# Banking

October 3, 2024

[50]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

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

[51]:

```
# Reading data from csv file and examine size of data as well as few rows of␣
  ↪data
df = pd.read_csv("LTFS_data.csv")
print(df.shape)
df.head()
```

```
(233154, 41)
```

[51]:

```
   UniqueID  disbursed_amount  asset_cost    ltv  branch_id  supplier_id  \
0    420825             50578       58400  89.55         67        22807
1    537409             47145       65550  73.23         67        22807
2    417566             53278       61360  89.63         67        22807
3    624493             57513       66113  88.48         67        22807
4    539055             52378       60300  88.39         67        22807

   manufacturer_id  Current_pincode_ID Date.of.Birth Employment.Type  …  \
0               45                1441      01-01-84        Salaried  …
1               45                1502      31-07-85   Self employed  …
2               45                1497      24-08-85   Self employed  …
3               45                1501      30-12-93   Self employed  …
4               45                1495      09-12-77   Self employed  …

  SEC.SANCTIONED.AMOUNT  SEC.DISBURSED.AMOUNT  PRIMARY.INSTAL.AMT  \
0                     0                     0                   0
1                     0                     0                1991
2                     0                     0                   0
3                     0                     0                  31
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                                    1        1yrs 11mon
2                                    0         0yrs 0mon
3                                    0         0yrs 8mon
4                                    0         0yrs 0mon

   CREDIT.HISTORY.LENGTH  NO.OF_INQUIRIES  loan_default
0              0yrs 0mon                0             0
1             1yrs 11mon                0             1
2              0yrs 0mon                0             0
3              1yrs 3mon                1             1
4              0yrs 0mon                1             1

[5 rows x 41 columns]
```

[52]:

```python
df.info() # Information about the dataset
```

```
<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  object
 9   Employment.Type                      225493 non-null  object
 10  DisbursalDate                        233154 non-null  object
 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: float64(1), int64(34), object(6)
memory usage: 72.9+ MB
```

[53]:

```python
df.isnull().sum() # To check for null values
```

[53]:

```
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                        7661
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
```

[54]:
```python
df.dropna(inplace=True)
df.isnull().sum()
```

[54]:
```
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
```

Getting Numerical Columns from dataset

[55]:

```
numcol = list(df._get_numeric_data().columns)
numcol
```

[55]:

```
['UniqueID',
 'disbursed_amount',
 'asset_cost',
 'ltv',
 'branch_id',
 'supplier_id',
 'manufacturer_id',
 'Current_pincode_ID',
 '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',
 'NO.OF_INQUIRIES',
 'loan_default']
```

Getting Categorical columns from data set

[56]:

```
catcol = list(set(df.columns) - set(numcol))
catcol
```

[56]:

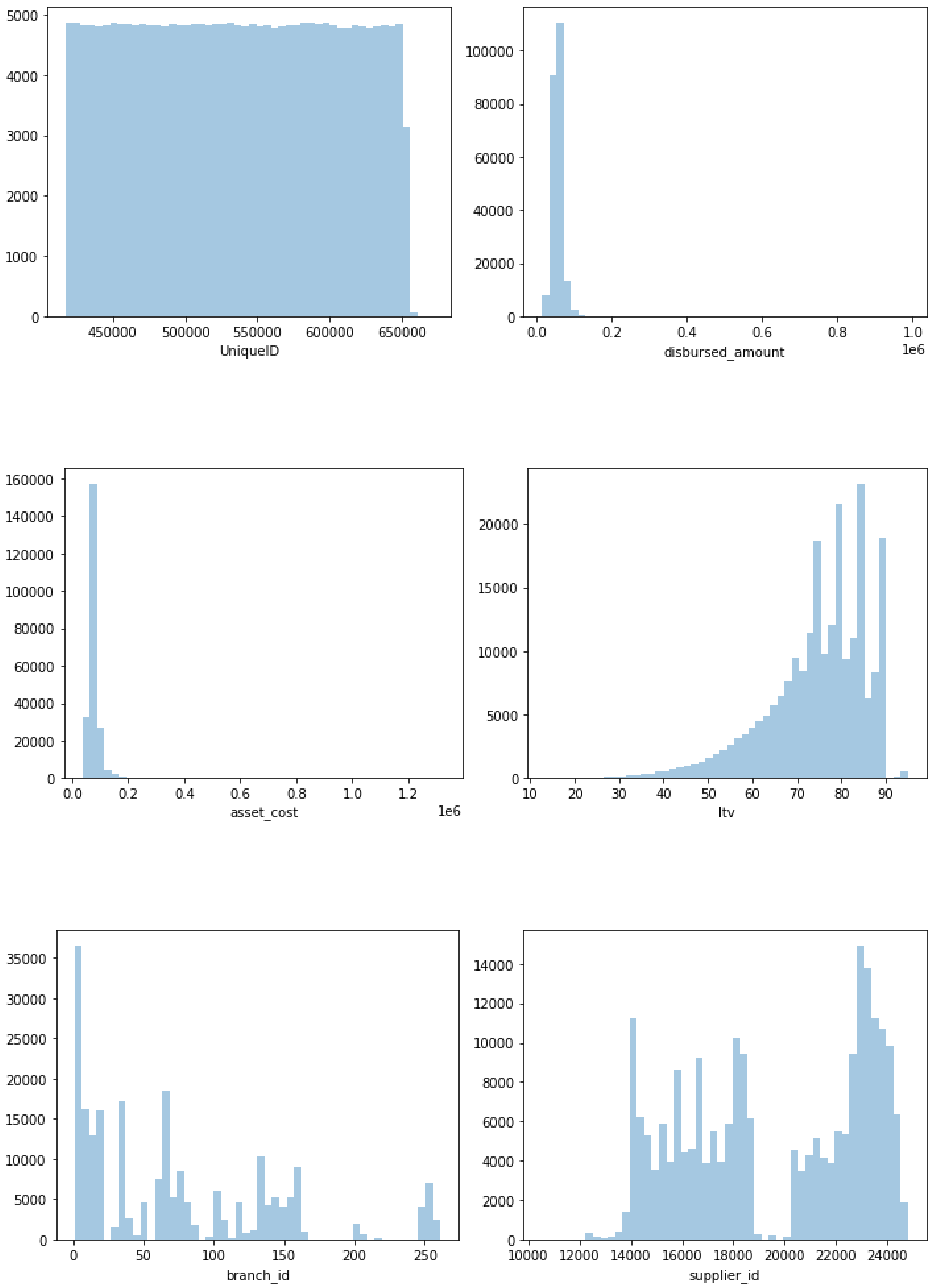
```
['Employment.Type',
 'PERFORM_CNS.SCORE.DESCRIPTION',
 'DisbursalDate',
 'Date.of.Birth',
 'AVERAGE.ACCT.AGE',
 'CREDIT.HISTORY.LENGTH']
```

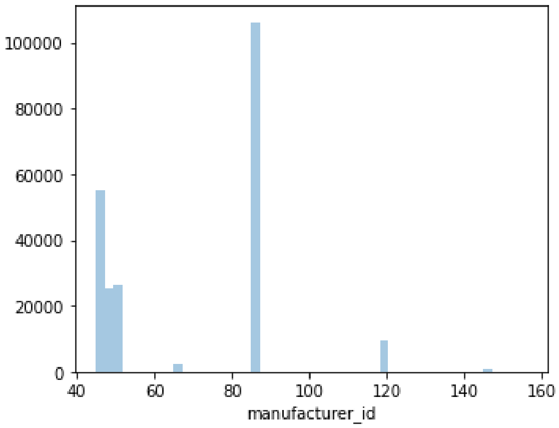
[57]:

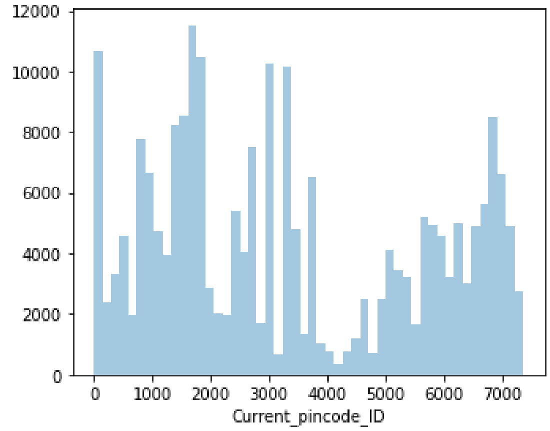
```
# Distribution plots for numerical categories
```

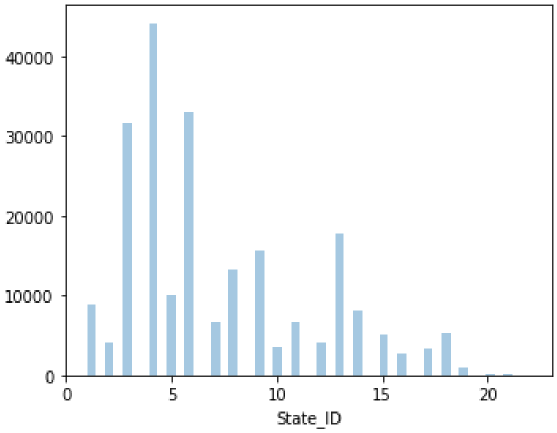
[58]:

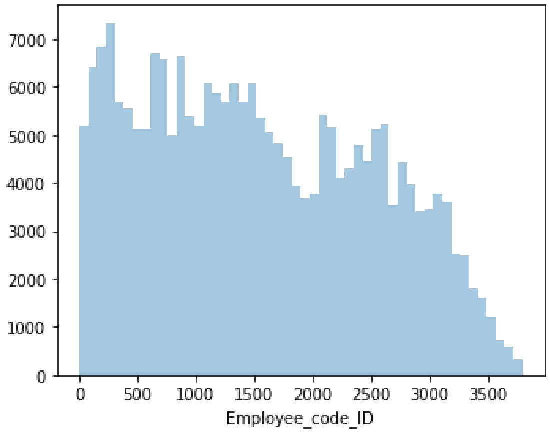
```
for i in range(0, len(numcol), 2):
    plt.figure(figsize=(10,4))
    plt.subplot(121)
    sns.distplot(df[numcol[i]], kde=False)
    plt.subplot(122)
    sns.distplot(df[numcol[i+1]], kde=False)
    plt.tight_layout()
    plt.show()
```

