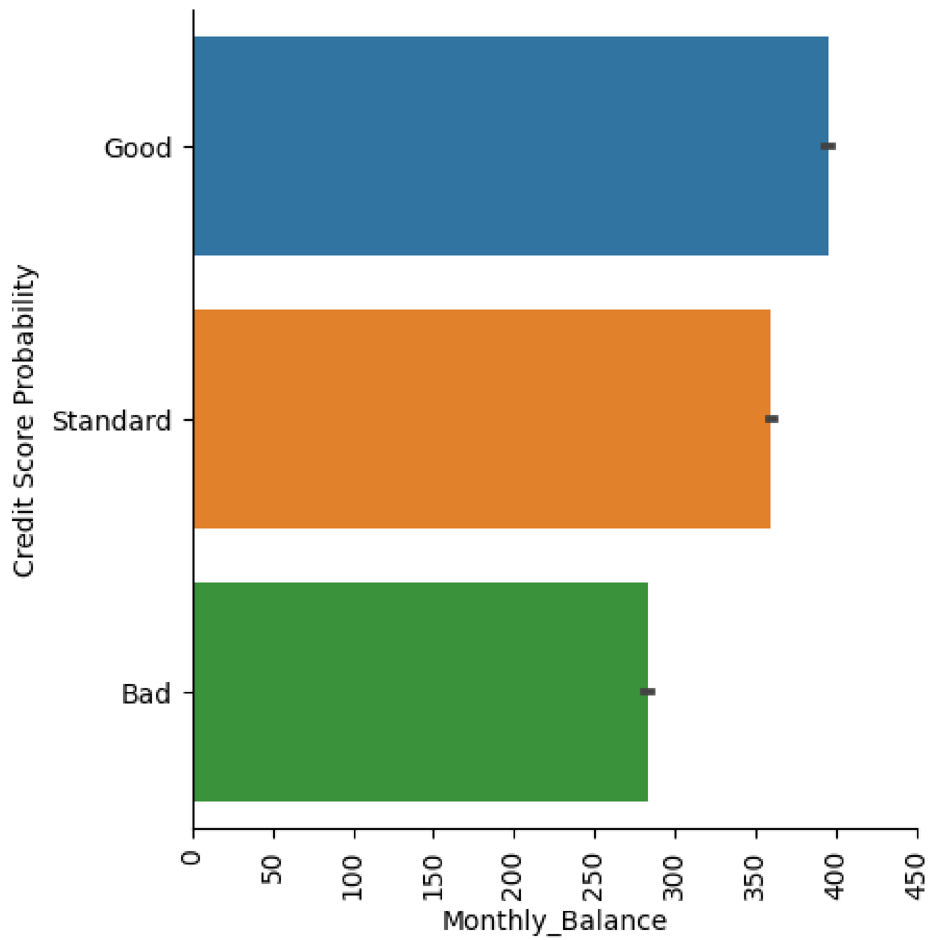

Credit Score Probability
Good
Standard
Bad
0
50
100
150
200
250
300
350
Credit_History_Age



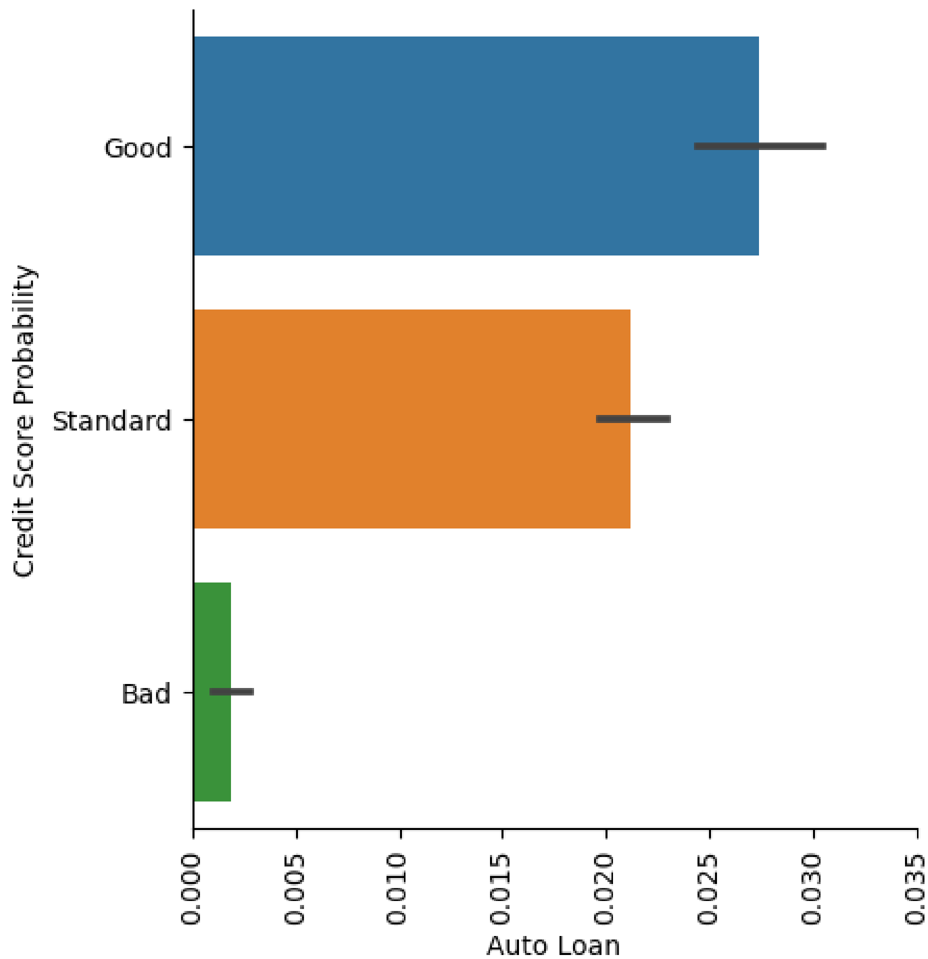
Good
Standard
Bad
Credit Score Probability
0
20
40
60
80
100
120
140
Total_EMI_per_month

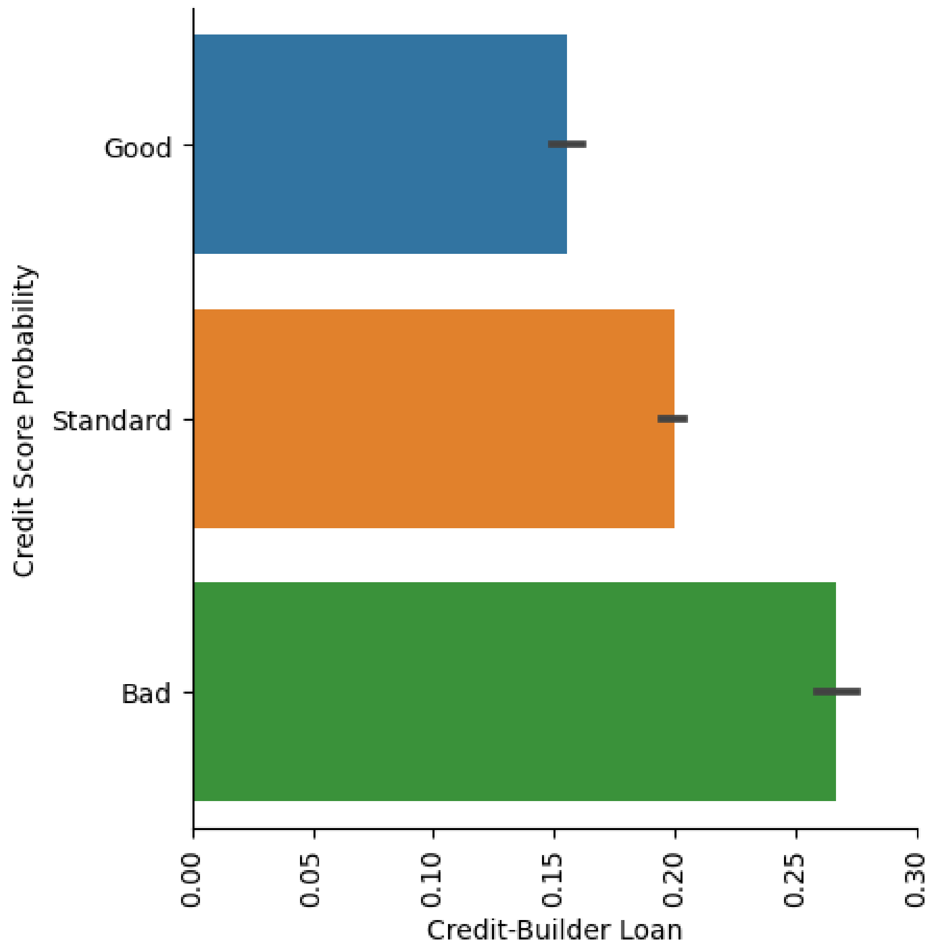


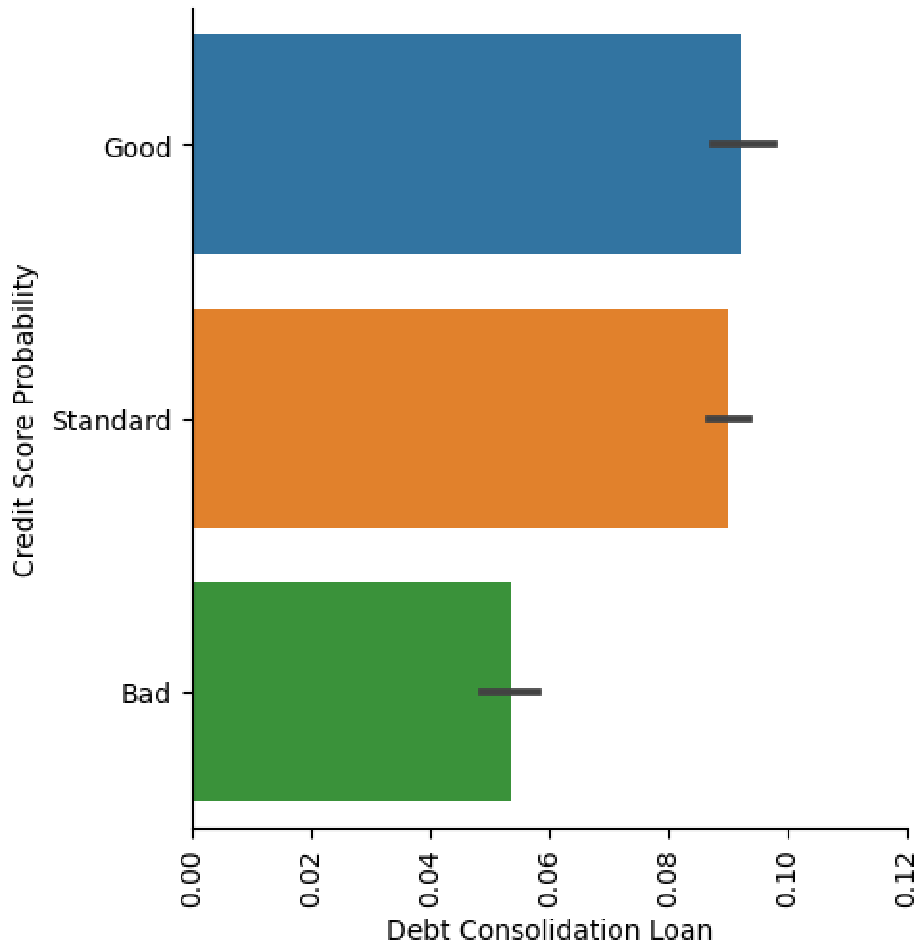
Good
Standard
Bad
Credit Score Probability
0
25
50
75
100
125
150
175
200
Amount_invested_monthly

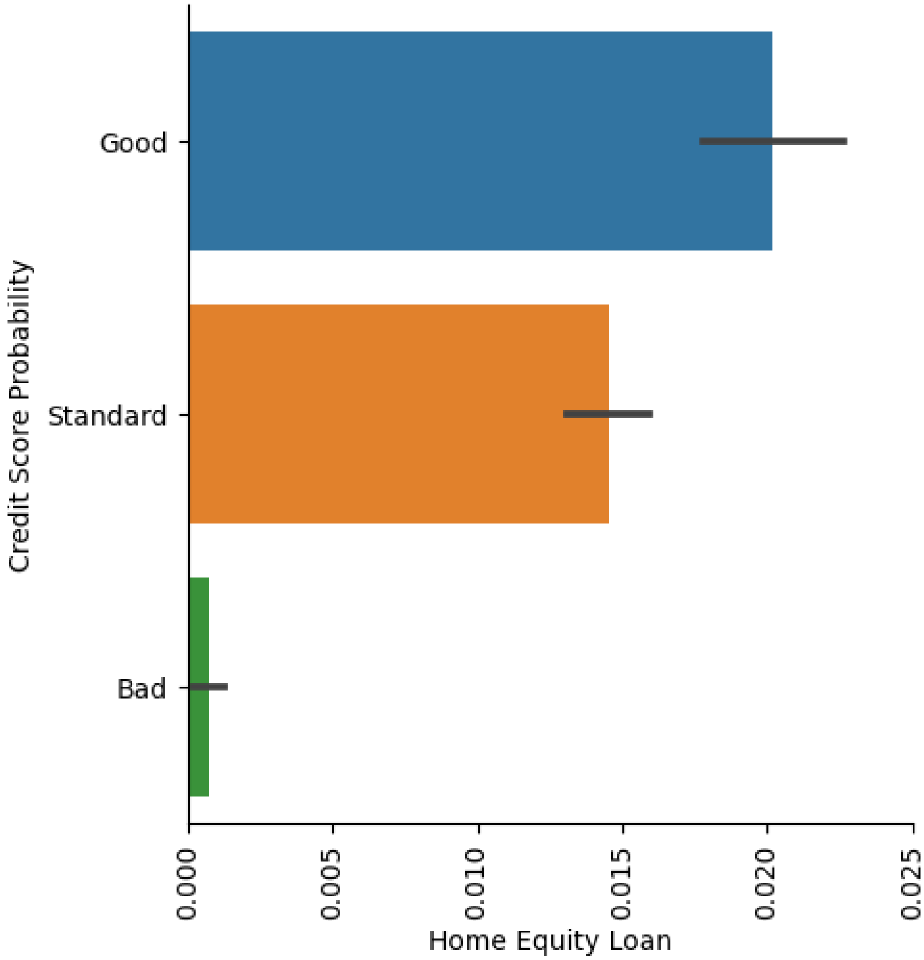


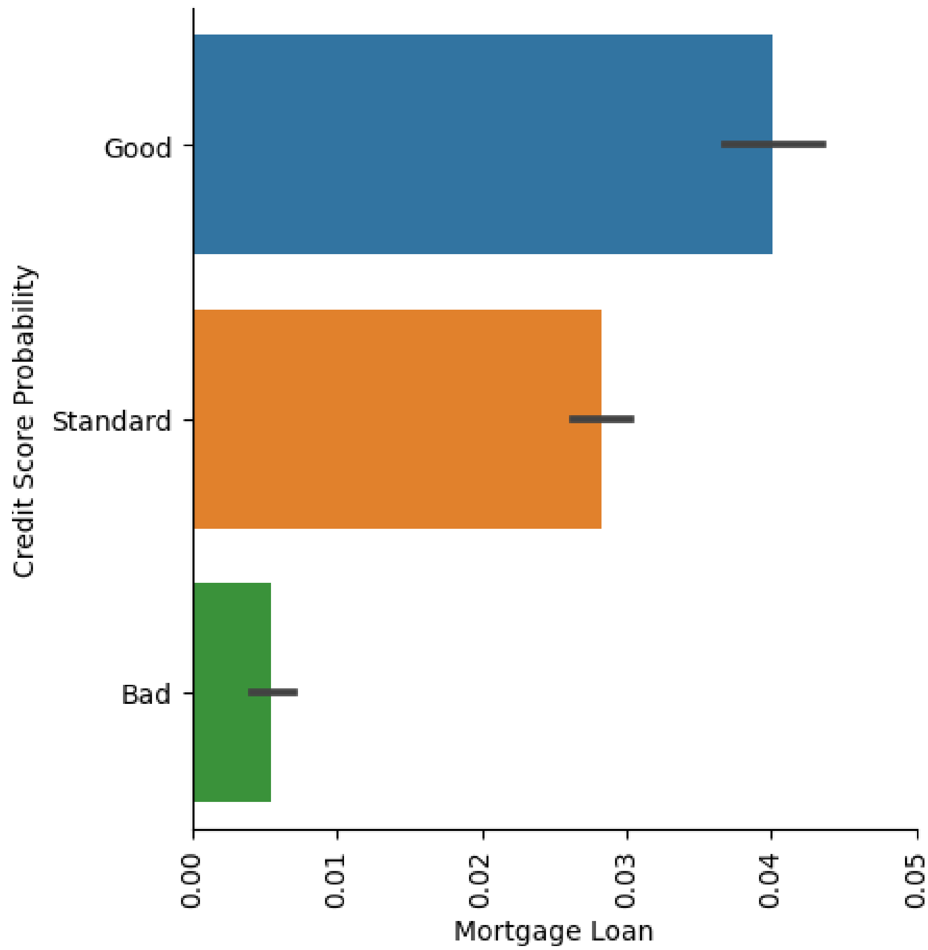
Good
Standard
Bad
Credit Score Probability
0
50
100
150
200
250
300
350
400
450
Monthly_Balance

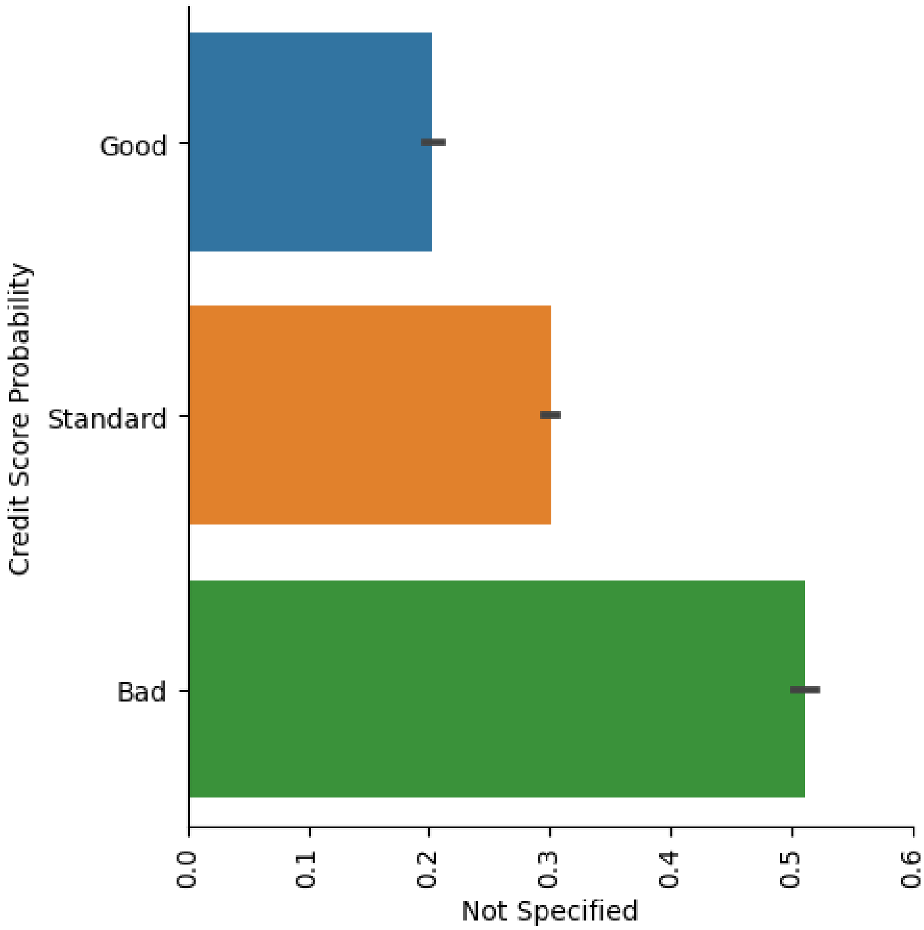

Credit Score Probability
Good
Standard
Bad
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
Auto Loan


Good
Standard
Bad
Credit Score Probability
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Credit-Builder Loan

