Master Capstone Banking Project

October 6, 2024

1 Imported All the Libraries

[594]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2 Perform preliminary data inspection and report the findings like the structure of the data, missing values, duplicates, etc.

[596]:
```
df = pd.read_excel('Downloads/data.xlsx')
df.head()
```

[596]:
```
   UniqueID  disbursed_amount  asset_cost    ltv  branch_id  supplier_id  \
0    420825             50578       58400  89.55         67        22807
1    417566             53278       61360  89.63         67        22807
2    539055             52378       60300  88.39         67        22807
3    529269             46349       61500  76.42         67        22807
4    563215             43594       78256  57.50         67        22744

   manufacturer_id  Current_pincode_ID Date.of.Birth Employment.Type  …  \
0               45                1441    1984-01-01        Salaried  …
1               45                1497    1985-08-24   Self employed  …
2               45                1495    1977-12-09   Self employed  …
3               45                1502    1988-06-01        Salaried  …
4               86                1499    1994-07-14   Self employed  …

  SEC.SANCTIONED.AMOUNT  SEC.DISBURSED.AMOUNT  PRIMARY.INSTAL.AMT  \
0                     0                     0                   0
1                     0                     0                   0
2                     0                     0                   0
3                     0                     0                   0
4                     0                     0                   0

   SEC.INSTAL.AMT  NEW.ACCTS.IN.LAST.SIX.MONTHS  \
0               0                             0
```

```
1                0                              0
2                0                              0
3                0                              0
4                0                              0

   DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS  AVERAGE.ACCT.AGE  \
0                                    0         0yrs 0mon
1                                    0         0yrs 0mon
2                                    0         0yrs 0mon
3                                    0         0yrs 0mon
4                                    0         0yrs 0mon

   CREDIT.HISTORY.LENGTH  NO.OF_INQUIRIES  loan_default
0              0yrs 0mon                0             0
1              0yrs 0mon                0             0
2              0yrs 0mon                1             1
3              0yrs 0mon                0             0
4              0yrs 0mon                0             0

[5 rows x 41 columns]
```

[597]:
```python
null_unique = pd.DataFrame()
null_unique['nulls'] = pd.Series(df.isnull().sum())
null_unique['unique'] = pd.Series(df.nunique())
null_unique
```

[597]:

	nulls	unique
UniqueID	0	233154
disbursed_amount	0	24565
asset_cost	0	46252
ltv	0	6579
branch_id	0	82
supplier_id	0	2953
manufacturer_id	0	11
Current_pincode_ID	0	6698
Date.of.Birth	0	15433
Employment.Type	7661	2
DisbursalDate	0	84
State_ID	0	22
Employee_code_ID	0	3270
MobileNo_Avl_Flag	0	1
Aadhar_flag	0	2
PAN_flag	0	2
VoterID_flag	0	2
Driving_flag	0	2
Passport_flag	0	2
PERFORM_CNS.SCORE	0	573

```
PERFORM_CNS.SCORE.DESCRIPTION          0      20
PRI.NO.OF.ACCTS                        0     108
PRI.ACTIVE.ACCTS                       0      40
PRI.OVERDUE.ACCTS                      0      22
PRI.CURRENT.BALANCE                    0   71341
PRI.SANCTIONED.AMOUNT                  0   44390
PRI.DISBURSED.AMOUNT                   0   47909
SEC.NO.OF.ACCTS                        0      37
SEC.ACTIVE.ACCTS                       0      23
SEC.OVERDUE.ACCTS                      0       9
SEC.CURRENT.BALANCE                    0    3246
SEC.SANCTIONED.AMOUNT                  0    2223
SEC.DISBURSED.AMOUNT                   0    2553
PRIMARY.INSTAL.AMT                     0   28067
SEC.INSTAL.AMT                         0    1918
NEW.ACCTS.IN.LAST.SIX.MONTHS           0      26
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS    0      14
AVERAGE.ACCT.AGE                       0     192
CREDIT.HISTORY.LENGTH                  0     294
NO.OF_INQUIRIES                        0      25
loan_default                           0       2
```

[598]:
```
round((df.isnull().sum()/df.shape[0]*100),2)
```

[598]:
```
UniqueID                               0.00
disbursed_amount                       0.00
asset_cost                             0.00
ltv                                    0.00
branch_id                              0.00
supplier_id                            0.00
manufacturer_id                        0.00
Current_pincode_ID                     0.00
Date.of.Birth                          0.00
Employment.Type                        3.29
DisbursalDate                          0.00
State_ID                               0.00
Employee_code_ID                       0.00
MobileNo_Avl_Flag                      0.00
Aadhar_flag                            0.00
PAN_flag                               0.00
VoterID_flag                           0.00
Driving_flag                           0.00
Passport_flag                          0.00
PERFORM_CNS.SCORE                      0.00
PERFORM_CNS.SCORE.DESCRIPTION          0.00
PRI.NO.OF.ACCTS                        0.00
PRI.ACTIVE.ACCTS                       0.00
```

```
PRI.OVERDUE.ACCTS                      0.00
PRI.CURRENT.BALANCE                    0.00
PRI.SANCTIONED.AMOUNT                  0.00
PRI.DISBURSED.AMOUNT                   0.00
SEC.NO.OF.ACCTS                        0.00
SEC.ACTIVE.ACCTS                       0.00
SEC.OVERDUE.ACCTS                      0.00
SEC.CURRENT.BALANCE                    0.00
SEC.SANCTIONED.AMOUNT                  0.00
SEC.DISBURSED.AMOUNT                   0.00
PRIMARY.INSTAL.AMT                     0.00
SEC.INSTAL.AMT                         0.00
NEW.ACCTS.IN.LAST.SIX.MONTHS           0.00
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS    0.00
AVERAGE.ACCT.AGE                       0.00
CREDIT.HISTORY.LENGTH                  0.00
NO.OF_INQUIRIES                        0.00
loan_default                           0.00
dtype: float64
```

[599]:

```
df[df.duplicated()]
```

[599]:

```
Empty DataFrame
Columns: [UniqueID, disbursed_amount, asset_cost, ltv, branch_id, supplier_id,
manufacturer_id, Current_pincode_ID, Date.of.Birth, Employment.Type,
DisbursalDate, State_ID, Employee_code_ID, MobileNo_Avl_Flag, Aadhar_flag,
PAN_flag, VoterID_flag, Driving_flag, Passport_flag, PERFORM_CNS.SCORE,
PERFORM_CNS.SCORE.DESCRIPTION, PRI.NO.OF.ACCTS, PRI.ACTIVE.ACCTS,
PRI.OVERDUE.ACCTS, PRI.CURRENT.BALANCE, PRI.SANCTIONED.AMOUNT,
PRI.DISBURSED.AMOUNT, SEC.NO.OF.ACCTS, SEC.ACTIVE.ACCTS, SEC.OVERDUE.ACCTS,
SEC.CURRENT.BALANCE, SEC.SANCTIONED.AMOUNT, SEC.DISBURSED.AMOUNT,
PRIMARY.INSTAL.AMT, SEC.INSTAL.AMT, NEW.ACCTS.IN.LAST.SIX.MONTHS,
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS, AVERAGE.ACCT.AGE, CREDIT.HISTORY.LENGTH,
NO.OF_INQUIRIES, loan_default]
Index: []

[0 rows x 41 columns]
```

[600]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233154 entries, 0 to 233153
Data columns (total 41 columns):
 #   Column                               Non-Null Count   Dtype
---  ------                               --------------   -----
 0   UniqueID                             233154 non-null  int64
 1   disbursed_amount                     233154 non-null  int64
```

```
 2   asset_cost                          233154 non-null  int64
 3   ltv                                 233154 non-null  float64
 4   branch_id                           233154 non-null  int64
 5   supplier_id                         233154 non-null  int64
 6   manufacturer_id                     233154 non-null  int64
 7   Current_pincode_ID                  233154 non-null  int64
 8   Date.of.Birth                       233154 non-null  datetime64[ns]
 9   Employment.Type                     225493 non-null  object
 10  DisbursalDate                       233154 non-null  datetime64[ns]
 11  State_ID                            233154 non-null  int64
 12  Employee_code_ID                    233154 non-null  int64
 13  MobileNo_Avl_Flag                   233154 non-null  int64
 14  Aadhar_flag                         233154 non-null  int64
 15  PAN_flag                            233154 non-null  int64
 16  VoterID_flag                        233154 non-null  int64
 17  Driving_flag                        233154 non-null  int64
 18  Passport_flag                       233154 non-null  int64
 19  PERFORM_CNS.SCORE                   233154 non-null  int64
 20  PERFORM_CNS.SCORE.DESCRIPTION       233154 non-null  object
 21  PRI.NO.OF.ACCTS                     233154 non-null  int64
 22  PRI.ACTIVE.ACCTS                    233154 non-null  int64
 23  PRI.OVERDUE.ACCTS                   233154 non-null  int64
 24  PRI.CURRENT.BALANCE                 233154 non-null  int64
 25  PRI.SANCTIONED.AMOUNT               233154 non-null  int64
 26  PRI.DISBURSED.AMOUNT                233154 non-null  int64
 27  SEC.NO.OF.ACCTS                     233154 non-null  int64
 28  SEC.ACTIVE.ACCTS                    233154 non-null  int64
 29  SEC.OVERDUE.ACCTS                   233154 non-null  int64
 30  SEC.CURRENT.BALANCE                 233154 non-null  int64
 31  SEC.SANCTIONED.AMOUNT               233154 non-null  int64
 32  SEC.DISBURSED.AMOUNT                233154 non-null  int64
 33  PRIMARY.INSTAL.AMT                  233154 non-null  int64
 34  SEC.INSTAL.AMT                      233154 non-null  int64
 35  NEW.ACCTS.IN.LAST.SIX.MONTHS        233154 non-null  int64
 36  DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS 233154 non-null  int64
 37  AVERAGE.ACCT.AGE                    233154 non-null  object
 38  CREDIT.HISTORY.LENGTH               233154 non-null  object
 39  NO.OF_INQUIRIES                     233154 non-null  int64
 40  loan_default                        233154 non-null  int64
dtypes: datetime64[ns](2), float64(1), int64(34), object(4)
memory usage: 72.9+ MB
```

3 Variable names in the data may not be in accordance with the identifier naming in Python. Change the variable names accordingly.

[601]:
```
def to_valid_symbol(name):
    return name.lower().replace('.','_').replace(' ','_')

df.columns = [to_valid_symbol(col) for col in df.columns]
df.head()
```

[601]:
```
   uniqueid  disbursed_amount  asset_cost    ltv  branch_id  supplier_id  \
0    420825             50578       58400  89.55         67        22807
1    417566             53278       61360  89.63         67        22807
2    539055             52378       60300  88.39         67        22807
3    529269             46349       61500  76.42         67        22807
4    563215             43594       78256  57.50         67        22744

   manufacturer_id  current_pincode_id date_of_birth employment_type  …  \
0               45                1441    1984-01-01        Salaried  …
1               45                1497    1985-08-24   Self employed  …
2               45                1495    1977-12-09   Self employed  …
3               45                1502    1988-06-01        Salaried  …
4               86                1499    1994-07-14   Self employed  …

  sec_sanctioned_amount  sec_disbursed_amount  primary_instal_amt  \
0                     0                     0                   0
1                     0                     0                   0
2                     0                     0                   0
3                     0                     0                   0
4                     0                     0                   0

   sec_instal_amt  new_accts_in_last_six_months  \
0               0                             0
1               0                             0
2               0                             0
3               0                             0
4               0                             0

   delinquent_accts_in_last_six_months  average_acct_age  \
0                                    0         0yrs 0mon
1                                    0         0yrs 0mon
2                                    0         0yrs 0mon
3                                    0         0yrs 0mon
4                                    0         0yrs 0mon

   credit_history_length  no_of_inquiries  loan_default
```

```
0             0yrs 0mon            0             0
1             0yrs 0mon            0             0
2             0yrs 0mon            1             1
3             0yrs 0mon            0             0
4             0yrs 0mon            0             0

[5 rows x 41 columns]
```

4 The presented data might also contain missing values, therefore, exploration will also lead to devising strategies to fill in the missing values. Devise strategies while exploring the data

[602]:
```
numerical_col = df.select_dtypes(include =['int64', 'float64']).columns
categorical_col = df.select_dtypes(include =['object']).columns

df[numerical_col] = df[numerical_col].fillna(df[numerical_col].mean())
df[categorical_col] = df[categorical_col].fillna(df[categorical_col].mode().
 ↪iloc[0])

print(df.isnull().sum())
```

```
uniqueid                               0
disbursed_amount                       0
asset_cost                             0
ltv                                    0
branch_id                              0
supplier_id                            0
manufacturer_id                        0
current_pincode_id                     0
date_of_birth                          0
employment_type                        0
disbursaldate                          0
state_id                               0
employee_code_id                       0
mobileno_avl_flag                      0
aadhar_flag                            0
pan_flag                               0
voterid_flag                           0
driving_flag                           0
passport_flag                          0
perform_cns_score                      0
perform_cns_score_description          0
pri_no_of_accts                        0
pri_active_accts                       0
pri_overdue_accts                      0
pri_current_balance                    0
```

```
pri_sanctioned_amount                  0
pri_disbursed_amount                   0
sec_no_of_accts                        0
sec_active_accts                       0
sec_overdue_accts                      0
sec_current_balance                    0
sec_sanctioned_amount                  0
sec_disbursed_amount                   0
primary_instal_amt                     0
sec_instal_amt                         0
new_accts_in_last_six_months           0
delinquent_accts_in_last_six_months    0
average_acct_age                       0
credit_history_length                  0
no_of_inquiries                        0
loan_default                           0
dtype: int64
```

5 Provide the statistical description of the quantitative data variables

[603]:
```
df.describe()
```

[603]:
```
            uniqueid  disbursed_amount    asset_cost             ltv  \
count  233154.000000     233154.000000  2.331540e+05  233154.000000
mean   535917.573376      54356.993528  7.586507e+04      74.746530
min    417428.000000      13320.000000  3.700000e+04      10.030000
25%    476786.250000      47145.000000  6.571700e+04      68.880000
50%    535978.500000      53803.000000  7.094600e+04      76.800000
75%    595039.750000      60413.000000  7.920175e+04      83.670000
max    671084.000000     990572.000000  1.628992e+06      95.000000
std     68315.693711      12971.314171  1.894478e+04      11.456636

           branch_id    supplier_id  manufacturer_id  current_pincode_id  \
count  233154.000000  233154.000000    233154.000000       233154.000000
mean       72.936094   19638.635035        69.028054         3396.880247
min         1.000000   10524.000000        45.000000            1.000000
25%        14.000000   16535.000000        48.000000         1511.000000
50%        61.000000   20333.000000        86.000000         2970.000000
75%       130.000000   23000.000000        86.000000         5677.000000
max       261.000000   24803.000000       156.000000         7345.000000
std        69.834995    3491.949566        22.141304         2238.147502

                       date_of_birth                 disbursaldate  …  \
count                         233154                        233154  …
mean   1984-04-04 04:32:39.947502400  2018-09-23 09:57:53.079595520  …
```

```
min             1949-09-15 00:00:00         2018-08-01 00:00:00  …
25%             1977-05-04 00:00:00         2018-08-30 00:00:00  …
50%             1986-01-01 00:00:00         2018-09-25 00:00:00  …
75%             1992-05-19 00:00:00         2018-10-21 00:00:00  …
max             2000-10-20 00:00:00         2018-10-31 00:00:00  …
std                             NaN                         NaN  …

       sec_overdue_accts  sec_current_balance  sec_sanctioned_amount  \
count      233154.000000         2.331540e+05           2.331540e+05
mean            0.007244         5.427793e+03           7.295923e+03
min             0.000000        -5.746470e+05           0.000000e+00
25%             0.000000         0.000000e+00           0.000000e+00
50%             0.000000         0.000000e+00           0.000000e+00
75%             0.000000         0.000000e+00           0.000000e+00
max             8.000000         3.603285e+07           3.000000e+07
std             0.111079         1.702370e+05           1.831560e+05

       sec_disbursed_amount  primary_instal_amt  sec_instal_amt  \
count          2.331540e+05        2.331540e+05    2.331540e+05
mean           7.179998e+03        1.310548e+04    3.232684e+02
min            0.000000e+00        0.000000e+00    0.000000e+00
25%            0.000000e+00        0.000000e+00    0.000000e+00
50%            0.000000e+00        0.000000e+00    0.000000e+00
75%            0.000000e+00        1.999000e+03    0.000000e+00
max            3.000000e+07        2.564281e+07    4.170901e+06
std            1.825925e+05        1.513679e+05    1.555369e+04

       new_accts_in_last_six_months  delinquent_accts_in_last_six_months  \
count                 233154.000000                        233154.000000
mean                       0.381833                             0.097481
min                        0.000000                             0.000000
25%                        0.000000                             0.000000
50%                        0.000000                             0.000000
75%                        0.000000                             0.000000
max                       35.000000                            20.000000
std                        0.955107                             0.384439

       no_of_inquiries   loan_default
count    233154.000000  233154.000000
mean          0.206615       0.217071
min           0.000000       0.000000
25%           0.000000       0.000000
50%           0.000000       0.000000
75%           0.000000       0.000000
max          36.000000       1.000000
std           0.706498       0.412252
```