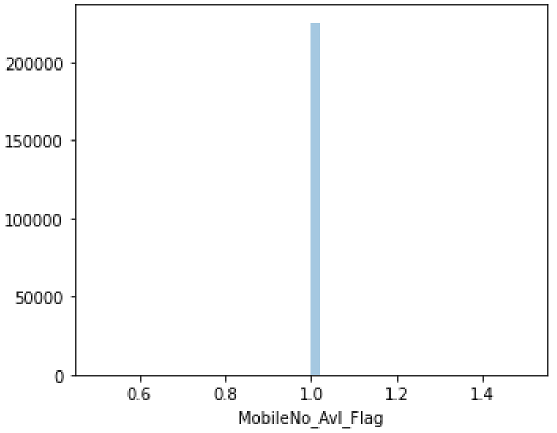

5000
4000
3000
2000
1000
0
450000
500000
550000
600000
650000
UniqueID
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
disbursed_amount
160000
140000
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
asset_cost
20000
15000
10000
5000
0
10
20
30
40
50
60
70
80
90
ltv
35000
30000
25000
20000
15000
10000
5000
0
0
50
100
150
200
250
branch_id
14000
12000
10000
8000
6000
4000
2000
0
10000
12000
14000
16000
18000
20000
22000
24000
supplier_id

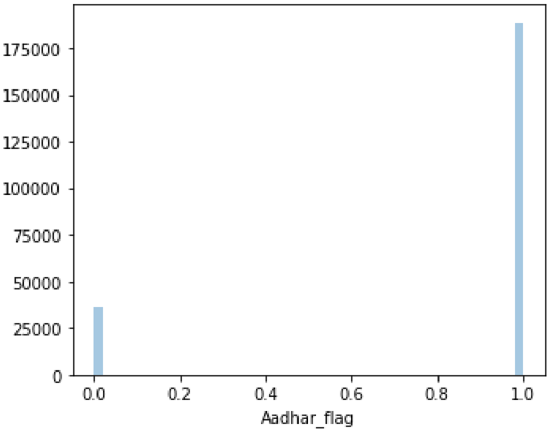

100000
80000
60000
40000
20000
0
40
60
80
100
120
140
160
manufacturer_id

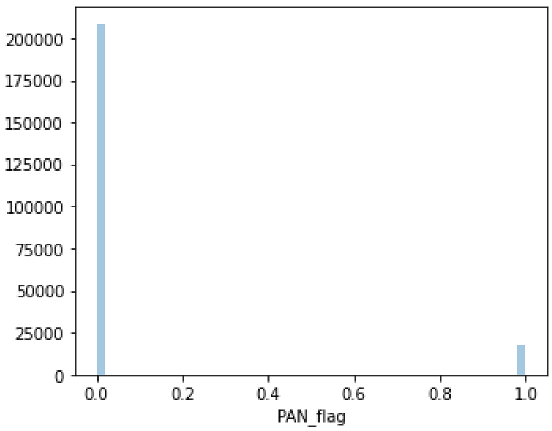

12000
10000
8000
6000
4000
2000
0
0
1000
2000
3000
4000
5000
6000
7000
Current_pincode_ID

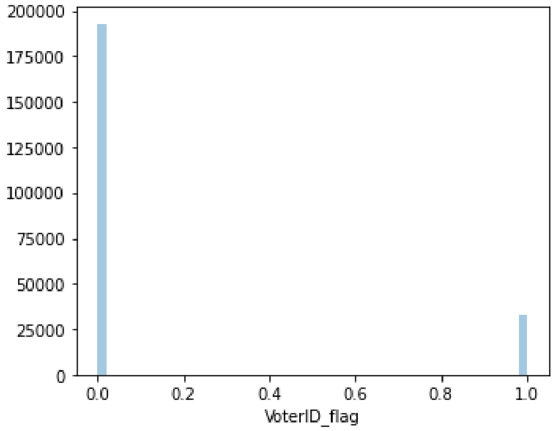

40000
30000
20000
10000
0
0
5
10
15
20
State_ID

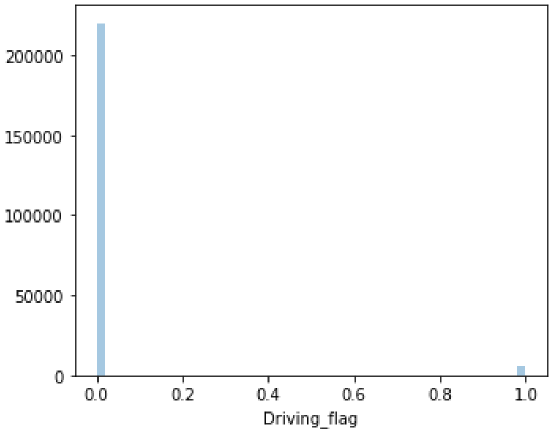

7000
6000
5000
4000
3000
2000
1000
0
0
500
1000
1500
2000
2500
3000
3500
Employee_code_ID

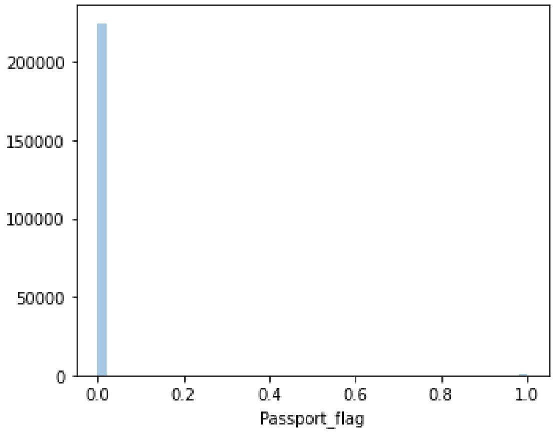

200000
150000
100000
50000
0
0.6
0.8
1.0
1.2
1.4
MobileNo_Avl_Flag

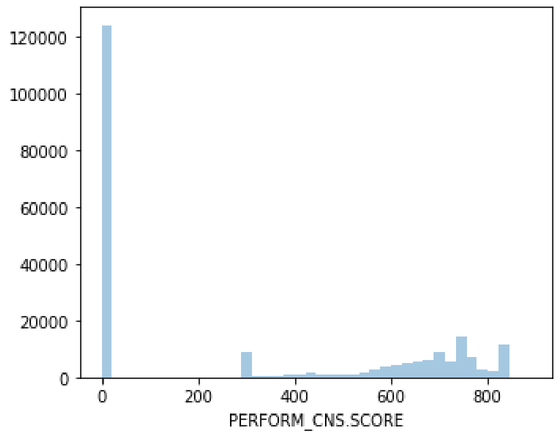

175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0
Aadhar_flag

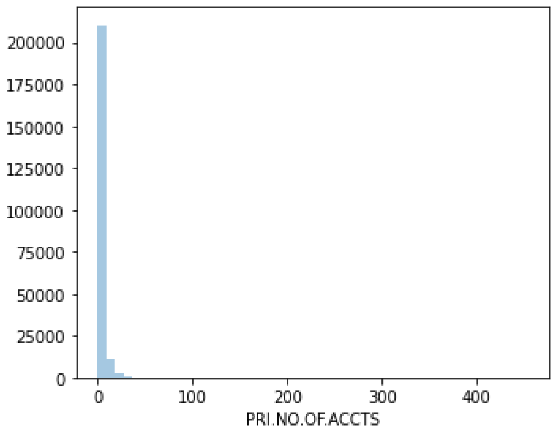

200000
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0
PAN_flag

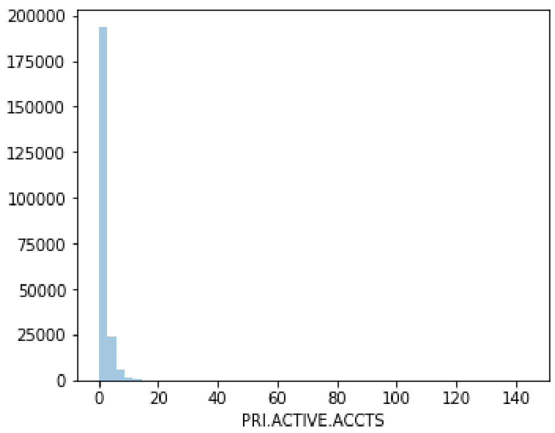

200000
175000
150000
125000
100000
75000
50000
25000
0
0.0
0.2
0.4
0.6
0.8
1.0
VoterID_flag