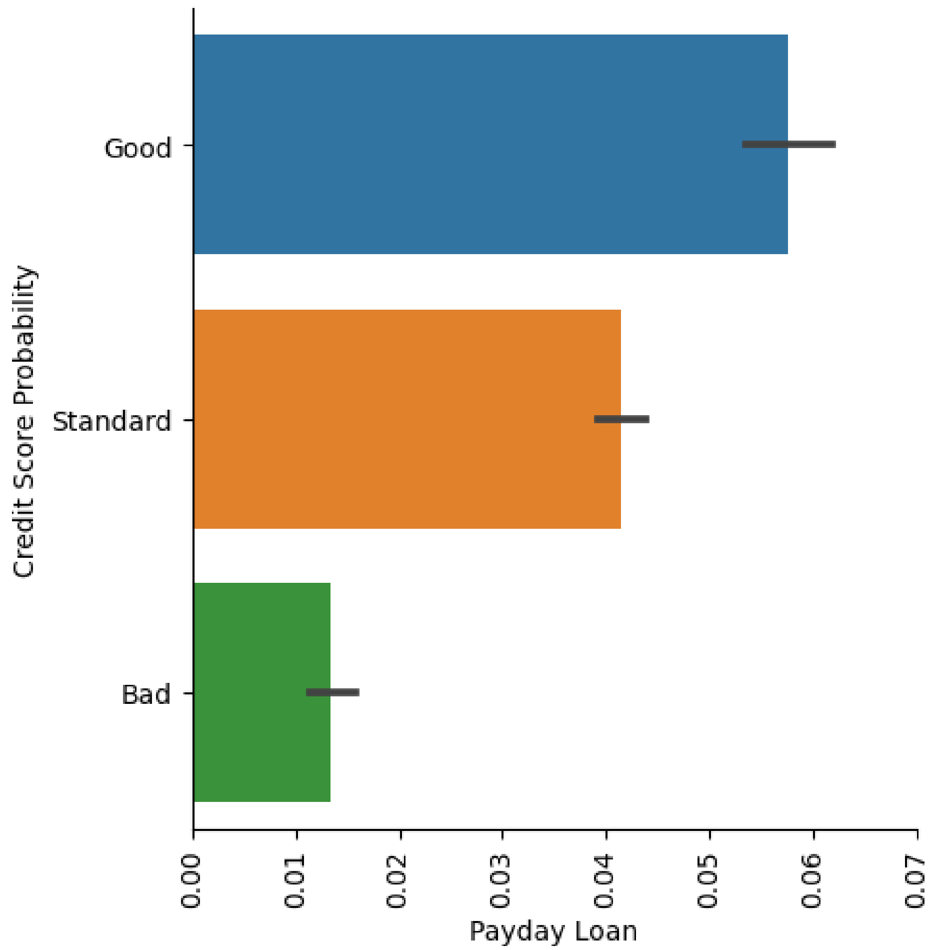

Good
Standard
Bad
Credit Score Probability
0.00
0.02
0.04
0.06
0.08
0.10
0.12
Debt Consolidation Loan

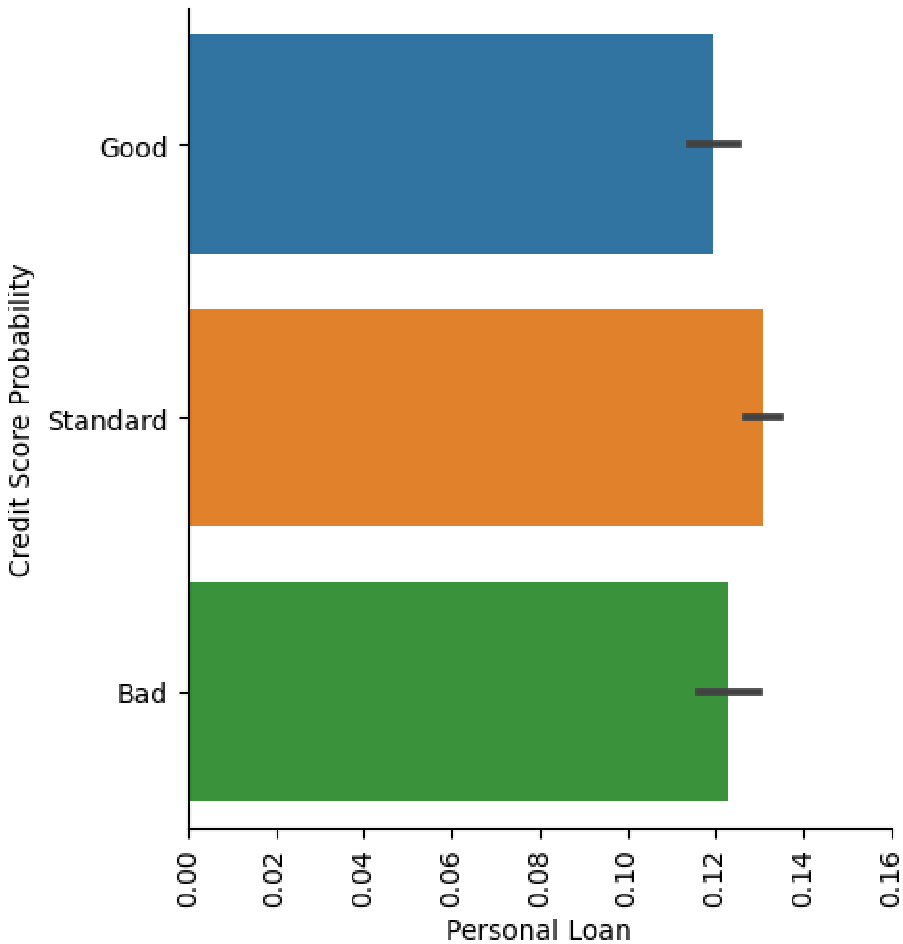

Good
Standard
Bad
Credit Score Probability
0.000
0.005
0.010
0.015
0.020
0.025
Home Equity Loan


Good
Standard
Bad
Credit Score Probability
0.00
0.01
0.02
0.03
0.04
0.05
Mortgage Loan

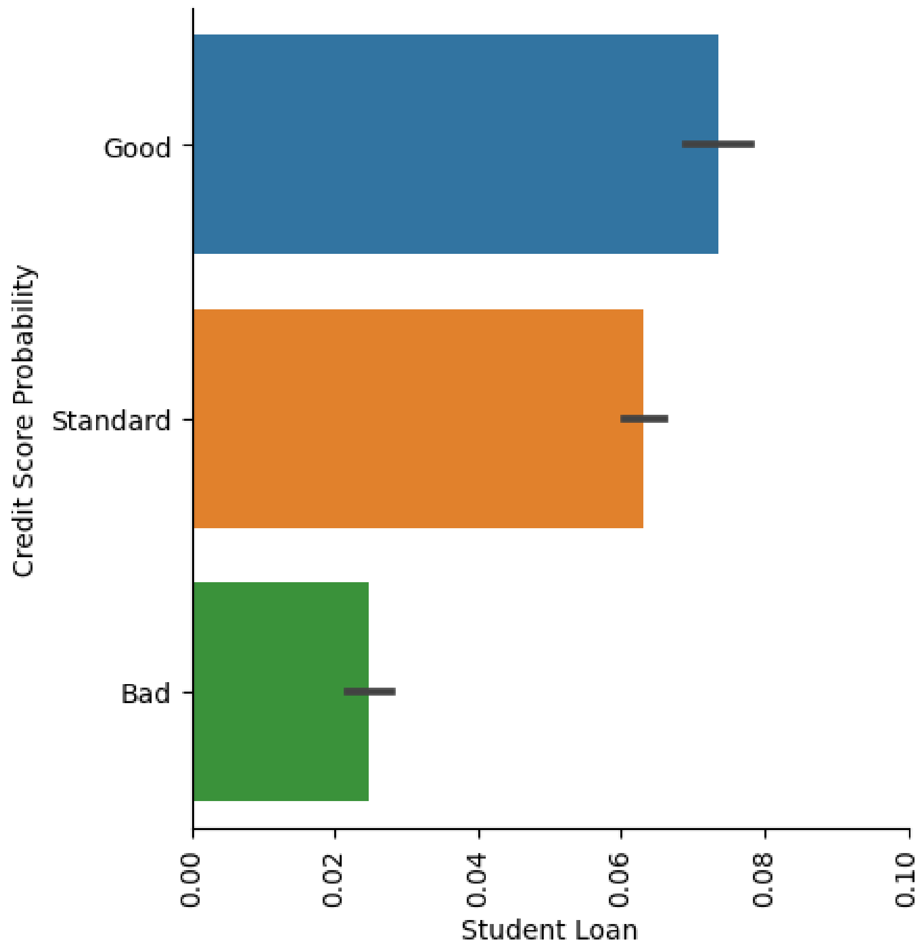


Good
Standard
Bad
Credit Score Probability
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Not Specified

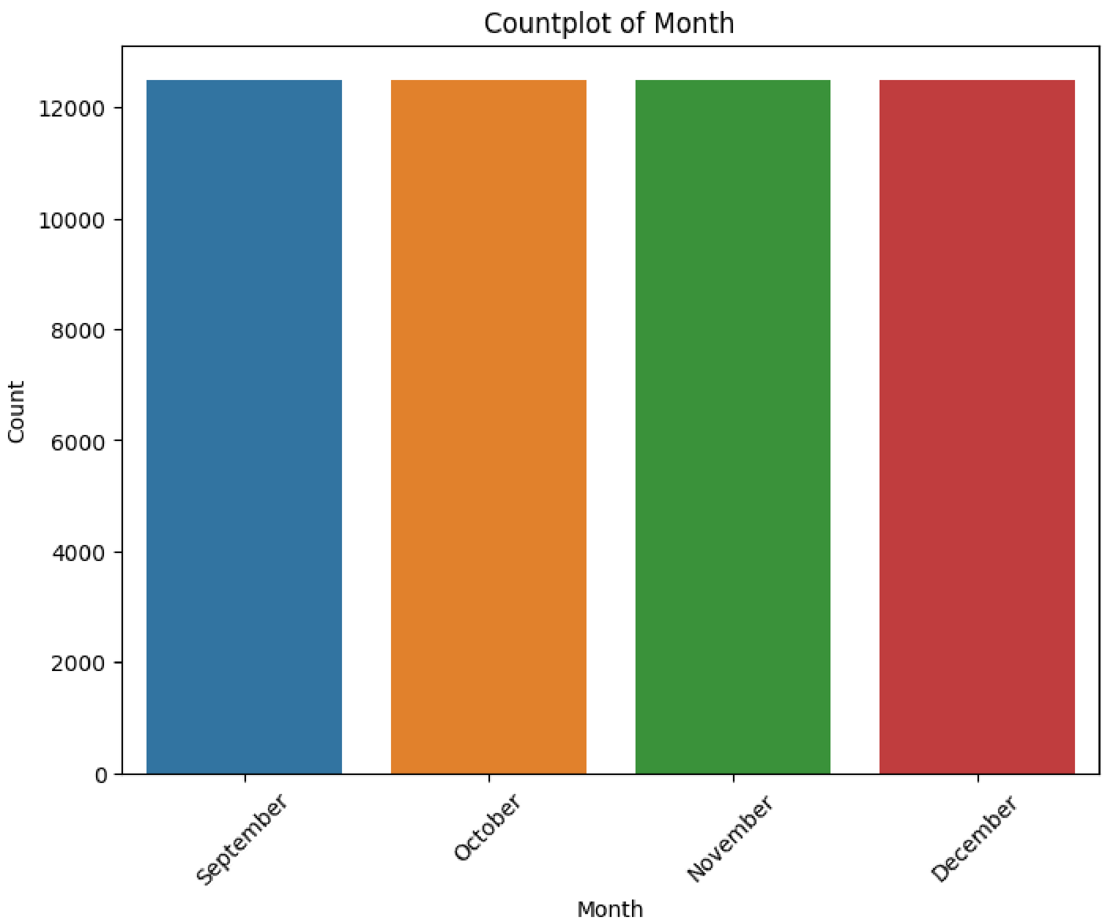

Good
Standard
Bad
Credit Score Probability
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
Payday Loan

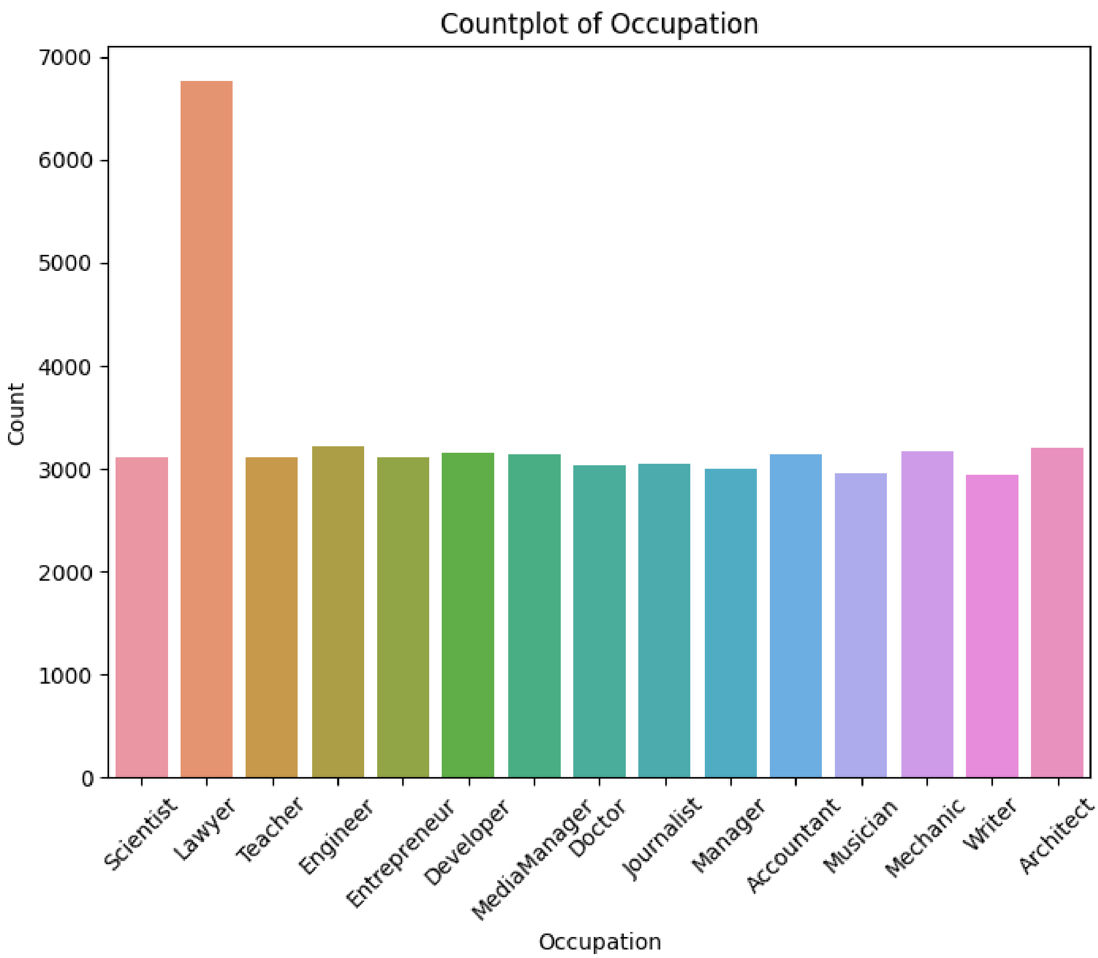


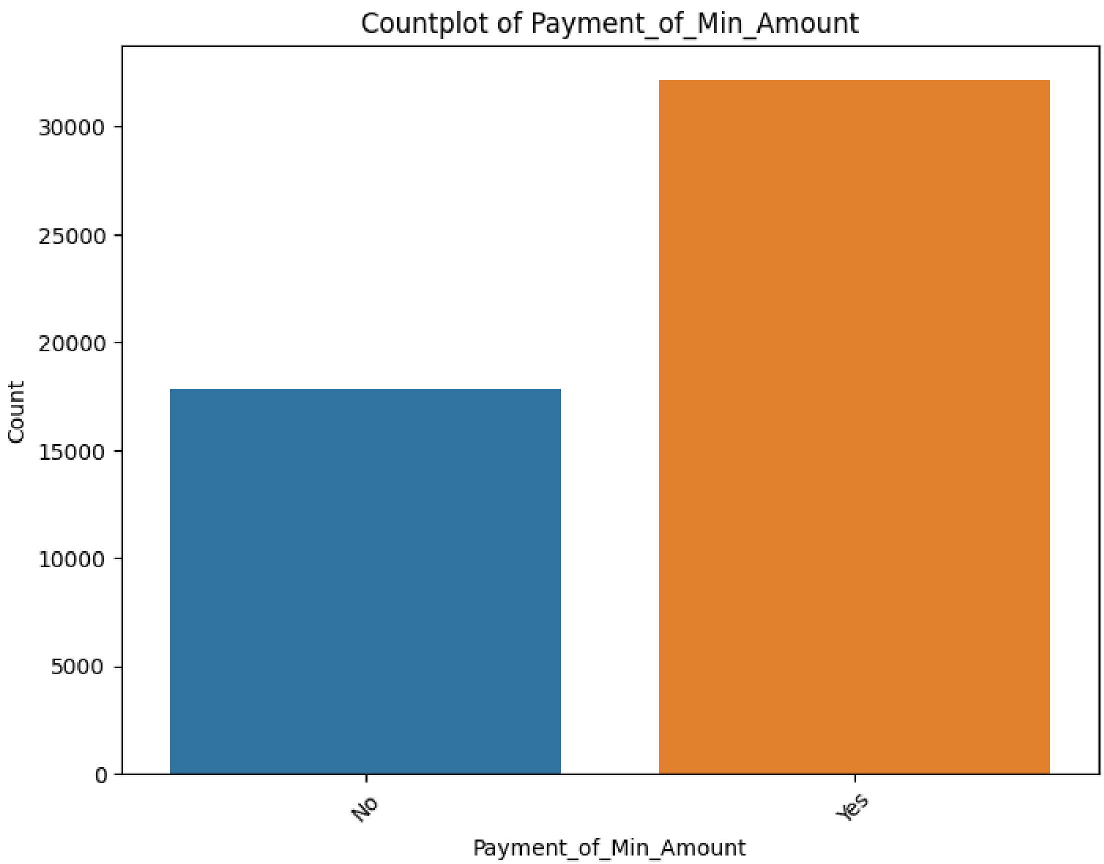
Good
Standard
Bad
Credit Score Probability
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Personal Loan

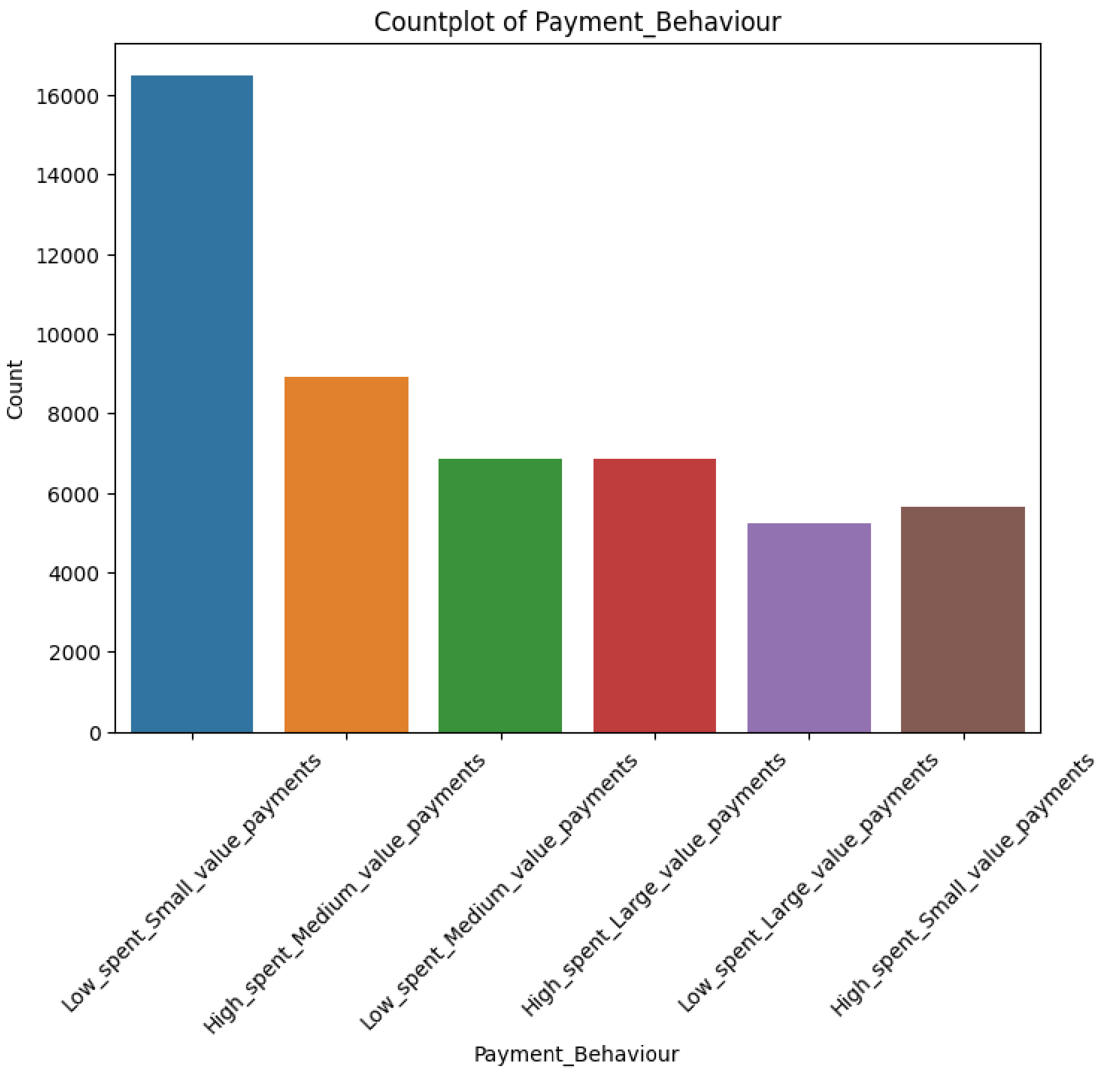


[71]:
```
for col in x.select_dtypes(['object']):
    plt.figure(figsize=(8, 6))
    sns.countplot(x=col, data=df)
    plt.title(f"Countplot of {col}")
    plt.xlabel(col)
    plt.ylabel("Count")
    plt.xticks(rotation=45)
    plt.show()
```