```
[8 rows x 37 columns]
```

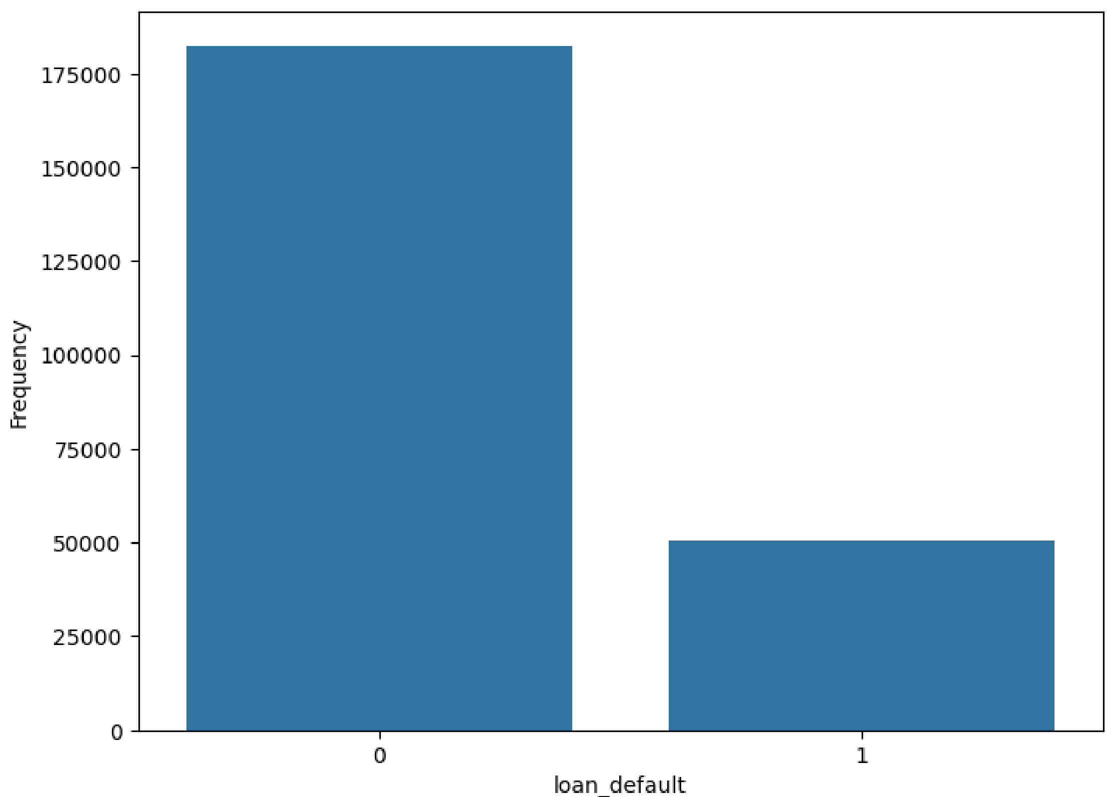
[604]:
```
target_distribution = df['loan_default'].value_counts()
target_distribution
```

[604]:
```
loan_default
0    182543
1     50611
Name: count, dtype: int64
```

6 How is the target variable distributed overall?

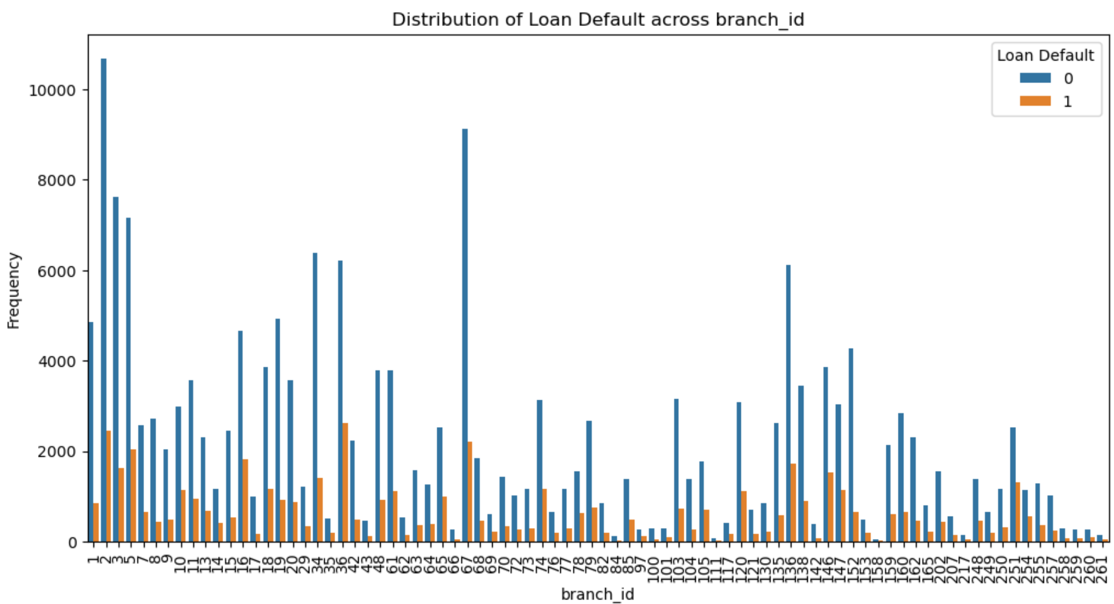
[605]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

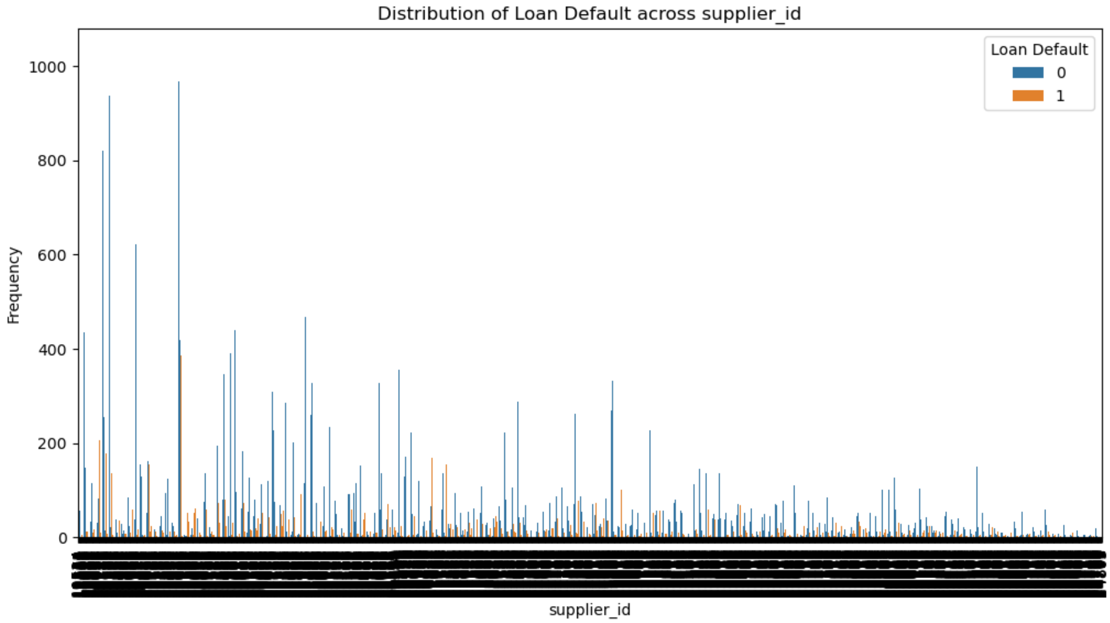
[606]:
```
plt.figure(figsize =(8,6))
sns.countplot(x = 'loan_default', data = df)
plt.xlabel('loan_default')
plt.ylabel('Frequency')
plt.show()
```

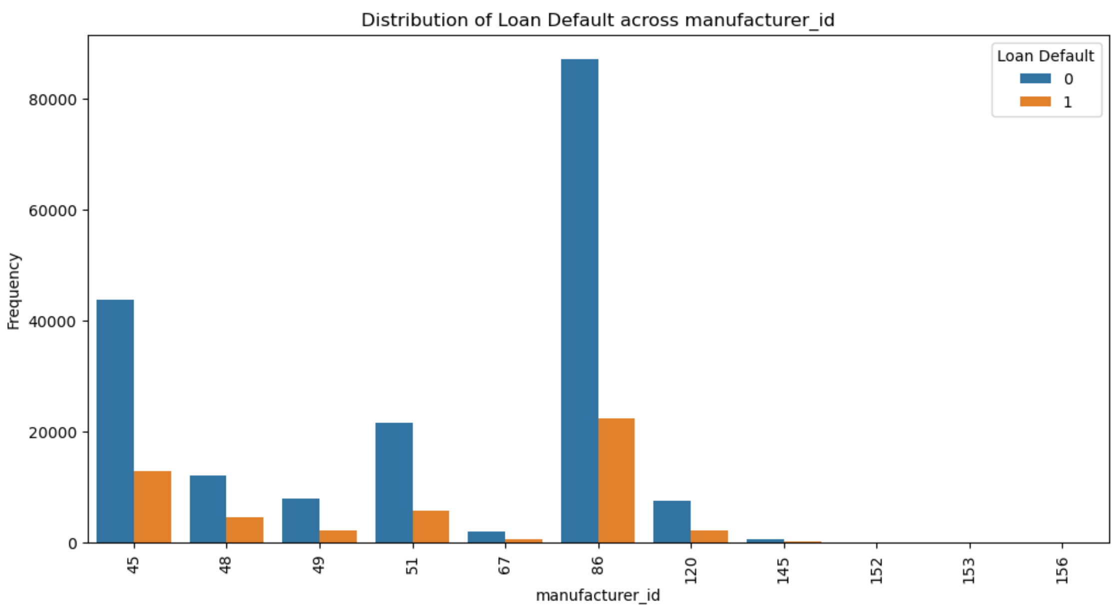


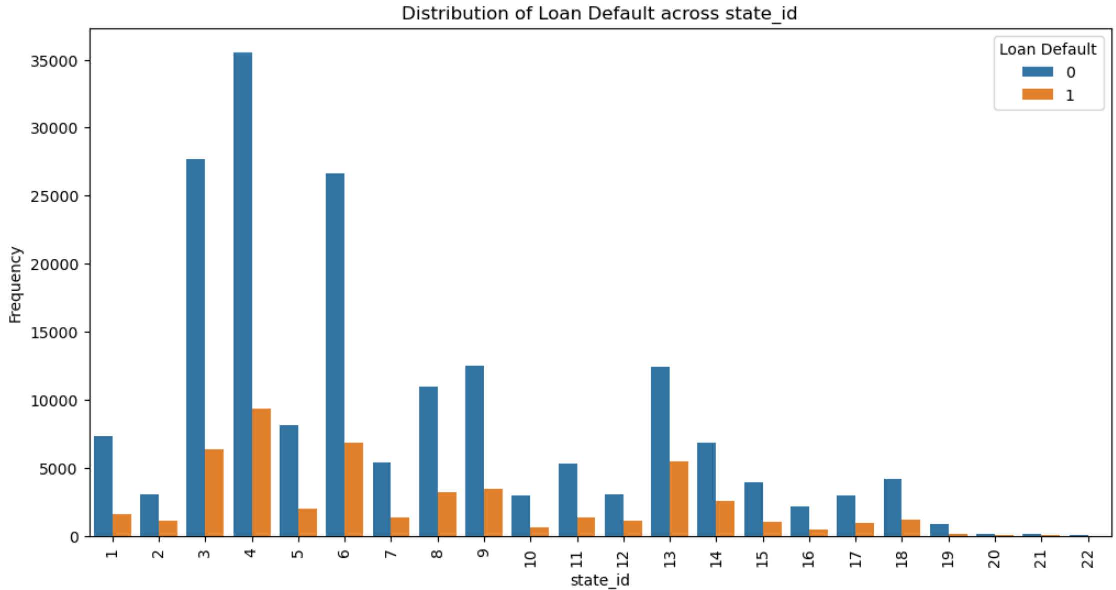
7 Study the distribution of the target variable across the various categories like branch, city, state, branch, supplier, manufacturer, etc.

[608]:
```python
categorical_columns = ['branch_id', 'supplier_id', 'manufacturer_id', 'state_id']
for col in categorical_columns:
    plt.figure(figsize =(12,6))
    sns.countplot(data =df, x =col, hue ='loan_default')
    plt.title(f'Distribution of Loan Default across {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.legend(title ='Loan Default', loc ='upper right')
    plt.xticks(rotation =90)
    plt.show()
```



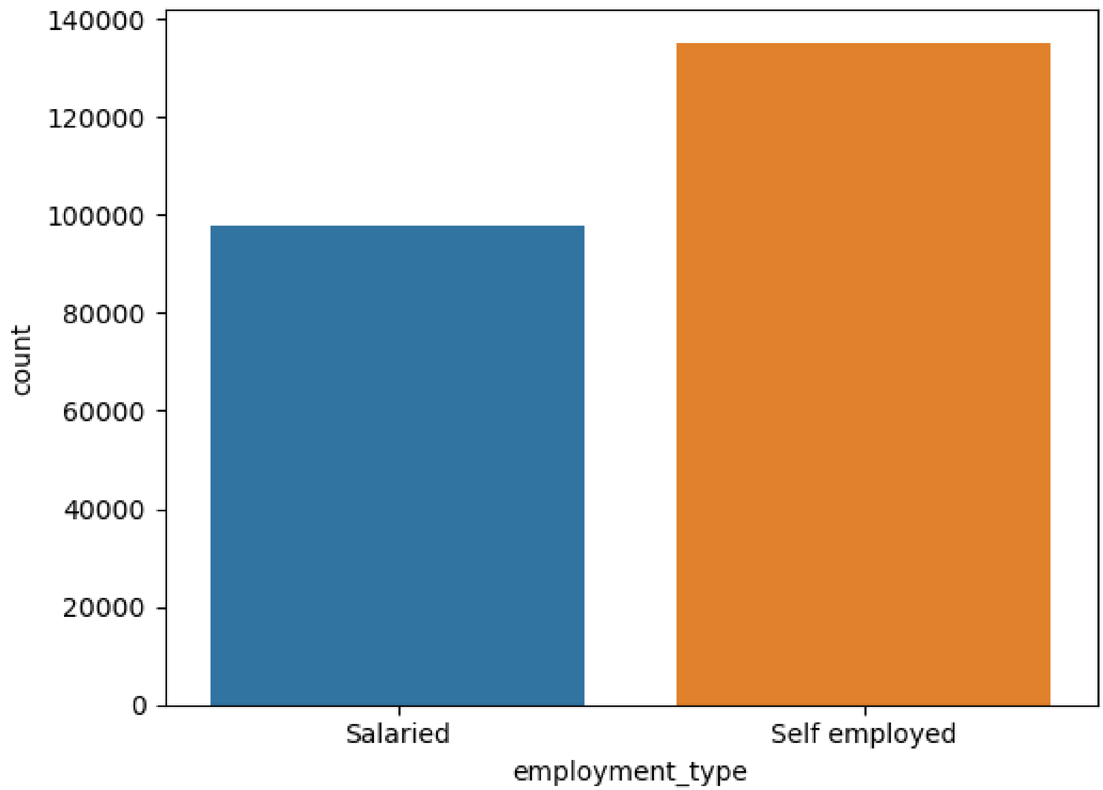

Distribution of Loan Default across supplier_id
Loan Default
0
1
Frequency
1000
800
600
400
200
0
supplier_id


Distribution of Loan Default across manufacturer_id
Loan Default
0
1
Frequency
80000
60000
40000
20000
0
45
48
49
51
67
86
120
145
152
153
156
manufacturer_id



8 What are the different employment types given in the data? Can a strategy be developed to fill in the missing values (if any)? Use pie charts to express how different types of employment defines defaulter and non-defaulters

[610]:
```
sns.countplot(x='employment_type', data = df, hue ='employment_type')
plt.show()
```



[611]:
```
missing_values =df['employment_type'].isnull().sum()
missing_values
```

[611]: 0

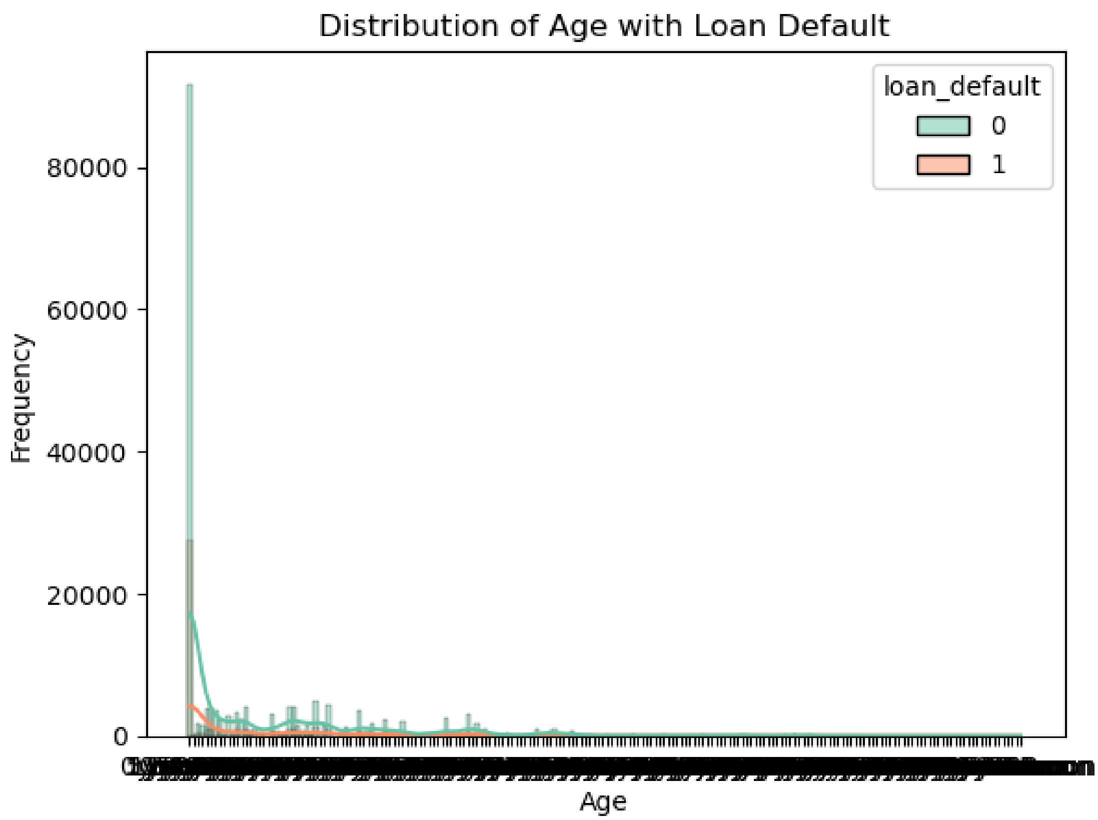
[896]:
```
mode_employement_type =df['employment_type'].mode()[0]
df['employment_type'].fillna(mode_employement_type, inplace =True)
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_7952\3903509892.py:2: FutureWarning: A
value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df['employment_type'].fillna(mode_employement_type, inplace =True)
```

[898]:
```
def plot_pie_charts(data, column, target):
    non_defaulters = data[data[target] == 0][column].value_counts()
    defaulters = data[data[target] == 1][column].value_counts()

    fig, ax = plt.subplots(1, 2, figsize=(14, 7))

    ax[0].pie(non_defaulters, labels=non_defaulters.index, autopct='%1.1f%%',␣
↪startangle=140)
    ax[0].set_title('Non-Defaulters by Employment Type')

    ax[1].pie(defaulters, labels=defaulters.index, autopct='%1.1f%%',␣
↪startangle=140)
    ax[1].set_title('Defaulters by Employment Type')

    plt.show()
```

[614]:
```
import seaborn as sns
```

[615]:
```
sns.histplot(data = df, x ='average_acct_age', hue= 'loan_default', kde␣
↪='True', palette ='Set2')
plt.xlabel ('Age')
plt.ylabel ('Frequency')
plt.title('Distribution of Age with Loan Default')
plt.show()
```



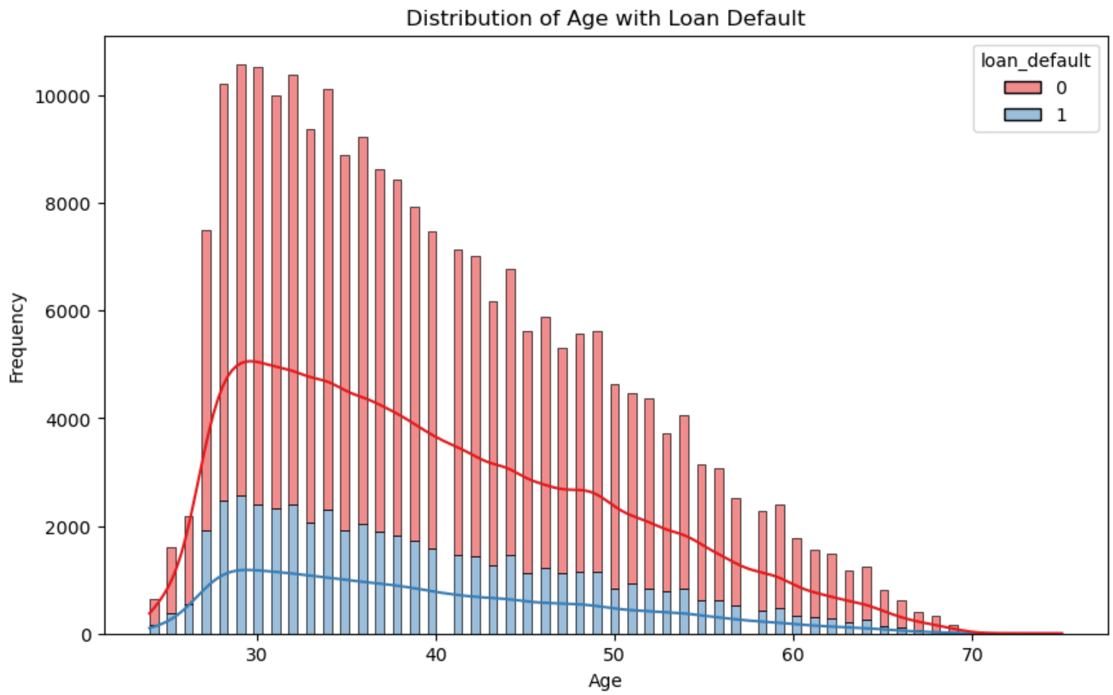
9 Has age got something to do with defaulting? What is the distribution of age w.r.t. to defaulters and non-defaulters?

from datetime import datetime

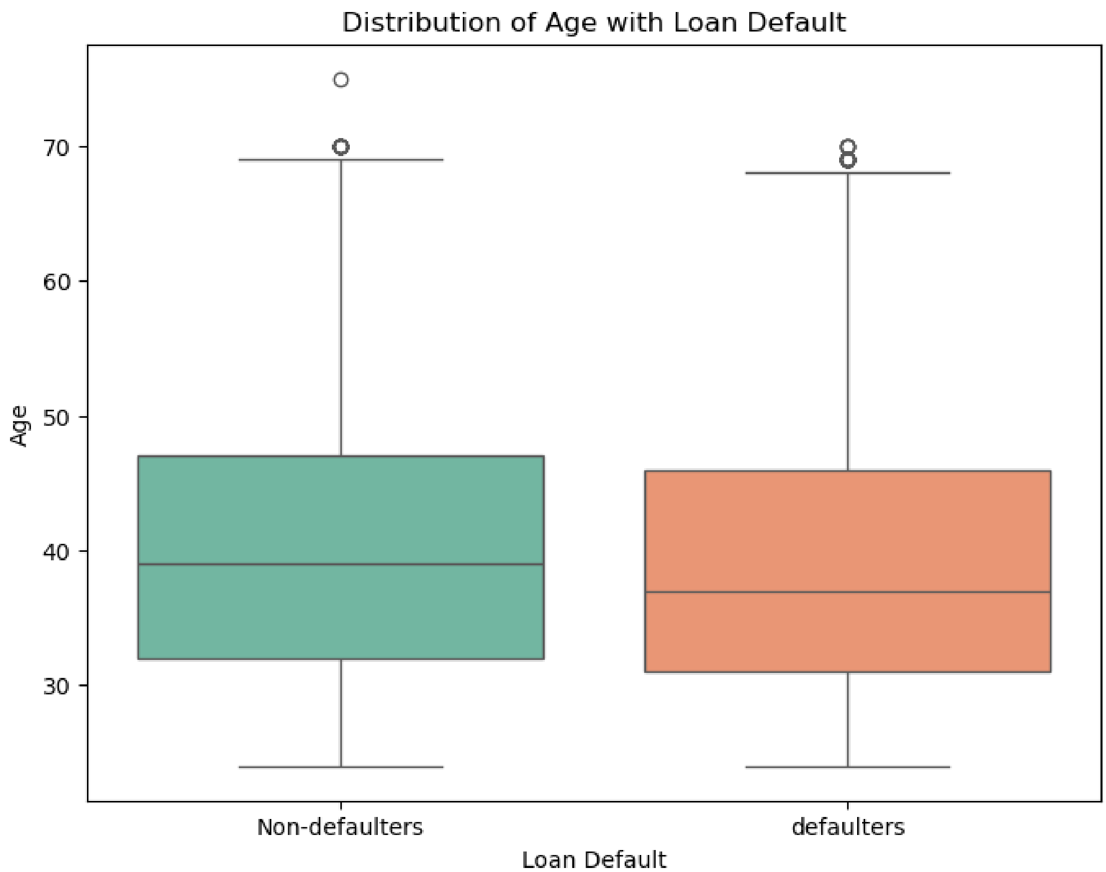
[617]:
```
current_year = datetime.now().year
df['date_of_birth'] = pd.to_datetime(df['date_of_birth'])
df['age'] =current_year-df['date_of_birth'].dt.year
```

[618]:
```
# Histogram
plt.figure(figsize=(10,6))
sns.histplot(data = df, x ='age', hue= 'loan_default', kde ='True', palette␣
↪='Set1', multiple ='stack')
plt.xlabel ('Age')
plt.ylabel ('Frequency')
plt.title('Distribution of Age with Loan Default')
plt.show()