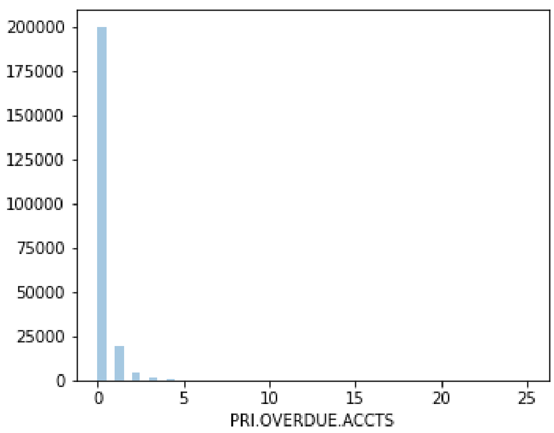

200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
Driving_flag

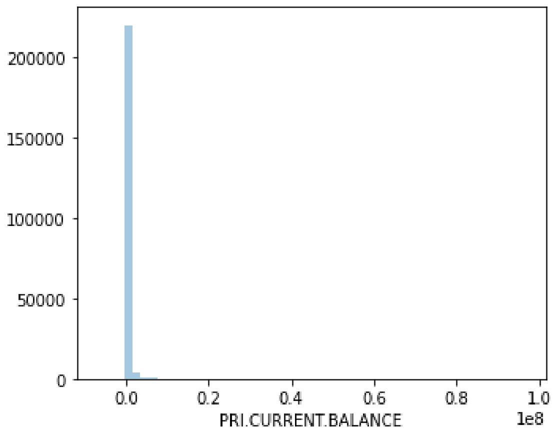

200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
Passport_flag

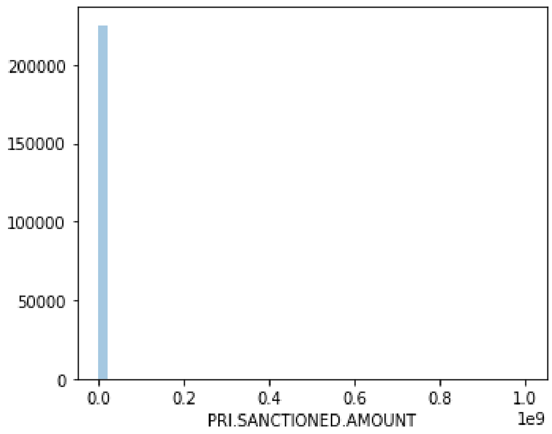

120000
100000
80000
60000
40000
20000
0
0
200
400
600
800
PERFORM_CNS.SCORE

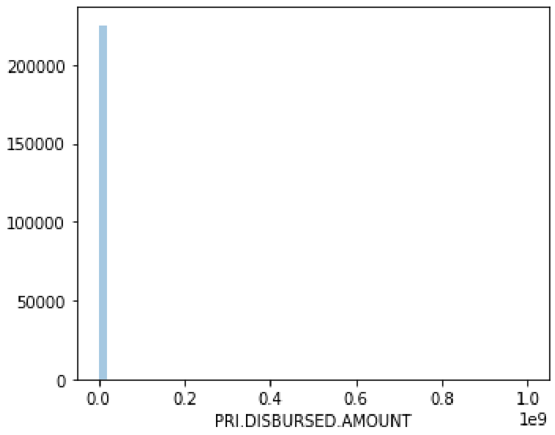

200000
175000
150000
125000
100000
75000
50000
25000
0
0
100
200
300
400
PRI.NO.OF.ACCTS

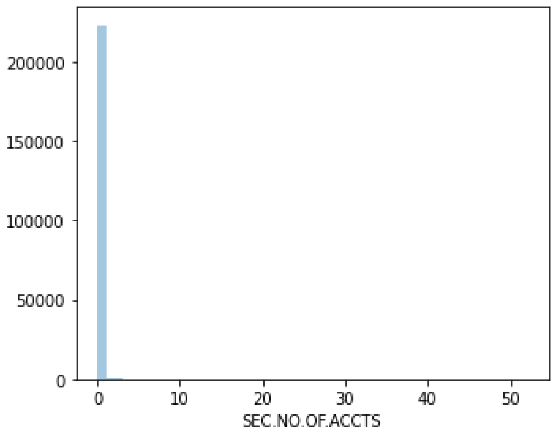

200000
175000
150000
125000
100000
75000
50000
25000
0
0
20
40
60
80
100
120
140
PRI.ACTIVE.ACCTS

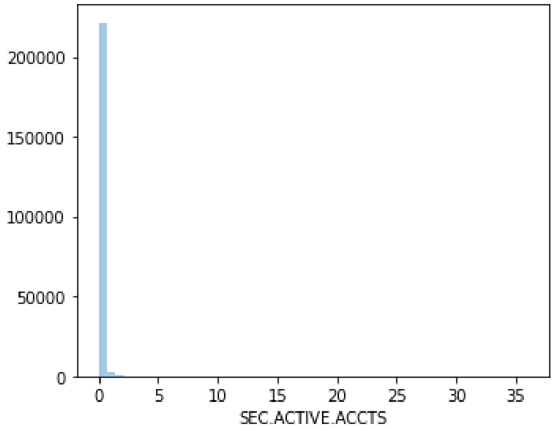

200000
175000
150000
125000
100000
75000
50000
25000
0
0
5
10
15
20
25
PRI.OVERDUE.ACCTS

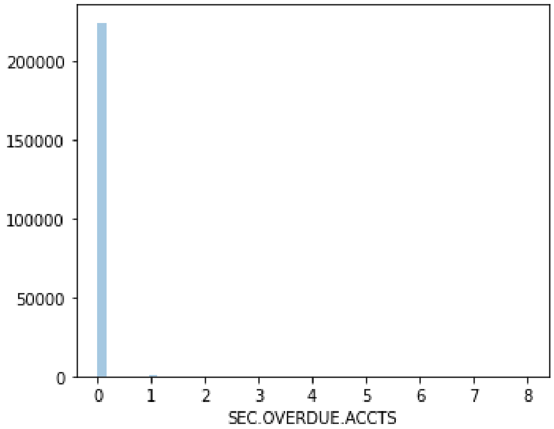

200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e8
PRI.CURRENT.BALANCE

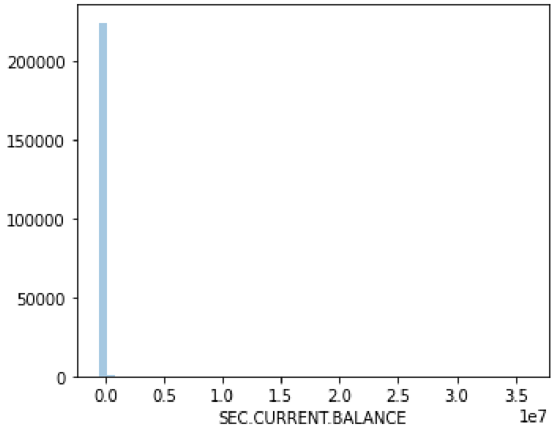

200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e9
PRI.SANCTIONED.AMOUNT

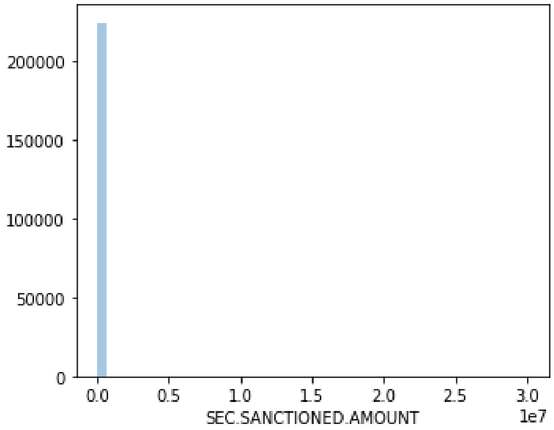

200000
150000
100000
50000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e9
PRI.DISBURSED.AMOUNT

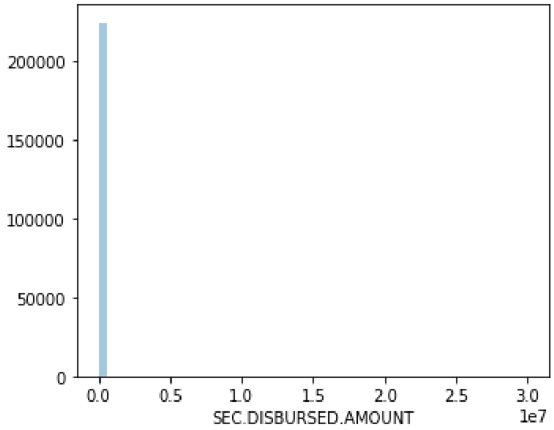

200000
150000
100000
50000
0
0
10
20
30
40
50
SEC.NO.OF.ACCTS

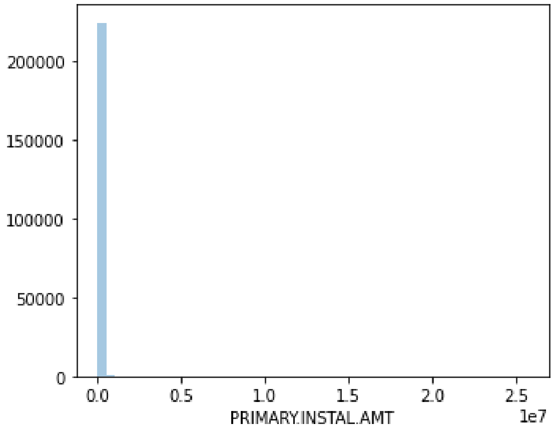

200000
150000
100000
50000
0
0
5
10
15
20
25
30
35
SEC.ACTIVE.ACCTS

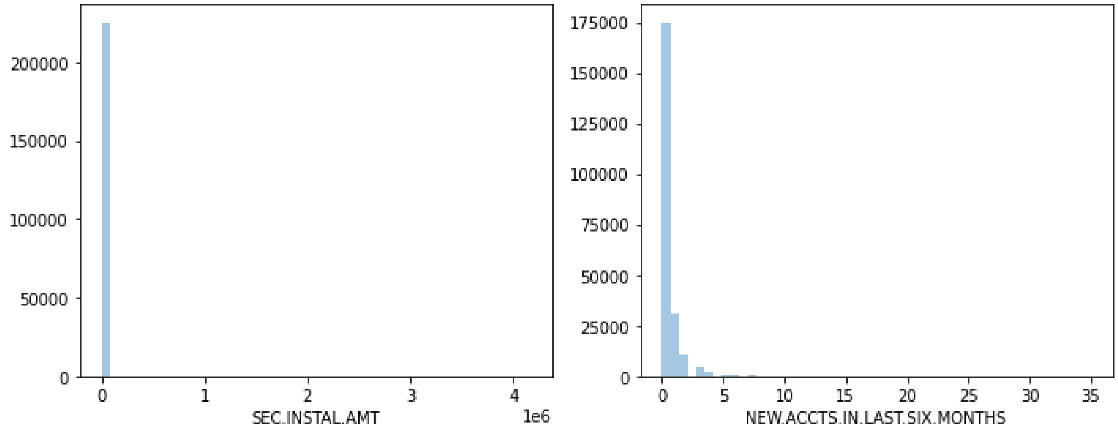

200000
150000
100000
50000
0
0
1
2
3
4
5
6
7
8
SEC.OVERDUE.ACCTS

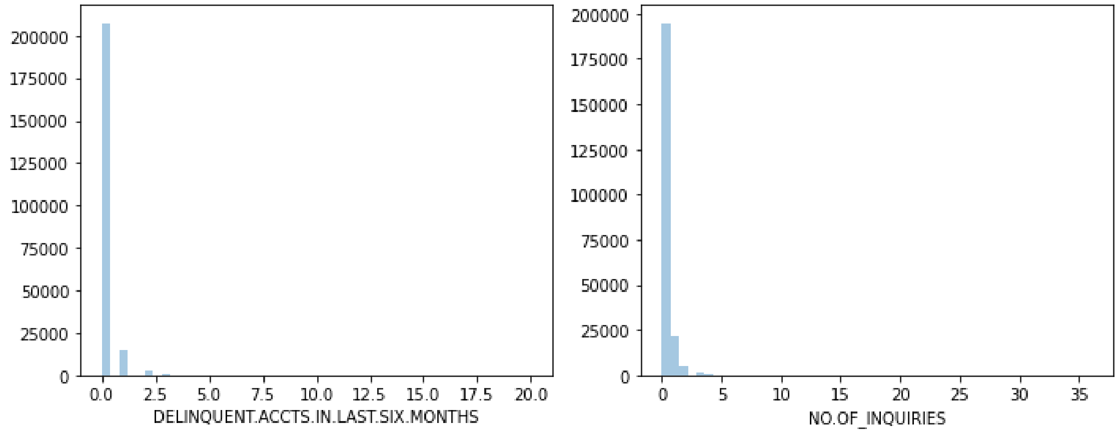

200000
150000
100000
50000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
SEC.CURRENT.BALANCE
1e7


200000
150000
100000
50000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
SEC.SANCTIONED.AMOUNT
1e7