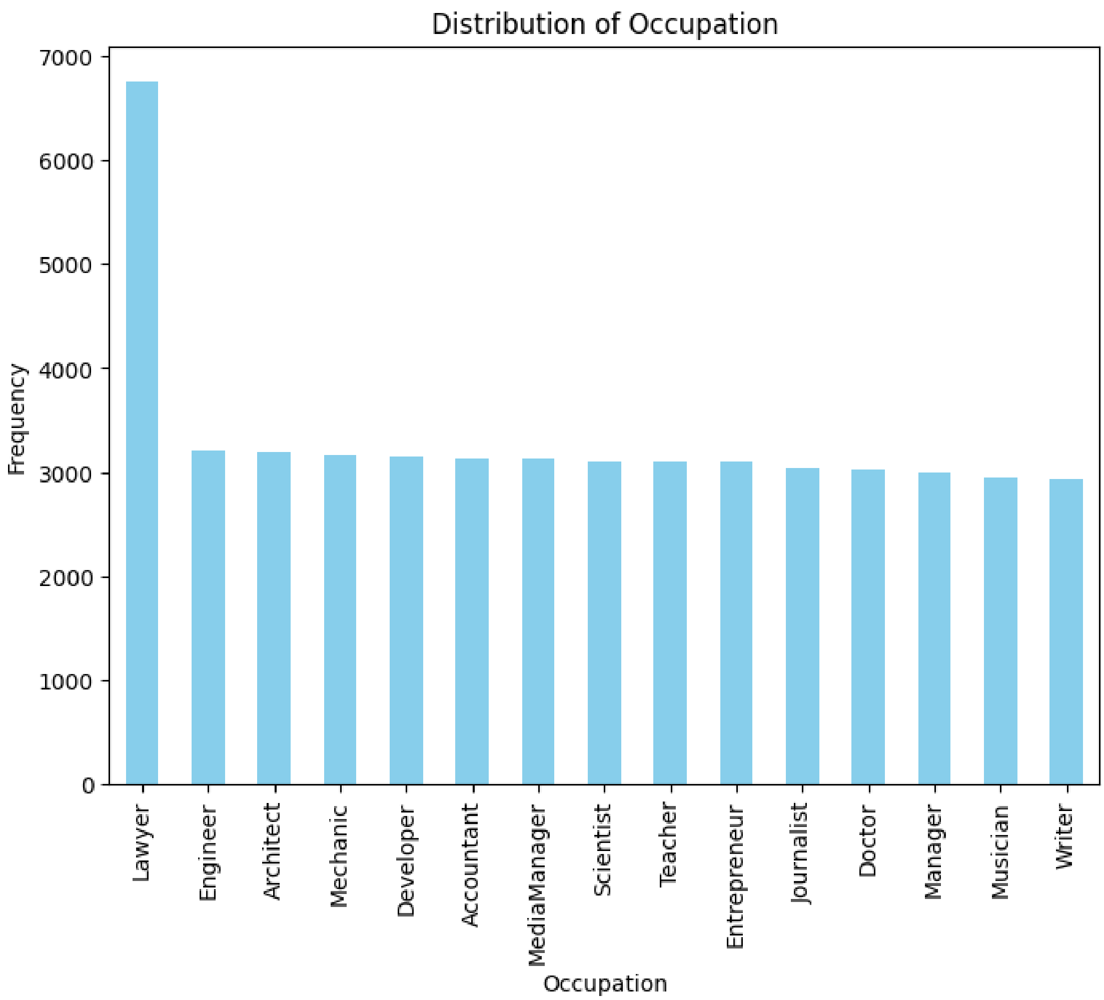

Countplot of Month
Count
12000
10000
8000
6000
4000
2000
0
September
October
November
December
Month


Countplot of Occupation
Count
7000
6000
5000
4000
3000
2000
1000
0
Scientist
Lawyer
Teacher
Engineer
Entrepreneur
Developer
MediaManager
Doctor
Journalist
Manager
Accountant
Musician
Mechanic
Writer
Architect
Occupation

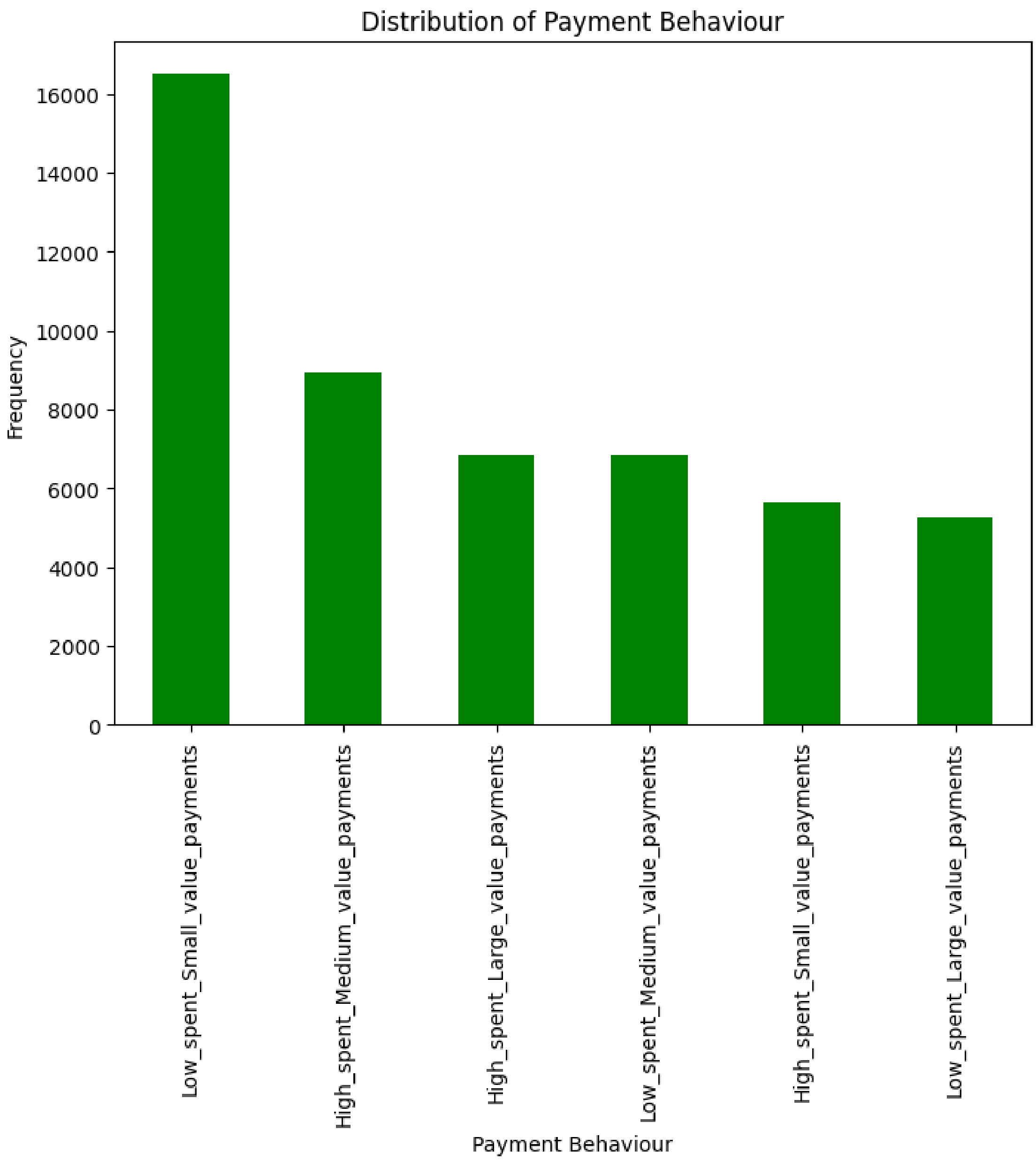

Countplot of Payment_of_Min_Amount
30000
25000
20000
15000
10000
5000
0
Count
No
Yes
Payment_of_Min_Amount



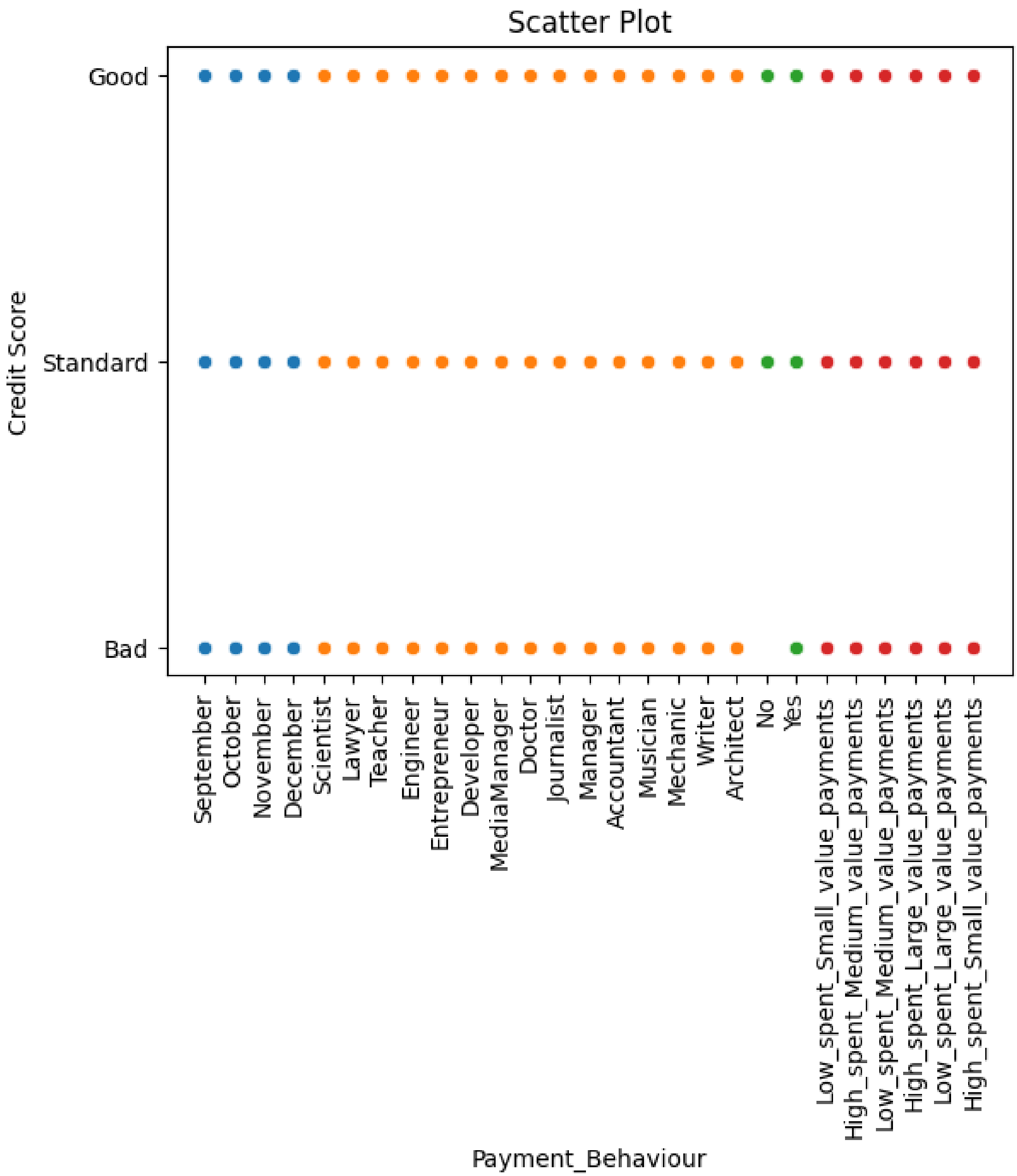
[72]:
```
plt.figure(figsize=(8, 6))
df['Occupation'].value_counts().plot(kind='bar', color='skyblue')
plt.xlabel('Occupation')
plt.ylabel('Frequency')
plt.title('Distribution of Occupation')
plt.show()
```



[73]:
```
plt.figure(figsize=(8, 6))
df['Payment_Behaviour'].value_counts().plot(kind='bar', color='green')
plt.xlabel('Payment Behaviour')
plt.ylabel('Frequency')
plt.title('Distribution of Payment Behaviour')
plt.show()
```



[74]:
```
for col in x.select_dtypes(['object']):
    sns.scatterplot(data=df, x=col, y=df['Credit_Score'])
    plt.title('Scatter Plot')
    plt.xlabel(col)
    plt.ylabel('Credit Score')
    plt.xticks(rotation=90)
```



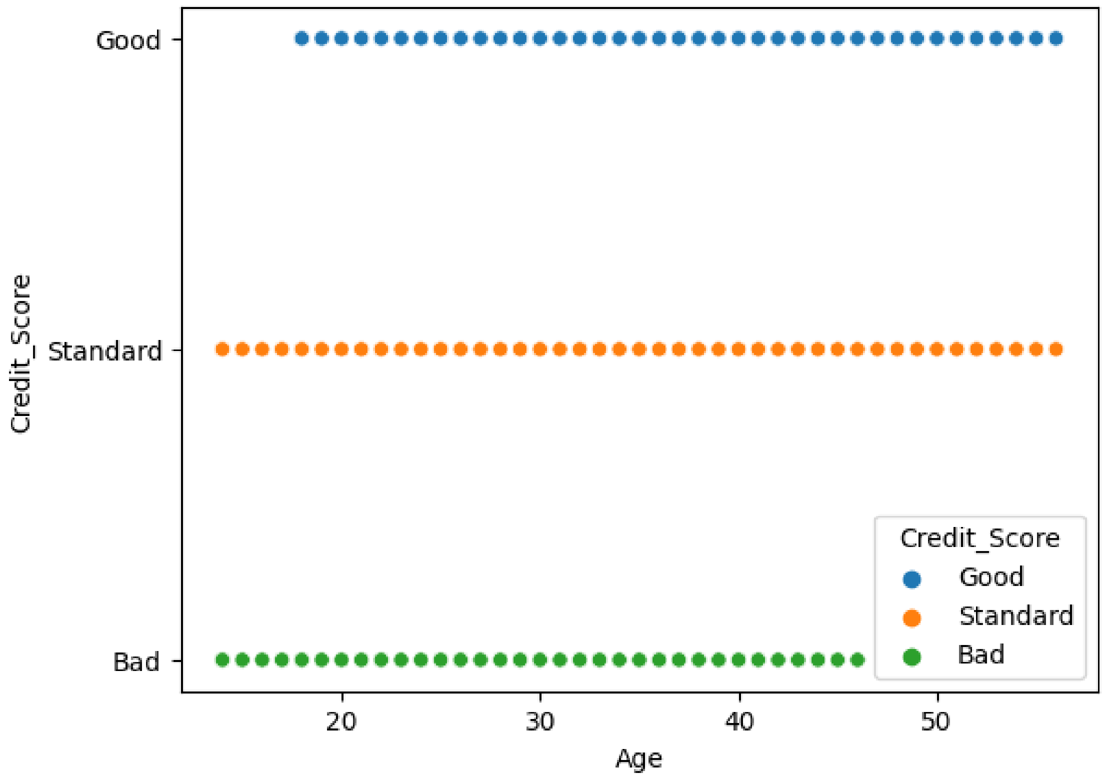
[75]:
```
df.columns
```

[75]:
```
Index(['Customer_ID', 'Month', 'Age', 'Occupation', 'Annual_Income',
       'Monthly_Inhand_Salary', 'Num_Bank_Accounts', 'Num_Credit_Card',
       'Interest_Rate', 'Num_of_Loan', 'Delay_from_due_date',
       'Num_of_Delayed_Payment', 'Changed_Credit_Limit',
       'Num_Credit_Inquiries', 'Credit_Score', 'Outstanding_Debt',
       'Credit_Utilization_Ratio', 'Credit_History_Age',
```

```
       'Payment_of_Min_Amount', 'Total_EMI_per_month',
       'Amount_invested_monthly', 'Payment_Behaviour', 'Monthly_Balance',
       'Auto Loan', 'Credit-Builder Loan', 'Debt Consolidation Loan',
       'Home Equity Loan', 'Mortgage Loan', 'Not Specified', 'Payday Loan',
       'Personal Loan', 'Student Loan'],
      dtype='object')
```

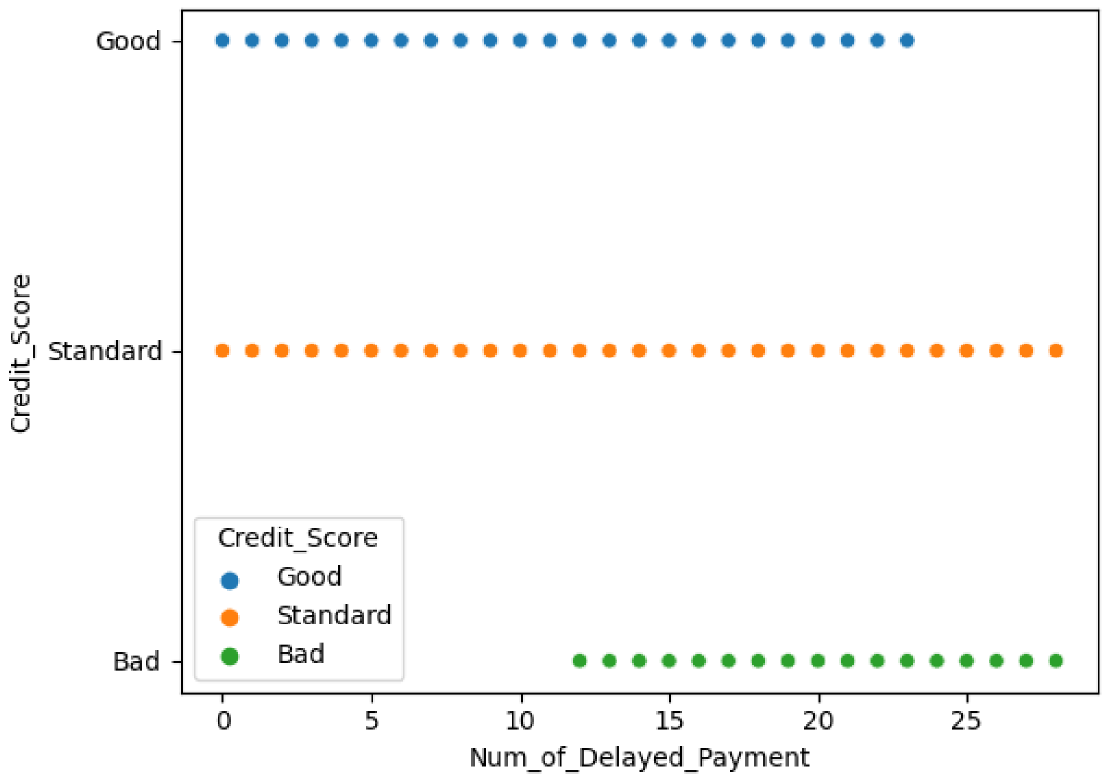
[76]:
```
sns.scatterplot( x = df['Age'],
                 y = df["Credit_Score"],
                 data = df, hue="Credit_Score")
```

[76]: <Axes: xlabel='Age', ylabel='Credit_Score'>



[77]:
```
sns.scatterplot( x = df['Num_of_Delayed_Payment'],
                 y = df["Credit_Score"],
                 data = df, hue="Credit_Score")
```