# Boxplot
plt.figure(figsize=(8,6))
```

```python
sns.boxplot(data = df, x ='loan_default', y= 'age',  palette ='Set2')
plt.xlabel ('Loan Default')
plt.ylabel ('Age')
plt.xticks ([0,1], ['Non-defaulters', 'defaulters'])
plt.title('Distribution of Age with Loan Default')
plt.show()
```




```
C:\Users\HP\AppData\Local\Temp\ipykernel_7952\145288895.py:11: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(data = df, x ='loan_default', y= 'age',  palette ='Set2')
```



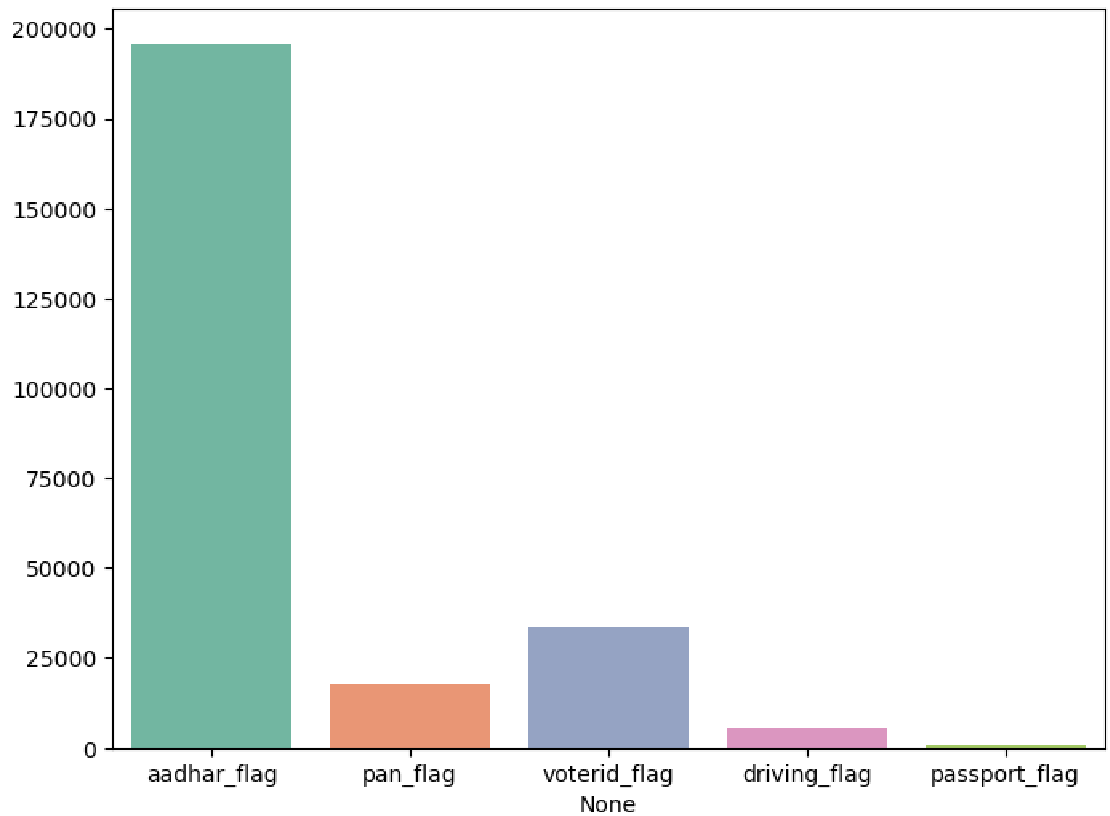
10 What type of ID is presented by most of the customers as proofs?

[620]:
```
id_proof_columns =␣
↪['aadhar_flag','pan_flag','voterid_flag','driving_flag','passport_flag']
id_proof_counts =df[id_proof_columns].sum()
id_proof_counts
```

[620]:
```
aadhar_flag      195924
pan_flag          17621
voterid_flag      33794
driving_flag       5419
passport_flag       496
dtype: int64
```

[621]:
```
plt.figure(figsize=(8,6))
sns.barplot(x =id_proof_counts.index, y=id_proof_counts.values, palette='Set2')
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_7952\2604259343.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.barplot(x =id_proof_counts.index, y=id_proof_counts.values,
palette='Set2')
```



11 Study the credit bureau score distribution. How is the distribution for defaulters vs. non-defaulters? Explore in detail.

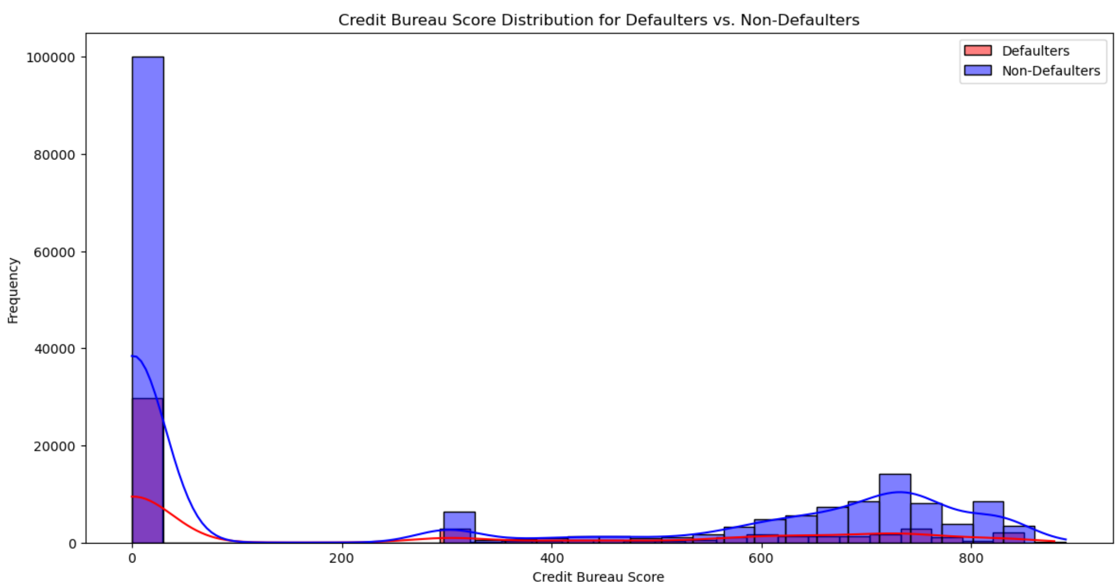
[623]:
```python
unique_scores = df['perform_cns_score'].unique()
unique_scores
```

[623]:
```
array([  0, 890, 884, 879, 878, 873, 870, 869, 868, 867, 864, 863, 862,
       859, 858, 855, 853, 852, 850, 849, 847, 845, 844, 843, 842, 841,
       840, 839, 838, 837, 836, 835, 834, 833, 832, 831, 830, 829, 828,
       827, 826, 825, 824, 823, 822, 821, 820, 819, 818, 817, 816, 815,
       814, 813, 812, 811, 810, 809, 808, 807, 806, 805, 804, 803, 802,
```

```
       801, 800, 799, 798, 797, 796, 795, 794, 793, 792, 791, 790, 789,
       788, 787, 786, 785, 784, 783, 782, 781, 780, 779, 778, 777, 776,
       775, 774, 773, 772, 771, 770, 769, 768, 767, 766, 765, 764, 763,
       762, 761, 760, 759, 758, 757, 756, 755, 754, 753, 752, 751, 750,
       749, 748, 747, 746, 745, 744, 743, 742, 741, 740, 739, 738, 737,
       736, 735, 734, 733, 732, 731, 730, 729, 728, 727, 726, 725, 724,
       723, 722, 721, 720, 719, 718, 717, 716, 715, 714, 713, 712, 711,
       710, 709, 708, 707, 706, 705, 704, 703, 702, 701, 700, 699, 698,
       697, 696, 695, 694, 693, 692, 691, 690, 689, 688, 687, 686, 685,
       684, 683, 682, 681, 680, 679, 678, 677, 676, 675, 674, 673, 672,
       671, 670, 669, 668, 667, 666, 665, 664, 663, 662, 661, 660, 659,
       658, 657, 656, 655, 654, 653, 652, 651, 650, 649, 648, 647, 646,
       645, 644, 643, 642, 641, 640, 639, 638, 637, 636, 635, 634, 633,
       632, 631, 630, 629, 628, 627, 626, 625, 624, 623, 622, 621, 620,
       619, 618, 617, 616, 615, 614, 613, 612, 611, 610, 609, 608, 607,
       606, 605, 604, 603, 602, 601, 600, 599, 598, 597, 596, 595, 594,
       593, 592, 591, 590, 589, 588, 587, 586, 585, 584, 583, 582, 581,
       580, 579, 578, 577, 576, 575, 574, 573, 572, 571, 570, 569, 568,
       567, 566, 565, 564, 563, 562, 561, 560, 559, 558, 557, 556, 555,
       554, 553, 552, 551, 550, 549, 548, 547, 546, 545, 544, 543, 542,
       541, 540, 539, 538, 537, 536, 535, 534, 533, 532, 531, 530, 529,
       528, 527, 526, 525, 524, 523, 522, 521, 520, 519, 518, 517, 516,
       515, 514, 513, 512, 511, 510, 509, 508, 507, 506, 505, 504, 503,
       502, 501, 500, 499, 498, 497, 496, 495, 494, 493, 492, 491, 490,
       489, 488, 487, 486, 485, 484, 483, 482, 481, 480, 479, 478, 477,
       476, 475, 474, 473, 472, 471, 470, 469, 468, 467, 466, 465, 464,
       463, 462, 461, 460, 459, 458, 457, 456, 455, 454, 453, 452, 451,
       450, 449, 448, 447, 446, 445, 444, 443, 442, 441, 440, 439, 438,
       437, 436, 435, 434, 433, 432, 431, 430, 429, 428, 427, 426, 425,
       424, 423, 422, 421, 420, 419, 418, 417, 416, 415, 414, 413, 412,
       411, 410, 409, 408, 407, 406, 405, 404, 403, 402, 401, 400, 399,
       398, 397, 396, 395, 394, 393, 392, 391, 390, 389, 388, 387, 386,
       385, 384, 383, 382, 381, 380, 379, 378, 377, 376, 375, 374, 373,
       372, 371, 370, 369, 368, 367, 366, 365, 364, 363, 362, 361, 360,
       359, 358, 357, 356, 355, 354, 353, 352, 351, 350, 349, 348, 347,
       346, 345, 344, 343, 342, 341, 340, 339, 338, 337, 336, 335, 334,
       333, 332, 331, 330, 329, 328, 327, 326, 325, 324, 323, 322, 321,
       320, 319, 318, 317, 316, 315, 314, 313, 312, 311, 310, 309, 308,
       307, 306, 305, 304, 303, 302, 301, 300,  18,  17,  16,  15,  14,
        11], dtype=int64)
```

[624]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Check unique values in the 'PERFORM_CNS.SCORE' column
unique_scores = df['perform_cns_score'].unique()
```

```
#print("Unique values in 'PERFORM_CNS.SCORE':", unique_scores)

# If there are valid scores, proceed with the analysis
if len(unique_scores) > 1 or unique_scores[0] != 0:
    # Separate the data into defaulters and non-defaulters
    defaulters = df[df['loan_default'] == 1]
    non_defaulters = df[df['loan_default'] == 0]

    # Set up the matplotlib figure
    plt.figure(figsize=(14, 7))

    # Plot the distribution of credit bureau scores for defaulters
    sns.histplot(defaulters['perform_cns_score'], kde=True, color='red',␣
 ↪label='Defaulters', bins=30)

    # Plot the distribution of credit bureau scores for non-defaulters
    sns.histplot(non_defaulters['perform_cns_score'], kde=True, color='blue',␣
 ↪label='Non-Defaulters', bins=30)

    # Add labels and title
    plt.title('Credit Bureau Score Distribution for Defaulters vs.␣
 ↪Non-Defaulters')
    plt.xlabel('Credit Bureau Score')
    plt.ylabel('Frequency')
    plt.legend()

    # Show the plot
    plt.show()
```



12 Explore the primary and secondary account details. Is the information in some way related to the loan default probability?

[625]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
```

[626]:
```
primary_features =['pri_no_of_accts', 'pri_active_accts', 'pri_overdue_accts',␣
↪'pri_current_balance', 'pri_sanctioned_amount', 'pri_disbursed_amount']
secondary_features =␣
↪['sec_no_of_accts','sec_active_accts','sec_overdue_accts','sec_current_balance','sec_sancti

primary_stats =df[primary_features].describe()
seconadry_stats =df[secondary_features].describe()
```

[627]:
```
primary_stats
```

[627]:
```
       pri_no_of_accts  pri_active_accts  pri_overdue_accts  \
count    233154.000000     233154.000000      233154.000000
mean          2.440636          1.039896           0.156549
std           5.217233          1.941496           0.548787
min           0.000000          0.000000           0.000000
25%           0.000000          0.000000           0.000000
50%           0.000000          0.000000           0.000000
75%           3.000000          1.000000           0.000000
max         453.000000        144.000000          25.000000

       pri_current_balance  pri_sanctioned_amount  pri_disbursed_amount
count         2.331540e+05           2.331540e+05          2.331540e+05
mean          1.659001e+05           2.185039e+05          2.180659e+05
std           9.422736e+05           2.374794e+06          2.377744e+06
min          -6.678296e+06           0.000000e+00          0.000000e+00
25%           0.000000e+00           0.000000e+00          0.000000e+00
50%           0.000000e+00           0.000000e+00          0.000000e+00
75%           3.500650e+04           6.250000e+04          6.080000e+04
max           9.652492e+07           1.000000e+09          1.000000e+09
```

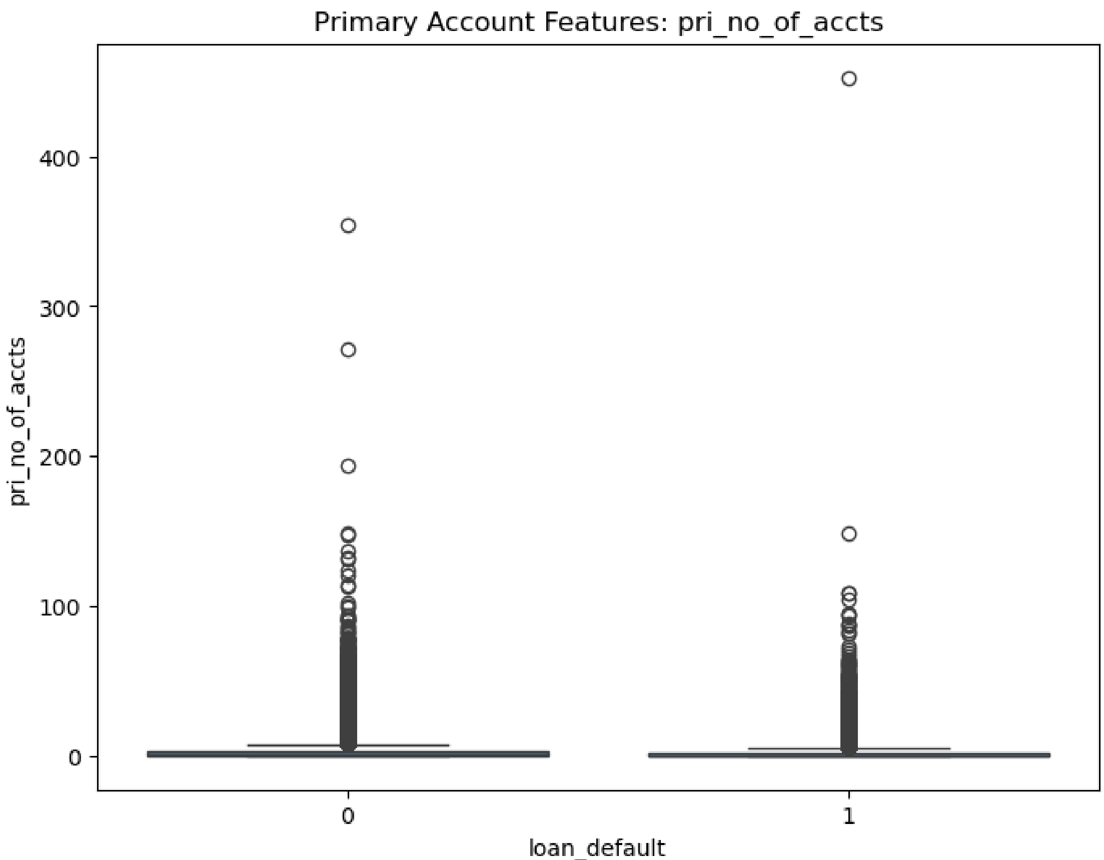
[628]:
```
seconadry_stats
```

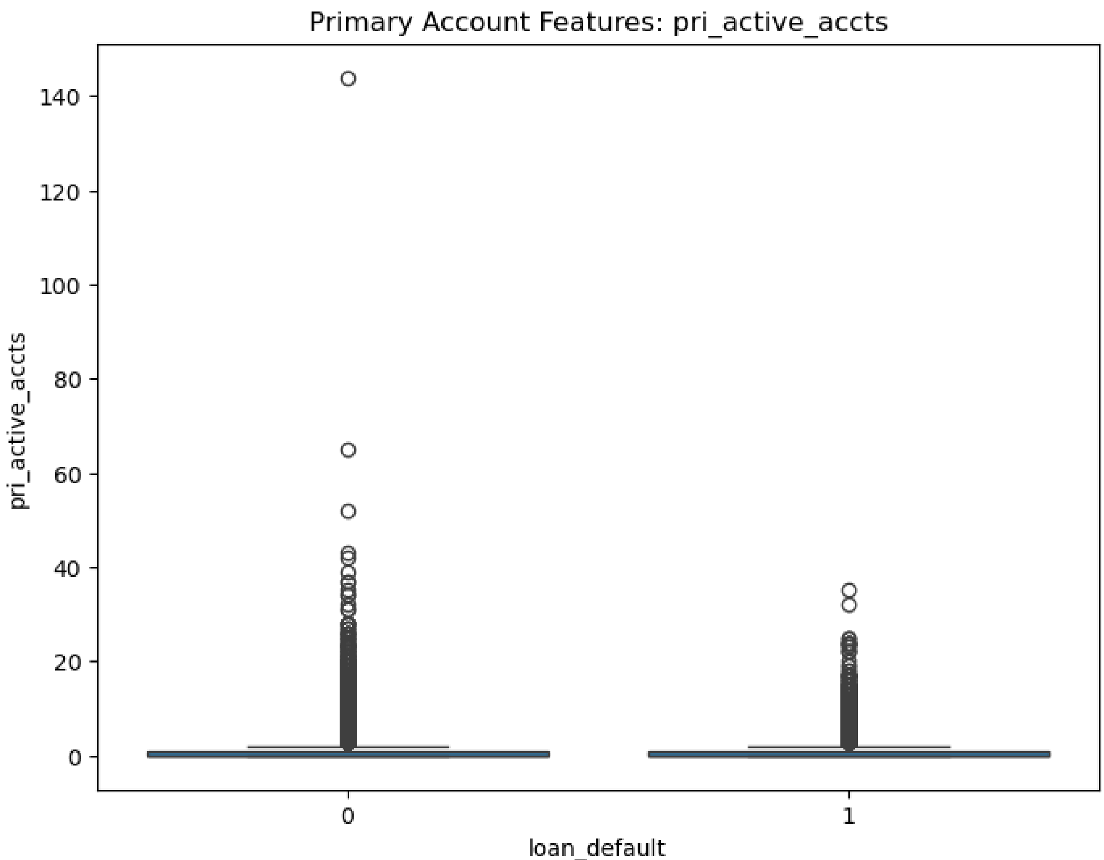
[628]:
```
       sec_no_of_accts  sec_active_accts  sec_overdue_accts  \
count    233154.000000     233154.000000      233154.000000
mean          0.059081          0.027703           0.007244
std           0.626795          0.316057           0.111079
```

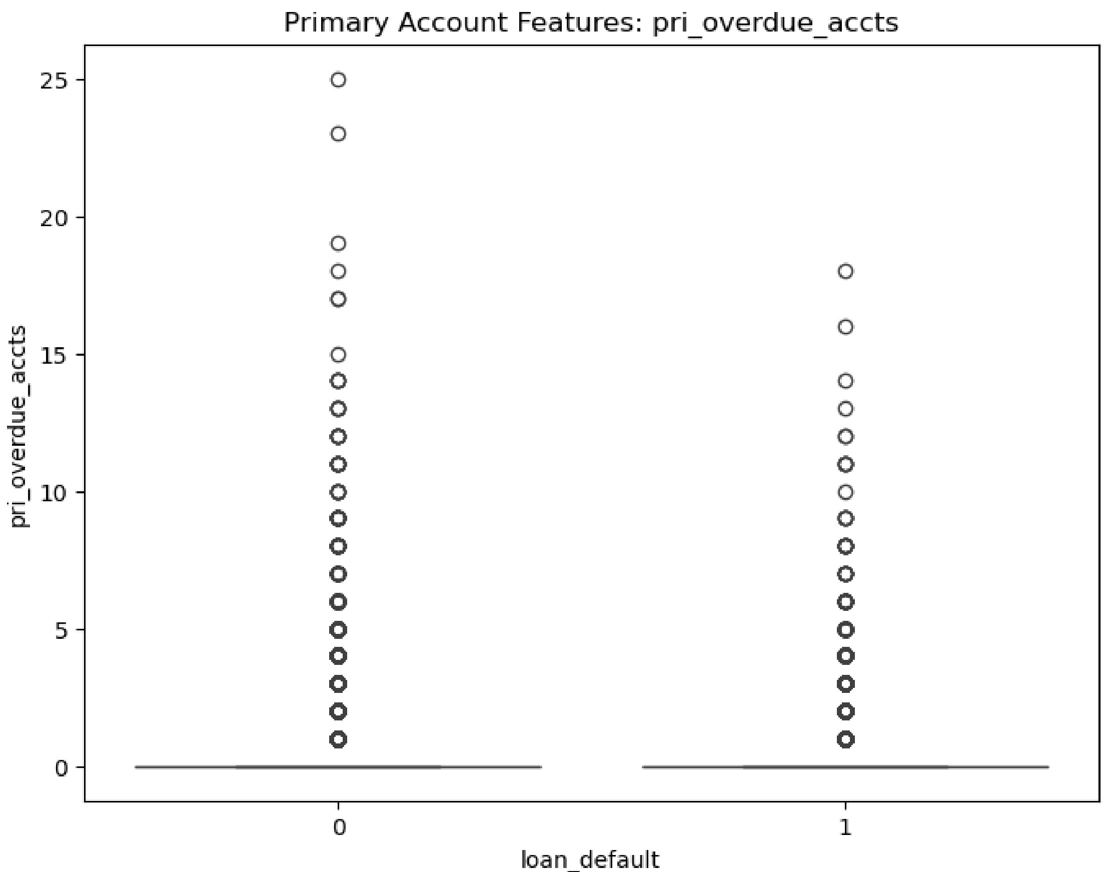
```
min          0.000000          0.000000          0.000000
25%          0.000000          0.000000          0.000000
50%          0.000000          0.000000          0.000000
75%          0.000000          0.000000          0.000000
max         52.000000         36.000000          8.000000