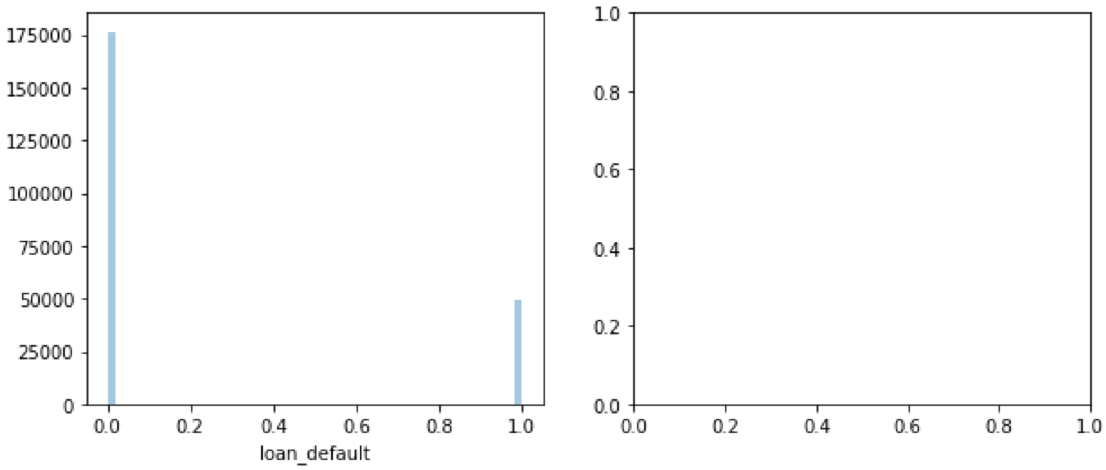

200000
150000
100000
50000
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
SEC.DISBURSED.AMOUNT
1e7


200000
150000
100000
50000
0
0.0
0.5
1.0
1.5
2.0
2.5
PRIMARY.INSTAL.AMT
1e7





```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/tmp/ipykernel_208/3886017166.py in <cell line: 1>()
      4     sns.distplot(df[numcol[i]], kde=False)
      5     plt.subplot(122)
----> 6     sns.distplot(df[numcol[i+1]], kde=False)
      7     plt.tight_layout()
      8     plt.show()

IndexError: list index out of range
```



Categorical Data Analysis

[59]:
```
df[catcol].head()
```

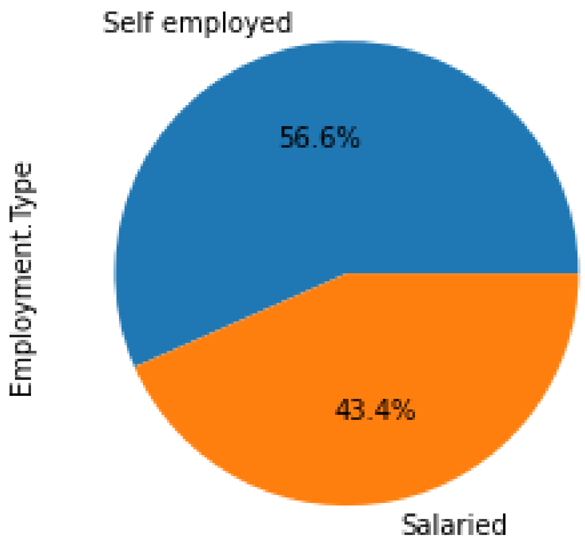
[59]:
```
  Employment.Type PERFORM_CNS.SCORE.DESCRIPTION DisbursalDate Date.of.Birth  \
0        Salaried    No Bureau History Available      03-08-18      01-01-84
1   Self employed                  I-Medium Risk      26-09-18      31-07-85
2   Self employed    No Bureau History Available      01-08-18      24-08-85
3   Self employed               L-Very High Risk      26-10-18      30-12-93
4   Self employed    No Bureau History Available      26-09-18      09-12-77

  AVERAGE.ACCT.AGE CREDIT.HISTORY.LENGTH
0        0yrs 0mon             0yrs 0mon
1       1yrs 11mon            1yrs 11mon
2        0yrs 0mon             0yrs 0mon
3        0yrs 8mon             1yrs 3mon
4        0yrs 0mon             0yrs 0mon
```

Two features AVERAGE.ACCT.AGE and CREDIT.HISTORY.LENGTH need to convert in terms of years.

[60]:
```
df['AVERAGE.ACCT.AGE'] = df['AVERAGE.ACCT.AGE'].str.replace('yrs ','.
 ↪',regex=False)
df['AVERAGE.ACCT.AGE'] = df['AVERAGE.ACCT.AGE'].str.
 ↪replace('mon','',regex=False).astype(float)
df['CREDIT.HISTORY.LENGTH'] = df['CREDIT.HISTORY.LENGTH'].str.replace('yrs ','.
 ↪',regex=False)
df['CREDIT.HISTORY.LENGTH'] = df['CREDIT.HISTORY.LENGTH'].str.
 ↪replace('mon','',regex=False).astype(float)
df[catcol].head()
```

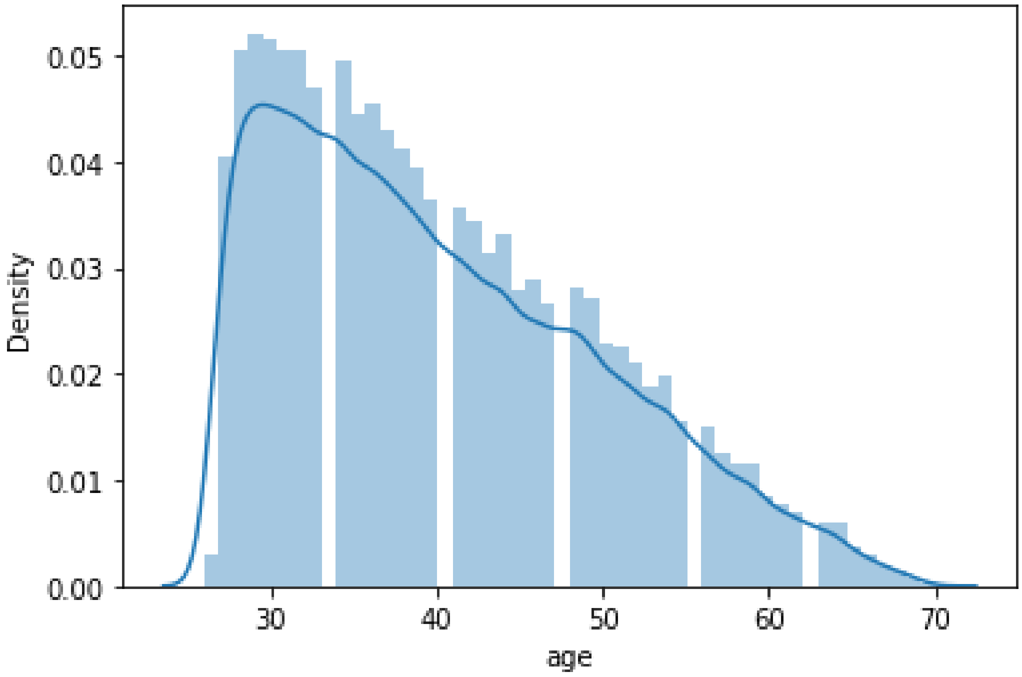
[60]:
```
  Employment.Type PERFORM_CNS.SCORE.DESCRIPTION DisbursalDate Date.of.Birth  \
0        Salaried   No Bureau History Available      03-08-18      01-01-84
1   Self employed                 I-Medium Risk      26-09-18      31-07-85
2   Self employed   No Bureau History Available      01-08-18      24-08-85
3   Self employed              L-Very High Risk      26-10-18      30-12-93
4   Self employed   No Bureau History Available      26-09-18      09-12-77

   AVERAGE.ACCT.AGE  CREDIT.HISTORY.LENGTH
0              0.00                   0.00
1              1.11                   1.11
2              0.00                   0.00
3              0.80                   1.30
4              0.00                   0.00
```

[61]:
```python
# Count the each category values from feature
df['Employment.Type'].value_counts().plot(kind='pie',autopct='%1.1f%%')
plt.show()
```



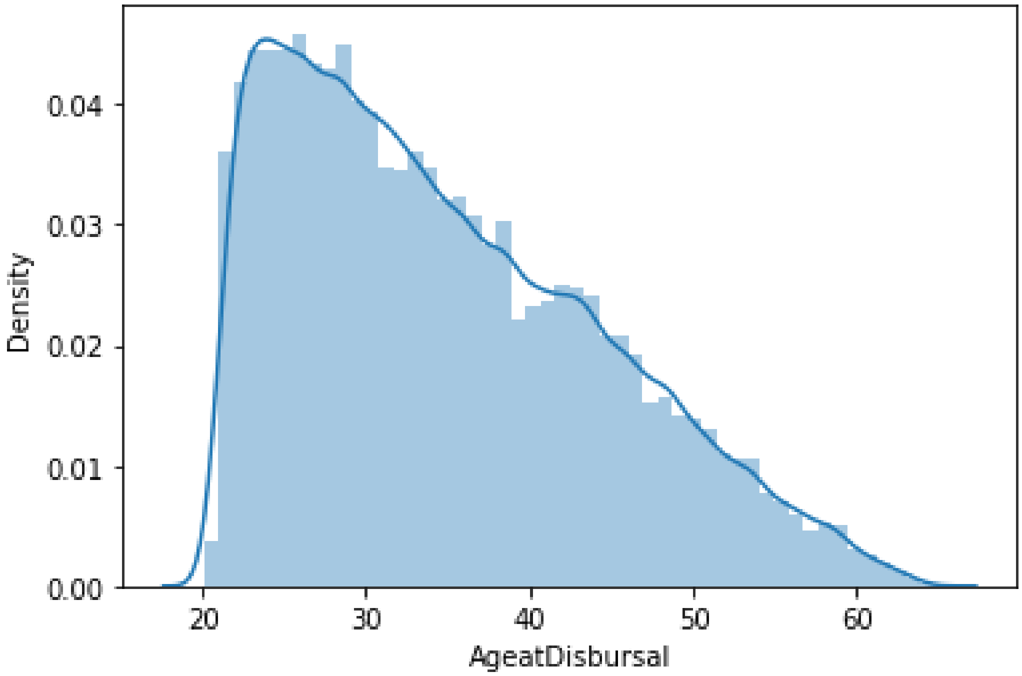
Age distribution of customers

[63]:
```python
df['Date.of.Birth'] = pd.to_datetime(df['Date.of.Birth'])
now = pd.Timestamp.now()
df['Date.of.Birth'] = df['Date.of.Birth'].where(df['Date.of.Birth'] < now,␣
 ↪df['Date.of.Birth'] -  np.timedelta64(100, 'Y'))   # 2
df['age'] = (now - df['Date.of.Birth']).astype('<m8[Y]')
sns.distplot(df['age'])
plt.show()
```