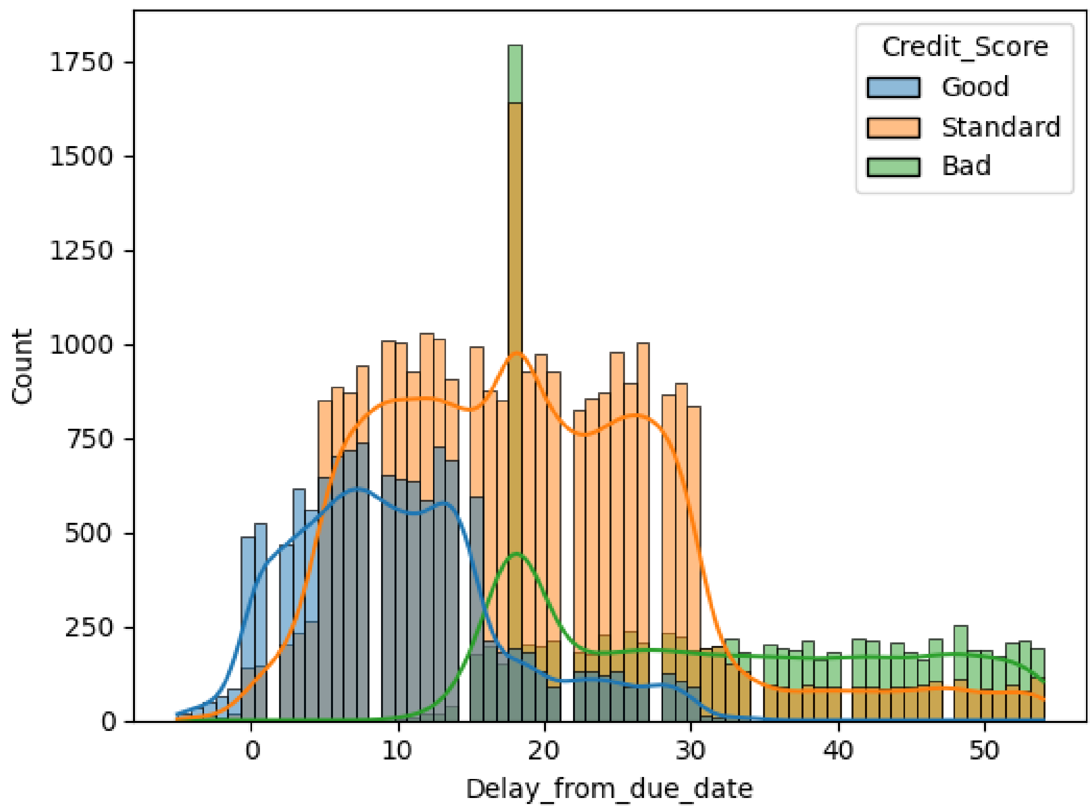
[77]: <Axes: xlabel='Num_of_Delayed_Payment', ylabel='Credit_Score'>



[78]:
```
sns.histplot(x = "Delay_from_due_date", data = df, kde=True, hue="Credit_Score")
```

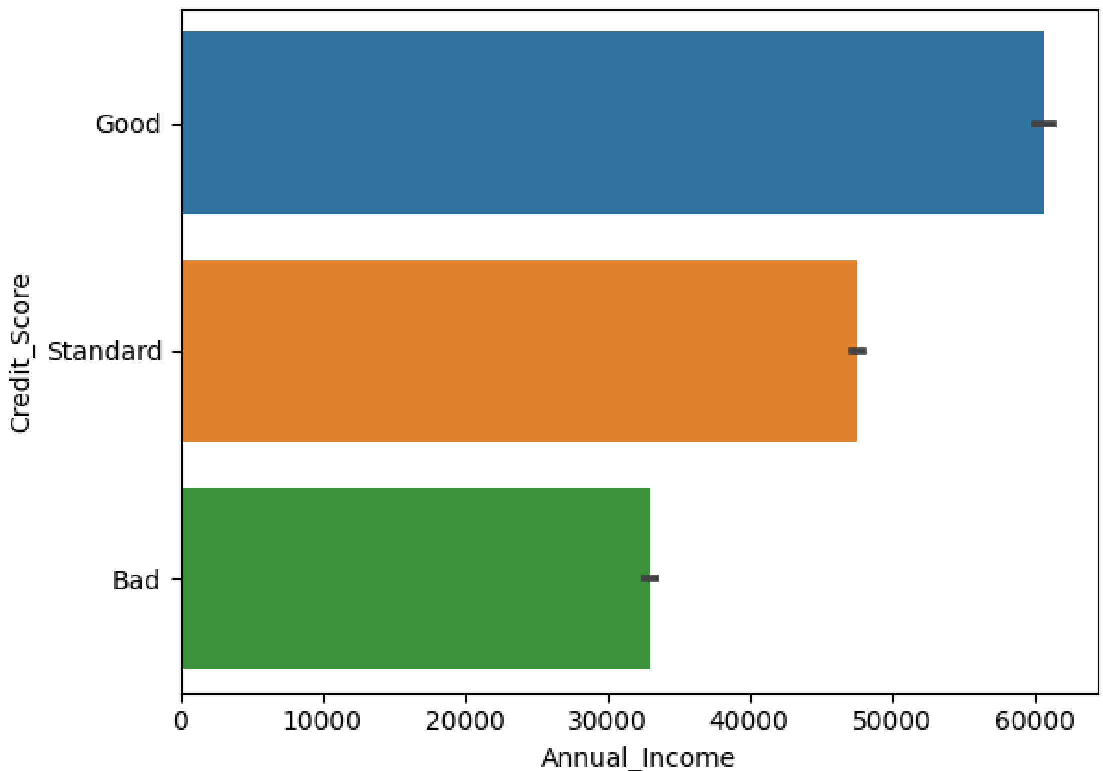
```
/home/applehx7/anaconda3/lib/python3.11/site-packages/seaborn/_oldcore.py:1119:
FutureWarning:

use_inf_as_na option is deprecated and will be removed in a future version.
Convert inf values to NaN before operating instead.
```

[78]:
```
<Axes: xlabel='Delay_from_due_date', ylabel='Count'>
```



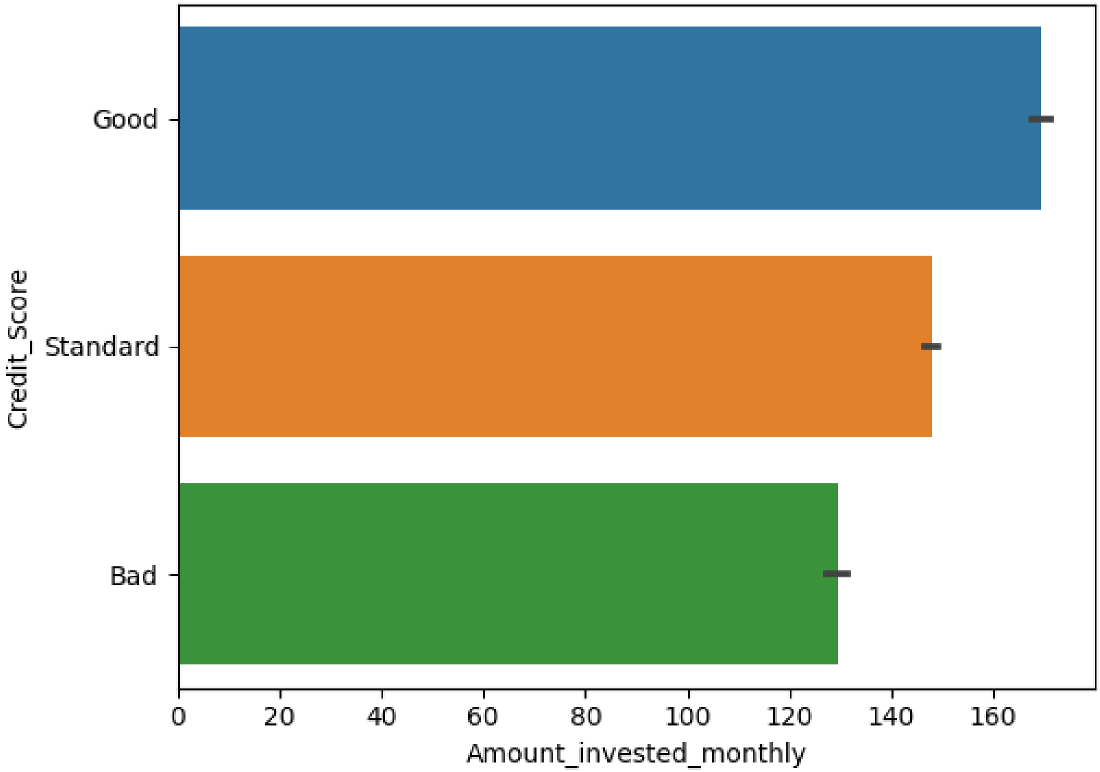
[79]:
```
sns.barplot(x = 'Annual_Income', y ="Credit_Score", data =df)
```

[79]: <Axes: xlabel='Annual_Income', ylabel='Credit_Score'>



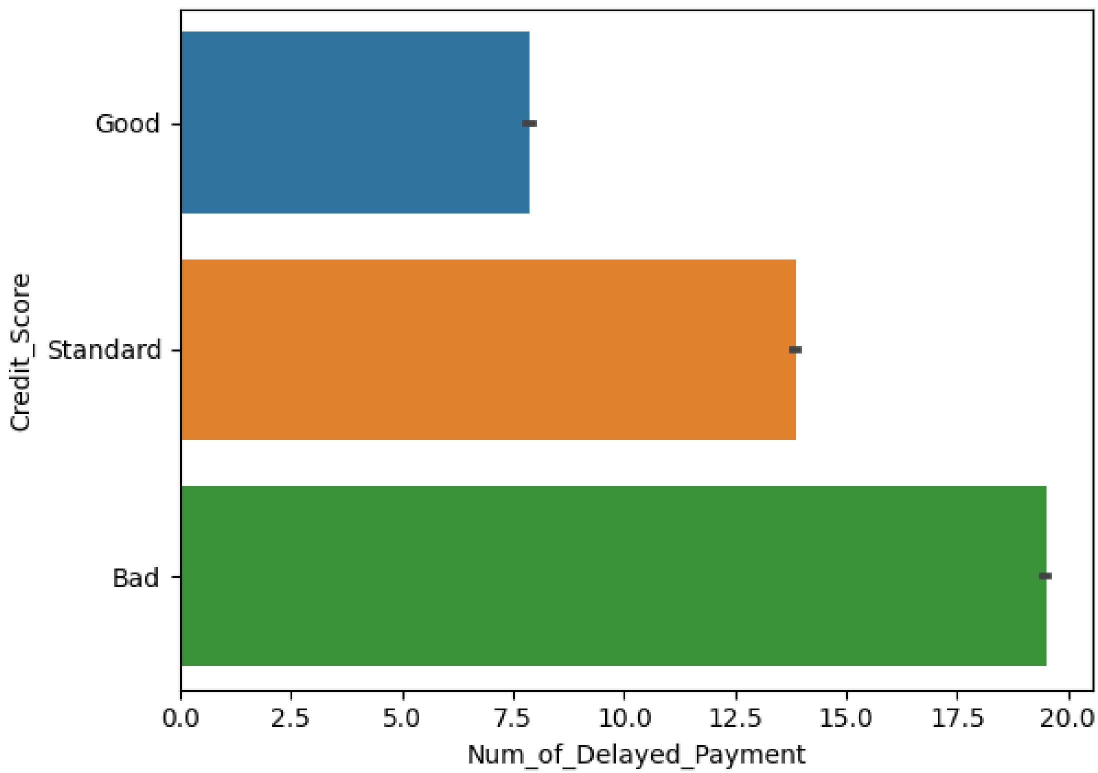
[80]:
```
sns.barplot(x = 'Amount_invested_monthly', y ="Credit_Score", data =df)
```

[80]: <Axes: xlabel='Amount_invested_monthly', ylabel='Credit_Score'>

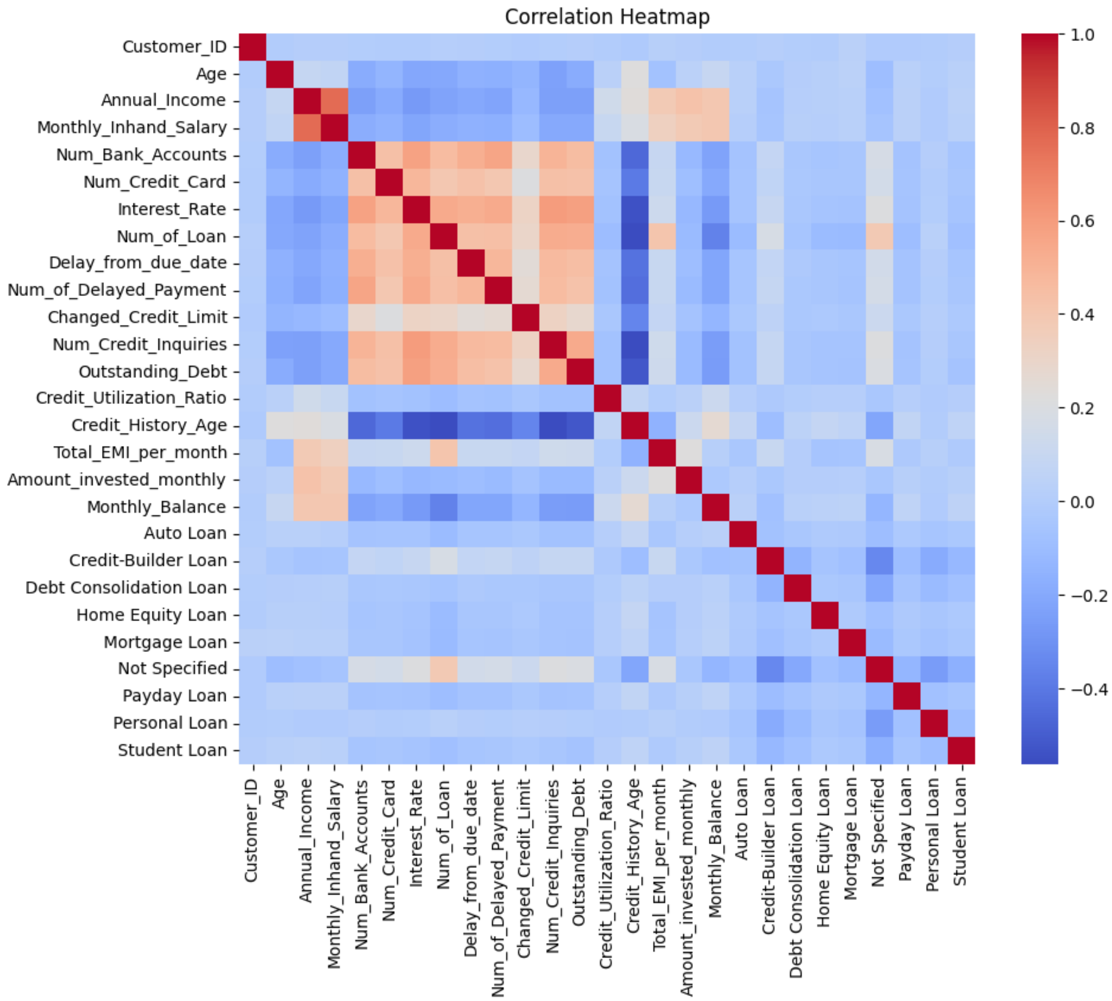


[81]:
```
sns.barplot(x = 'Num_of_Delayed_Payment', y ="Credit_Score", data =df)
```

[81]: <Axes: xlabel='Num_of_Delayed_Payment', ylabel='Credit_Score'>

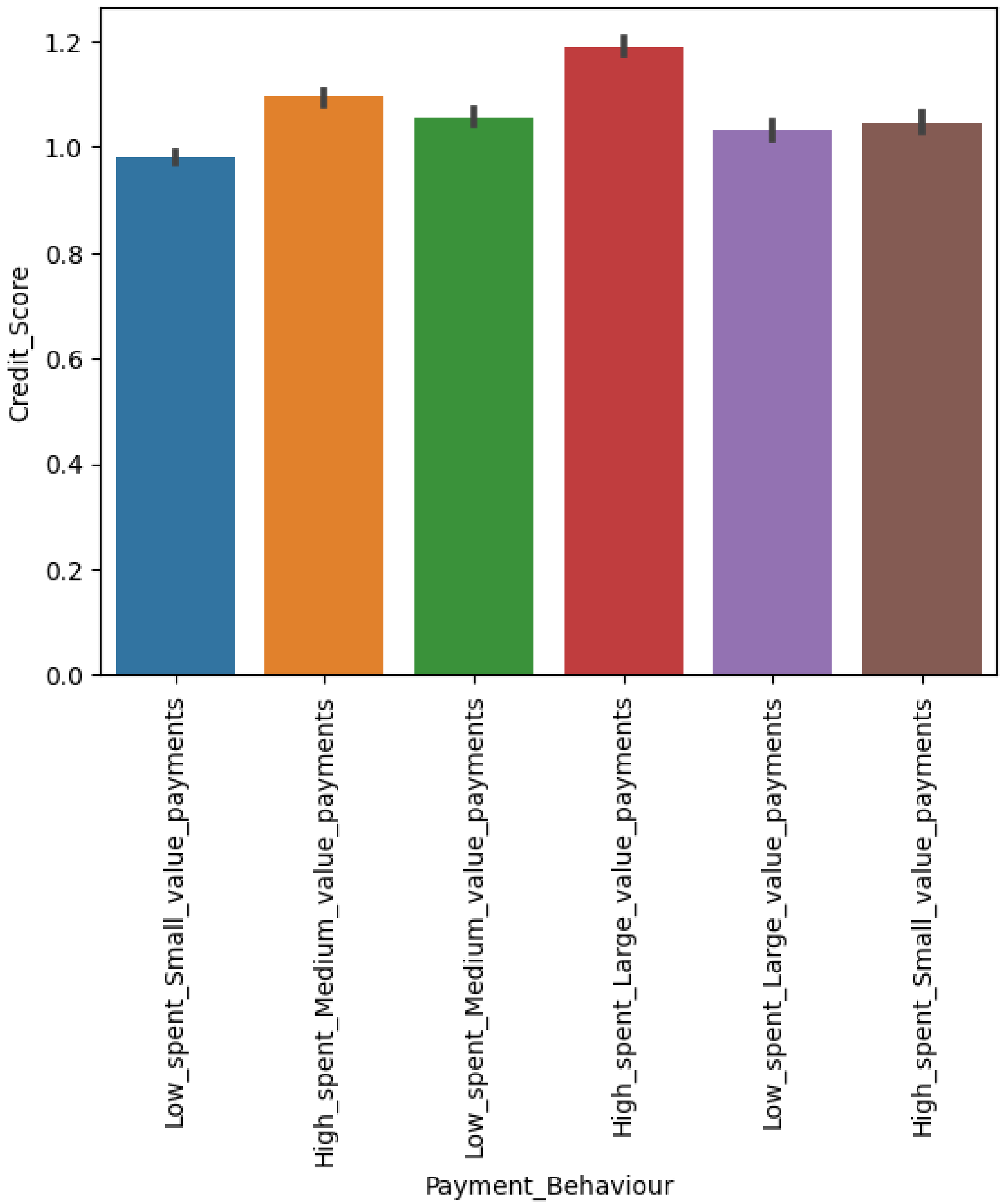


[82]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(df.select_dtypes(['int', 'float']).corr(), fmt='.2f',␣
 ↪cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

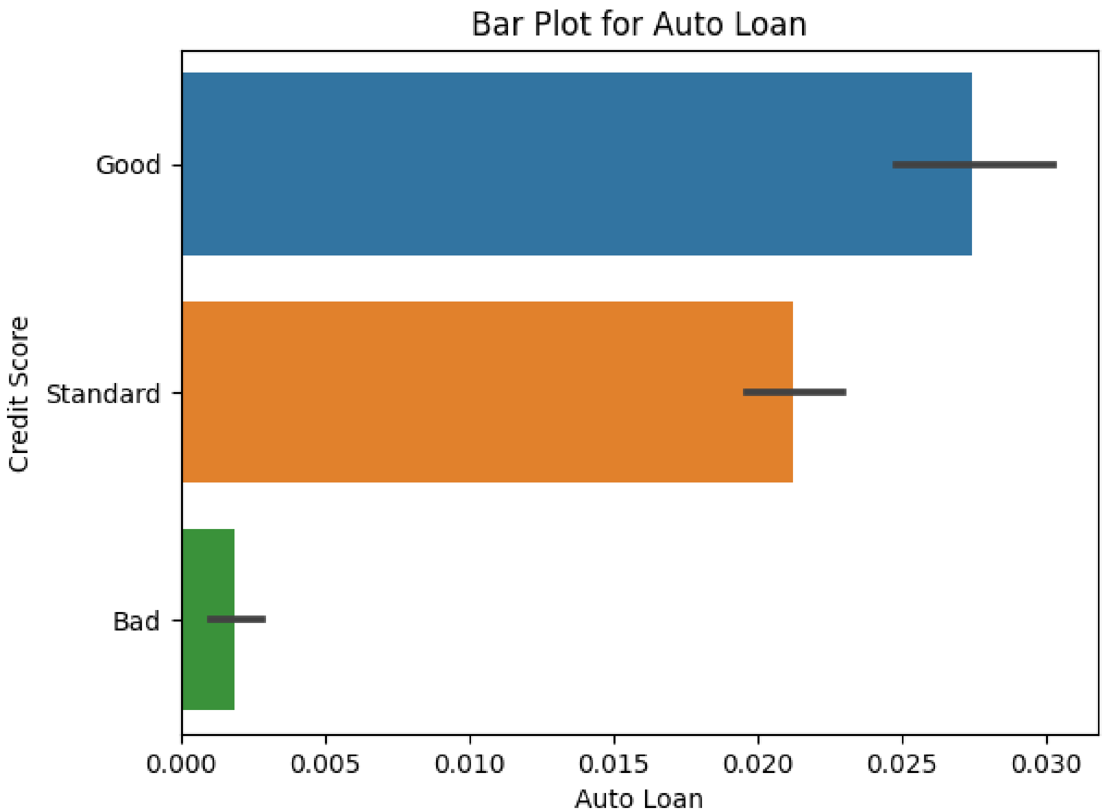


[83]:
```
sns.barplot(x = 'Payment_Behaviour', y =y, data =df)
plt.xticks(rotation=90)
```