       sec_current_balance  sec_sanctioned_amount  sec_disbursed_amount
count         2.331540e+05           2.331540e+05          2.331540e+05
mean          5.427793e+03           7.295923e+03          7.179998e+03
std           1.702370e+05           1.831560e+05          1.825925e+05
min          -5.746470e+05           0.000000e+00          0.000000e+00
25%           0.000000e+00           0.000000e+00          0.000000e+00
50%           0.000000e+00           0.000000e+00          0.000000e+00
75%           0.000000e+00           0.000000e+00          0.000000e+00
max           3.603285e+07           3.000000e+07          3.000000e+07
```

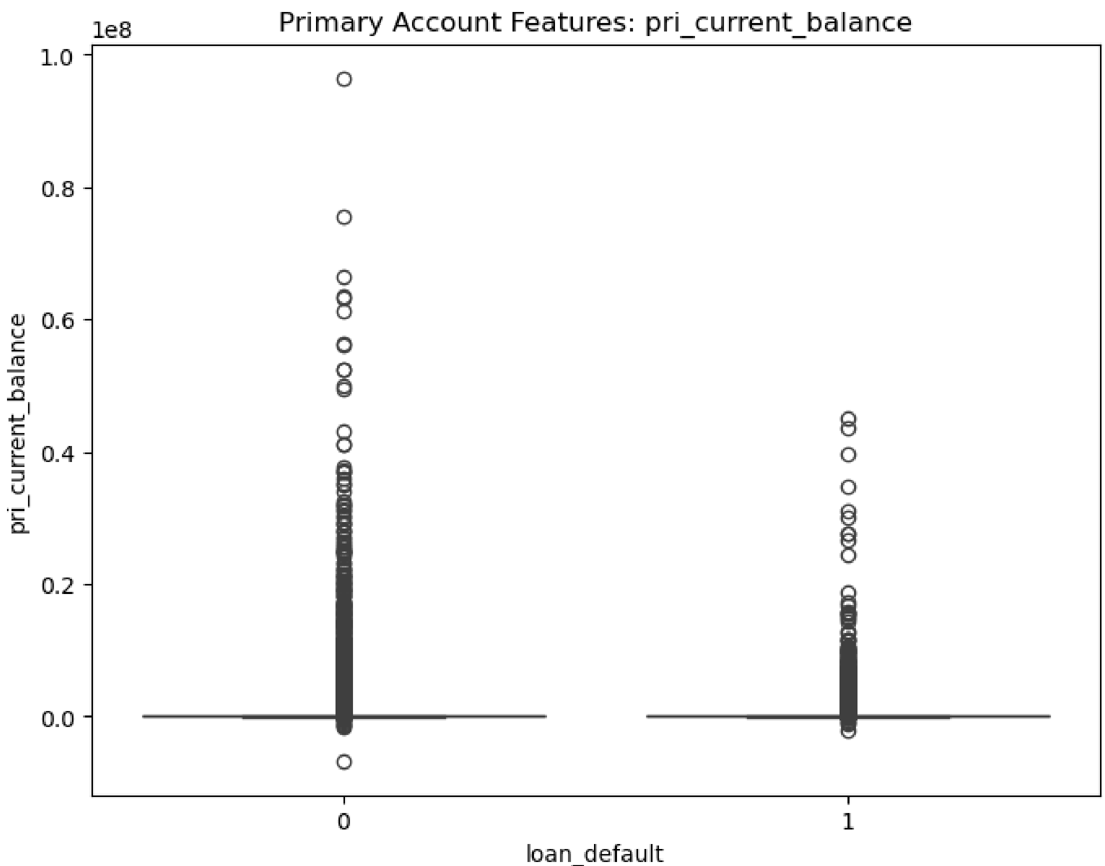
[629]:

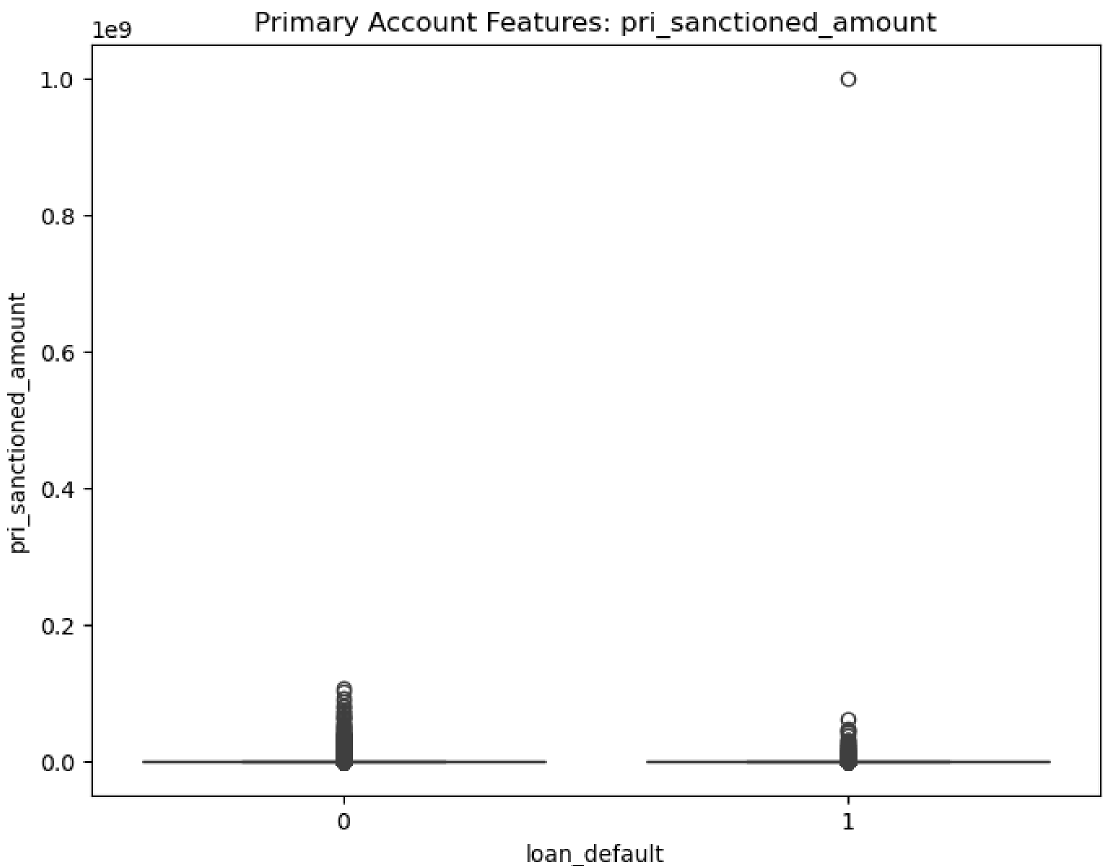
```
def plot_features(features, target, title):
    for feature in features:
        plt.figure (figsize=(8,6))
        sns.boxplot(data = df, x= target, y =feature)
        plt.title (f'{title}: {feature}')
        plt.show()