Age at Disbursal

```
[64]: df['DisbursalDate'] = pd.to_datetime(df['DisbursalDate'])
      df['Date.of.Birth'] = pd.to_datetime(df['Date.of.Birth'])

      # Calculate Age at Disbursal in years
      df['AgeatDisbursal'] = (df['DisbursalDate'] - df['Date.of.Birth']).dt.days /␣
        ↪365.25

      # Plot age at disbursal distribution
      sns.distplot(df['AgeatDisbursal'])
      plt.show()
```



[65]:
```
# Encode the values in terms of 0 and 1
df['Employment.Type'].replace({'Salaried': 0, 'Self employed': 1}, inplace=True)
```

[66]:
```
# Identify unique values in each features
df.nunique()
```

[66]:
```
UniqueID                                225493
disbursed_amount                         24228
asset_cost                               45415
ltv                                       6541
branch_id                                   82
supplier_id                               2945
manufacturer_id                             11
Current_pincode_ID                        6659
Date.of.Birth                            14417
Employment.Type                              2
DisbursalDate                               84
State_ID                                    22
Employee_code_ID                          3269
MobileNo_Avl_Flag                            1
Aadhar_flag                                  2
PAN_flag                                     2
VoterID_flag                                 2
Driving_flag                                 2
Passport_flag                                2
```

```
PERFORM_CNS.SCORE                         573
PERFORM_CNS.SCORE.DESCRIPTION              20
PRI.NO.OF.ACCTS                           107
PRI.ACTIVE.ACCTS                           40
PRI.OVERDUE.ACCTS                          22
PRI.CURRENT.BALANCE                     70044
PRI.SANCTIONED.AMOUNT                   43743
PRI.DISBURSED.AMOUNT                    47206
SEC.NO.OF.ACCTS                            37
SEC.ACTIVE.ACCTS                           23
SEC.OVERDUE.ACCTS                           9
SEC.CURRENT.BALANCE                      3197
SEC.SANCTIONED.AMOUNT                    2195
SEC.DISBURSED.AMOUNT                     2519
PRIMARY.INSTAL.AMT                      27608
SEC.INSTAL.AMT                           1890
NEW.ACCTS.IN.LAST.SIX.MONTHS               26
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS        14
AVERAGE.ACCT.AGE                          178
CREDIT.HISTORY.LENGTH                     269
NO.OF_INQUIRIES                            25
loan_default                                2
age                                        45
AgeatDisbursal                          14996
dtype: int64
```

Check Correlation

[112]:

```python
# Create correlation matrix
corr = df.corr()

# Set threshold for correlation
threshold = 0.5
corrlist = []

# Get the list of column names
cols = df.columns.tolist()

# Iterate through the correlation matrix
for i in range(1, corr.shape[0]):  # Ensure i doesn't go beyond the matrix size
    for j in range(i):
        # Check if the correlation is above the threshold and not equal to 1
        if (abs(corr.iloc[i, j]) > threshold) and (abs(corr.iloc[i, j]) < 1):
            corrlist.append([corr.iloc[i, j], i, j])

# Sort the list by correlation value, higher correlations first
sort_corr_list = sorted(corrlist, key=lambda x: abs(x[0]), reverse=True)
```

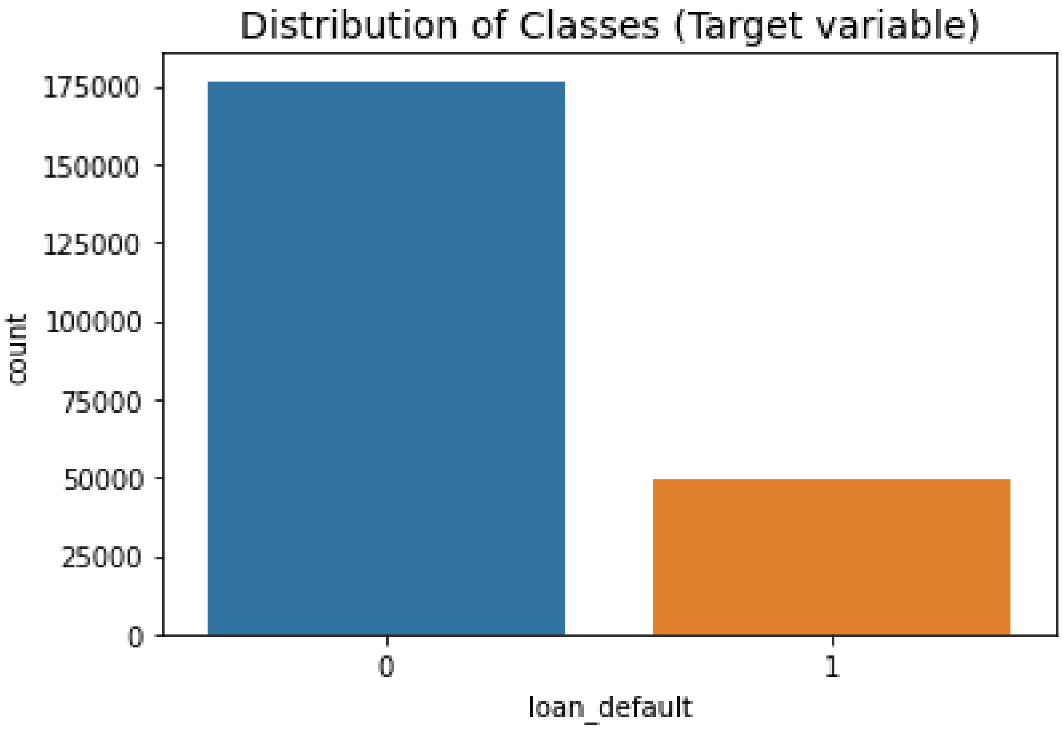
```
# Print correlations and column names
for corr_value, i, j in sort_corr_list:
    print(f"{cols[i]} and {cols[j]} = {round(corr_value, 2)}")
```

```
SEC.OVERDUE.ACCTS and SEC.ACTIVE.ACCTS = 1.0
NO.OF_INQUIRIES and CREDIT.HISTORY.LENGTH = 1.0
PRI.OVERDUE.ACCTS and PRI.ACTIVE.ACCTS = 1.0
SEC.OVERDUE.ACCTS and SEC.NO.OF.ACCTS = 0.93
SEC.ACTIVE.ACCTS and SEC.NO.OF.ACCTS = 0.93
Aadhar_flag and Employee_code_ID = -0.87
PRI.SANCTIONED.AMOUNT and PRI.CURRENT.BALANCE = 0.83
NEW.ACCTS.IN.LAST.SIX.MONTHS and SEC.INSTAL.AMT = 0.82
PERFORM_CNS.SCORE and Passport_flag = 0.75
asset_cost and disbursed_amount = 0.75
SEC.DISBURSED.AMOUNT and PERFORM_CNS.SCORE = 0.7
SEC.DISBURSED.AMOUNT and Passport_flag = 0.54
PRI.DISBURSED.AMOUNT and PRI.SANCTIONED.AMOUNT = 0.53
PRI.DISBURSED.AMOUNT and PRI.CURRENT.BALANCE = 0.51
```

Plotting distribution of classes of target variable

```
[69]: print('Distribution of the loan_default in the dataset')
print(df['loan_default'].value_counts()/len(df))

sns.countplot(data=df, x='loan_default')
plt.title('Distribution of Classes (Target variable)', fontsize=14)
plt.show()
```

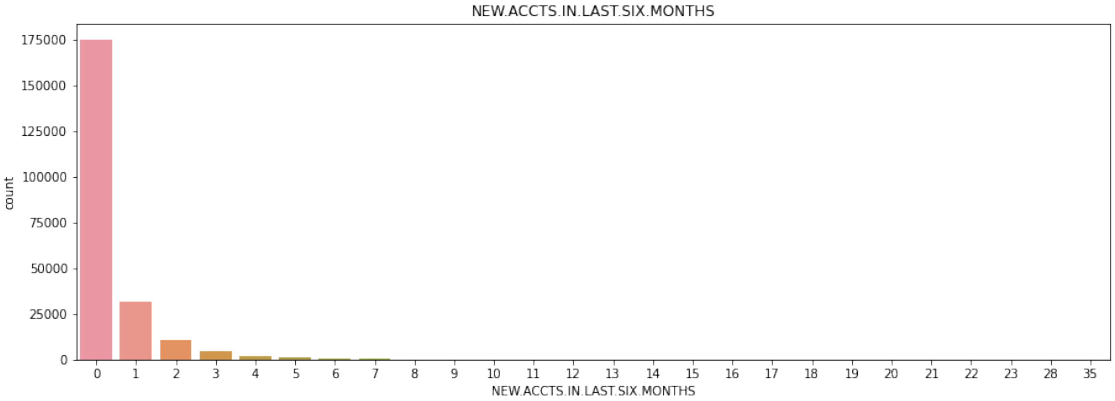
```
Distribution of the loan_default in the dataset
0    0.782845
1    0.217155
Name: loan_default, dtype: float64
```



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

No of Accounts created in last 6 months

```
[70]: plt.figure(figsize=(15,5))
      sns.countplot(data=df,x='NEW.ACCTS.IN.LAST.SIX.MONTHS')
      plt.title("NEW.ACCTS.IN.LAST.SIX.MONTHS")
      plt.show()
```

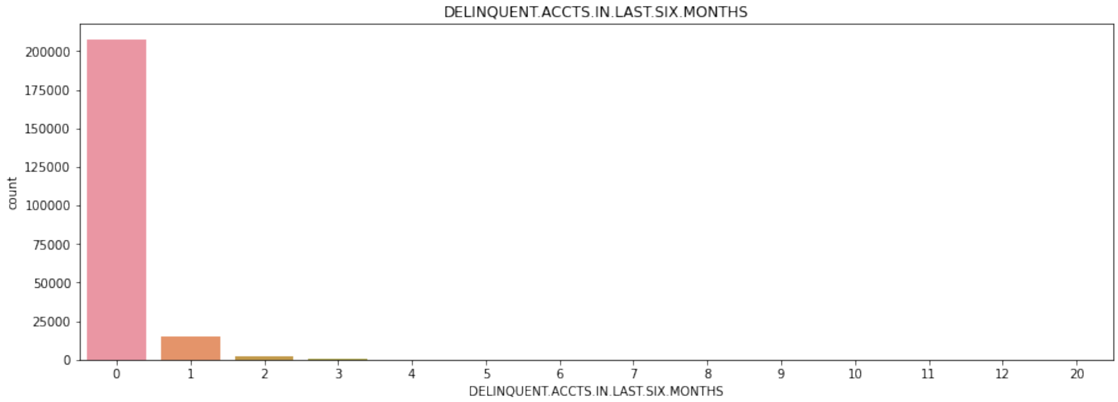


[71]:
```
df['NEW.ACCTS.IN.LAST.SIX.MONTHS'].value_counts(normalize=100).head()
```

[71]:
```
0    0.775829
1    0.139077
2    0.047922
3    0.019402
4    0.008506
Name: NEW.ACCTS.IN.LAST.SIX.MONTHS, dtype: float64
```

It can be infered around 77% of the customers have zero new loans taken by the them in last 6 months before the disbursment

[72]:
```
plt.figure(figsize=(15,5))
sns.countplot(data=df, x='DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS')
plt.title("DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS")
plt.show()
```

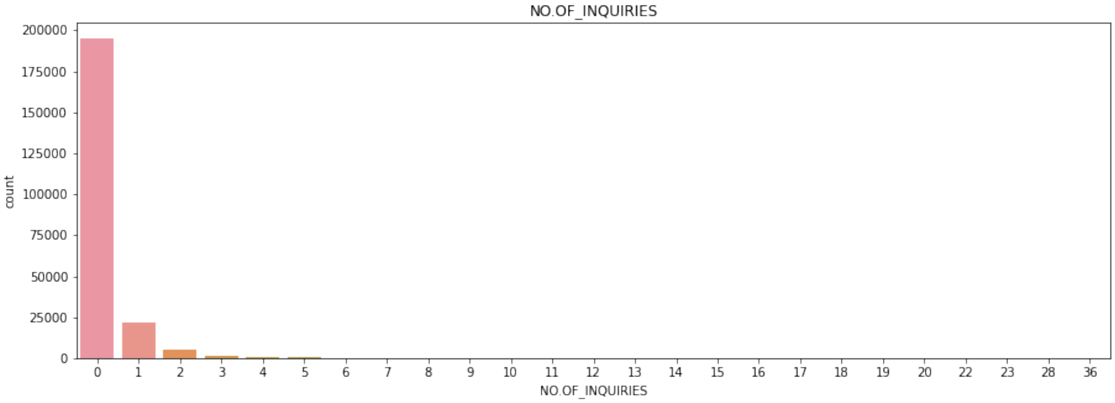


[73]:
```
df['DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS'].value_counts(normalize=100).head()
```

[73]:
```
0    0.920858
1    0.065102
2    0.010666
3    0.002302
4    0.000603
Name: DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS, dtype: float64
```

It can be infered that around 86% of the customers have no loans defaulted in the last six months

[74]:
```
# No of Enquiries
plt.figure(figsize=(15,5))
sns.countplot(data=df,x='NO.OF_INQUIRIES')
plt.title("NO.OF_INQUIRIES")
plt.show()
```



```
[75]: df['NO.OF_INQUIRIES'].value_counts(normalize=100).head()
```

```
[75]: 0    0.864728
      1    0.096650
      2    0.023477
      3    0.007645
      4    0.003304
      Name: NO.OF_INQUIRIES, dtype: float64
```

Around 86% of the customers have not made any enquiries before taking the loans

```
[77]: df = df.replace({'PERFORM_CNS.SCORE.DESCRIPTION':{'C-Very Low Risk':'Low',␣
        ↪'A-Very Low Risk':'Low',
                                                        'B-Very Low Risk':'Low',␣
        ↪'D-Very Low Risk':'Low',
                                                        'F-Low Risk':'Low',␣
        ↪'E-Low Risk':'Low', 'G-Low Risk':'Low',
                                                        'H-Medium Risk':␣
        ↪'Medium', 'I-Medium Risk': 'Medium',
                                                        'J-High Risk':'High',␣
        ↪'K-High Risk':'High','L-Very High Risk':'Very High',
                                                        'M-Very High Risk':'Very␣
        ↪High','Not Scored: More than 50 active Accounts found':'Not Scored',
                                                        'Not Scored: Only a␣
        ↪Guarantor':'Not Scored','Not Scored: Not Enough Info available on the␣
        ↪customer':'Not Scored',
                                                         'Not Scored: No␣
        ↪Activity seen on the customer (Inactive)':'Not Scored','Not Scored: No␣
        ↪Updates available in last 36 months':'Not Scored',
                                                        'Not Scored: Sufficient␣
        ↪History Not Available':'Not Scored', 'No Bureau History Available':'Not␣
        ↪Scored'
```

```
                                                                }})
```

[78]:

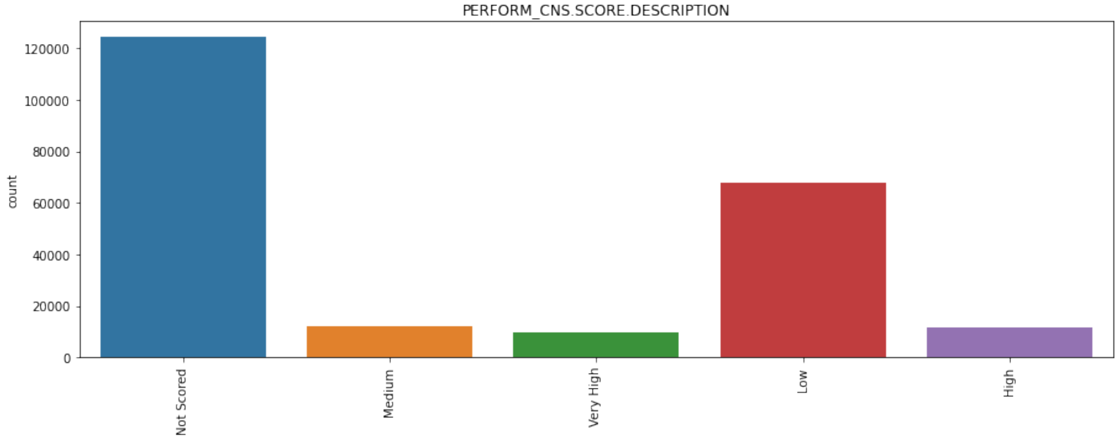
```
df['PERFORM_CNS.SCORE.DESCRIPTION'].value_counts()
```

[78]:

```
Not Scored    124253
Low            67577
Medium         12135
High           11774
Very High       9754
Name: PERFORM_CNS.SCORE.DESCRIPTION, dtype: int64
```

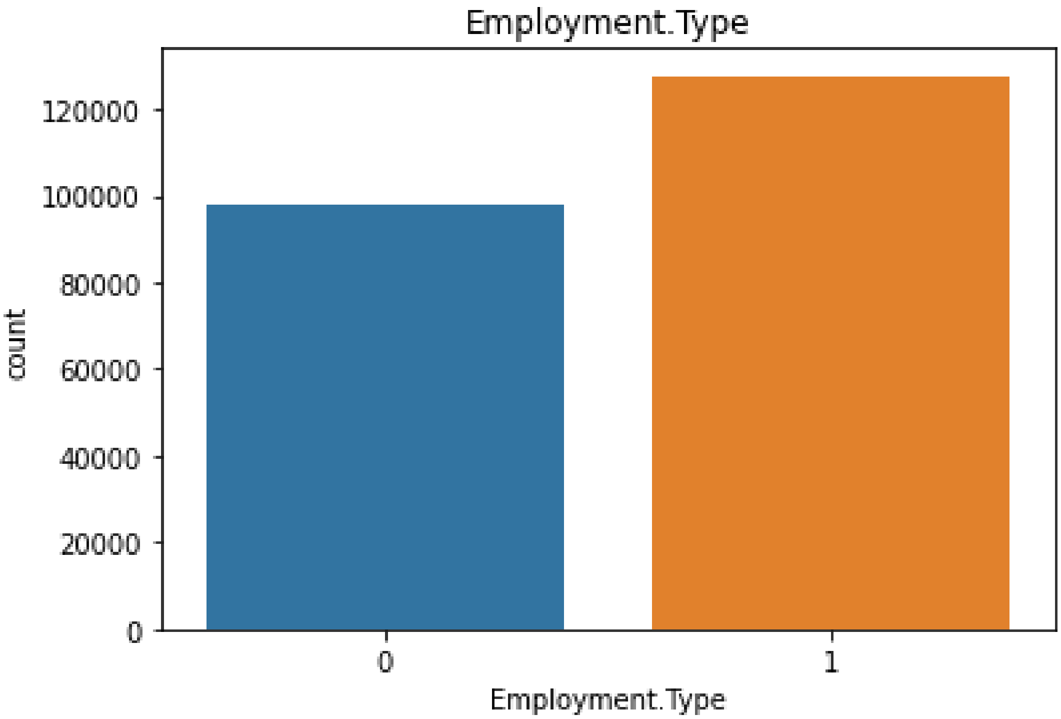
[79]:

```
plt.figure(figsize=(15,5))
sns.countplot(x=df['PERFORM_CNS.SCORE.DESCRIPTION'])
plt.title("PERFORM_CNS.SCORE.DESCRIPTION")
plt.xticks(rotation=90)
plt.show()
```



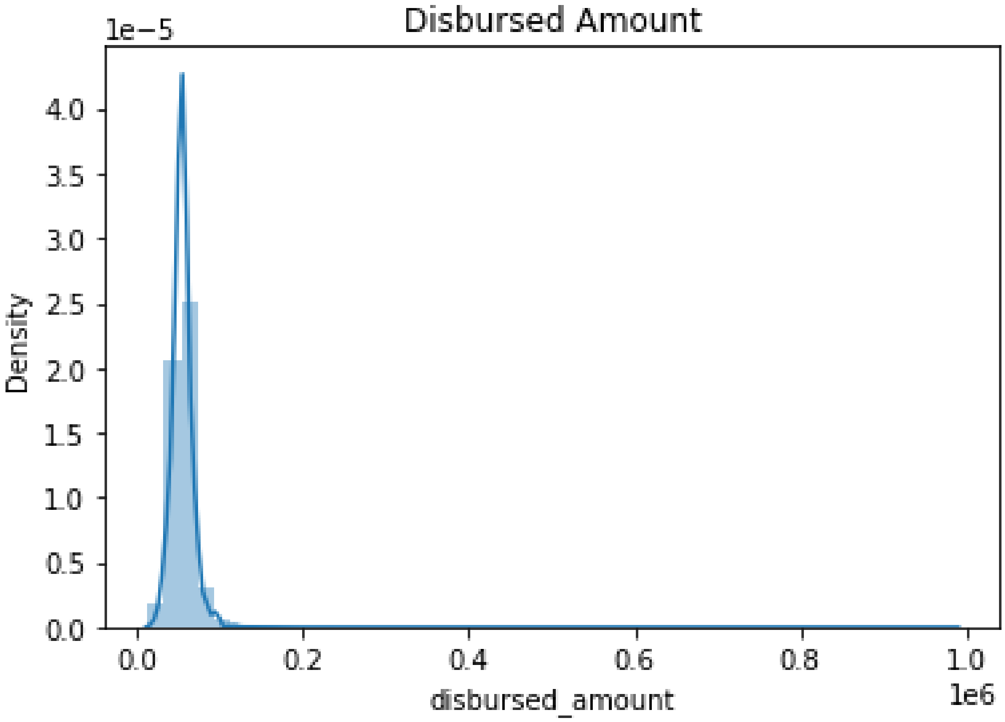
[80]:

```
sns.countplot(data=df,x='Employment.Type')
plt.title("Employment.Type")
plt.show()
```

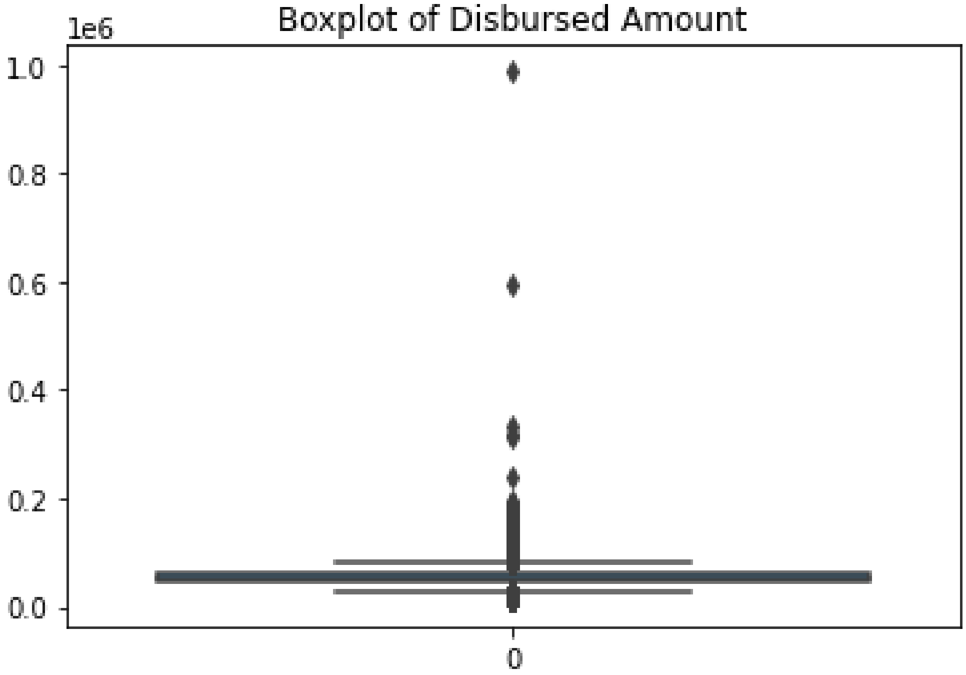


[81]:

```
#disbursed_amount = Amount of Loan disbursed
sns.distplot(df['disbursed_amount'])
plt.title("Disbursed Amount")
plt.show()
```

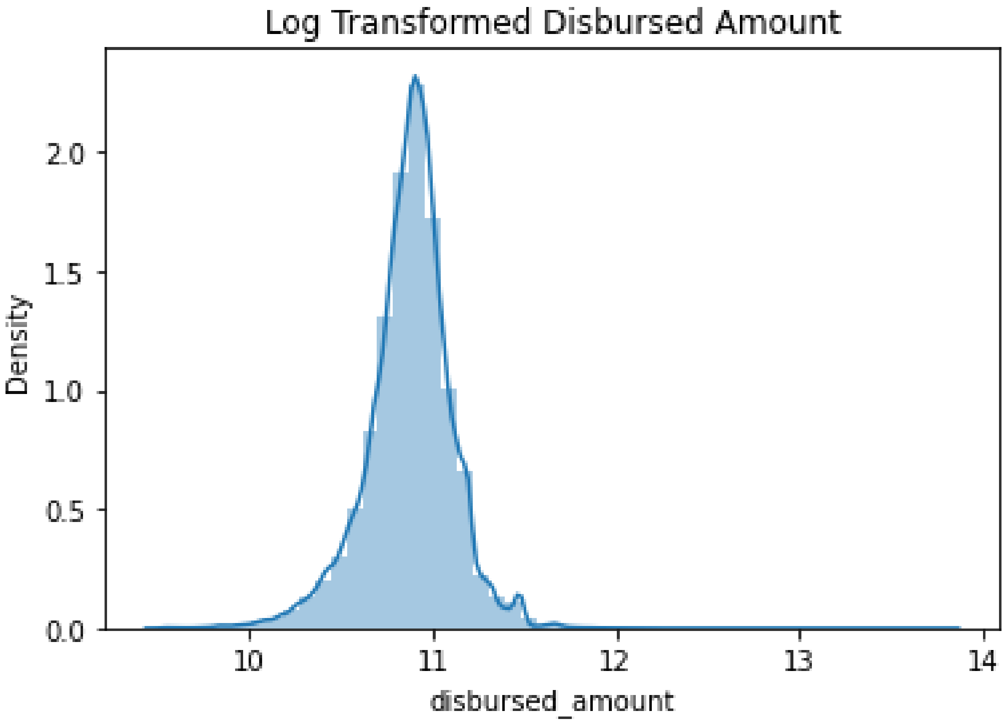


[82]:
```
sns.boxplot(df['disbursed_amount'])
plt.title("Boxplot of Disbursed Amount")
plt.show()
```

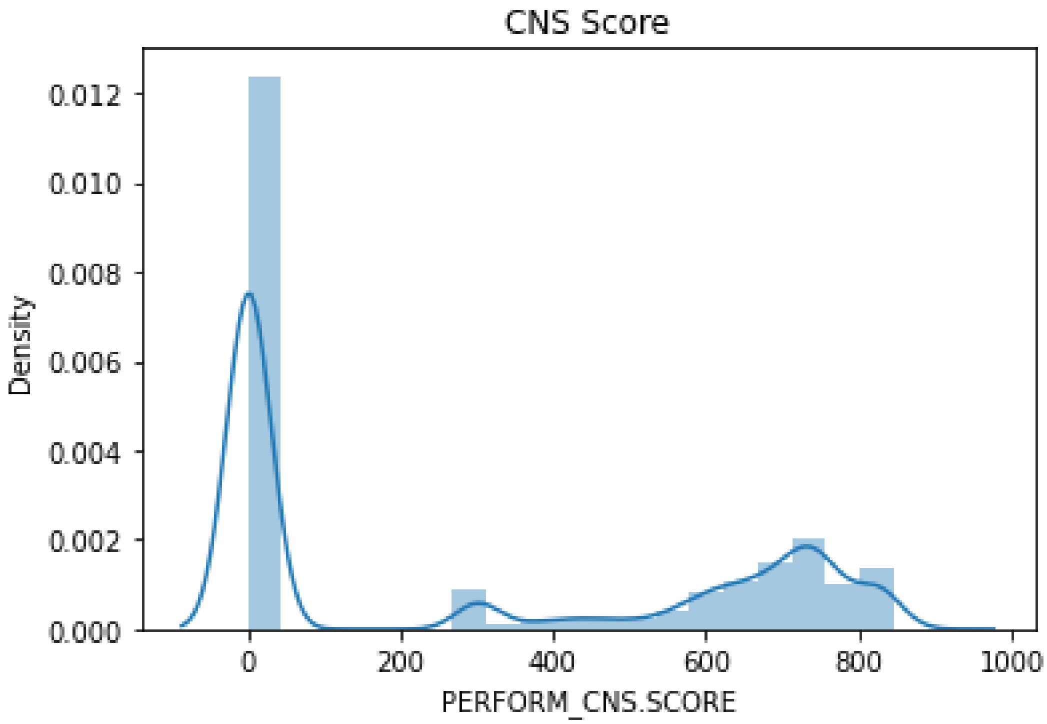


It is heavily right skewed data.

```
[83]: sns.distplot(np.log(df['disbursed_amount']))
plt.title("Log Transformed Disbursed Amount")
plt.show()
```



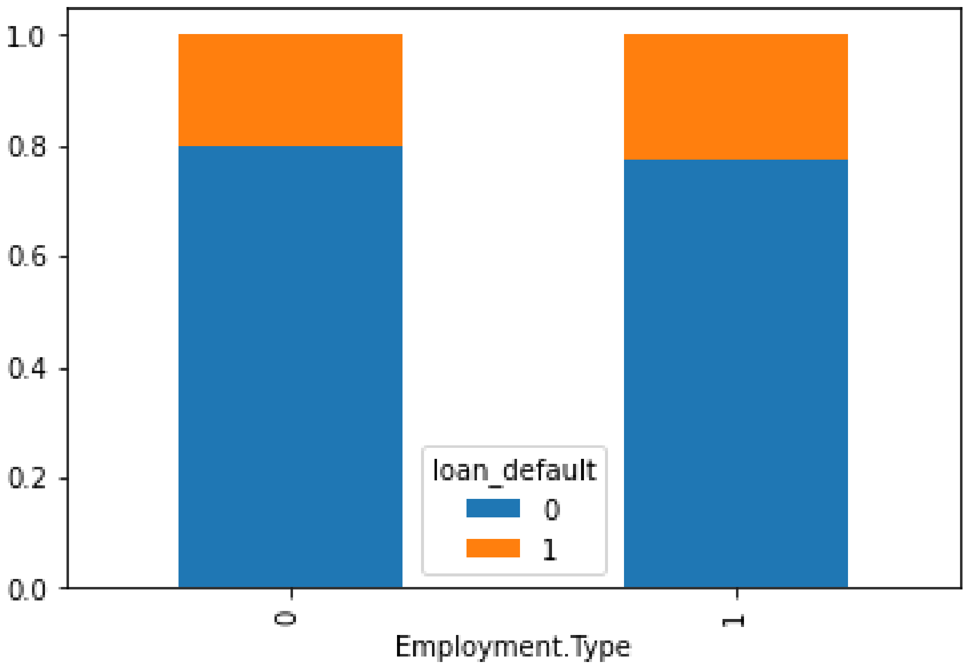
```
[84]: sns.distplot(df['PERFORM_CNS.SCORE'],bins=20)
plt.title("CNS Score")
plt.show()
```



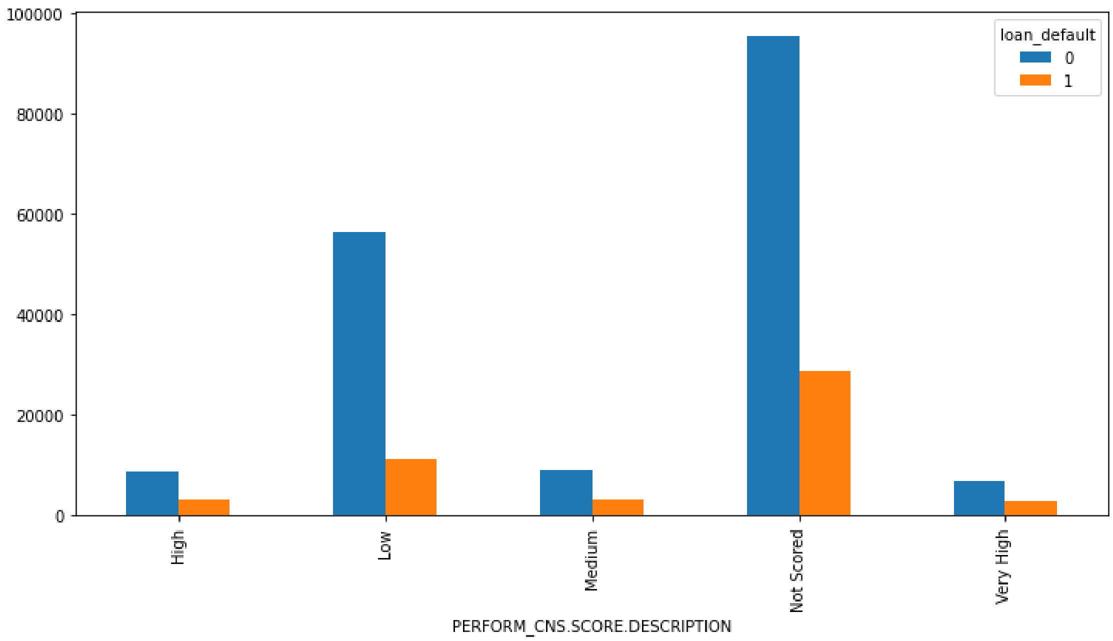
Bivariate Analysis

Employment type Vs Loan Default

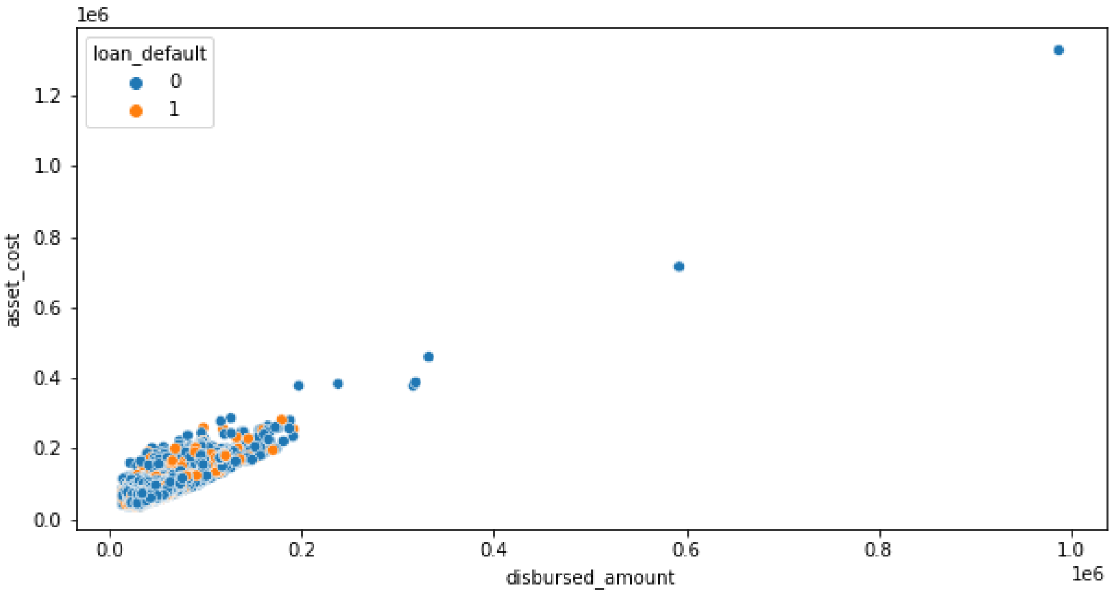
[85]:
```
a = pd.crosstab(df['Employment.Type'],df['loan_default'],normalize="index")
a.plot(kind='bar', stacked = True)
plt.show()
```



[86]:
```
b = pd.crosstab(df['PERFORM_CNS.SCORE.DESCRIPTION'],df['loan_default'])
b.plot.bar(figsize=(12,6))
plt.show()
```



[87]:
```
plt.figure(figsize=(10,5))
sns.
 ↪scatterplot(x=df["disbursed_amount"],y=df["asset_cost"],hue=df['loan_default'])
plt.show()
```

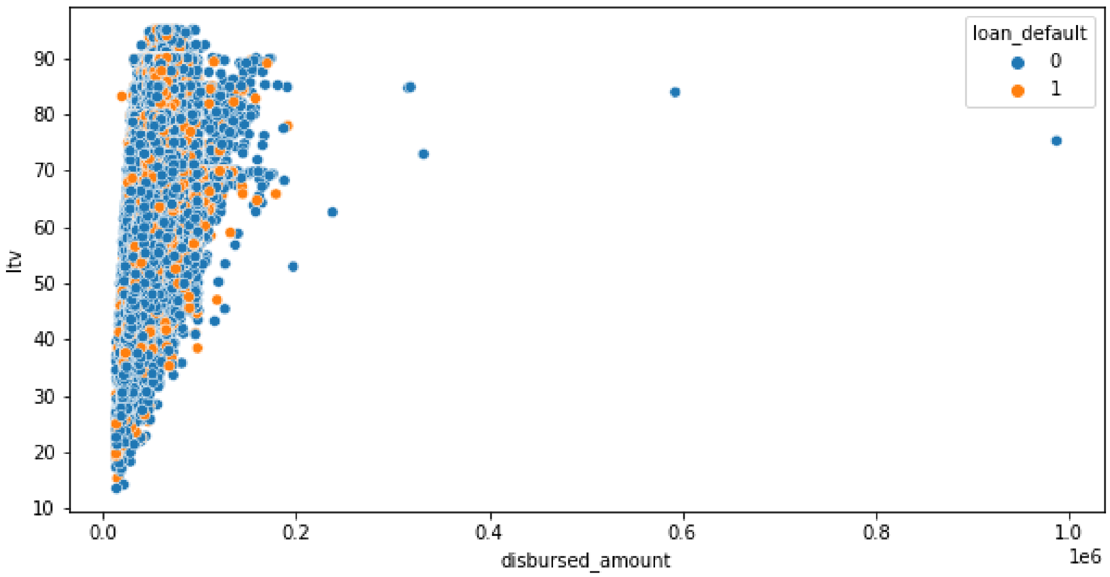


[88]:
```
df[["disbursed_amount","asset_cost"]].corr()
```

[88]:
```
                  disbursed_amount  asset_cost
disbursed_amount          1.000000    0.746037
asset_cost                0.746037    1.000000
```

Strong correlation between disbursed amount and asset cost

[89]:
```
plt.figure(figsize=(10,5))
sns.
 ↪scatterplot(x=df["disbursed_amount"],y=df["ltv"],hue=df['loan_default'],markers="x")
plt.show()
```



Moderate Correlation

```
[ ]: Multivariate Analysis
```

```
[98]: new_df = df.copy()
      new_df.columns
```

```
[98]: Index(['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', 'age', 'AgeatDisbursal'],
            dtype='object')
```

Modelling

```
[104]: X = new_df.drop(['loan_default','Date.of.Birth','PERFORM_CNS.SCORE.
        ↪DESCRIPTION','DisbursalDate'], axis=1)
       y = new_df['loan_default'].copy()
```

[105]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
 ↪3,random_state=42)
print("X_train size: ", X_train.shape)
print("X_test size: ", X_test.shape)
```

```
X_train size:  (157845, 39)
X_test size:  (67648, 39)
```

Build and evaluate models Define evaluation function which calculates following metrics:

Confusion matrix Accuracy score Precision Recall F1 score ROC AUC score

[102]:

```
def evaluate_model(y_test, y_pred):
    print("Confusion Matrix: \n", metrics.confusion_matrix(y_test, y_pred))
    print("Accuracy: ",metrics.accuracy_score(y_test, y_pred))
    print("Precision: ",metrics.precision_score(y_test, y_pred))
    print("Recall: ",metrics.recall_score(y_test, y_pred))
    print("f1 score: ",metrics.f1_score(y_test, y_pred))
    print("roc_auc_score: ",metrics.roc_auc_score(y_test, y_pred))
```

[106]:

```
# Scaling training and testing data
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

1. Logistic Regression

[107]:

```
# Find best parameters using grid search
params = {'C':[0.1, 0.5, 1, 5]}

lr = LogisticRegression()
grid = GridSearchCV(estimator=lr, param_grid=params)
grid.fit(X_train, y_train)
y_pred = grid.predict(X_test)
evaluate_model(y_test, y_pred)
```

```
Confusion Matrix:
 [[52636    86]
 [14848    78]]
Accuracy:  0.7792395931882686
Precision:  0.47560975609756095
Recall:  0.005225780517218277
f1 score:  0.010337972166998012
roc_auc_score:  0.5017972914573496
```

2. Decision Tree

[108]:

```
params = {'criterion':['gini','entropy'], 'max_depth': [2,3,4,5]}
dt = DecisionTreeClassifier()
dt_clf = GridSearchCV(dt, params)
dt_clf.fit(X_train, y_train)
y_pred = dt_clf.predict(X_test)
evaluate_model(y_test, y_pred)
```

```
Confusion Matrix:
 [[52722     0]
 [14926     0]]
Accuracy:  0.7793578524124882
Precision:  0.0
Recall:  0.0
f1 score:  0.0
roc_auc_score:  0.5
```

3. Random Forest

[109]:

```
rf = RandomForestClassifier(n_estimators=250, random_state=42)
rf.fit(X_train,y_train)
y_pred = rf.predict(X_test)
evaluate_model(y_test, y_pred)
```

```
Confusion Matrix:
 [[52222   500]
 [14494   432]]
Accuracy:  0.7783526490066225
Precision:  0.463519313304721
Recall:  0.028942784403055507
f1 score:  0.054483541430192954
roc_auc_score:  0.509729538705833
```