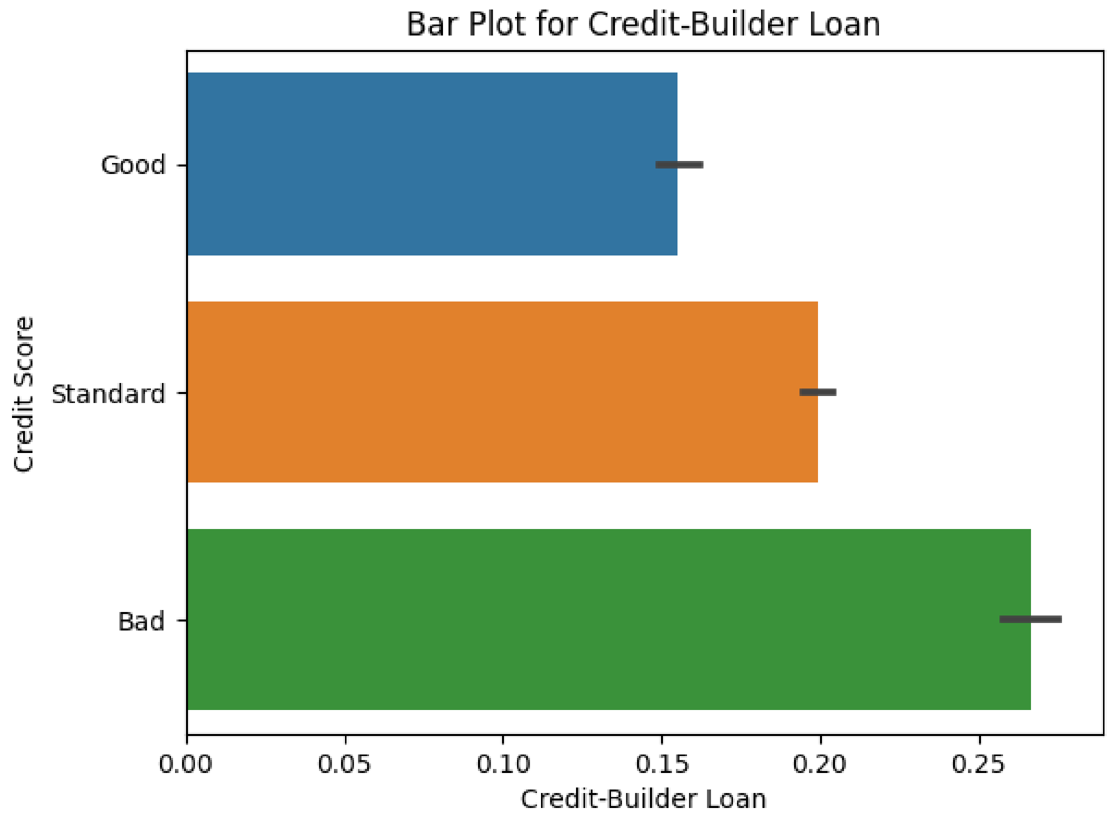
[83]:
```
(array([0, 1, 2, 3, 4, 5]),
 [Text(0, 0, 'Low_spent_Small_value_payments'),
  Text(1, 0, 'High_spent_Medium_value_payments'),
  Text(2, 0, 'Low_spent_Medium_value_payments'),
  Text(3, 0, 'High_spent_Large_value_payments'),
  Text(4, 0, 'Low_spent_Large_value_payments'),
  Text(5, 0, 'High_spent_Small_value_payments')])
```

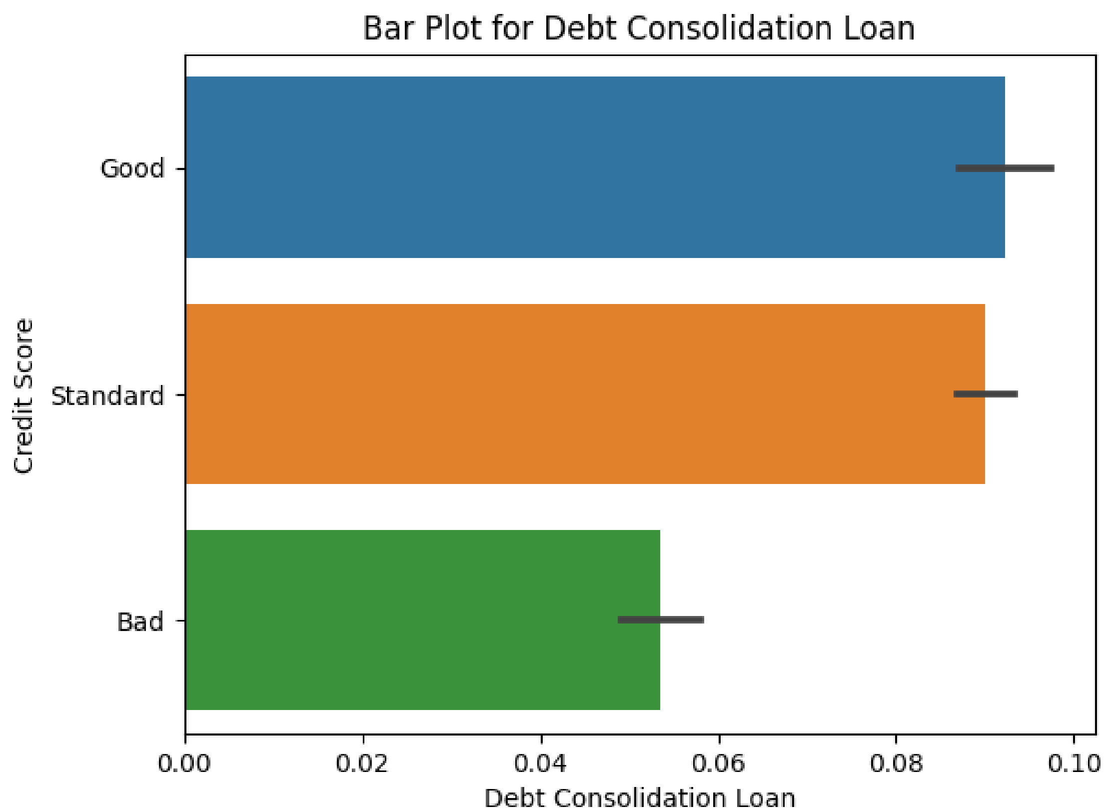


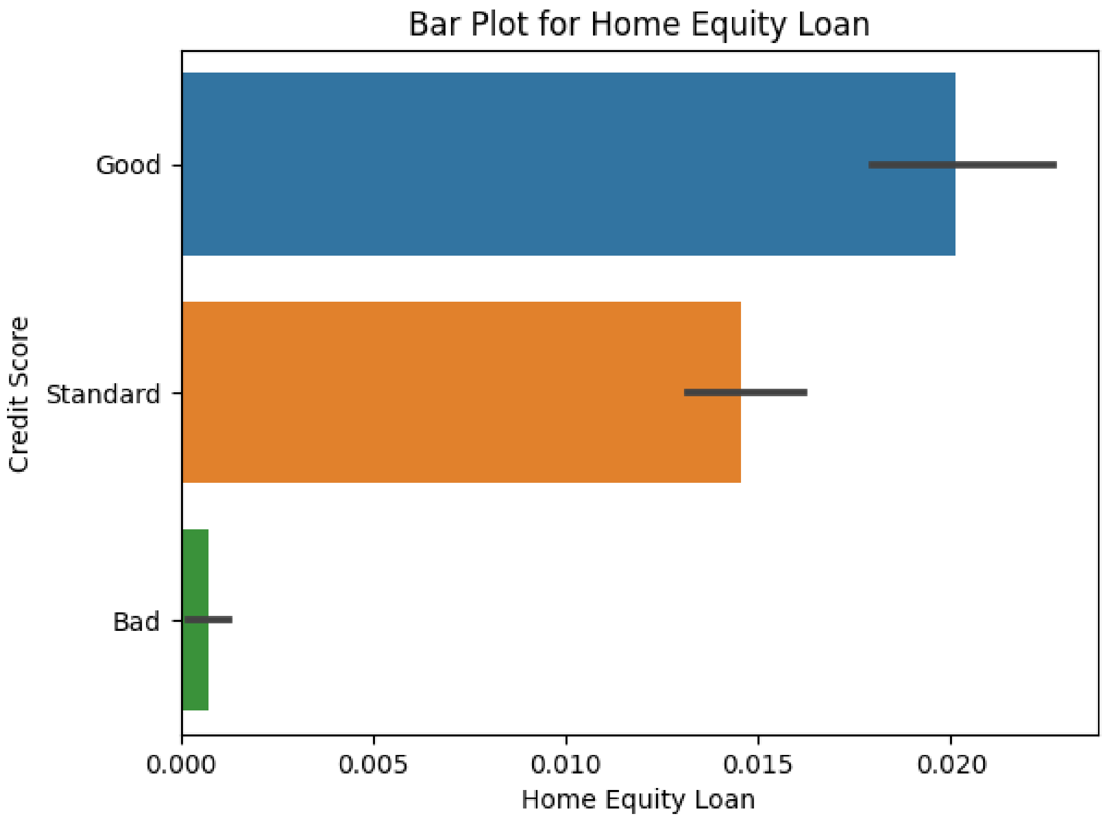
[84]:
```
for col in df.columns[-9:]:
    sns.barplot(x = col, y ="Credit_Score", data =df)
    plt.title(f'Bar Plot for {col}')
    plt.xlabel(col)
    plt.ylabel('Credit Score')
    plt.show()
```

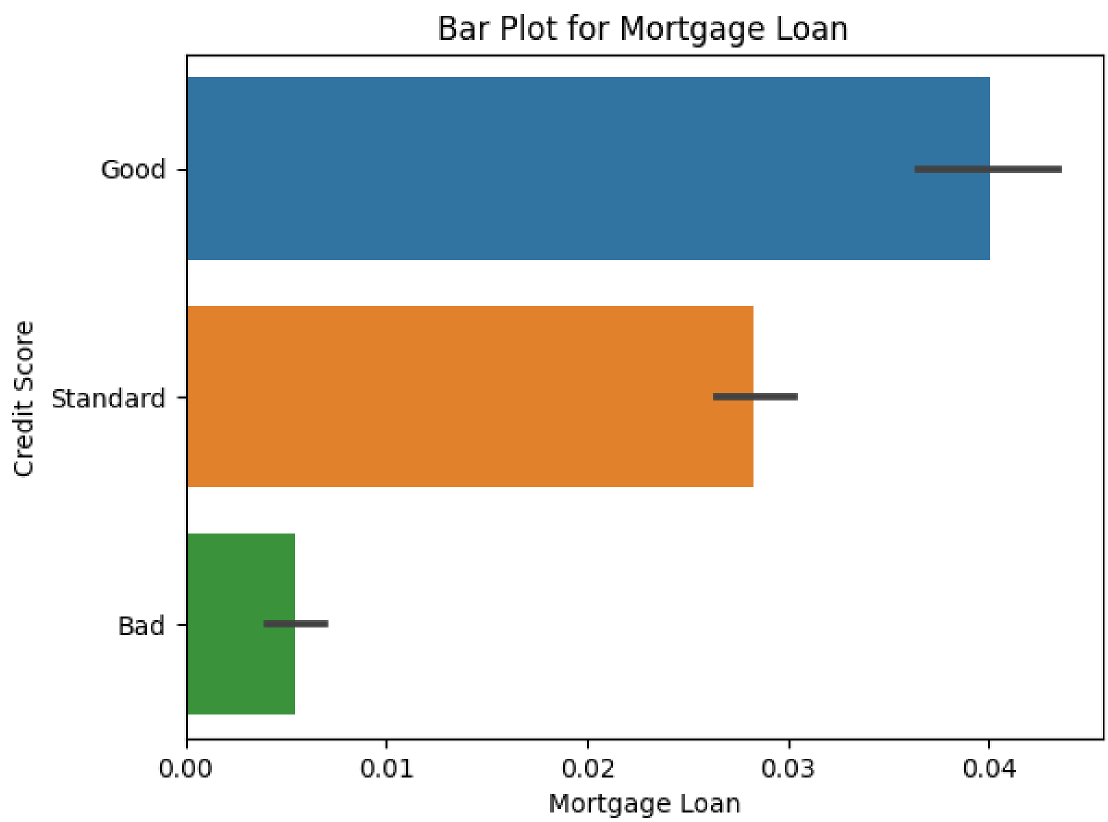


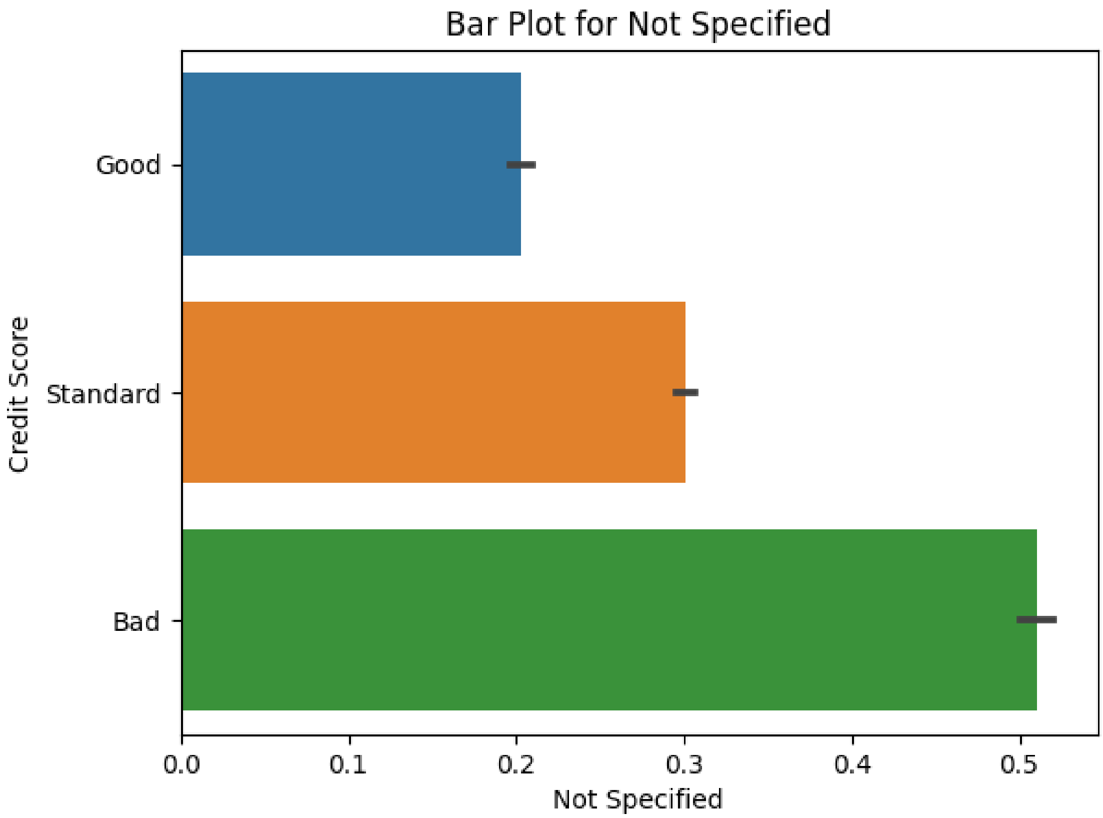
Bar Plot for Auto Loan
Good
Standard
Bad
Credit Score
0.000
0.005
0.010
0.015
0.020
0.025
0.030
Auto Loan


Bar Plot for Credit-Builder Loan
Good
Standard
Bad
Credit Score
0.00
0.05
0.10
0.15
0.20
0.25
Credit-Builder Loan

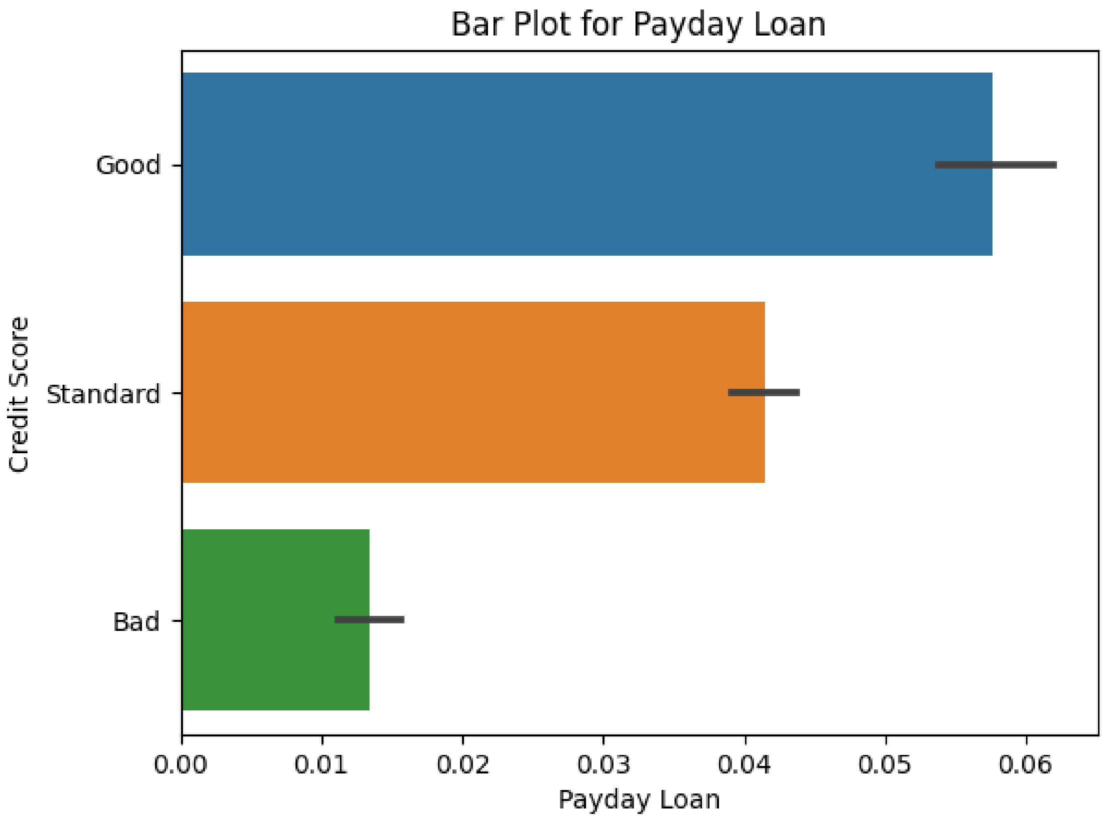

Bar Plot for Debt Consolidation Loan
Good
Standard
Bad
Credit Score
0.00
0.02
0.04
0.06
0.08
0.10
Debt Consolidation Loan