plot_features (primary_features,'loan_default', 'Primary Account Features')
plot_features (secondary_features, 'loan_default', 'Secondary Account Features')
```

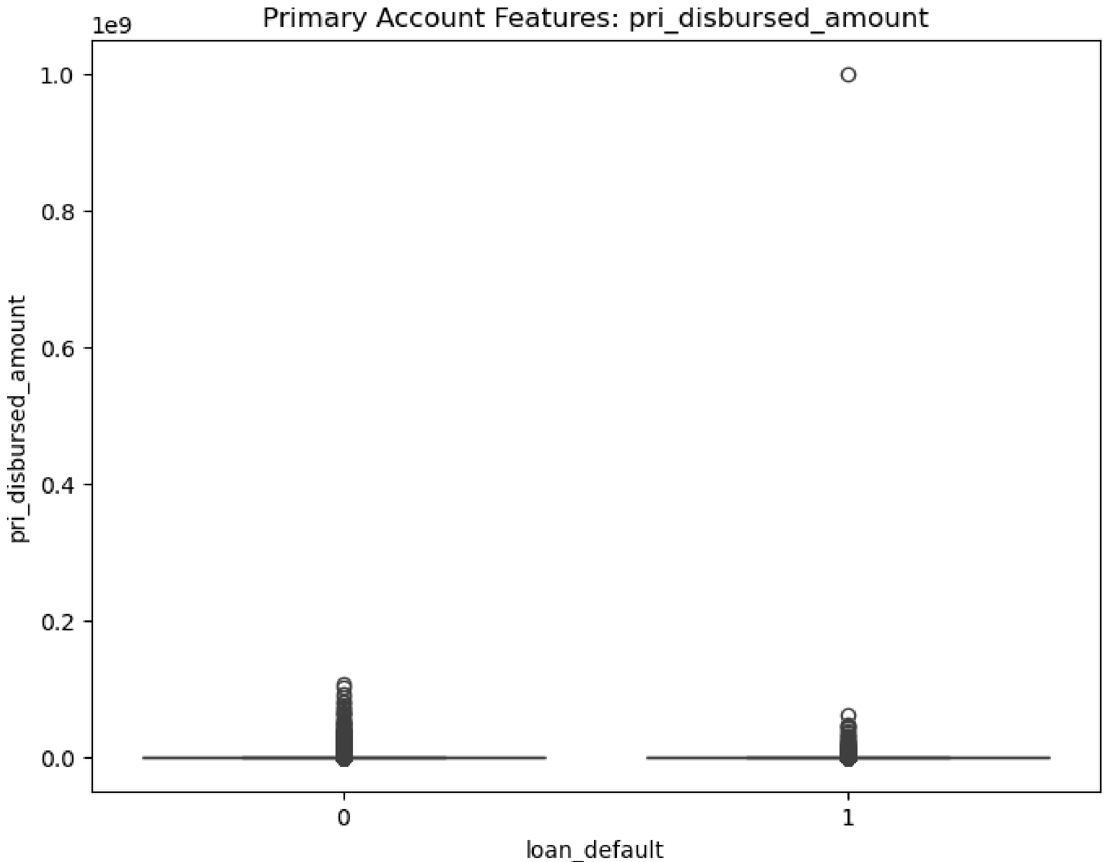

Primary Account Features: pri_no_of_accts
pri_no_of_accts
400
300
200
100
0
0
1
loan_default

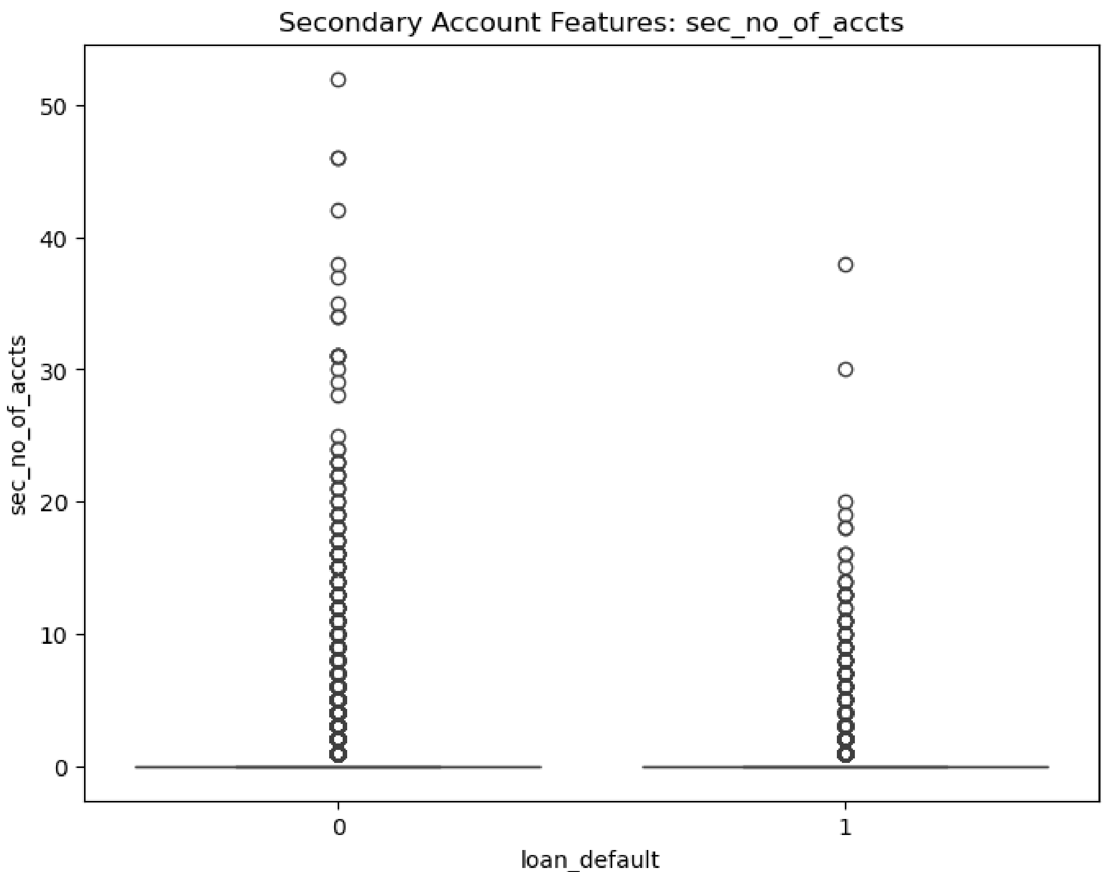

Primary Account Features: pri_active_accts
pri_active_accts
140
120
100
80
60
40
20
0
0
1
loan_default

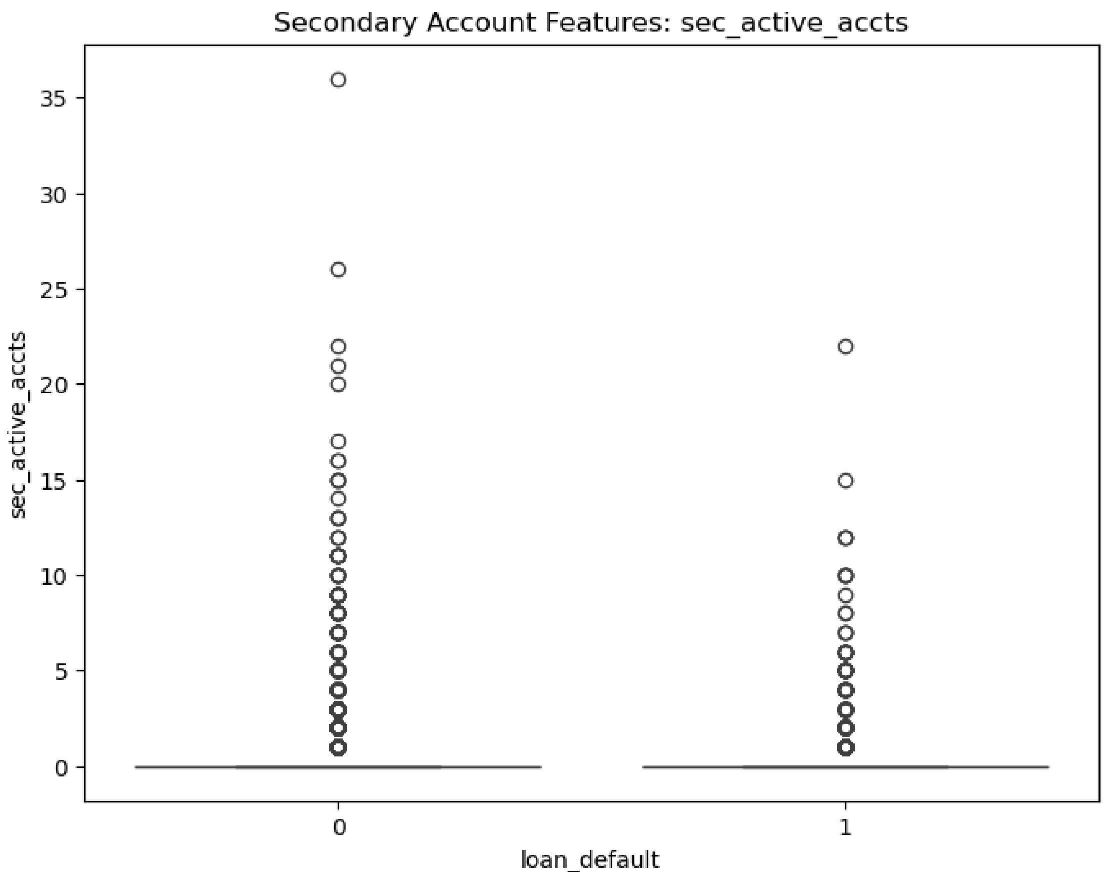

Primary Account Features: pri_overdue_accts
pri_overdue_accts
25
20
15
10
5
0
0
1
loan_default

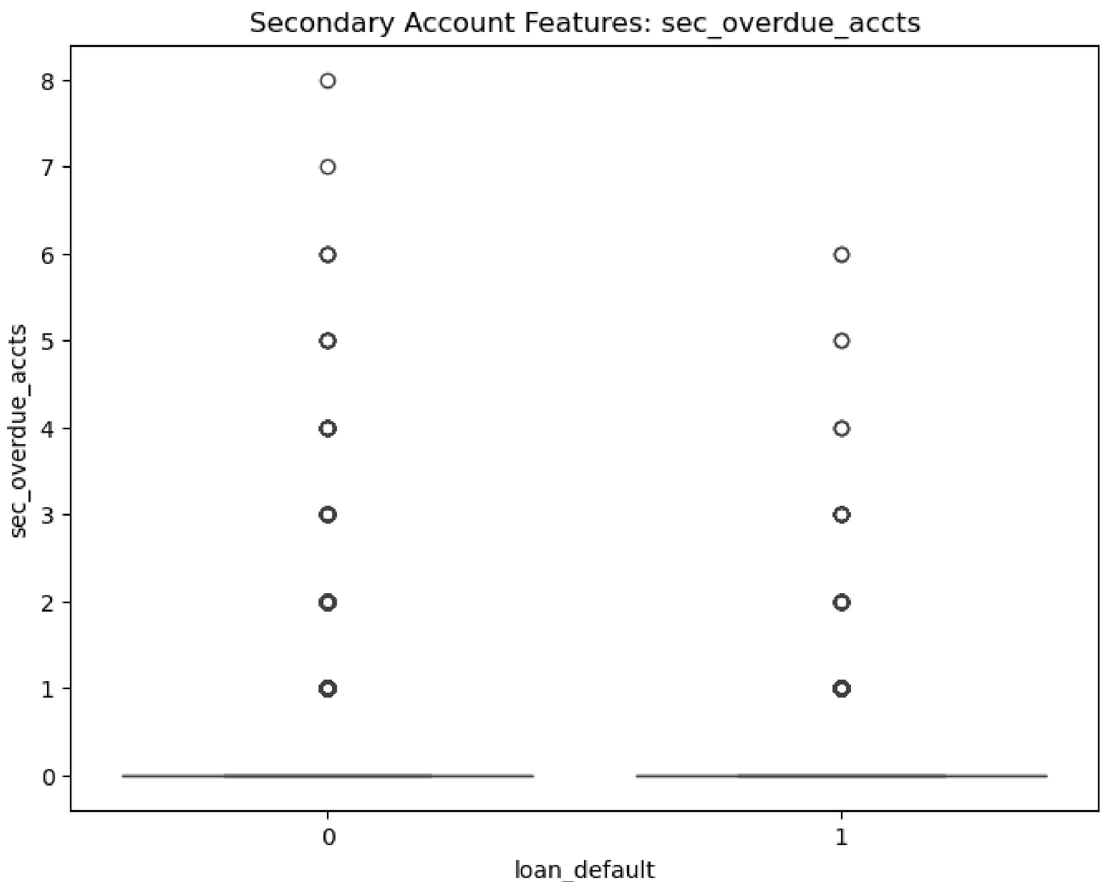

Primary Account Features: pri_current_balance
1e8
1.0
0.8
0.6
0.4
0.2
0.0
pri_current_balance
0
1
loan_default


Primary Account Features: pri_sanctioned_amount
1e9
1.0
0.8
0.6
0.4
0.2
0.0
pri_sanctioned_amount
0
1
loan_default

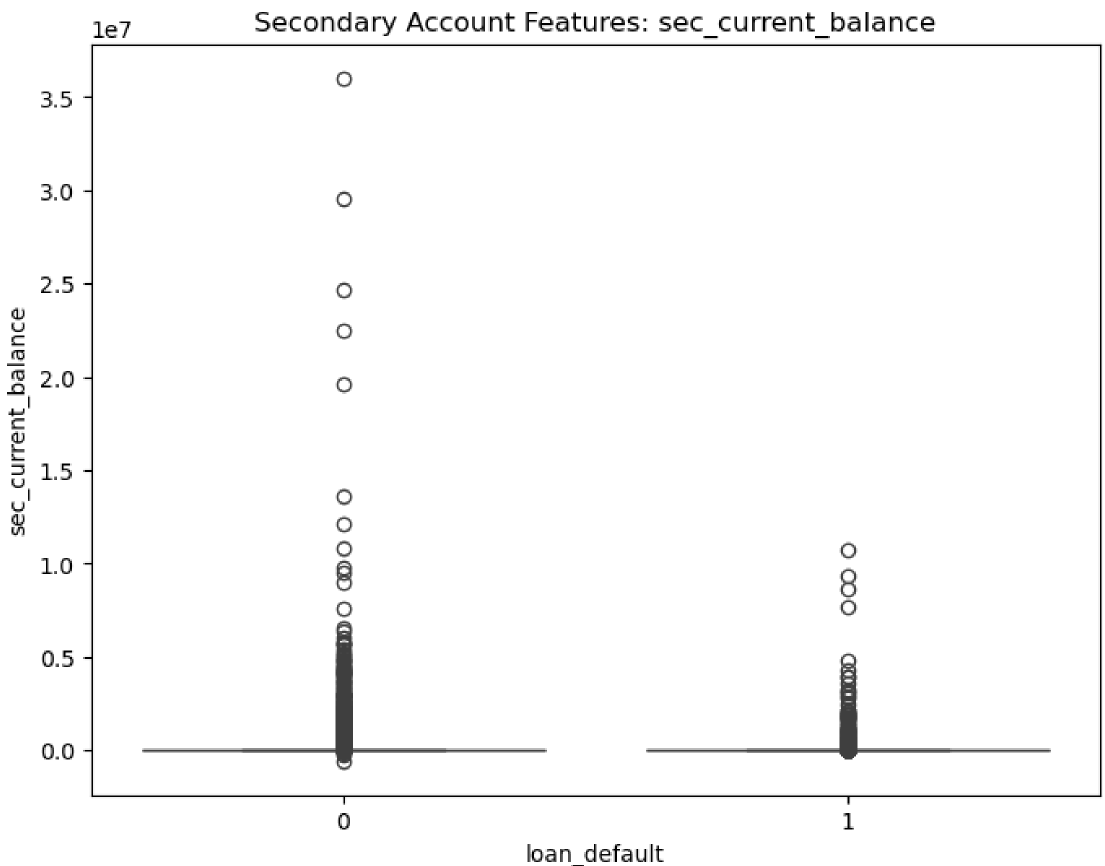

Primary Account Features: pri_disbursed_amount
1e9
1.0
0.8
0.6
0.4
0.2
0.0
pri_disbursed_amount
0
1
loan_default

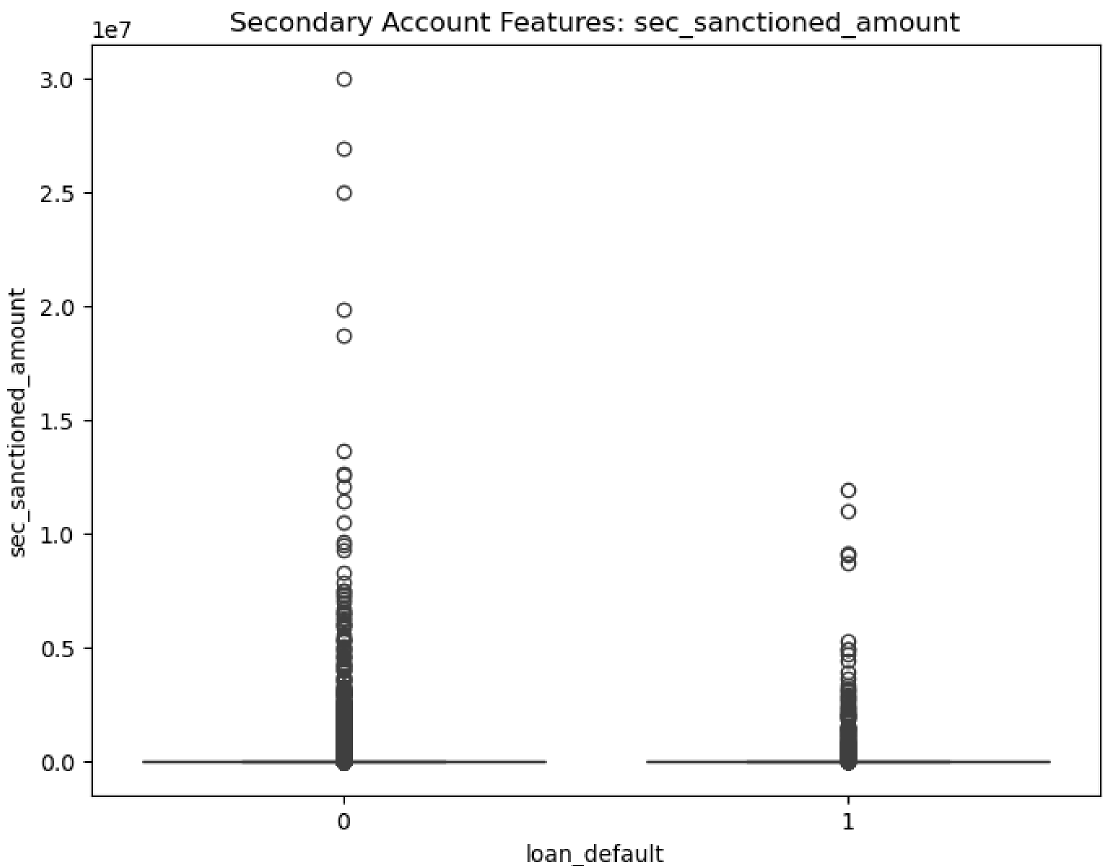

Secondary Account Features: sec_no_of_accts
sec_no_of_accts
0
10
20
30
40
50
0
1
loan_default


Secondary Account Features: sec_active_accts
sec_active_accts
loan_default
0
1
0
5
10
15
20
25
30
35

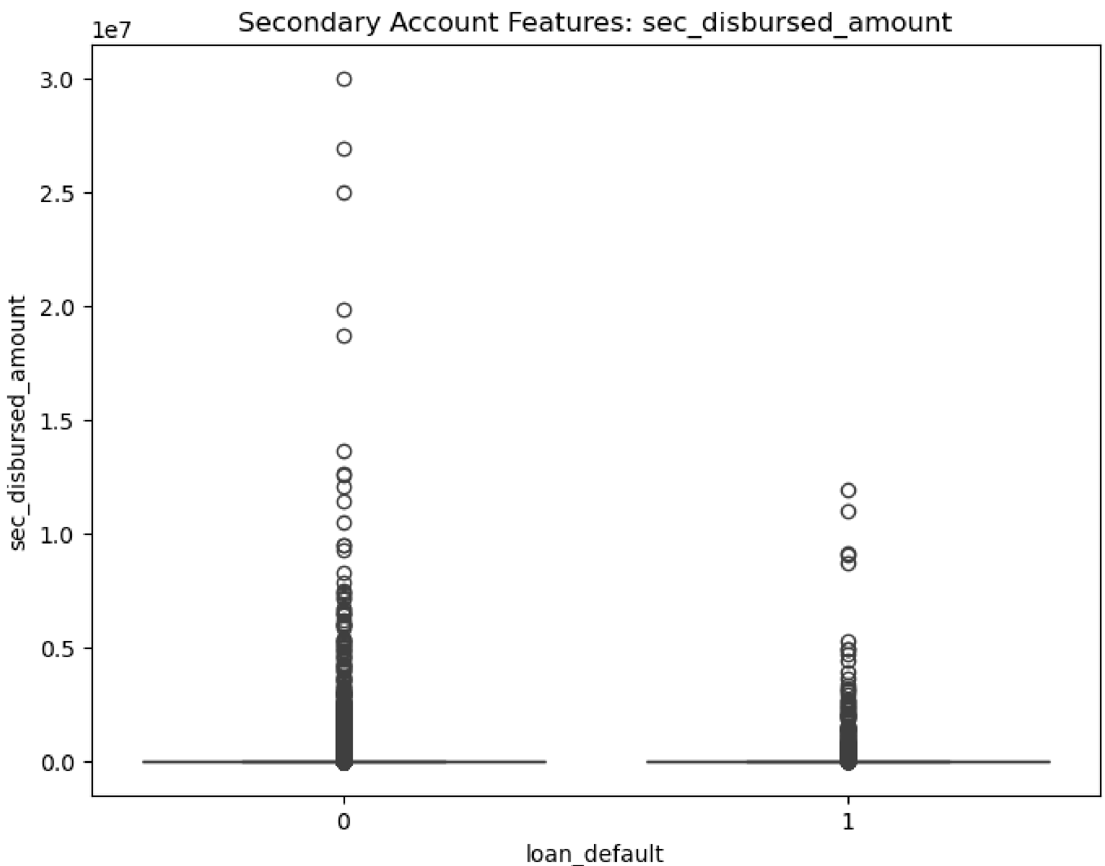


Secondary Account Features: sec_overdue_accts
sec_overdue_accts
8
7
6
5
4
3
2
1
0
0
1
loan_default


Secondary Account Features: sec_current_balance
1e7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
sec_current_balance
0
1
loan_default

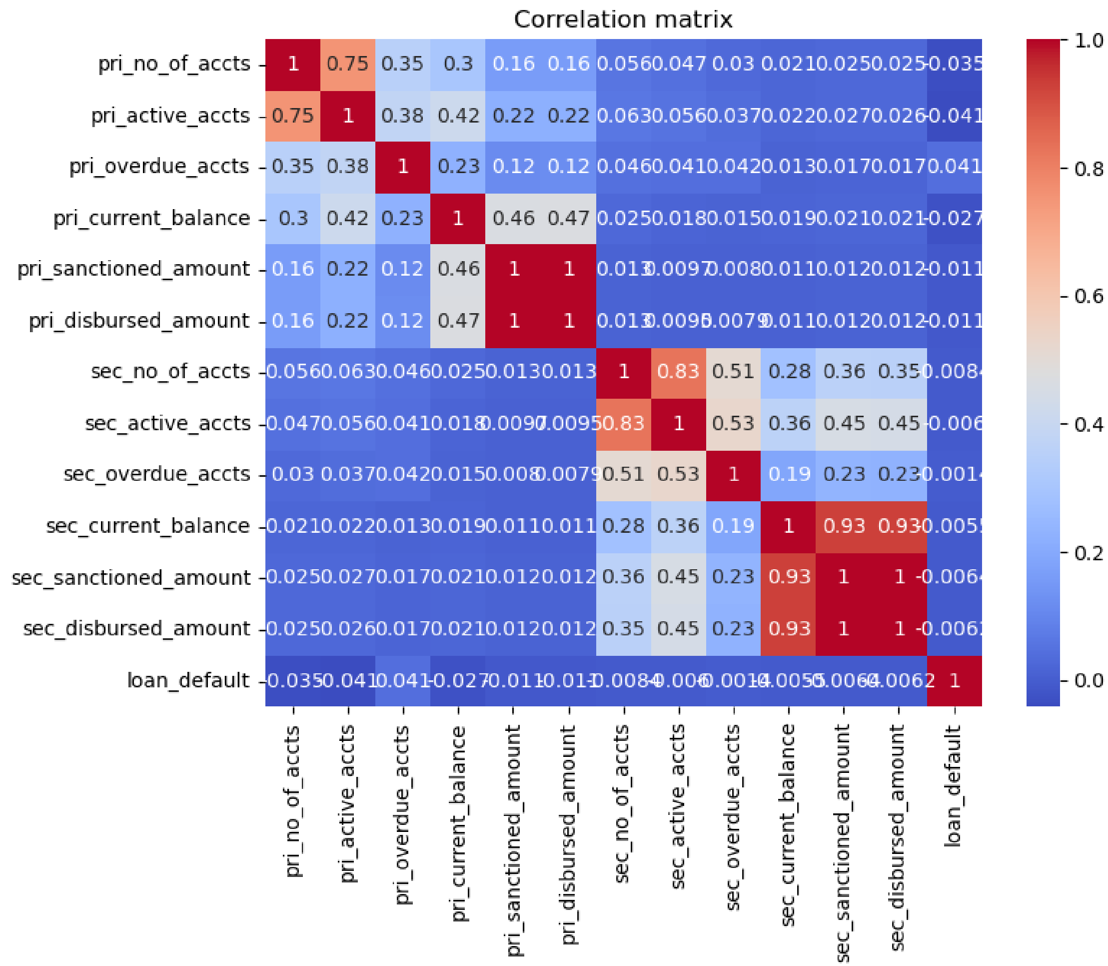


Secondary Account Features: sec_sanctioned_amount
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
sec_sanctioned_amount
0
1
loan_default



[630]:
```
correlation_matrix = df[primary_features + secondary_features+
    ['loan_default']].corr()
```

[631]:
```
plt.figure(figsize=(8,6))
sns.heatmap( correlation_matrix, annot =True, cmap='coolwarm')
plt.title('Correlation matrix')
plt.show()
```



13 Is there a difference between the sanctioned and disbursed amount of primary and secondary loans? Study the difference by providing appropriate statistics and graphs.

[632]:
```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

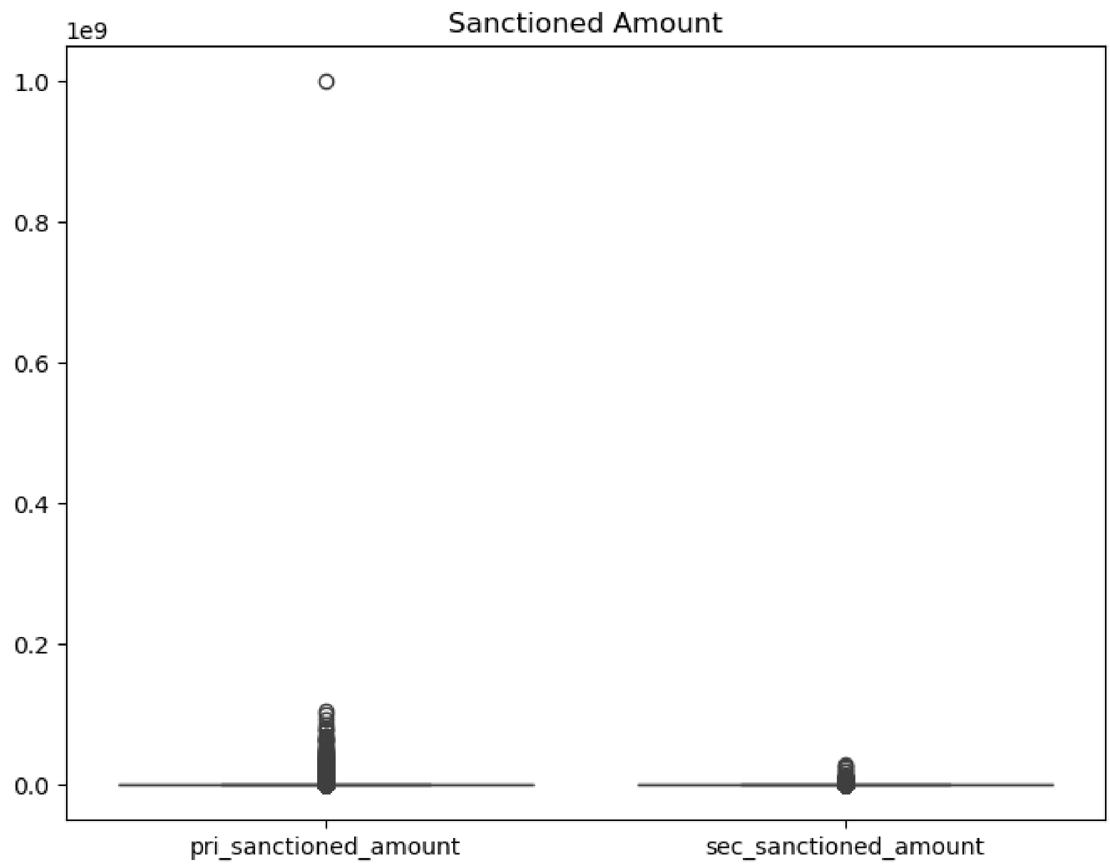
[633]:
```
sanctioned_amount = ['pri_sanctioned_amount', 'sec_sanctioned_amount']
disbursed_amount =['pri_disbursed_amount', 'sec_disbursed_amount']
```

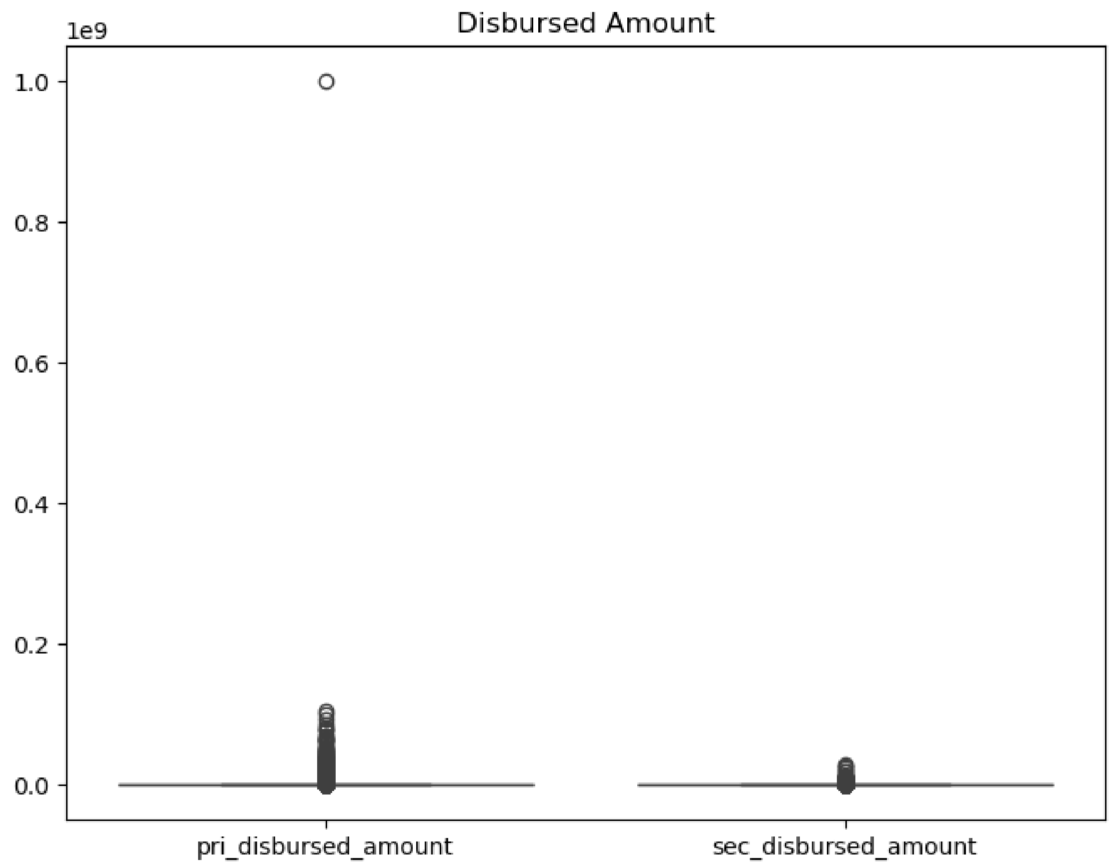
[634]:
```
sanctioned_amount_stat =df[sanctioned_amount].describe()
disbursed_amount_stat =df[disbursed_amount].describe()
print(sanctioned_amount_stat)
```

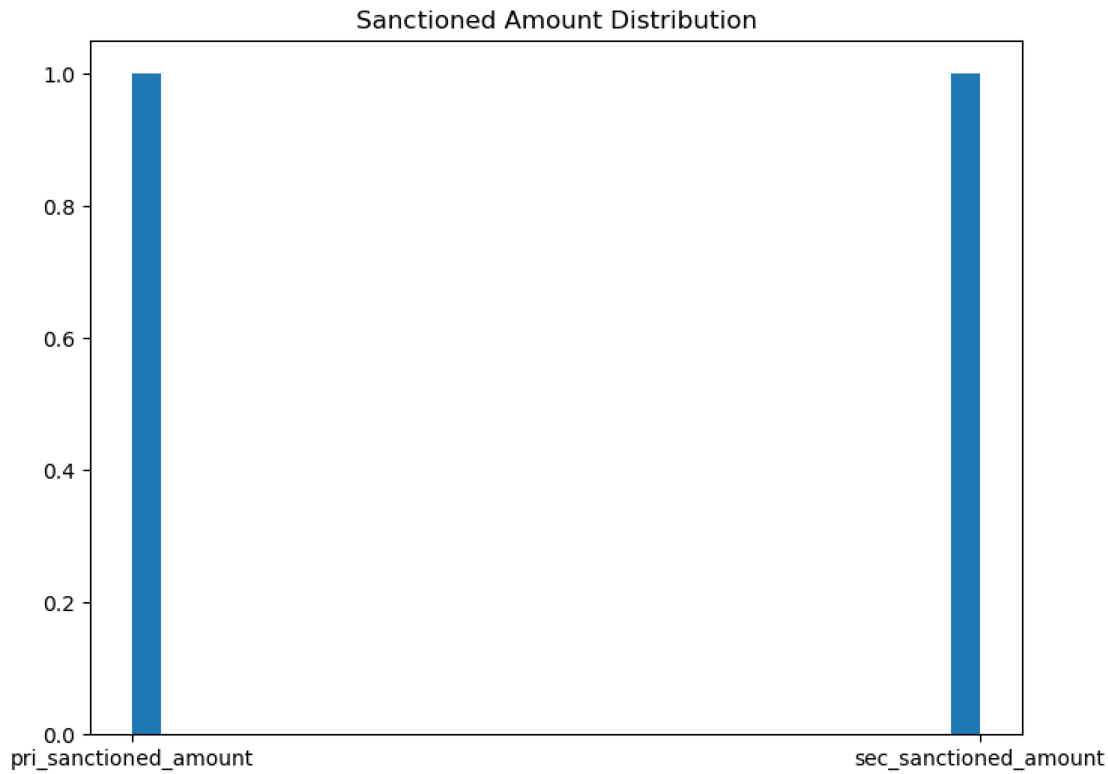
```python
print(disbursed_amount_stat)
```

```
       pri_sanctioned_amount  sec_sanctioned_amount
count           2.331540e+05           2.331540e+05
mean            2.185039e+05           7.295923e+03
std             2.374794e+06           1.831560e+05
min             0.000000e+00           0.000000e+00
25%             0.000000e+00           0.000000e+00
50%             0.000000e+00           0.000000e+00
75%             6.250000e+04           0.000000e+00
max             1.000000e+09           3.000000e+07
       pri_disbursed_amount  sec_disbursed_amount
count          2.331540e+05          2.331540e+05
mean           2.180659e+05          7.179998e+03
std            2.377744e+06          1.825925e+05
min            0.000000e+00          0.000000e+00
25%            0.000000e+00          0.000000e+00
50%            0.000000e+00          0.000000e+00
75%            6.080000e+04          0.000000e+00
max            1.000000e+09          3.000000e+07
```

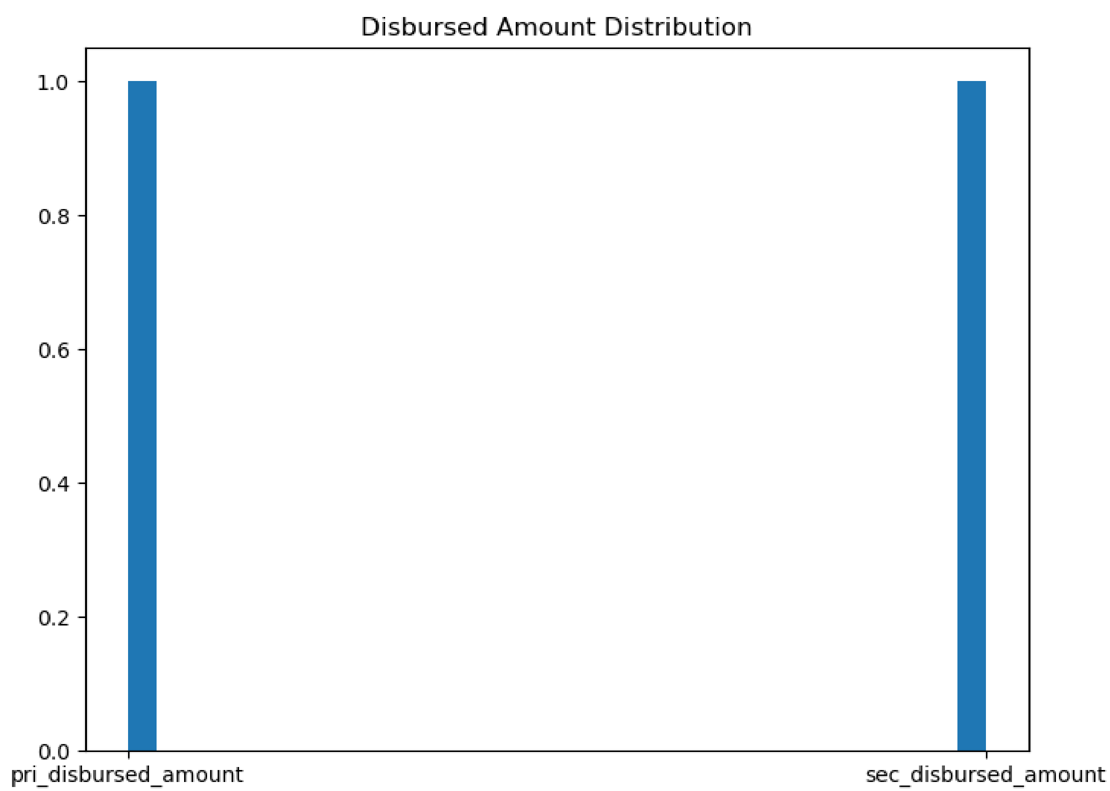
[635]:
```python
#Visualize the data with boxplot
def plot_boxplot (features, title):
    plt.figure(figsize =(8,6))
    sns.boxplot(data =df[features])
    plt.title(title)
    plt.show()

plot_boxplot(sanctioned_amount, 'Sanctioned Amount')
plot_boxplot(disbursed_amount, 'Disbursed Amount')

def plot_histogram(features, title):
    plt.figure(figsize=(8,6))
    plt.hist(data = df[features],  x= features, bins = 30)
    plt.title(title)
    plt.show()

plot_histogram(sanctioned_amount, 'Sanctioned Amount Distribution')
plot_histogram(disbursed_amount,'Disbursed Amount Distribution')
```


Sanctioned Amount
1e9
1.0
0.8
0.6
0.4
0.2
0.0
pri_sanctioned_amount
sec_sanctioned_amount


Disbursed Amount
1e9
1.0
0.8
0.6
0.4
0.2
0.0
pri_disbursed_amount
sec_disbursed_amount



Sanctioned Amount Distribution
1.0
0.8
0.6
0.4
0.2
0.0
pri_sanctioned_amount
sec_sanctioned_amount

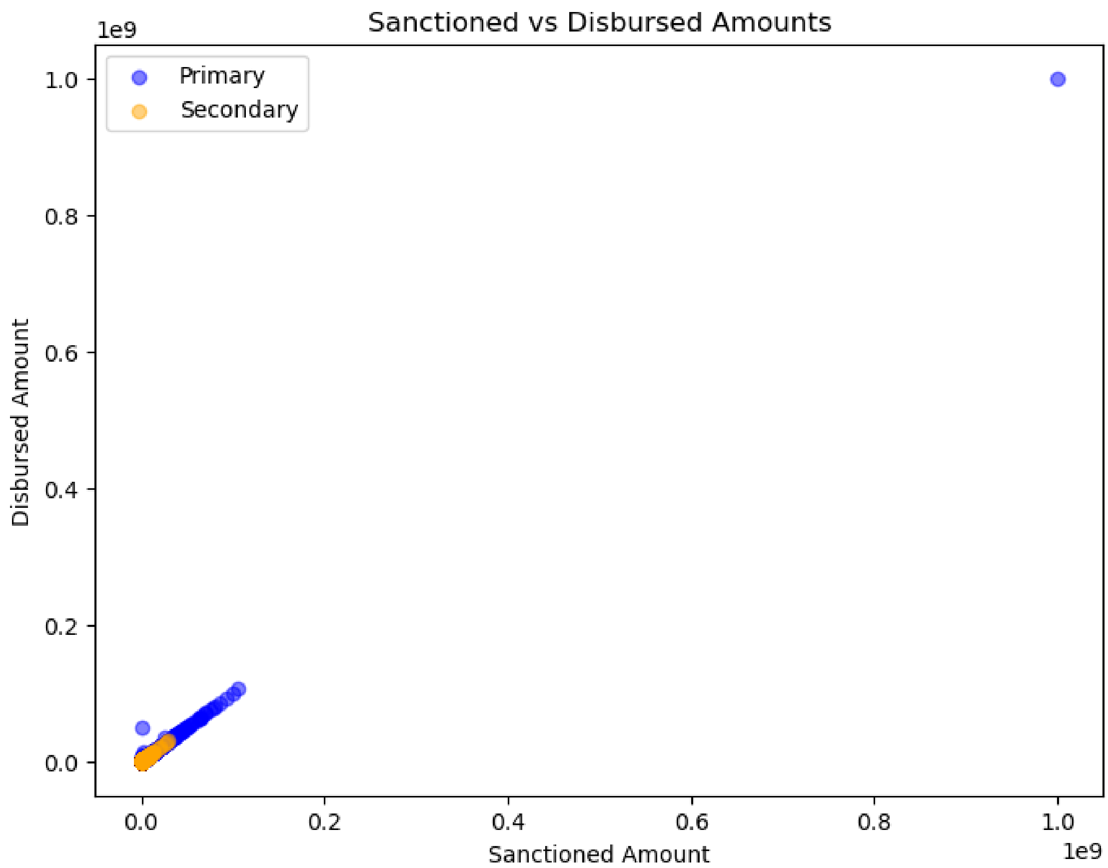


Disbursed Amount Distribution
1.0
0.8
0.6
0.4
0.2
0.0
pri_disbursed_amount
sec_disbursed_amount

[636]:
```
t_stat_sanctioned, p_value_sanctioned = stats.
↪ttest_rel(df['pri_sanctioned_amount'], df['sec_sanctioned_amount'],␣
↪nan_policy ='omit')
t_stat_disbursed, p_value_disbursed = stats.
↪ttest_rel(df['pri_disbursed_amount'], df['sec_disbursed_amount'],␣
↪nan_policy='omit')
print(f'T-test for Sanctioned Amount: t-statistic = {t_stat_sanctioned},␣
↪p-value = {p_value_sanctioned}')
print(f'T-test for Disbursed Amount : t-statistic= {t_stat_disbursed}, p-value␣
↪={p_value_disbursed}')
```

```
T-test for Sanctioned Amount: t-statistic = 42.856146083926376, p-value = 0.0
T-test for Disbursed Amount : t-statistic= 42.738305407591284, p-value =0.0
```

[637]:
```
# Relationship between Disbursed & Sanctioned Amount
plt.figure(figsize=(8,6))
plt.scatter(df ['pri_sanctioned_amount'], df['pri_disbursed_amount'], alpha =0.
↪5, label ='Primary', color ='blue')
plt.scatter (df['sec_sanctioned_amount'], df['sec_disbursed_amount'], alpha =0.
↪5, label ='Secondary', color ='orange')
plt.xlabel('Sanctioned Amount')
plt.ylabel('Disbursed Amount')
plt.title('Sanctioned vs Disbursed Amounts')
plt.legend()
plt.show()
```

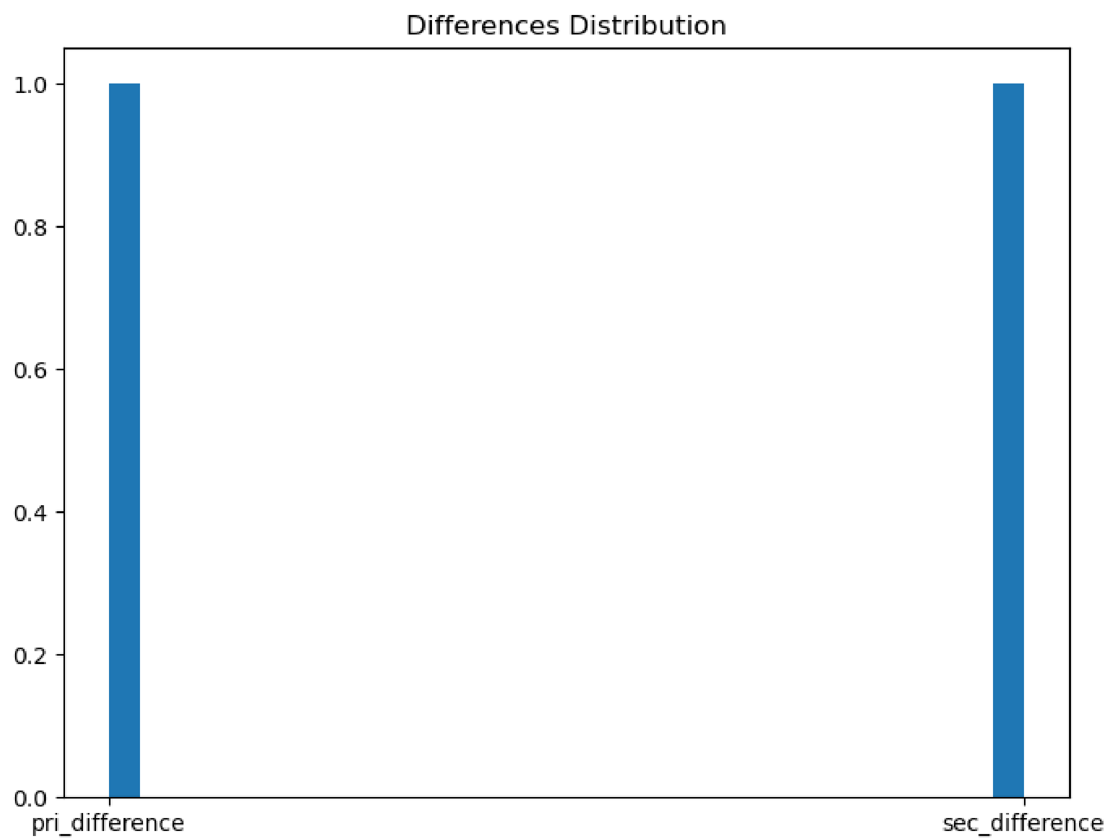


[638]:
```
# Calculate the difference between sanctioned & disbursed amount
df['pri_difference'] =df['pri_sanctioned_amount'] - df['pri_disbursed_amount']
df['sec_difference'] =df['sec_sanctioned_amount']- df['sec_disbursed_amount']
print('Pri Difference',df['pri_difference'], '\n Sec␣
↪Difference',df['sec_difference'])
```

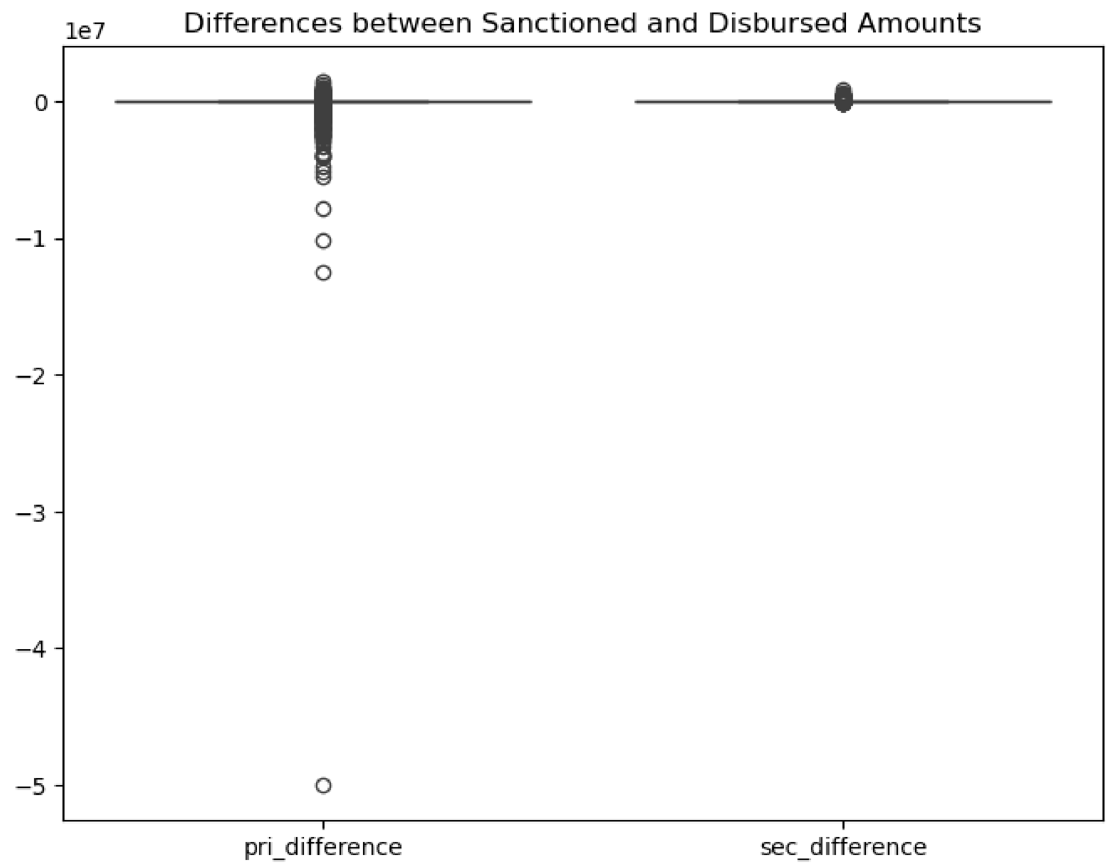
```
Pri Difference 0               0
1              0
2              0
3              0
4              0
          …
233149         0
233150         0
233151    110000
233152         0
233153     -5912
Name: pri_difference, Length: 233154, dtype: int64
 Sec Difference 0          0
1          0
```