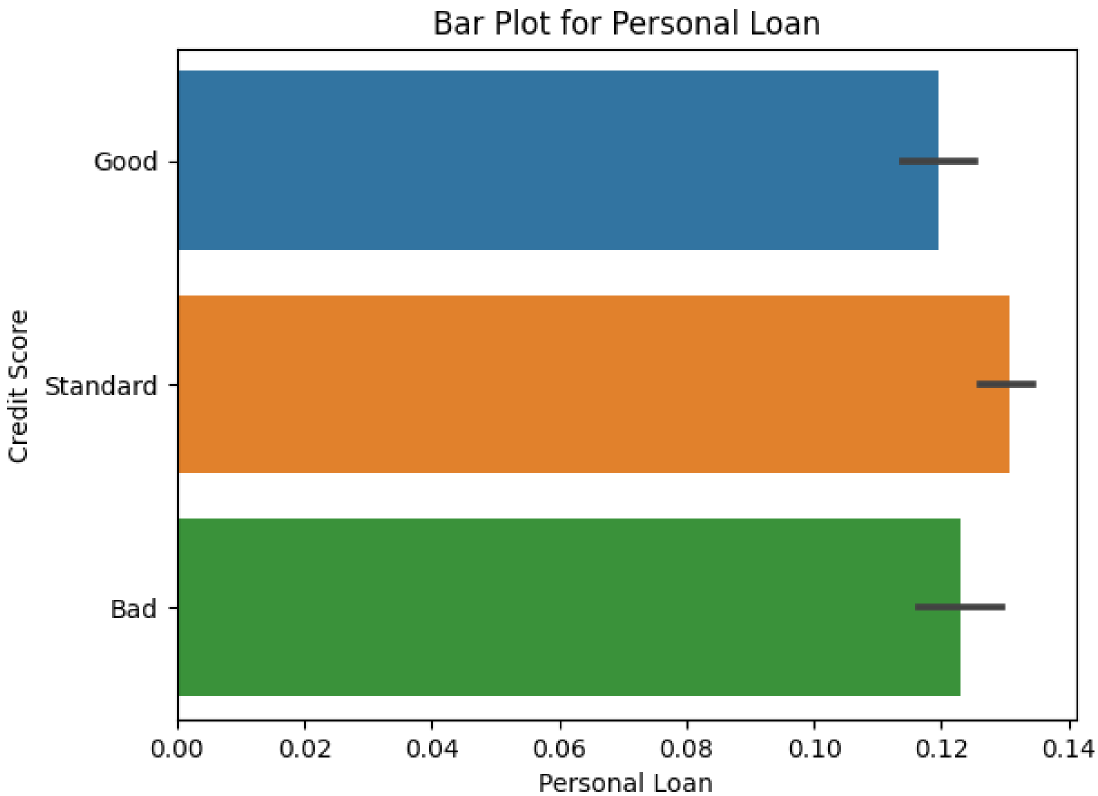

Bar Plot for Home Equity Loan
Good
Standard
Bad
Credit Score
0.000
0.005
0.010
0.015
0.020
Home Equity Loan

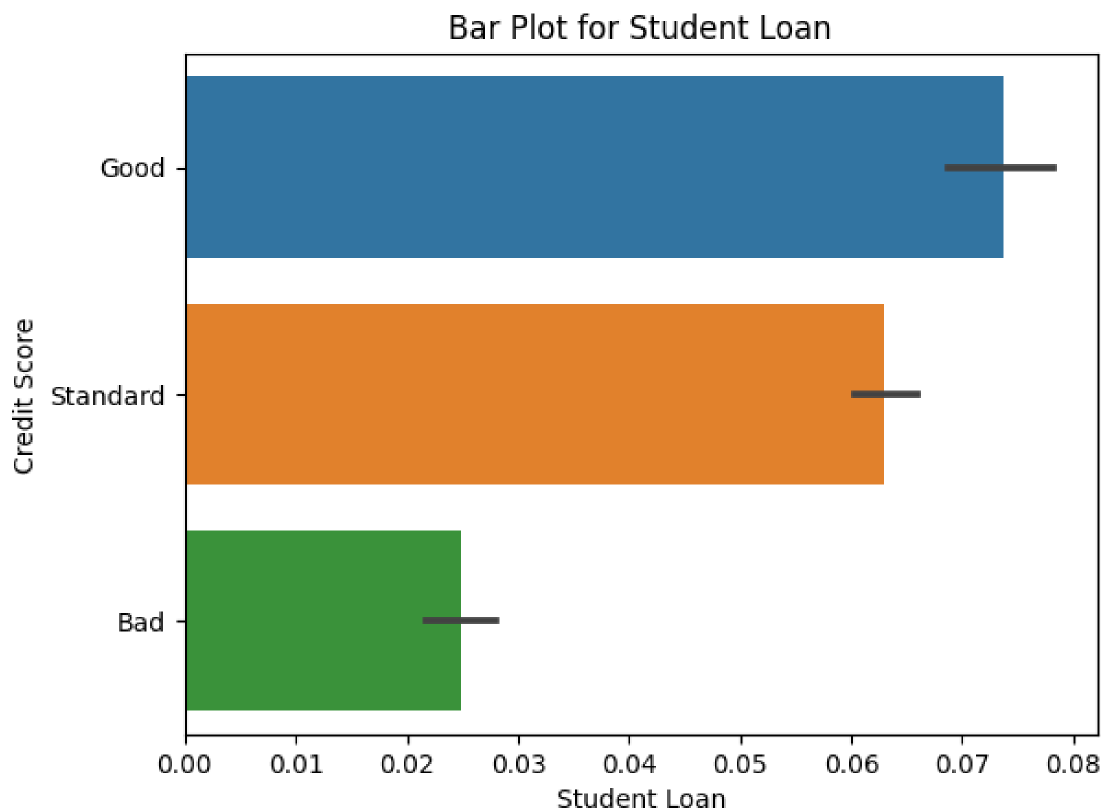

Bar Plot for Mortgage Loan
Credit Score
Good
Standard
Bad
0.00
0.01
0.02
0.03
0.04
Mortgage Loan



Bar Plot for Not Specified
Good
Standard
Bad
Credit Score
0.0
0.1
0.2
0.3
0.4
0.5
Not Specified


Bar Plot for Payday Loan
Good
Standard
Bad
Credit Score
0.00
0.01
0.02
0.03
0.04
0.05
0.06
Payday Loan


Bar Plot for Personal Loan
Good
Standard
Bad
Credit Score
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Personal Loan



0.1.6 Text Preprocessing and encoding

[85]:
```
df.dtypes
```

[85]:
```
Customer_ID                   int64
Month                        object
Age                         float64
Occupation                   object
Annual_Income               float64
Monthly_Inhand_Salary       float64
Num_Bank_Accounts           float64
Num_Credit_Card             float64
Interest_Rate               float64
Num_of_Loan                 float64
Delay_from_due_date           int64
Num_of_Delayed_Payment      float64
Changed_Credit_Limit        float64
Num_Credit_Inquiries        float64
Credit_Score                 object
Outstanding_Debt            float64
```

```
Credit_Utilization_Ratio    float64
Credit_History_Age          float64
Payment_of_Min_Amount        object
Total_EMI_per_month         float64
Amount_invested_monthly     float64
Payment_Behaviour            object
Monthly_Balance             float64
Auto Loan                     int64
Credit-Builder Loan           int64
Debt Consolidation Loan       int64
Home Equity Loan              int64
Mortgage Loan                 int64
Not Specified                 int64
Payday Loan                   int64
Personal Loan                 int64
Student Loan                  int64
dtype: object
```

[86]:
```
df.drop(['Customer_ID', 'Credit_Score'], axis=1, inplace=True)
```

[87]:
```
df
```

[87]:
```
           Month   Age Occupation  Annual_Income  Monthly_Inhand_Salary  \
0      September  23.0  Scientist       19114.12            1824.843333
1        October  24.0  Scientist       19114.12            1824.843333
2       November  24.0  Scientist       19114.12            1824.843333
3       December  24.0  Scientist       19114.12            3086.305000
4      September  28.0     Lawyer       34847.84            3037.986667
…            …    …        …            …                     …
49995   December  33.0  Architect       20002.88            1929.906667
49996  September  25.0   Mechanic       39628.99            3086.305000
49997    October  25.0   Mechanic       39628.99            3359.415833
49998   November  25.0   Mechanic       39628.99            3086.305000
49999   December  25.0   Mechanic       39628.99            3359.415833

       Num_Bank_Accounts  Num_Credit_Card  Interest_Rate  Num_of_Loan  \
0                    3.0              4.0            3.0          4.0
1                    3.0              4.0            3.0          4.0
2                    3.0              4.0            3.0          4.0
3                    3.0              4.0            3.0          4.0
4                    2.0              4.0            6.0          1.0
…                  …                …              …            …
49995               10.0              8.0           29.0          5.0
49996                4.0              6.0            7.0          2.0
49997                4.0              6.0            7.0          2.0
49998                4.0              6.0            7.0          2.0
49999                4.0              6.0            7.0          2.0
```

```
       Delay_from_due_date  Num_of_Delayed_Payment  Changed_Credit_Limit  \
0                        3                     7.0                 11.27
1                        3                     9.0                 13.27
2                       -1                     4.0                 12.27
3                        4                     5.0                 11.27
4                        3                     1.0                  5.42
…                        …                       …                     …
49995                   33                    25.0                 18.31
49996                   20                    14.0                 11.50
49997                   23                     5.0                 13.50
49998                   21                     6.0                 11.50
49999                   22                     5.0                 11.50

       Num_Credit_Inquiries  Outstanding_Debt  Credit_Utilization_Ratio  \
0                       7.0            809.98                 35.030402
1                       4.0            809.98                 33.053114
2                       4.0            809.98                 33.811894
3                       4.0            809.98                 32.430559
4                       5.0            605.03                 25.926822
…                         …                 …                         …
49995                  12.0           3571.70                 34.780553
49996                   7.0            502.38                 27.758522
49997                   7.0            502.38                 36.858542
49998                   7.0            502.38                 39.139840
49999                   7.0            502.38                 34.108530

       Credit_History_Age Payment_of_Min_Amount  Total_EMI_per_month  \
0                   273.0                    No            49.574949
1                   274.0                    No            49.574949
2                   225.0                    No            49.574949
3                   276.0                    No            49.574949
4                   327.0                    No            18.816215
…                       …                     …                    …
49995               225.0                   Yes            60.964772
49996               383.0                   Yes            35.104023
49997               384.0                    No            35.104023
49998               385.0                    No            35.104023
49999               386.0                    No            35.104023

       Amount_invested_monthly                 Payment_Behaviour  \
0                   236.642682    Low_spent_Small_value_payments
1                    21.465380  High_spent_Medium_value_payments
2                   148.233938   Low_spent_Medium_value_payments
3                    39.082511  High_spent_Medium_value_payments
4                    39.684018   High_spent_Large_value_payments
…                            …                                 …
```

```
49995              146.486325    Low_spent_Small_value_payments
49996              181.442999    Low_spent_Small_value_payments
49997              135.590430    Low_spent_Large_value_payments
49998               97.598580   High_spent_Small_value_payments
49999              220.457878   Low_spent_Medium_value_payments

       Monthly_Balance  Auto Loan  Credit-Builder Loan  \
0           186.266702          0                    1
1           361.444004          0                    1
2           264.675446          0                    1
3           343.826873          0                    1
4           485.298434          0                    1
…                    …          …                    …
49995       275.539570          0                    0
49996       409.394562          0                    0
49997       349.726332          0                    0
49998       463.238981          0                    0
49999       360.379683          0                    0

       Debt Consolidation Loan  Home Equity Loan  Mortgage Loan  \
0                            0                 0              0
1                            0                 0              0
2                            0                 0              0
3                            0                 0              0
4                            0                 0              0
…                            …                 …              …
49995                        0                 0              0
49996                        0                 0              0
49997                        0                 0              0
49998                        0                 0              0
49999                        0                 0              0

       Not Specified  Payday Loan  Personal Loan  Student Loan
0                  0            0              0             0
1                  0            0              0             0
2                  0            0              0             0
3                  0            0              0             0
4                  0            0              0             0
…                  …            …              …             …
49995              0            0              1             0
49996              0            0              0             1
49997              0            0              0             1
49998              0            0              0             1
49999              0            0              0             1

[50000 rows x 30 columns]
```

[88]:
```python
def OH_encode(dataframe, column):
    dummy = pd.get_dummies(dataframe[column], prefix=str(column),␣
↪drop_first=True).astype(int)
    dataframe.drop(column, axis=1, inplace=True)
    return dataframe, dummy
```

[89]:
```python
encode_cols = df.select_dtypes('object')
```

[90]:
```python
for col in encode_cols:
    df, dummy = OH_encode(dataframe=df, column=col)
    df = pd.concat([df, dummy], axis=1)
```

Feature transformation

[91]:
```python
X = df
```

[92]:
```python
scaler = StandardScaler()
columns_toScale = X.columns[:17]
```

[93]:
```python
X[columns_toScale] = scaler.fit_transform(X[columns_toScale])
```

[94]:
```python
X
```

[94]:
```
            Age  Annual_Income  Monthly_Inhand_Salary  Num_Bank_Accounts  \
0     -1.017213      -0.833538              -0.764202          -0.917781
1     -0.923032      -0.833538              -0.764202          -0.917781
2     -0.923032      -0.833538              -0.764202          -0.917781
3     -0.923032      -0.833538              -0.222372          -0.917781
4     -0.546307      -0.380347              -0.243126          -1.305902
…           …            …                     …                 …
49995 -0.075401      -0.807938              -0.719074           1.799063
49996 -0.828851      -0.242631              -0.222372          -0.529660
49997 -0.828851      -0.242631              -0.105064          -0.529660
49998 -0.828851      -0.242631              -0.222372          -0.529660
49999 -0.828851      -0.242631              -0.105064          -0.529660

       Num_Credit_Card  Interest_Rate  Num_of_Loan  Delay_from_due_date  \
0            -0.741688      -1.327974     0.204078            -1.311033
1            -0.741688      -1.327974     0.204078            -1.311033
2            -0.741688      -1.327974     0.204078            -1.635023
3            -0.741688      -1.327974     0.204078            -1.230036
4            -0.741688      -0.981605    -1.049318            -1.311033
…                  …              …          …                    …
49995         1.210020       1.673890     0.621877             1.118888
49996         0.234166      -0.866149    -0.631520             0.065922
49997         0.234166      -0.866149    -0.631520             0.308914
49998         0.234166      -0.866149    -0.631520             0.146920
```

```
49999         0.234166      -0.866149    -0.631520             0.227917

       Num_of_Delayed_Payment  Changed_Credit_Limit  Num_Credit_Inquiries  \
0                   -1.087484              0.157452             -0.071593
1                   -0.751126              0.509143             -0.845009
2                   -1.592022              0.333298             -0.845009
3                   -1.423843              0.157452             -0.845009
4                   -2.096560             -0.871241             -0.587203
…                       …                     …                     …
49995                1.939743              1.395402              1.217434
49996                0.089771              0.197897             -0.071593
49997               -1.423843              0.549587             -0.071593
49998               -1.255664              0.197897             -0.071593
49999               -1.423843              0.197897             -0.071593

       Outstanding_Debt  Credit_Utilization_Ratio  Credit_History_Age  \
0             -0.491741                  0.538723            0.483679
1             -0.491741                  0.151489            0.494205
2             -0.491741                  0.300089           -0.021578
3             -0.491741                  0.029568            0.515257
4             -0.720976                 -1.244130            1.052092
…                 …                         …                  …
49995          2.597221                  0.489792           -0.021578
49996         -0.835790                 -0.885408            1.641558
49997         -0.835790                  0.896748            1.652084
49998         -0.835790                  1.343519            1.662611
49999         -0.835790                  0.358183            1.673137

       Total_EMI_per_month  Amount_invested_monthly  Monthly_Balance  \
0                -0.524272                 0.840685        -1.252744
1                -0.524272                -1.239900         0.055237
2                -0.524272                -0.014154        -0.667296
3                -0.524272                -1.069557        -0.076304
4                -0.882296                -1.063741         0.980009
…                    …                        …                …
49995            -0.391698                -0.031052        -0.586178
49996            -0.692710                 0.306950         0.413265
49997            -0.692710                -0.136406        -0.032255
49998            -0.692710                -0.503756         0.815300
49999            -0.692710                 0.684192         0.047290

       Auto Loan  Credit-Builder Loan  Debt Consolidation Loan  \
0              0                    1                        0
1              0                    1                        0
2              0                    1                        0
3              0                    1                        0
4              0                    1                        0
```

```
…          …                  …                        …
49995        0                    0                          0
49996        0                    0                          0
49997        0                    0                          0
49998        0                    0                          0
49999        0                    0                          0

       Home Equity Loan  Mortgage Loan  Not Specified  Payday Loan  \
0                     0              0              0            0
1                     0              0              0            0
2                     0              0              0            0
3                     0              0              0            0
4                     0              0              0            0
…                   …            …            …          …
49995                 0              0              0            0
49996                 0              0              0            0
49997                 0              0              0            0
49998                 0              0              0            0
49999                 0              0              0            0

       Personal Loan  Student Loan  Month_November  Month_October  \
0                  0             0               0              0
1                  0             0               0              1
2                  0             0               1              0
3                  0             0               0              0
4                  0             0               0              0
…                …           …             …              …
49995              1             0               0              0
49996              0             1               0              0
49997              0             1               0              1
49998              0             1               1              0
49999              0             1               0              0

       Month_September  Occupation_Architect  Occupation_Developer  \
0                    1                     0                     0
1                    0                     0                     0
2                    0                     0                     0
3                    0                     0                     0
4                    1                     0                     0
…                  …                     …                    …
49995                0                     1                     0
49996                1                     0                     0
49997                0                     0                     0
49998                0                     0                     0
49999                0                     0                     0

       Occupation_Doctor  Occupation_Engineer  Occupation_Entrepreneur  \
```

```
0                      0                   0                         0
1                      0                   0                         0
2                      0                   0                         0
3                      0                   0                         0
4                      0                   0                         0
…                  …                   …                         …
49995                  0                   0                         0
49996                  0                   0                         0
49997                  0                   0                         0
49998                  0                   0                         0
49999                  0                   0                         0

       Occupation_Journalist  Occupation_Lawyer  Occupation_Manager  \
0                          0                  0                   0
1                          0                  0                   0
2                          0                  0                   0
3                          0                  0                   0
4                          0                  1                   0
…                        …                …                 …
49995                      0                  0                   0
49996                      0                  0                   0
49997                      0                  0                   0
49998                      0                  0                   0
49999                      0                  0                   0

       Occupation_Mechanic  Occupation_MediaManager  Occupation_Musician  \
0                        0                        0                    0
1                        0                        0                    0
2                        0                        0                    0
3                        0                        0                    0
4                        0                        0                    0
…                    …                      …                  …
49995                    0                        0                    0
49996                    1                        0                    0
49997                    1                        0                    0
49998                    1                        0                    0
49999                    1                        0                    0

       Occupation_Scientist  Occupation_Teacher  Occupation_Writer  \
0                         1                   0                  0
1                         1                   0                  0
2                         1                   0                  0
3                         1                   0                  0
4                         0                   0                  0
…                     …                 …                …
49995                     0                   0                  0
49996                     0                   0                  0
```

```
49997                           0                         0                        0
49998                           0                         0                        0
49999                           0                         0                        0

       Payment_of_Min_Amount_Yes  \
0                              0
1                              0
2                              0
3                              0
4                              0
…                            …
49995                          1
49996                          1
49997                          0
49998                          0
49999                          0

       Payment_Behaviour_High_spent_Medium_value_payments  \
0                                                      0
1                                                      1
2                                                      0
3                                                      1
4                                                      0
…                                                    …
49995                                                  0
49996                                                  0
49997                                                  0
49998                                                  0
49999                                                  0

       Payment_Behaviour_High_spent_Small_value_payments  \
0                                                      0
1                                                      0
2                                                      0
3                                                      0
4                                                      0
…                                                    …
49995                                                  0
49996                                                  0
49997                                                  0
49998                                                  1
49999                                                  0

       Payment_Behaviour_Low_spent_Large_value_payments  \
0                                                     0
1                                                     0
2                                                     0
```

```
3                                                      0
4                                                      0
…                                                  …
49995                                                  0
49996                                                  0
49997                                                  1
49998                                                  0
49999                                                  0

       Payment_Behaviour_Low_spent_Medium_value_payments  \
0                                                       0
1                                                       0
2                                                       1
3                                                       0
4                                                       0
…                                                   …
49995                                                   0
49996                                                   0
49997                                                   0
49998                                                   0
49999                                                   1

       Payment_Behaviour_Low_spent_Small_value_payments
0                                                      1
1                                                      0
2                                                      0
3                                                      0
4                                                      0
…                                                  …
49995                                                  1
49996                                                  1
49997                                                  0
49998                                                  0
49999                                                  0

[50000 rows x 49 columns]
```

0.1.7 Model Splitting

[95]:
```
Y = pd.DataFrame(y)
```

[96]:
```
xTrain, xTest, yTrain, yTest = train_test_split(X, Y, test_size=.3,␣
↪random_state=42)
```

Balancing Dataset

[97]:
```
smote = SMOTE(random_state=42)
xTrain, yTrain = smote.fit_resample(xTrain, yTrain)
```

Logistic Regression

[98]:
```
LR_model = LogisticRegression()
```

[99]:
```
LR_model.fit(xTrain, yTrain)
```

```
/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning:

lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

[99]:
```
LogisticRegression()
```

RandomForest Classifier

[100]:
```
RFC = RandomForestClassifier()
```

[101]:
```
RFC.fit(xTrain, yTrain)
```

```
/tmp/ipykernel_37599/469850128.py:1: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
```

[101]:
```
RandomForestClassifier()
```

XGboost

[102]:
```
XGB = xgb()
```

[103]:
```
XGB.fit(xTrain, yTrain)
```

[103]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, objective='multi:softprob', …)
```

SVC

[104]:
```
svc = SVC(kernel='rbf', gamma='scale', probability=True)
```

[105]:
```
svc.fit(xTrain, yTrain)
```

```
/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
```

[105]:
```
SVC(probability=True)
```

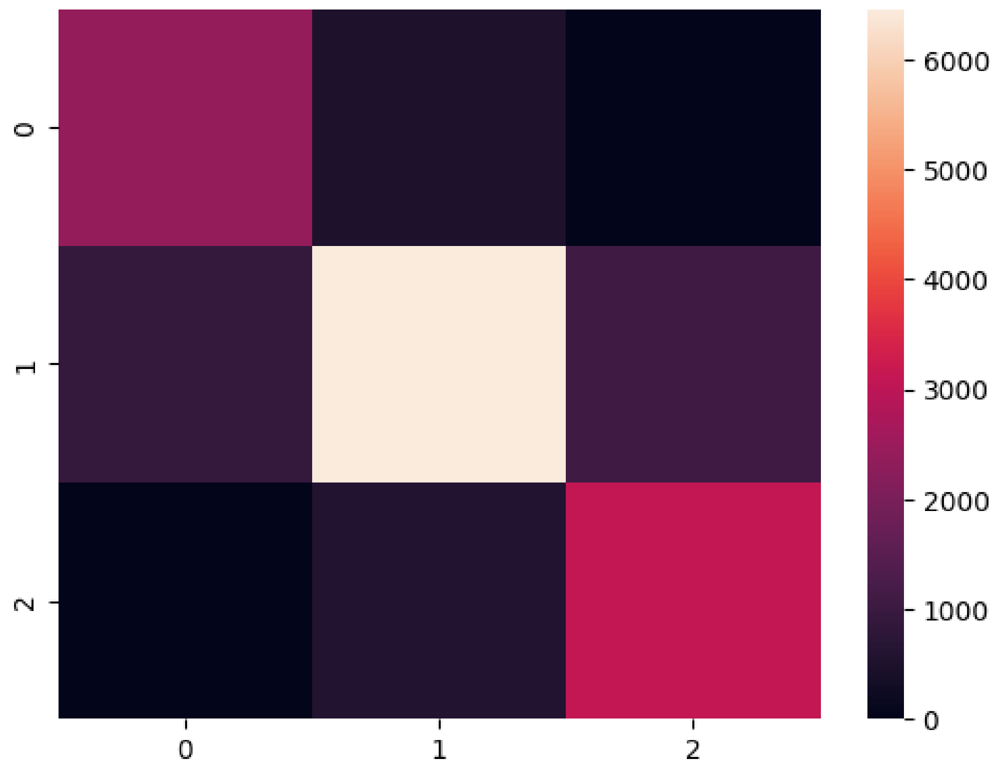
Predcting and Evaluation Metrics

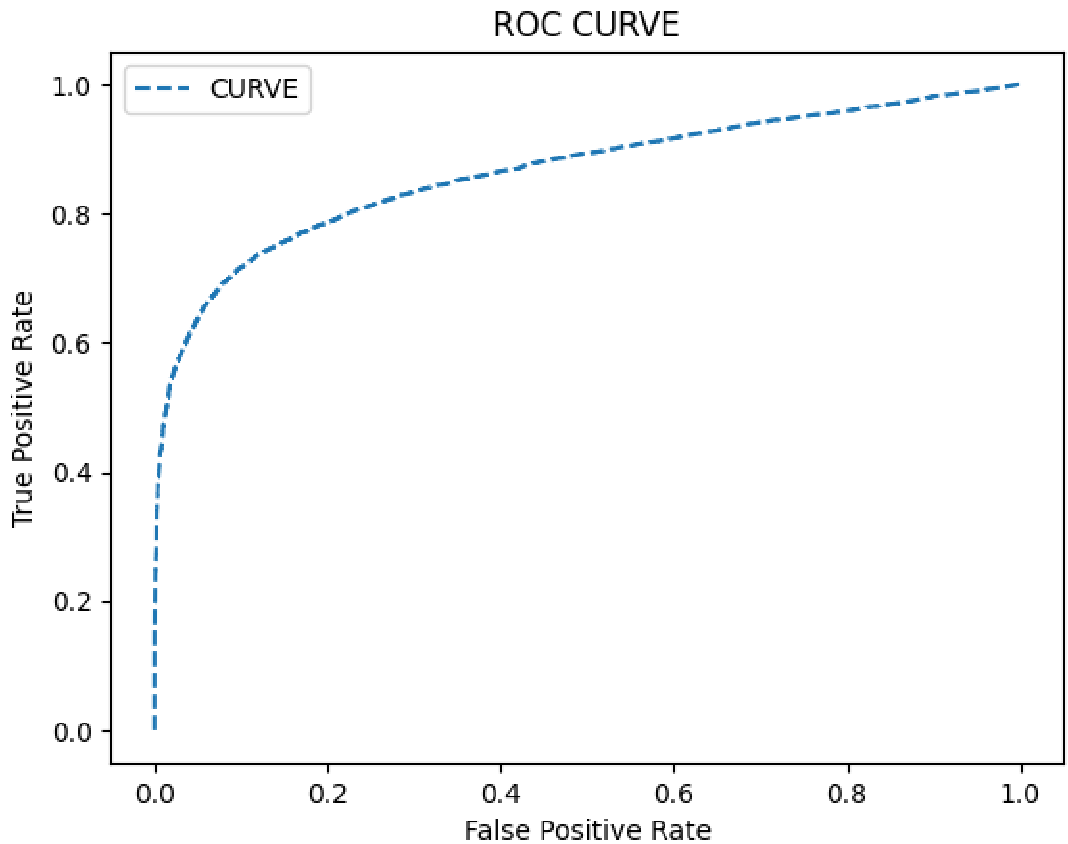
[106]:
```
trained_models = [
    LR_model,
    RFC,
    XGB,
    svc
]
```

[107]:
```
for model in trained_models:
    y_pred = model.predict(xTest)
    accuracy = accuracy_score(yTest, y_pred)
    precision = precision_score(yTest, y_pred, average='micro')
    recall = recall_score(yTest, y_pred, average='micro')
    f1 = f1_score(yTest, y_pred, average='micro')
    cm = confusion_matrix(yTest, y_pred)


    proba = model.predict_proba(xTest)
```

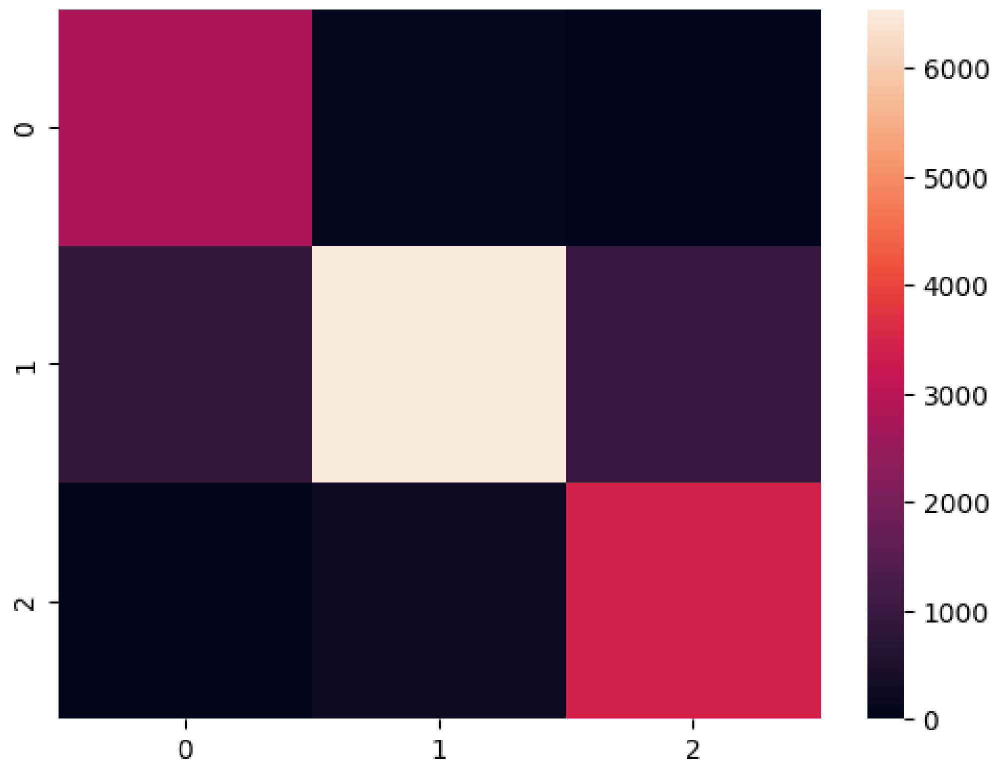
```python
    auc = roc_auc_score(yTest, proba, multi_class='ovr')

    print(f"\t ********* {model.__class__.__name__} *********")
    print("Accuracy Score: ", accuracy)
    print("Precision Score: ", precision)
    print("Recall Score: ", recall)
    print("F1 Score: ", f1)
    print("AUC Score: ", auc)
    print("Confusion Matrix: \n", cm)
    sns.heatmap(cm)
    plt.show()

    fpr, tpr, thresold = roc_curve(yTest, proba[:,1], pos_label=1)
    plt.plot(fpr, tpr, linestyle="--", label="CURVE", )
    plt.title("ROC CURVE")
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.legend()
    plt.show()
```

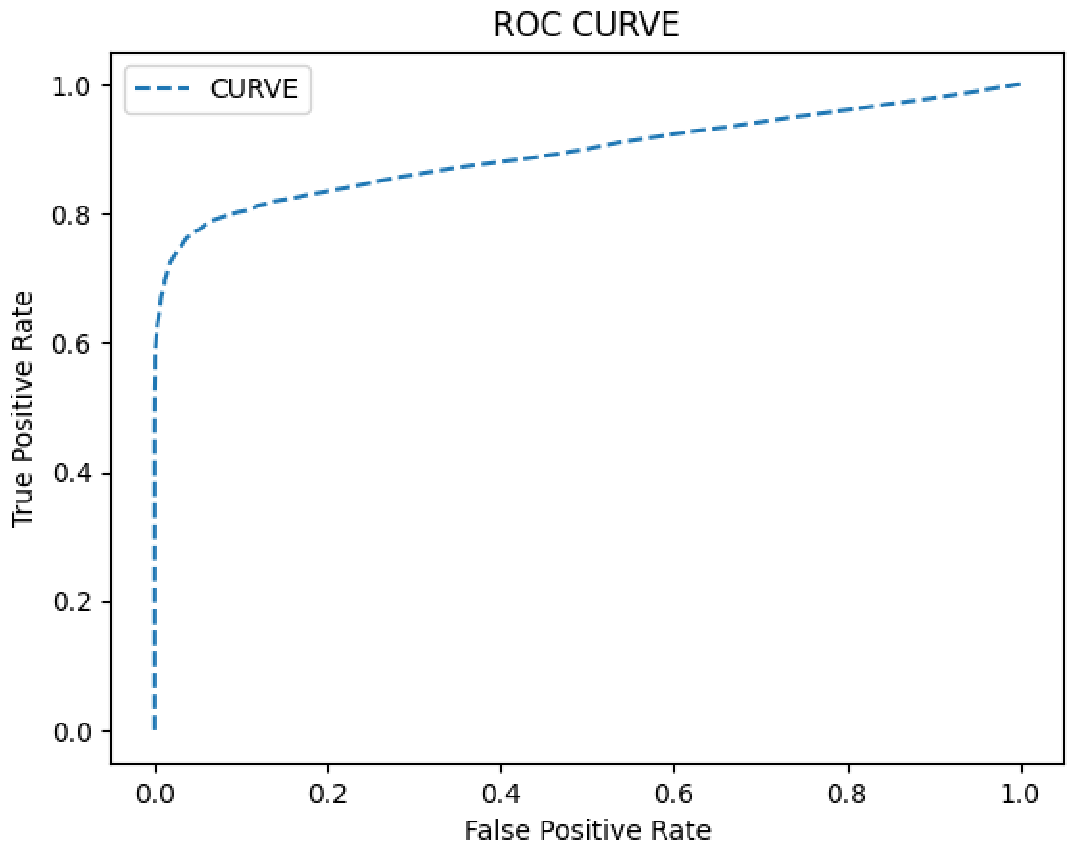
```
         ********* LogisticRegression *********
Accuracy Score:  0.7966666666666666
Precision Score:  0.7966666666666666
Recall Score:  0.7966666666666666
F1 Score:  0.7966666666666665
AUC Score:  0.9224079427090616
Confusion Matrix:
 [[2405  500    0]
 [ 896 6458 1063]
 [   0  591 3087]]
```


0
1
2
0
1
2
6000
5000
4000
3000
2000
1000
0

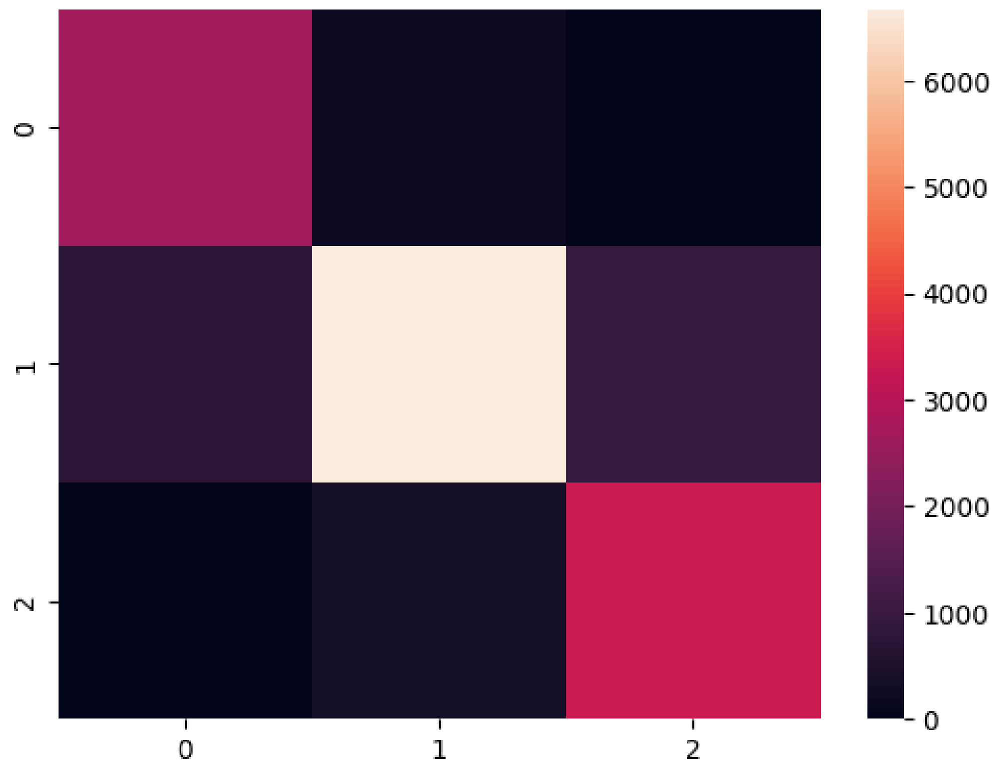


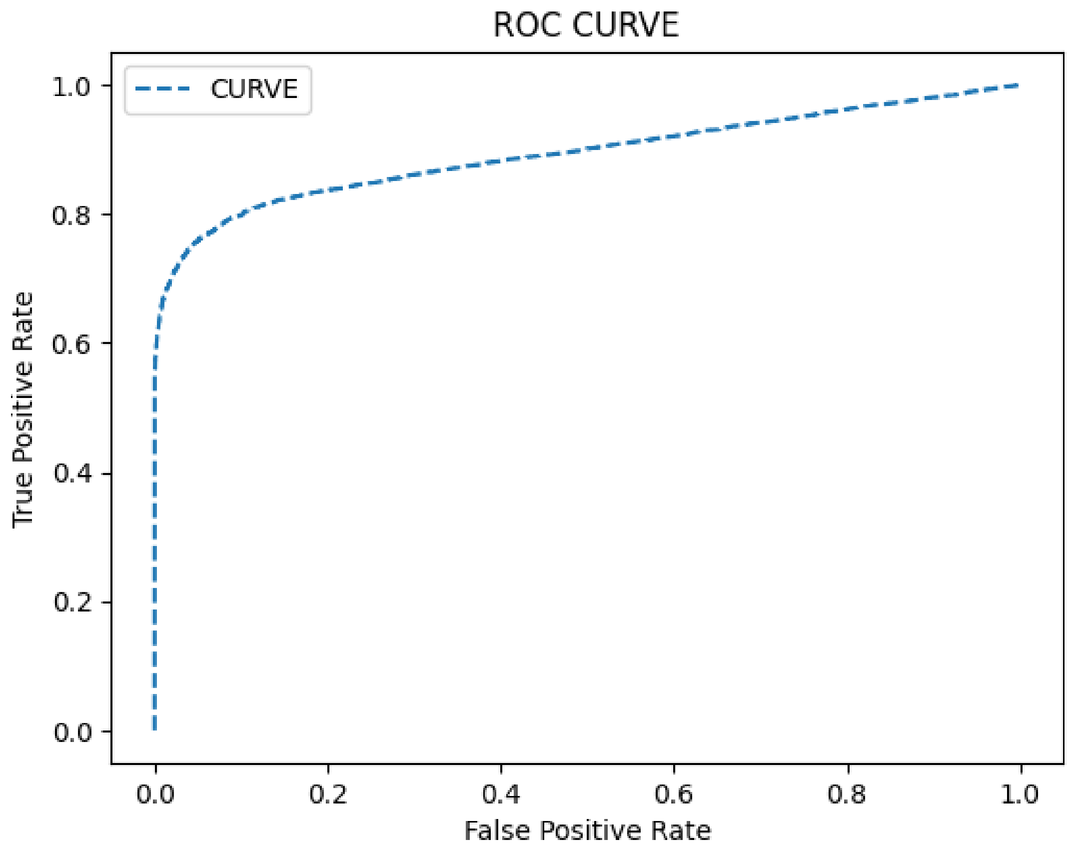

```
        ********* RandomForestClassifier *********
Accuracy Score:  0.8511333333333333
Precision Score:  0.8511333333333333
Recall Score:  0.8511333333333333
F1 Score:  0.8511333333333333
AUC Score:  0.9390130617000828
Confusion Matrix:
 [[2792  113    0]
 [ 902 6540  975]
 [   0  243 3435]]
```


0
1
2
0
1
2
6000
5000
4000
3000
2000
1000
0

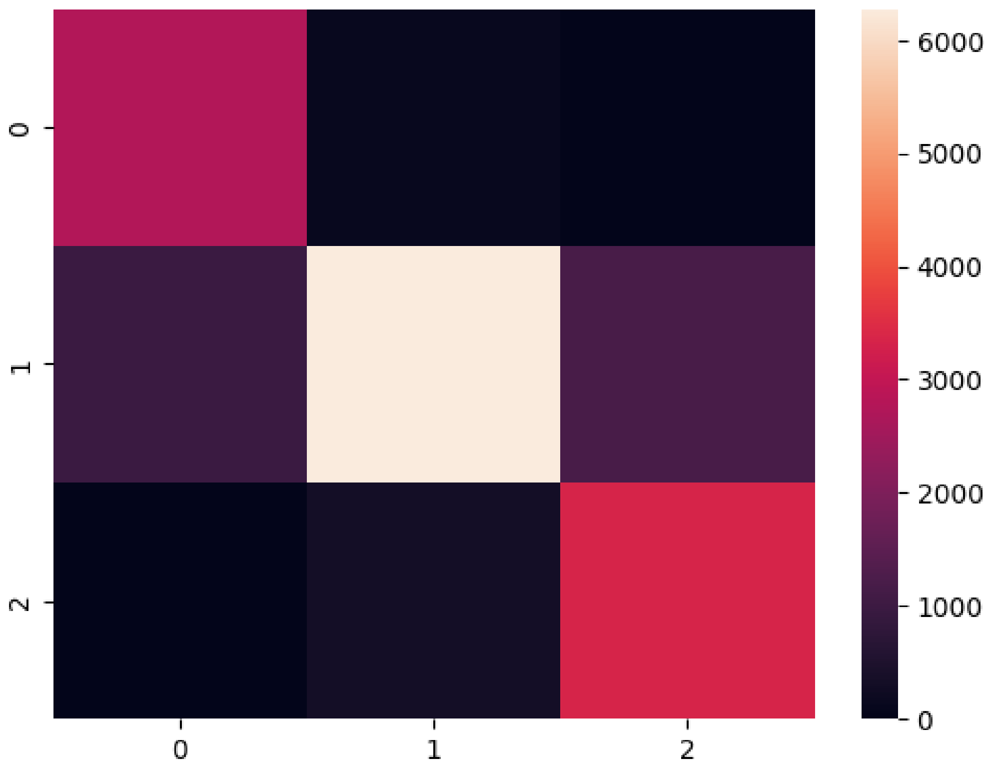


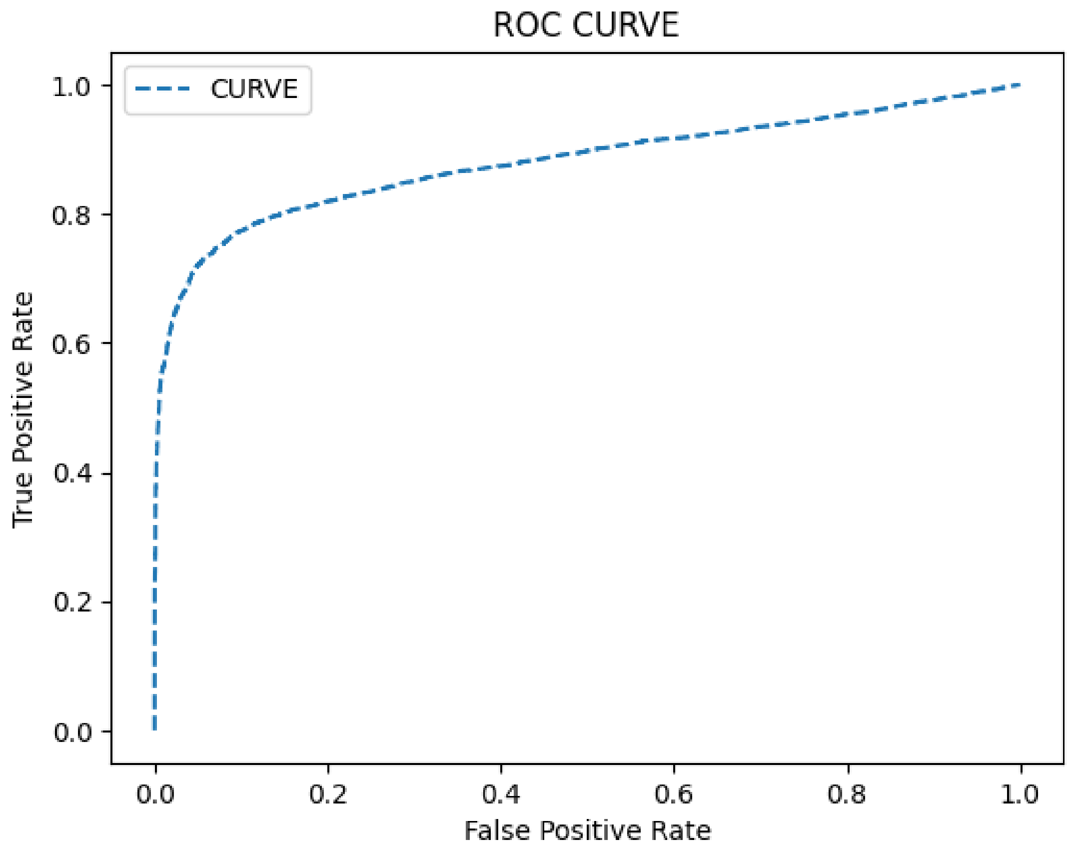

```
         ********* XGBClassifier *********
Accuracy Score:  0.8449333333333333
Precision Score:  0.8449333333333333
Recall Score:  0.8449333333333333
F1 Score:  0.8449333333333333
AUC Score:  0.938216526980666
Confusion Matrix:
 [[2679  226    0]
 [ 778 6668  971]
 [   0  351 3327]]
```


0
1
2
0
1
2
6000
5000
4000
3000
2000
1000
0




```
         ********* SVC *********
Accuracy Score:  0.8256666666666667
Precision Score:  0.8256666666666667
Recall Score:  0.8256666666666667
F1 Score:  0.8256666666666665
AUC Score:  0.9308429786852145
Confusion Matrix:
 [[2759  146    0]
 [ 966 6272 1179]
 [   0  324 3354]]
```


0
1
2
0
1
2
6000
5000
4000
3000
2000
1000
0



RandomForest Classifier has the highest accuracy. So we will use Gradient Boosting Classifier for hyper parameter tuning

Hyperparameter Tuning:

[108]:
```
params = {
    'n_estimators': [100, 200, 300],
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 3],
    'max_features': ['auto', 'sqrt', 'log2']
}
```

[109]:
```
# Choosing Best Model using RandomSearchCV
```

[110]:
```
model_param = {
    'svm': {
        'model': SVC(gamma='auto'),
        'params': {
```

```
            'C': [1.0, 5.0, 10.0],
            'kernel': ['rbf', 'linear']
        }
    },
    'LogReg': {
        'model': LogisticRegression(solver='liblinear'),
        'params': {
            'C': [1.0, 5.0, 10.0],
            'penalty': ['l1', 'l2'],
        }
    },
    'rf': {
        'model': RandomForestClassifier(),
        'params': {
            'n_estimators': [20, 50, 100],
            'criterion': ['gini', 'entropy'],
            'min_samples_leaf' : [1, 2],
            'max_features': ['sqrt', 'log2']
        }
    },
    'XGboost': {
        'model': xgb(),
        'params': {
            'learning_rate': [0.1, 0.01, 0.2],
            'n_estimators': [20, 50, 100],
        }
    }
}
```

[111]:
```
scores = []

for name, mp in model_param.items():
    RandomSearch = RandomizedSearchCV(estimator=mp['model'] ,␣
 ↪param_distributions=mp['params'], return_train_score=False, n_iter=1)
    RandomSearch.fit(xTrain, yTrain)
    scores.append({
        'model': mp['model'],
        'best_score': RandomSearch.best_score_,
        'best_param': RandomSearch.best_params_
    })
```

```
/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
```

```
/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().
```

```
/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/utils/validation.py:1143: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples, ), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().

/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/model_selection/_validation.py:686: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
```

```
/home/applehx7/anaconda3/lib/python3.11/site-
packages/sklearn/model_selection/_search.py:909: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
```

[112]:
```
scores
```

[112]:
```
[{'model': SVC(gamma='auto'),
  'best_score': 0.8704754397231678,
  'best_param': {'kernel': 'linear', 'C': 1.0}},
 {'model': LogisticRegression(solver='liblinear'),
  'best_score': 0.8442370485100532,
  'best_param': {'penalty': 'l2', 'C': 10.0}},
 {'model': RandomForestClassifier(),
  'best_score': 0.9104233993163928,
  'best_param': {'n_estimators': 100,
   'min_samples_leaf': 2,
   'max_features': 'log2',
   'criterion': 'gini'}},
 {'model': XGBClassifier(base_score=None, booster=None, callbacks=None,
                colsample_bylevel=None, colsample_bynode=None,
                colsample_bytree=None, device=None, early_stopping_rounds=None,
                enable_categorical=False, eval_metric=None, feature_types=None,
                gamma=None, grow_policy=None, importance_type=None,
                interaction_constraints=None, learning_rate=None, max_bin=None,
                max_cat_threshold=None, max_cat_to_onehot=None,
                max_delta_step=None, max_depth=None, max_leaves=None,
                min_child_weight=None, missing=nan, monotone_constraints=None,
                multi_strategy=None, n_estimators=None, n_jobs=None,
                num_parallel_tree=None, random_state=None, …),
  'best_score': 0.8930711332632789,
  'best_param': {'n_estimators': 20, 'learning_rate': 0.1}}]
```

RFC has the best score, so we will use RFC as well

Reason Behind choosing RandomForest Hyperparameter For hyperparameter tuning in the Gradient Boosting Classifier, Random Search was employed to explore various hyperparameter combinations efficiently, considering the model's performance and computational efficiency.The goal was to optimize an evaluation metric (e.g., accuracy, F1-score, AUC-ROC), and after evaluating various combinations, the selected hyperparameters were chosen based on the trade-offs between model performance, training time, and avoiding overfitting

Predict testing values with hyperparameter tuning

[113]:
```
rf = RandomForestClassifier(n_estimators=50, min_samples_leaf=1,␣
 ↪criterion='entropy')
rf.fit(xTrain, yTrain)
b_predicted = rf.predict(xTest)
```

```
/tmp/ipykernel_37599/835512972.py:2: DataConversionWarning:

A column-vector y was passed when a 1d array was expected. Please change the
shape of y to (n_samples,), for example using ravel().
```

[114]:
```
accuracy_score(yTest, b_predicted)
```

[114]: 0.8482666666666666

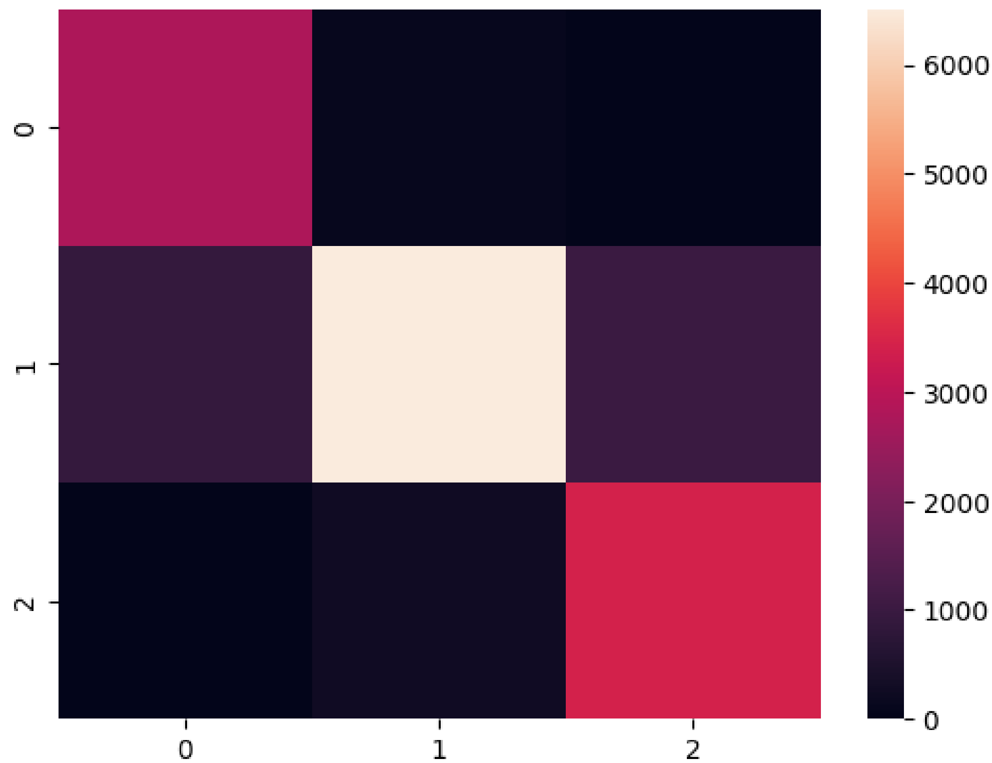
[115]:
```
accuracy = accuracy_score(yTest, b_predicted)
precision = precision_score(yTest, b_predicted, average='micro')
recall = recall_score(yTest, b_predicted, average='micro')
f1 = f1_score(yTest, b_predicted, average='micro')
cm = confusion_matrix(yTest, b_predicted)
```

[116]:
```
print("Accuracy Score: ", accuracy)
print("Precision Score: ", precision)
print("Recall Score: ", recall)
print("F1 Score: ", f1)
print("AUC Score: ", auc)
print("Confusion Matrix: \n", cm)
```

```
Accuracy Score:  0.8482666666666666
Precision Score:  0.8482666666666666
Recall Score:  0.8482666666666666
F1 Score:  0.8482666666666666
AUC Score:  0.9308429786852145
Confusion Matrix:
 [[2795  110    0]
 [ 903 6508 1006]
 [   0  257 3421]]
```

[117]:
```
sns.heatmap(cm)
```

[117]: <Axes: >



Model Evaluation We have done it before, after training the data. So no need to do it again

0.1.8 *Discuss the strengths and limitations of each model in the context of credit score classification.*

0.1.9 Logistic Regression:

1. **Strengths**:
    - Interpretability: Logistic Regression provides easily interpretable coefficients that show feature importance.
    - Efficient with large datasets and less prone to overfitting.
    - Works well when the relationship between features and target is linear.
2. **Limitations**:
    - Assumes linear relationship between features and target, might not capture complex patterns.
    - Sensitive to outliers and multicollinearity.
    - May not handle non-linear relationships effectively.

0.1.10 Random Forest Classifier:

1. **Strengths**:

- Robust to overfitting due to ensemble learning and decision tree structure.
- Handles non-linear relationships and interactions between features well.
- Provides feature importance metrics.

2. **Limitations**:

- Can be computationally expensive with a large number of trees or features.
- May not be as interpretable as simpler models like Logistic Regression.
- Prone to overfitting if not tuned properly.

0.1.11 Support Vector Machine (SVM):

1. **Strengths**:

- Effective in high-dimensional spaces, especially with non-linear kernel functions.
- Versatile with different kernel functions (linear, polynomial, radial basis function).

2. **Limitations**:

- Memory-intensive and might be slow on large datasets.
- Requires careful selection of kernel and tuning of hyperparameters.
- Interpretability can be challenging, especially with non-linear kernels.

0.1.12 Gradient Boosting Classifier (XGBoost):

1. **Strengths**:

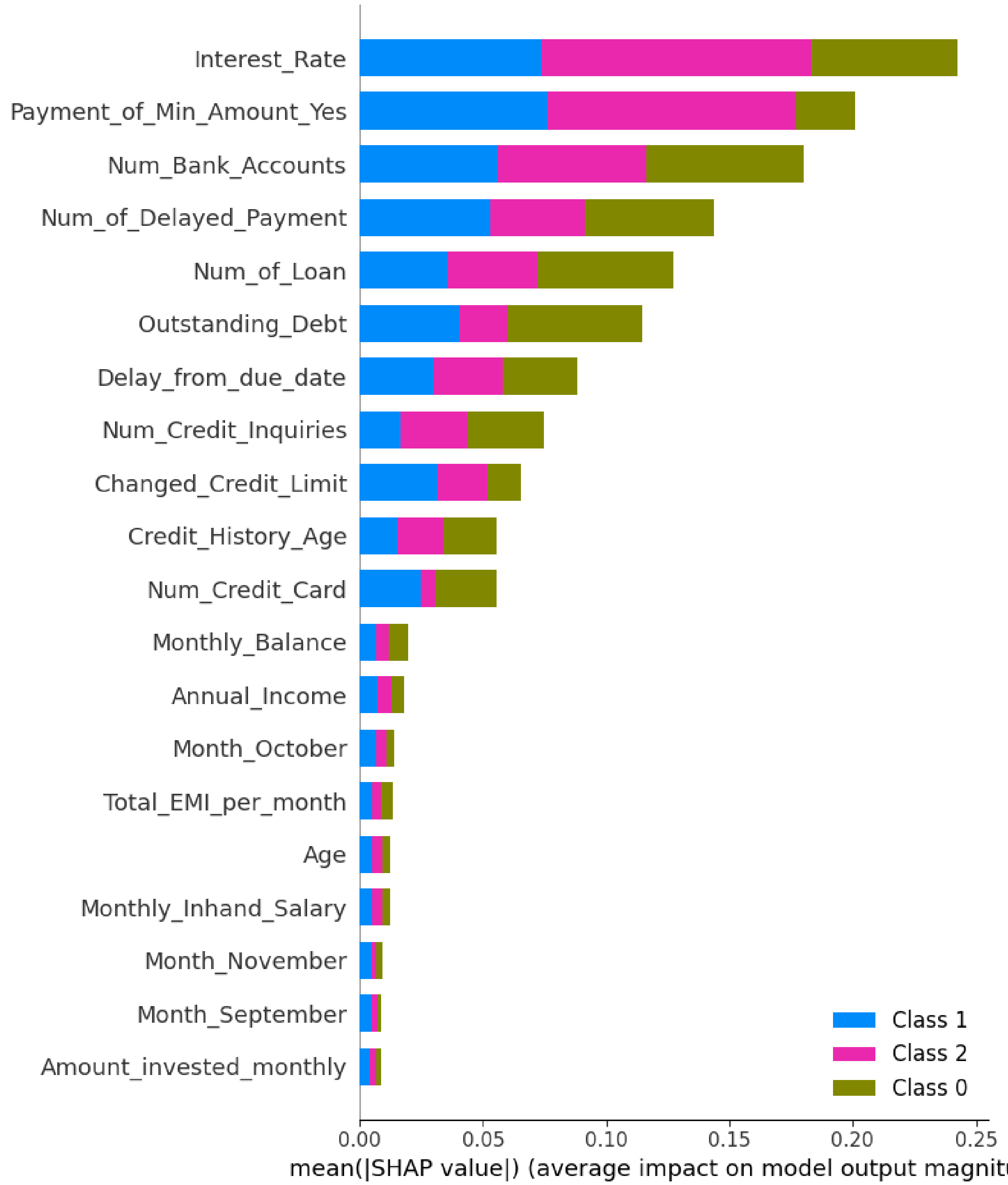
- High predictive accuracy due to sequential learning from weak learners.
- Handles complex interactions and non-linear relationships effectively.
- Good with handling missing data and irrelevant features.

2. **Limitations**:

- Prone to overfitting if hyperparameters are not properly tuned.
- Computationally expensive and may take longer to train.
- Interpretability might be a challenge due to the ensemble nature.

Interpretability:

[118]:
```
explainer = shap.Explainer(rf)
shap_values = explainer.shap_values(xTest)
shap.summary_plot(shap_values, xTest)
```



[119]:
```
explainer = lime.lime_tabular.LimeTabularExplainer(xTrain.values,
                                                    feature_names= xTrain.
 ↪columns.tolist(),
                                                    class_names=rf.classes_,
                                                    mode='classification')

sample_index = 0  # Choose any index from the test set
```

```
sample = xTest.iloc[[sample_index]]
true_label = yTest.iloc[sample_index]

explanation = explainer.explain_instance(sample.values[0],
                                         rf.predict_proba,
                                         num_features=len(xTrain.columns.
 ↪tolist()))

explanation.show_in_notebook()
```

```
/home/applehx7/anaconda3/lib/python3.11/site-packages/sklearn/base.py:439:
UserWarning:

X does not have valid feature names, but RandomForestClassifier was fitted with
feature names


<IPython.core.display.HTML object>
```

0.2 Conclusion

1. **Conclusion:**

    - In this project, our objective as a data scientist within a global finance company was to develop a machine learning model for predicting individuals' credit scores based on their financial and credit-related information. We embarked on a comprehensive process encompassing data acquisition, exploration, model selection, training, evaluation, and interpretability analysis.

2. **Data Exploration and Preprocessing:**

    - We started by acquiring a dataset containing relevant credit-related information, identifying key features such as income, outstanding debt, credit history, etc., and recognizing Credit_Score as the target variable. Through exploratory data analysis (EDA), we gained insights into feature distributions, handled missing values, outliers, and categorical variables, and explored the distribution of the target variable.

3. **Model Selection and Training:**

    - To build our predictive model, we selected several machine learning classification models suitable for credit score prediction. These included Logistic Regression, Random Forest Classifier, Support Vector Machine (SVM), and Gradient Boosting Classifier (e.g., XGBoost). Each model was trained using appropriate evaluation metrics such as accuracy, precision, recall, F1 score, and confusion matrix on the testing set.

4. **Hyperparameter Tuning and Model Evaluation:**

    - We conducted hyperparameter tuning for at least one model using methods like Grid Search or Random Search, aiming to optimize model performance. The chosen hyperparameters were reasoned based on their impact on model accuracy and generalization. We assessed and compared the models' performance on the testing set, highlighting their strengths and limitations concerning credit score classification.

5. **Interpretability:**

   - For interpretability, we explored LIME (Local Interpretable Model-agnostic Explanations) to understand the factors influencing credit score classifications. This allowed us to gain insights into how individual instances were being predicted by our models, aiding in understanding the models' decisions.

6. **Overall Insights:**

   - The models demonstrated varying degrees of performance and interpretability. For instance, Logistic Regression provided interpretability but might lack complexity for capturing nuanced relationships. Random Forest and Gradient Boosting exhibited higher accuracy but were relatively complex and computationally intensive. SVM showed versatility in handling non-linear relationships but might require careful tuning.

7. **Recommendations:**

   - In practical terms, the choice of model could depend on the trade-offs between interpretability, computational resources, and predictive performance. Logistic Regression might be suitable for interpretability, while Random Forest or Gradient Boosting could be preferred for higher accuracy, and SVM might be beneficial for handling non-linear relationships.

In summary, this project facilitated the development and evaluation of machine learning models for credit score prediction, providing insights into their strengths, limitations, and implications for real-world credit scoring systems.

[ ]:

```

```