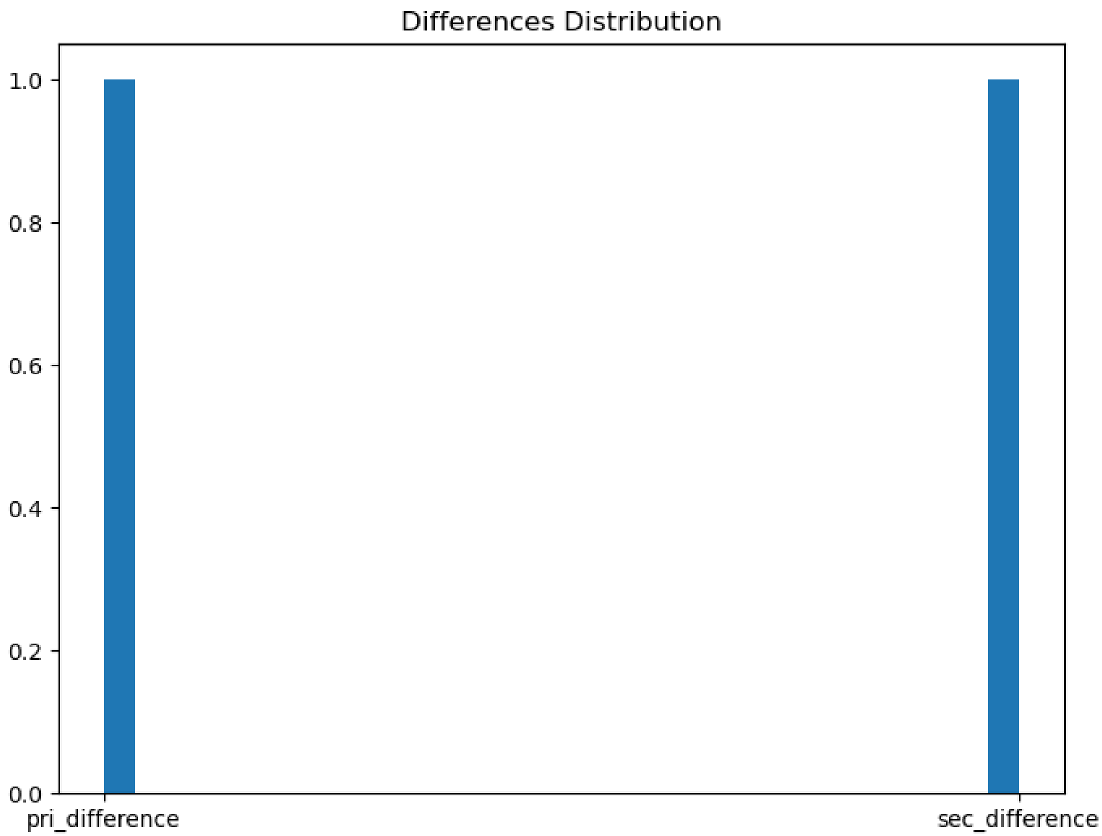
```
2          0
3          0
4          0
          ..
233149     0
233150     0
233151     0
233152     0
233153     0
Name: sec_difference, Length: 233154, dtype: int64
```

[639]:
```
plot_histogram(['pri_difference', 'sec_difference'], 'Differences Distribution')
```



[640]:
```
plot_boxplot(['pri_difference', 'sec_difference'], 'Differences between␣
↪Sanctioned and Disbursed Amounts')

plot_histogram(['pri_difference', 'sec_difference'], 'Differences Distribution')
```


Differences between Sanctioned and Disbursed Amounts
1e7
0
−1
−2
−3
−4
−5
pri_difference
sec_difference



[641]:
```
t_stat_difference, p_value_difference =stats.
    ↪ttest_rel(df['pri_difference'],df['sec_difference'], nan_policy ='omit')
print(f'T-test for Differences: t-statistic = {t_stat_difference}, p-value =␣
    ↪{p_value_difference}')
```

```
T-test for Differences: t-statistic = 1.3059103558376783, p-value =
0.19158433290608548
```

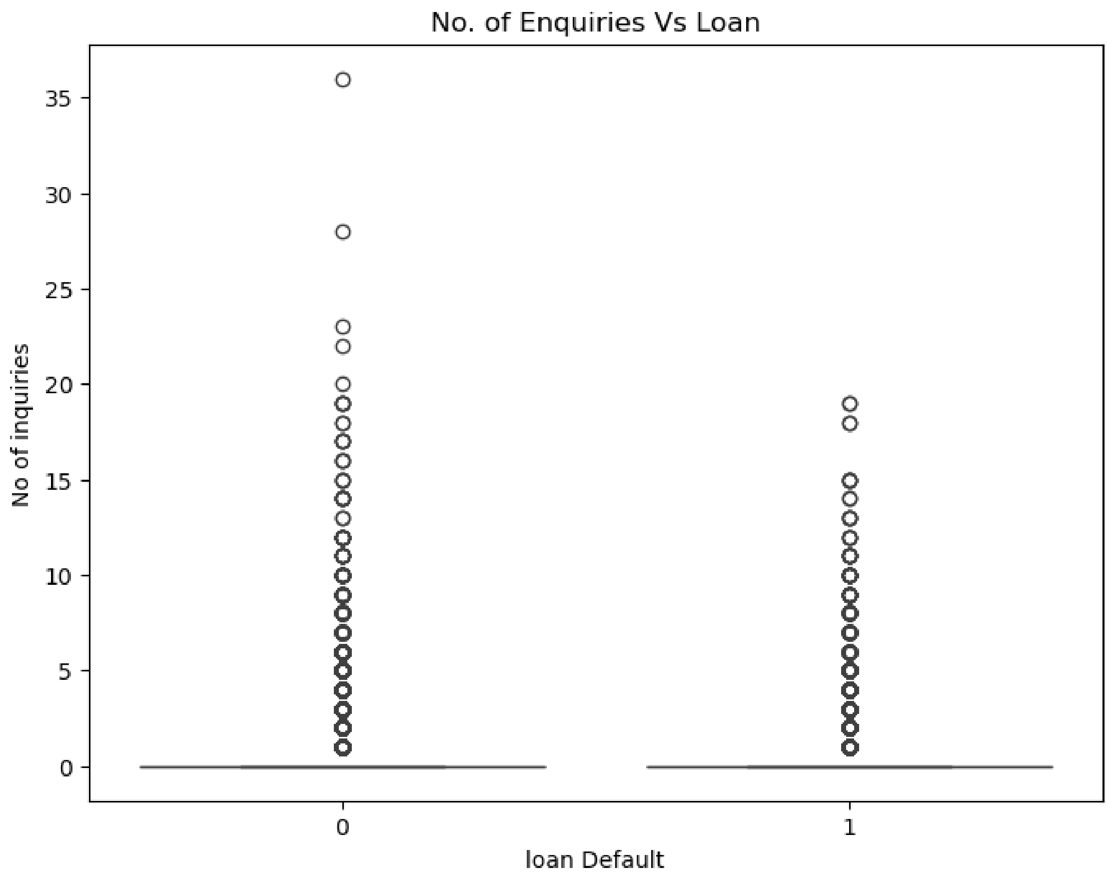
14 Customers who make higher numbers of inquiries end up being higher risk candidates.

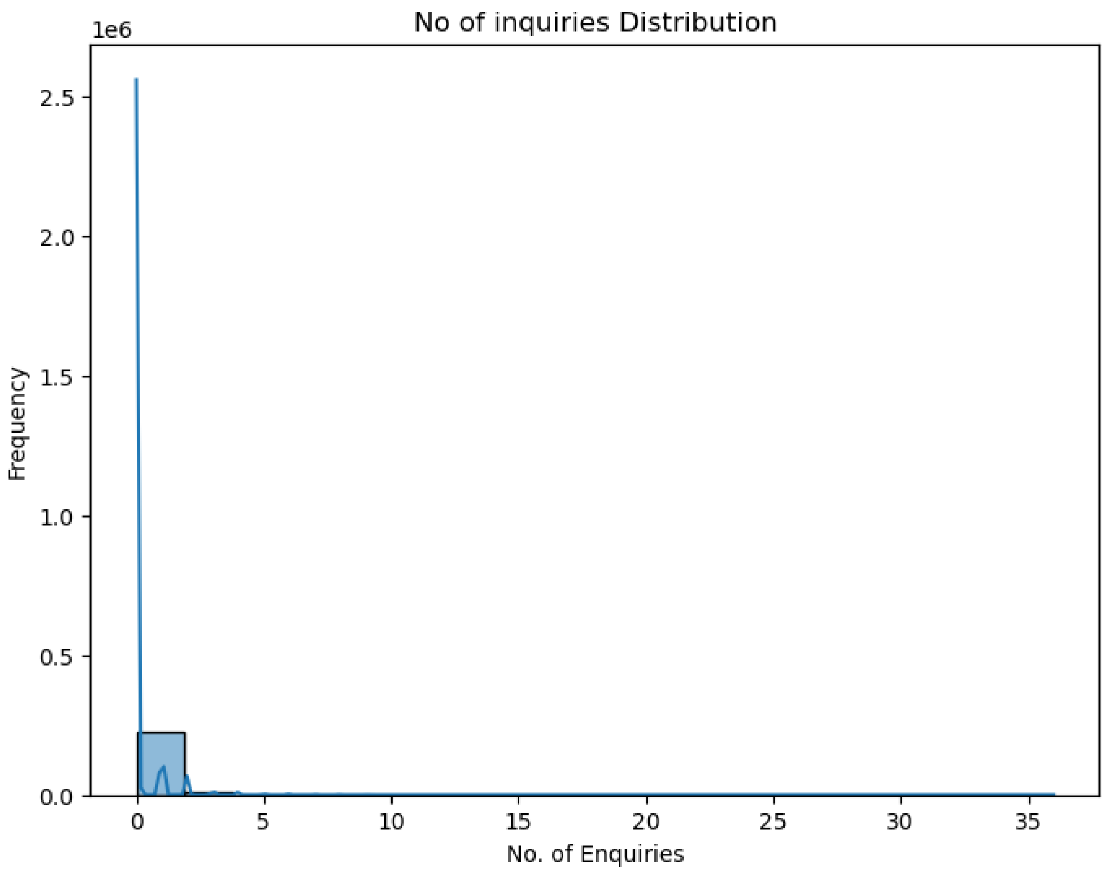
[690]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
print(df[['no_of_inquiries','loan_default']].describe())
```

```
       no_of_inquiries   loan_default
count    233154.000000  233154.000000
mean          0.206615       0.217071
```

```
std            0.706498       0.412252
min            0.000000       0.000000
25%            0.000000       0.000000
50%            0.000000       0.000000
75%            0.000000       0.000000
max           36.000000       1.000000
```

[710]:
```
# Visualization by Boxplot
plt.figure(figsize = (8,6))
sns.boxplot(x='loan_default', y ='no_of_inquiries', data =df)
plt.ylabel ('No of inquiries')
plt.xlabel('loan Default')
plt.title('No. of Enquiries Vs Loan')
plt.show()

# Visualization by Histogram
plt.figure(figsize =(8,6))
sns.histplot(df['no_of_inquiries'], kde =True)
plt.title('No of inquiries Distribution')
plt.xlabel('No. of Enquiries')
plt.ylabel('Frequency')
plt.show()

average_inquiries_default = df[df['loan_default'] ==1] ['no_of_inquiries'].
 ↪mean()
average_inquiries_non_default = df[df['loan_default']==0] ['no_of_inquiries'].
 ↪mean()

print(f'Default Average Enquiries : {average_inquiries_default}')
print(f'Non-Default Average Enquiries : {average_inquiries_non_default}')
```


No. of Enquiries Vs Loan
No of inquiries
loan Default
0
1
0
5
10
15
20
25
30
35




```
Default Average Enquiries : 0.26521902353243365
Non-Default Average Enquiries : 0.19036610552034317
```

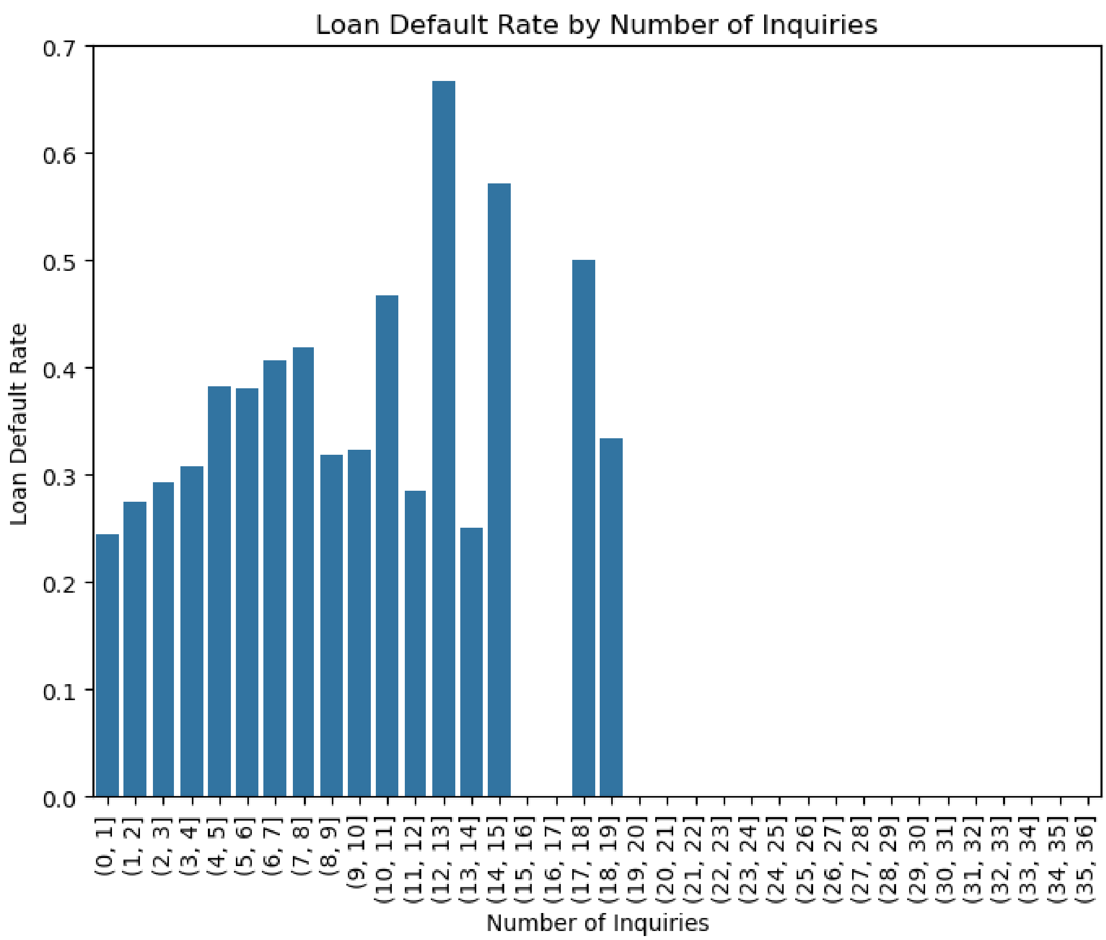
[714]:
```
# Perform T-test
inquiries_default = df[df['loan_default'] ==1] ['no_of_inquiries']
inquiries_non_default = df[df['loan_default']==0] ['no_of_inquiries']
t_stats, p_value =stats.ttest_ind(inquiries_default, inquiries_non_default,␣
↪nan_policy ='omit')
print(f't_test : Statistics = {t_stats}, Probablity = {p_value}')
```

```
t_test : Statistics = 21.110344428166638, Probablity = 7.912566786376203e-99
```

[722]:
```
df['inquiry_bins'] =pd.cut(df['no_of_inquiries'], bins =range(0,␣
↪df['no_of_inquiries'].max() +1,1))
default_rate_by_inquiry = df.groupby('inquiry_bins')['loan_default'].mean().
↪reset_index()
plt.figure(figsize=(8, 6))
sns.barplot(x='inquiry_bins', y='loan_default', data=default_rate_by_inquiry)
plt.title('Loan Default Rate by Number of Inquiries')
plt.xlabel('Number of Inquiries')
```

```
plt.ylabel('Loan Default Rate')
plt.xticks(rotation=90)
plt.show()
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_7952\1061419541.py:2: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  default_rate_by_inquiry =
df.groupby('inquiry_bins')['loan_default'].mean().reset_index()
```



[724]:
```
# Clean up the bin labels
df.drop(columns=['inquiry_bins'], inplace=True)
```

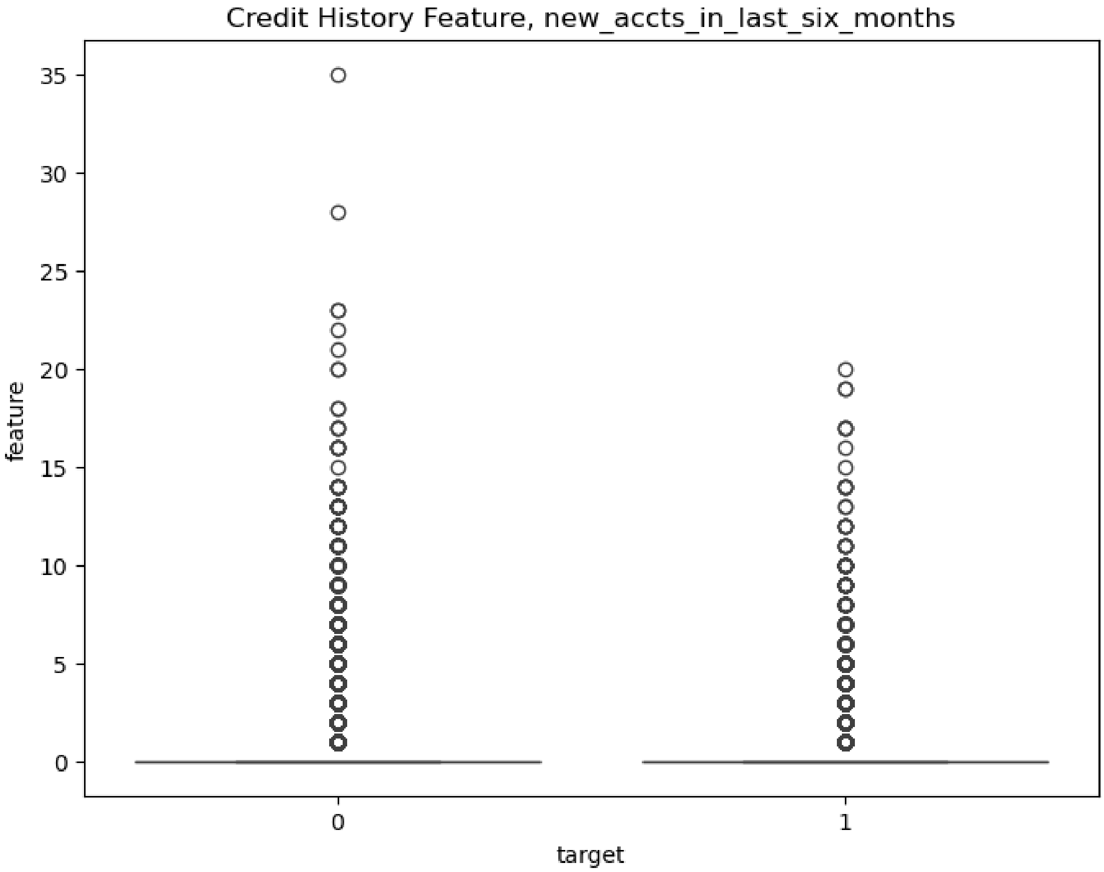
15 Is credit history, that is, new loans in the last six months, loans defaulted in the last six months, time since the first loan, etc., a significant factor in estimating the probability of loan defaulters?

[743]:
```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as splt
import seaborn as sns
import re
```

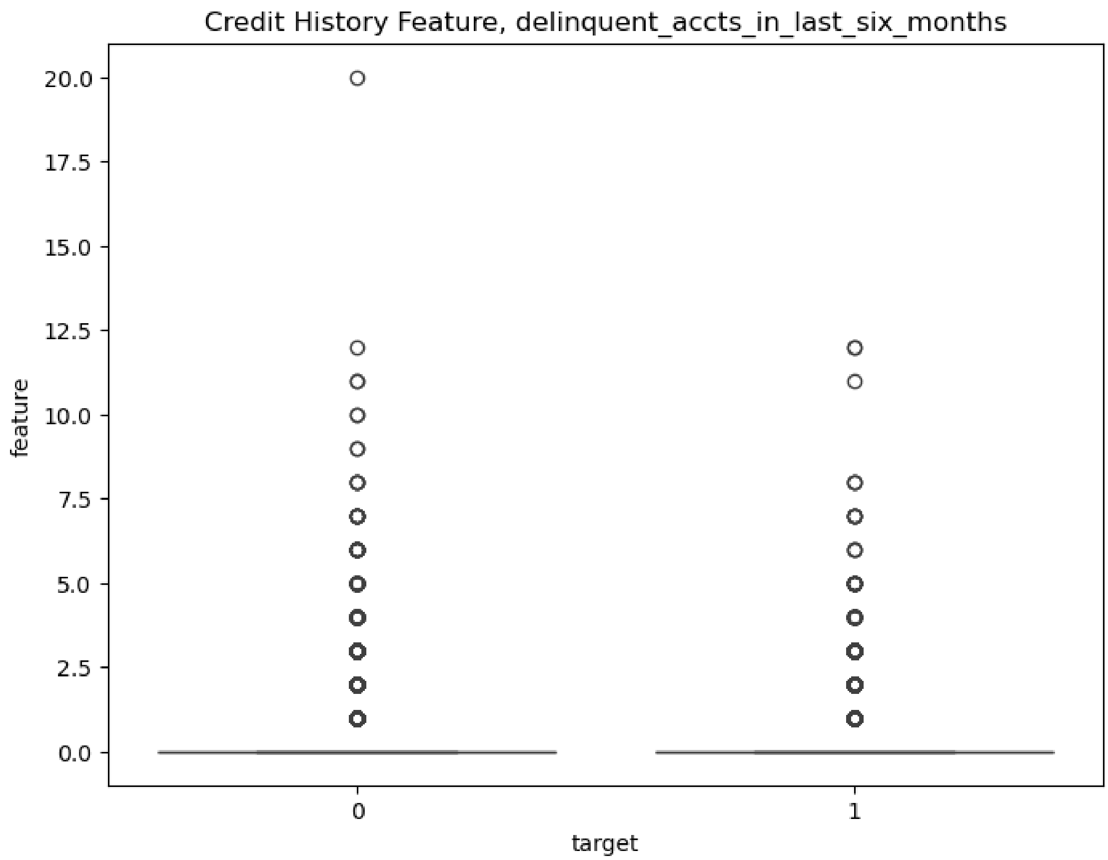
[783]:
```
def convert_to_months (age_str):
    if pd.isnull(age_str):
        return 0
    years = int(re.search (r'(\d+)yrs',age_str).group(1)) if 'yrs' in age_str␣
 ↪else 0
    months = int(re.search(r'(\d+)mon', age_str).group(1)) if 'mon' in age_str␣
 ↪else 0
    return years * 12 + months
df['credit_history_length_months'] = df['credit_history_length'].
 ↪apply(convert_to_months)
df['average_acct_age_months'] = df['average_acct_age'].apply(convert_to_months)
```

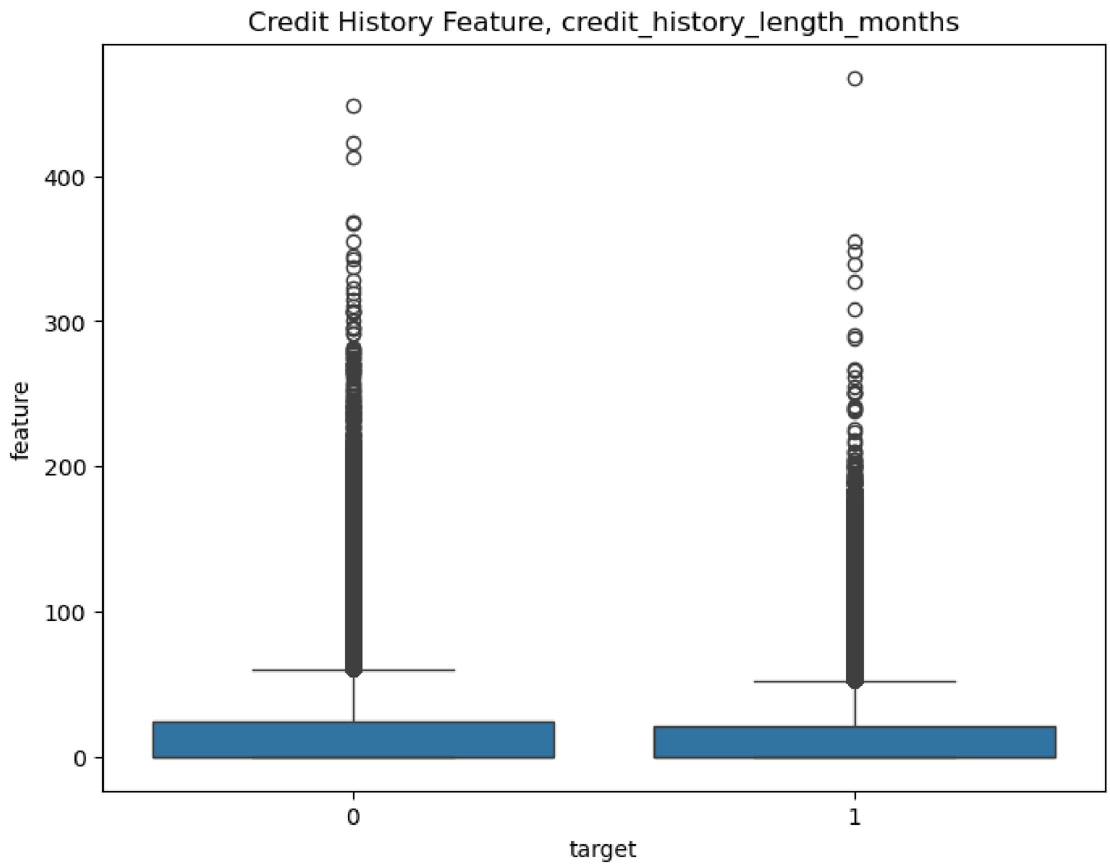
[785]:
```
credit_history_columns = ['new_accts_in_last_six_months',␣
 ↪'delinquent_accts_in_last_six_months', 'credit_history_length_months',␣
 ↪'average_acct_age_months']
```

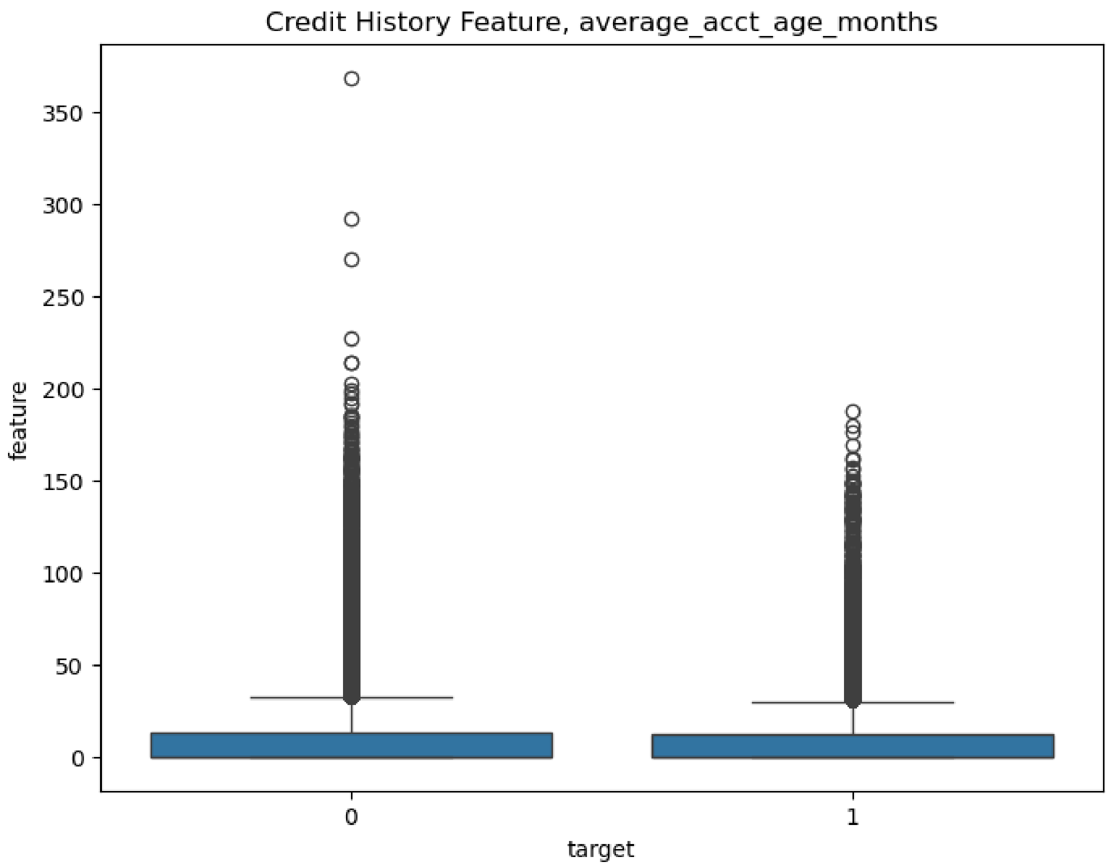
[791]:
```
# Visualize Credit History Vs Loan Default
def plot_features(features,target,title):
    for feature in features:
        plt.figure(figsize =(8,6))
        sns.boxplot(x =target, y=feature, data =df)
        plt.xlabel('target')
        plt.ylabel('feature')
        plt.title(f'{title}, {feature}')
        plt.show()

plot_features(credit_history_columns,'loan_default', 'Credit History Feature')
```



Credit History Feature, new_accts_in_last_six_months
feature
target
0
1
0
5
10
15
20
25
30
35


Credit History Feature, delinquent_accts_in_last_six_months
feature
target
0
1
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0


Credit History Feature, credit_history_length_months
400
300
200
100
0
feature
0
1
target



[795]:
```
# T Test
def t_test_features (features, target):
    for feature in features:
        default =df[df[target] == 1] [feature]
        non_default =df[df[target]==0] [feature]
        t_stats, p_value= stats.ttest_ind(default,non_default, nan_policy␣
↪='omit')
        print(f'Test of features : t_stats ={t_stats}, p_value={p_value}')

t_test_features(credit_history_columns,'loan_default')
```

```
Test of features : t_stats =-14.202014495523946, p_value=9.302293710212261e-46
Test of features : t_stats =16.649971389459616, p_value=3.2892517686894386e-62
Test of features : t_stats =-20.35898474060458, p_value=4.6500173864982836e-92
Test of features : t_stats =-11.969316978054648, p_value=5.261091482095606e-33
```

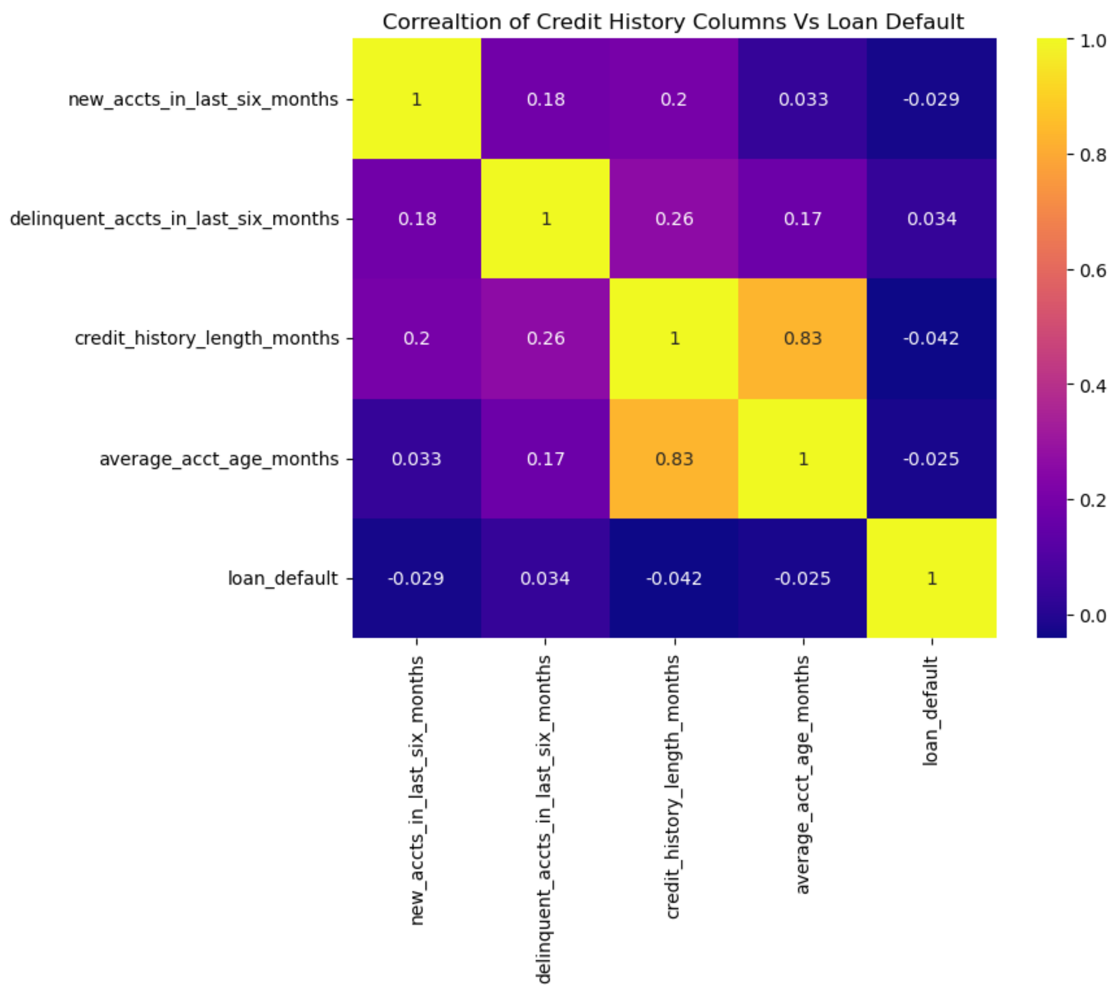
[805]:
```
correlation_matrix = df[credit_history_columns + ['loan_default']].corr()
print(f'Correlation Matrix:\n  {correlation_matrix}')
```

```
Correlation Matrix:
```

```
                                     new_accts_in_last_six_months  \
new_accts_in_last_six_months                             1.000000
delinquent_accts_in_last_six_months                      0.182769
credit_history_length_months                             0.200087
average_acct_age_months                                  0.033372
loan_default                                            -0.029400

                                     delinquent_accts_in_last_six_months  \
new_accts_in_last_six_months                                    0.182769
delinquent_accts_in_last_six_months                             1.000000
credit_history_length_months                                    0.262218
average_acct_age_months                                         0.171348
loan_default                                                    0.034462

                                     credit_history_length_months  \
new_accts_in_last_six_months                             0.200087
delinquent_accts_in_last_six_months                      0.262218
credit_history_length_months                             1.000000
average_acct_age_months                                  0.831952
loan_default                                            -0.042126

                                     average_acct_age_months  loan_default
new_accts_in_last_six_months                        0.033372     -0.029400
delinquent_accts_in_last_six_months                 0.171348      0.034462
credit_history_length_months                        0.831952     -0.042126
average_acct_age_months                             1.000000     -0.024781
loan_default                                       -0.024781      1.000000
```

[813]:

```
plt.figure(figsize =(8,6))
sns.heatmap(correlation_matrix, annot =True, cmap  = 'plasma')
plt.title('Correaltion of Credit History Columns Vs Loan Default')
plt.show()
```



[815]:
```
df.head()
```

[815]:
```
   uniqueid  disbursed_amount  asset_cost    ltv  branch_id  supplier_id  \
0    420825             50578       58400  89.55         67        22807
1    417566             53278       61360  89.63         67        22807
2    539055             52378       60300  88.39         67        22807
3    529269             46349       61500  76.42         67        22807
4    563215             43594       78256  57.50         67        22744

   manufacturer_id  current_pincode_id date_of_birth employment_type  …  \
0               45                1441    1984-01-01        Salaried  …
1               45                1497    1985-08-24   Self employed  …
2               45                1495    1977-12-09   Self employed  …
3               45                1502    1988-06-01        Salaried  …
4               86                1499    1994-07-14   Self employed  …
```

```
  average_acct_age  credit_history_length  no_of_inquiries  loan_default  age  \
0        0yrs 0mon             0yrs 0mon                0             0   40
1        0yrs 0mon             0yrs 0mon                0             0   39
2        0yrs 0mon             0yrs 0mon                1             1   47
3        0yrs 0mon             0yrs 0mon                0             0   36
4        0yrs 0mon             0yrs 0mon                0             0   30

   pri_difference  sec_difference  inquiry_bins  credit_history_length_months  \
0               0               0           NaN                             0
1               0               0           NaN                             0
2               0               0    (0.0, 1.0]                             0
3               0               0           NaN                             0
4               0               0           NaN                             0

   average_acct_age_months
0                        0
1                        0
2                        0
3                        0
4                        0

[5 rows x 47 columns]
```

16 Perform logistic regression modeling, predict the outcome for the test data, and validate the results using the confusion matrix.

[849]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score,␣
↪classification_report
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

[851]:

```
features =['disbursed_amount', 'asset_cost', 'ltv', 'branch_id', 'supplier_id',
            'manufacturer_id', 'current_pincode_id', 'date_of_birth',␣
↪'employment_type',
            'disbursaldate', 'state_id', 'employee_code_id',␣
↪'mobileno_avl_flag',
            'aadhar_flag', 'pan_flag', 'voterid_flag', 'driving_flag',␣
↪'passport_flag',
```

```
            'perform_cns_score', 'pri_no_of_accts', 'pri_active_accts',␣
↪'pri_overdue_accts',
            'pri_current_balance', 'pri_sanctioned_amount',␣
↪'pri_disbursed_amount',
            'sec_no_of_accts', 'sec_active_accts', 'sec_overdue_accts',␣
↪'sec_current_balance',
            'sec_sanctioned_amount', 'sec_disbursed_amount',␣
↪'primary_instal_amt', 'sec_instal_amt',
            'new_accts_in_last_six_months',␣
↪'delinquent_accts_in_last_six_months', 'average_acct_age_months',
            'credit_history_length_months', 'no_of_inquiries', 'age',␣
↪'pri_difference', 'sec_difference']
target = 'loan_default'
```

[853]:
```
label_encoders = {}
for column in ['employment_type', 'date_of_birth', 'disbursaldate']:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

[861]:
```
x =df[features]
y =df[target]

x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2,␣
↪random_state = 42)
scaler= StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[863]:
```
model =LogisticRegression(max_iter =1000)
model.fit(x_train, y_train)
```

[863]:
```
LogisticRegression(max_iter=1000)
```

[867]:
```
y_pred =model.predict(x_train)
```

[869]:
```
y_pred =model.predict(x_test)
```

[875]:
```
conf_matrix = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix : \n { conf_matrix}')
```

```
Confusion Matrix :
 [[36646    87]
 [ 9845    53]]
```

[881]:
```
class_report = classification_report(y_test,y_pred)
print(f'Classificatio Report : \n{class_report}')
```

```
Classificatio Report :
               precision    recall  f1-score   support

           0       0.79      1.00      0.88     36733
           1       0.38      0.01      0.01      9898

    accuracy                           0.79     46631
   macro avg       0.58      0.50      0.45     46631
weighted avg       0.70      0.79      0.70     46631
```

[883]:

```
acc_score =accuracy_score(y_test, y_pred)
print(f'Accuracy Score : \n{acc_score}')
```

```
Accuracy Score :
0.7870086423194871
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: