processing

August 1, 2025

[1]:
```python
import pandas as pd
from tabulate import tabulate

# Loading data
data = pd.read_csv('train.csv')

# Replacing periods in the column names with underscores to avoid syntax issues␣
↪when referencing columns
data.columns = data.columns.str.replace('.', '_')

# Loading data that provides descriptions for each column in the main data
data_information = pd.read_csv("data_dictionary.csv")

# Extracting only Variable Name and Description columns
column_information = data_information[["Variable Name", "Description"]]

# Printing the variable names and descriptions
print(tabulate(column_information, headers="keys", tablefmt="pretty",␣
↪showindex=False))
```

```
+--------------------------------------+----------------------------------------
--------------------------------------------+
|            Variable Name             |
Description                                          |
+--------------------------------------+----------------------------------------
--------------------------------------------+
|               UniqueID               |                             Identifier
for customers                                    |
|             loan_default             |                    Payment default in
the first EMI on due date                           |
|           disbursed_amount           |                             Amount of
Loan disbursed                                     |
|              asset_cost              |                                Cost of
the Asset                                        |
|                 ltv                  |                              Loan to
Value of the asset                                     |
|              branch_id               |                        Branch where the
loan was disbursed                               |
```

```
|             supplier_id             |                   Vehicle Dealer where
the loan was disbursed                     |
|           manufacturer_id           |                   Vehicle
manufacturer(Hero, Honda, TVS etc.)                     |
|           Current_pincode           |                         Current pincode
of the customer                           |
|            Date.of.Birth            |                          Date of birth
of the customer                            |
|           Employment.Type           |             Employment Type of the
customer (Salaried/Self Employed)              |
|            DisbursalDate            |                               Date of
disbursement                                 |
|              State_ID               |                              State of
disbursement                                 |
|          Employee_code_ID           |             Employee of the
organization who logged the disbursement             |
|          MobileNo_Avl_Flag          |            if Mobile no. was shared by
the customer then flagged as 1             |
|             Aadhar_flag             |              if aadhar was shared by
the customer then flagged as 1               |
|              PAN_flag               |               if pan was shared by the
customer then flagged as 1                 |
|            VoterID_flag             |              if voter  was shared by
the customer then flagged as 1               |
|            Driving_flag             |                 if DL was shared by the
customer then flagged as 1                 |
|            Passport_flag            |             if passport was shared by
the customer then flagged as 1              |
|          PERFORM_CNS.SCORE          |
Bureau Score                                     |
|    PERFORM_CNS.SCORE.DESCRIPTION    |                             Bureau
score description                           |
|           PRI.NO.OF.ACCTS           |       count of total loans taken by the
customer at the time of disbursement       |
|          PRI.ACTIVE.ACCTS           |      count of active loans taken by the
customer at the time of disbursement       |
|          PRI.OVERDUE.ACCTS          |              count of default accounts
at the time of disbursement                |
|         PRI.CURRENT.BALANCE         | total Principal outstanding amount of
the active loans at the time of disbursement |
|        PRI.SANCTIONED.AMOUNT        |   total amount that was sanctioned for
all the loans at the time of disbursement   |
|        PRI.DISBURSED.AMOUNT         |   total amount that was disbursed for
all the loans at the time of disbursement    |
|           SEC.NO.OF.ACCTS           |       count of total loans taken by the
customer at the time of disbursement       |
|          SEC.ACTIVE.ACCTS           |      count of active loans taken by the
customer at the time of disbursement       |
```

```
|          SEC.OVERDUE.ACCTS          |               count of default accounts
at the time of disbursement                 |
|         SEC.CURRENT.BALANCE         | total Principal outstanding amount of
the active loans at the time of disbursement |
|        SEC.SANCTIONED.AMOUNT        |   total amount that was sanctioned for
all the loans at the time of disbursement   |
|        SEC.DISBURSED.AMOUNT         |   total amount that was disbursed for
all the loans at the time of disbursement    |
|         PRIMARY.INSTAL.AMT          |                            EMI Amount of
the primary loan                            |
|           SEC.INSTAL.AMT            |                            EMI Amount of
the secondary loan                           |
|    NEW.ACCTS.IN.LAST.SIX.MONTHS     |      New loans taken by the customer in
last 6 months before the disbursment        |
| DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS |                        Loans defaulted
in the last 6 months                         |
|          AVERAGE.ACCT.AGE           |                                Average
loan tenure                                  |
|        CREDIT.HISTORY.LENGTH        |                               Time since
first loan                                 |
|           NO.OF_INQUIRIES           |                         Enquries done by
the customer for loans                        |
+-------------------------------------+----------------------------------------
-------------------------------------------+

C:\Users\jmand\AppData\Local\Temp\ipykernel_23584\3406735122.py:8:
FutureWarning: The default value of regex will change from True to False in a
future version. In addition, single character regular expressions will *not* be
treated as literal strings when regex=True.
  data.columns = data.columns.str.replace('.', '_')
```

[2]:
```
data.describe().T
```

[2]:

	count	mean	std \
UniqueID	233154.0	535917.573376	6.831569e+04
disbursed_amount	233154.0	54356.993528	1.297131e+04
asset_cost	233154.0	75865.068144	1.894478e+04
ltv	233154.0	74.746530	1.145664e+01
branch_id	233154.0	72.936094	6.983499e+01
supplier_id	233154.0	19638.635035	3.491950e+03
manufacturer_id	233154.0	69.028054	2.214130e+01
Current_pincode_ID	233154.0	3396.880247	2.238148e+03
State_ID	233154.0	7.262243	4.482230e+00
Employee_code_ID	233154.0	1549.477148	9.752613e+02
MobileNo_Avl_Flag	233154.0	1.000000	0.000000e+00
Aadhar_flag	233154.0	0.840320	3.663097e-01
PAN_flag	233154.0	0.075577	2.643201e-01
VoterID_flag	233154.0	0.144943	3.520439e-01

Driving_flag	233154.0	0.023242	1.506720e-01
Passport_flag	233154.0	0.002127	4.607421e-02
PERFORM_CNS_SCORE	233154.0	289.462994	3.383748e+02
PRI_NO_OF_ACCTS	233154.0	2.440636	5.217233e+00
PRI_ACTIVE_ACCTS	233154.0	1.039896	1.941496e+00
PRI_OVERDUE_ACCTS	233154.0	0.156549	5.487867e-01
PRI_CURRENT_BALANCE	233154.0	165900.076936	9.422736e+05
PRI_SANCTIONED_AMOUNT	233154.0	218503.855323	2.374794e+06
PRI_DISBURSED_AMOUNT	233154.0	218065.898655	2.377744e+06
SEC_NO_OF_ACCTS	233154.0	0.059081	6.267946e-01
SEC_ACTIVE_ACCTS	233154.0	0.027703	3.160566e-01
SEC_OVERDUE_ACCTS	233154.0	0.007244	1.110789e-01
SEC_CURRENT_BALANCE	233154.0	5427.792819	1.702370e+05
SEC_SANCTIONED_AMOUNT	233154.0	7295.923347	1.831560e+05
SEC_DISBURSED_AMOUNT	233154.0	7179.997873	1.825925e+05
PRIMARY_INSTAL_AMT	233154.0	13105.481720	1.513679e+05
SEC_INSTAL_AMT	233154.0	323.268449	1.555369e+04
NEW_ACCTS_IN_LAST_SIX_MONTHS	233154.0	0.381833	9.551067e-01
DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS	233154.0	0.097481	3.844390e-01
NO_OF_INQUIRIES	233154.0	0.206615	7.064977e-01
loan_default	233154.0	0.217071	4.122523e-01

	min	25%	50% \
UniqueID	417428.00	476786.25	535978.5
disbursed_amount	13320.00	47145.00	53803.0
asset_cost	37000.00	65717.00	70946.0
ltv	10.03	68.88	76.8
branch_id	1.00	14.00	61.0
supplier_id	10524.00	16535.00	20333.0
manufacturer_id	45.00	48.00	86.0
Current_pincode_ID	1.00	1511.00	2970.0
State_ID	1.00	4.00	6.0
Employee_code_ID	1.00	713.00	1451.0
MobileNo_Avl_Flag	1.00	1.00	1.0
Aadhar_flag	0.00	1.00	1.0
PAN_flag	0.00	0.00	0.0
VoterID_flag	0.00	0.00	0.0
Driving_flag	0.00	0.00	0.0
Passport_flag	0.00	0.00	0.0
PERFORM_CNS_SCORE	0.00	0.00	0.0
PRI_NO_OF_ACCTS	0.00	0.00	0.0
PRI_ACTIVE_ACCTS	0.00	0.00	0.0
PRI_OVERDUE_ACCTS	0.00	0.00	0.0
PRI_CURRENT_BALANCE	-6678296.00	0.00	0.0
PRI_SANCTIONED_AMOUNT	0.00	0.00	0.0
PRI_DISBURSED_AMOUNT	0.00	0.00	0.0
SEC_NO_OF_ACCTS	0.00	0.00	0.0

SEC_ACTIVE_ACCTS	0.00	0.00	0.0
SEC_OVERDUE_ACCTS	0.00	0.00	0.0
SEC_CURRENT_BALANCE	-574647.00	0.00	0.0
SEC_SANCTIONED_AMOUNT	0.00	0.00	0.0
SEC_DISBURSED_AMOUNT	0.00	0.00	0.0
PRIMARY_INSTAL_AMT	0.00	0.00	0.0
SEC_INSTAL_AMT	0.00	0.00	0.0
NEW_ACCTS_IN_LAST_SIX_MONTHS	0.00	0.00	0.0
DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS	0.00	0.00	0.0
NO_OF_INQUIRIES	0.00	0.00	0.0
loan_default	0.00	0.00	0.0

	75%	max
UniqueID	595039.75	6.710840e+05
disbursed_amount	60413.00	9.905720e+05
asset_cost	79201.75	1.628992e+06
ltv	83.67	9.500000e+01
branch_id	130.00	2.610000e+02
supplier_id	23000.00	2.480300e+04
manufacturer_id	86.00	1.560000e+02
Current_pincode_ID	5677.00	7.345000e+03
State_ID	10.00	2.200000e+01
Employee_code_ID	2362.00	3.795000e+03
MobileNo_Avl_Flag	1.00	1.000000e+00
Aadhar_flag	1.00	1.000000e+00
PAN_flag	0.00	1.000000e+00
VoterID_flag	0.00	1.000000e+00
Driving_flag	0.00	1.000000e+00
Passport_flag	0.00	1.000000e+00
PERFORM_CNS_SCORE	678.00	8.900000e+02
PRI_NO_OF_ACCTS	3.00	4.530000e+02
PRI_ACTIVE_ACCTS	1.00	1.440000e+02
PRI_OVERDUE_ACCTS	0.00	2.500000e+01
PRI_CURRENT_BALANCE	35006.50	9.652492e+07
PRI_SANCTIONED_AMOUNT	62500.00	1.000000e+09
PRI_DISBURSED_AMOUNT	60800.00	1.000000e+09
SEC_NO_OF_ACCTS	0.00	5.200000e+01
SEC_ACTIVE_ACCTS	0.00	3.600000e+01
SEC_OVERDUE_ACCTS	0.00	8.000000e+00
SEC_CURRENT_BALANCE	0.00	3.603285e+07
SEC_SANCTIONED_AMOUNT	0.00	3.000000e+07
SEC_DISBURSED_AMOUNT	0.00	3.000000e+07
PRIMARY_INSTAL_AMT	1999.00	2.564281e+07
SEC_INSTAL_AMT	0.00	4.170901e+06
NEW_ACCTS_IN_LAST_SIX_MONTHS	0.00	3.500000e+01
DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS	0.00	2.000000e+01
NO_OF_INQUIRIES	0.00	3.600000e+01

```
loan_default                                    0.00  1.000000e+00
```

[3]:
```
data.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  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
```

[4]:
```
data.isna().sum()
```

[4]:
```
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
```

[5]:
```
data.shape
```

[5]:
```
(233154, 41)
```

[6]:
```
# Removing rows from DataFrame where Employment_Type has missing values.
data = data.dropna(subset=['Employment_Type'])
print(f"Remaining rows after dropping missing values: {data.shape[0]}")
```

```
Remaining rows after dropping missing values: 225493
```

[7]:
```
data.isna().sum()
```

[7]:
```
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
```

[8]:

```
import pandas as pd

def print_unique_values(data):
    for column in data.columns:
        unique_values = data[column].unique()
        distinct_count = data[column].nunique()
# Printing details for each column
        print(f"Column: '{column}'")
        print(f"Distinct Count: {distinct_count}")
        print(f"Unique Values: {unique_values[:10]}{'...' if len(unique_values)␣
↪> 10 else ''}")
        # Showing first 10 unique values
        print("-" * 50)
print_unique_values(data)
```

```
Column: 'UniqueID'
Distinct Count: 225493
Unique Values: [420825 537409 417566 624493 539055 518279 529269 510278 490213
510980]…
--------------------------------------------------
Column: 'disbursed_amount'
Distinct Count: 24228
Unique Values: [50578 47145 53278 57513 52378 54513 46349 43894 53713 52603]…
--------------------------------------------------
Column: 'asset_cost'
Distinct Count: 45415
Unique Values: [58400 65550 61360 66113 60300 61900 61500 61973 61300 61230]…
--------------------------------------------------
Column: 'ltv'
Distinct Count: 6541
Unique Values: [89.55 73.23 89.63 88.48 88.39 89.66 76.42 71.89 89.56 86.95]…
--------------------------------------------------
Column: 'branch_id'
Distinct Count: 82
Unique Values: [ 67  78  34 130  74  11   5  20  63  48]…
--------------------------------------------------
Column: 'supplier_id'
Distinct Count: 2945
```

```
Unique Values: [22807 22744 17014 20700 15196 23069 21081 20520 15218 22637]…
--------------------------------------------------
Column: 'manufacturer_id'
Distinct Count: 11
Unique Values: [ 45  86  48  51 120  49 145  67 156 153]…
--------------------------------------------------
Column: 'Current_pincode_ID'
Distinct Count: 6659
Unique Values: [1441 1502 1497 1501 1495 1492 1493 1446 1440 1498]…
--------------------------------------------------
Column: 'Date_of_Birth'
Distinct Count: 14417
Unique Values: ['01-01-84' '31-07-85' '24-08-85' '30-12-93' '09-12-77'
'08-09-90'
 '01-06-88' '04-10-89' '15-11-91' '01-06-68']…
--------------------------------------------------
Column: 'Employment_Type'
Distinct Count: 2
Unique Values: ['Salaried' 'Self employed']
--------------------------------------------------
Column: 'DisbursalDate'
Distinct Count: 84
Unique Values: ['03-08-18' '26-09-18' '01-08-18' '26-10-18' '19-09-18'
'23-09-18'
 '16-09-18' '05-09-18' '29-09-18' '03-09-18']…
--------------------------------------------------
Column: 'State_ID'
Distinct Count: 22
Unique Values: [ 6  4  3  9  5 10 18 15  1 16]…
--------------------------------------------------
Column: 'Employee_code_ID'
Distinct Count: 3269
Unique Values: [1998 1646  115 1863 1570 1943 1835  864  958   83]…
--------------------------------------------------
Column: 'MobileNo_Avl_Flag'
Distinct Count: 1
Unique Values: [1]
--------------------------------------------------
Column: 'Aadhar_flag'
Distinct Count: 2
Unique Values: [1 0]
--------------------------------------------------
Column: 'PAN_flag'
Distinct Count: 2
Unique Values: [0 1]
--------------------------------------------------
Column: 'VoterID_flag'
Distinct Count: 2
```

```
Unique Values: [0 1]
--------------------------------------------------
Column: 'Driving_flag'
Distinct Count: 2
Unique Values: [0 1]
--------------------------------------------------
Column: 'Passport_flag'
Distinct Count: 2
Unique Values: [0 1]
--------------------------------------------------
Column: 'PERFORM_CNS_SCORE'
Distinct Count: 573
Unique Values: [  0 598 305 825  17 718 818 300 786 738]…
--------------------------------------------------
Column: 'PERFORM_CNS_SCORE_DESCRIPTION'
Distinct Count: 20
Unique Values: ['No Bureau History Available' 'I-Medium Risk' 'L-Very High Risk'
 'A-Very Low Risk' 'Not Scored: Not Enough Info available on the customer'
 'D-Very Low Risk' 'M-Very High Risk' 'B-Very Low Risk' 'C-Very Low Risk'
 'E-Low Risk']…
--------------------------------------------------
Column: 'PRI_NO_OF_ACCTS'
Distinct Count: 107
Unique Values: [ 0  1  3  2  7 10  5  6 13  9]…
--------------------------------------------------
Column: 'PRI_ACTIVE_ACCTS'
Distinct Count: 40
Unique Values: [ 0  1  2  5  4  8  3  6  9 11]…
--------------------------------------------------
Column: 'PRI_OVERDUE_ACCTS'
Distinct Count: 22
Unique Values: [ 0  1  3  2 14  4  5  8  6  7]…
--------------------------------------------------
Column: 'PRI_CURRENT_BALANCE'
Distinct Count: 70044
Unique Values: [      0   27600   72879     -41     676   79750   95597   29069
1076657
  134499]…
--------------------------------------------------
Column: 'PRI_SANCTIONED_AMOUNT'
Distinct Count: 43743
Unique Values: [      0   50200   74500  365384   36154   69900  187000  179252
1067200
 2277048]…
--------------------------------------------------
Column: 'PRI_DISBURSED_AMOUNT'
Distinct Count: 47206
Unique Values: [      0   50200   74500  365384   23374   69900  187000  179252
```

```
1067200
 2277048]…
--------------------------------------------------
Column: 'SEC_NO_OF_ACCTS'
Distinct Count: 37
Unique Values: [ 0  2  1  3 11  9  5  4  7 19]…
--------------------------------------------------
Column: 'SEC_ACTIVE_ACCTS'
Distinct Count: 23
Unique Values: [ 0  2  1  3 11  4  9  5  7  6]…
--------------------------------------------------
Column: 'SEC_OVERDUE_ACCTS'
Distinct Count: 9
Unique Values: [0 1 4 5 3 2 6 8 7]
--------------------------------------------------
Column: 'SEC_CURRENT_BALANCE'
Distinct Count: 3197
Unique Values: [      0 1171994 5787530   29870    4683  355631      28   28999
1900602
   91437]…
--------------------------------------------------
Column: 'SEC_SANCTIONED_AMOUNT'
Distinct Count: 2195
Unique Values: [      0 1690000   40000 6425000   34458   10935  400000   37000
33000
   26990]…
--------------------------------------------------
Column: 'SEC_DISBURSED_AMOUNT'
Distinct Count: 2519
Unique Values: [      0 1690000     361 6425000   34458   10935  400000   37000
77758
   26990]…
--------------------------------------------------
Column: 'PRIMARY_INSTAL_AMT'
Distinct Count: 27608
Unique Values: [    0  1991    31  1347  2608  2270  3300 23309  3514  7900]…
--------------------------------------------------
Column: 'SEC_INSTAL_AMT'
Distinct Count: 1890
Unique Values: [   0 9382 6485 1563 1065 1330 5185 2372 1345 3290]…
--------------------------------------------------
Column: 'NEW_ACCTS_IN_LAST_SIX_MONTHS'
Distinct Count: 26
Unique Values: [ 0  1  4  2  6  3  9  5 14  8]…
--------------------------------------------------
Column: 'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS'
Distinct Count: 14
Unique Values: [0 1 2 3 5 4 7 6 8 9]…
```

```
--------------------------------------------------
Column: 'AVERAGE_ACCT_AGE'
Distinct Count: 192
Unique Values: ['0yrs 0mon' '1yrs 11mon' '0yrs 8mon' '1yrs 9mon' '0yrs 2mon'
'4yrs 8mon'
 '1yrs 7mon' '0yrs 7mon' '2yrs 1mon' '1yrs 3mon']…
--------------------------------------------------
Column: 'CREDIT_HISTORY_LENGTH'
Distinct Count: 291
Unique Values: ['0yrs 0mon' '1yrs 11mon' '1yrs 3mon' '2yrs 0mon' '0yrs 2mon'
'4yrs 8mon'
 '1yrs 7mon' '0yrs 7mon' '2yrs 3mon' '2yrs 9mon']…
--------------------------------------------------
Column: 'NO_OF_INQUIRIES'
Distinct Count: 25
Unique Values: [ 0  1  4  2  3  5  6 12  9  8]…
--------------------------------------------------
Column: 'loan_default'
Distinct Count: 2
Unique Values: [0 1]
--------------------------------------------------
```

[9]:

```
def convert_to_months(column_value):
    # Extracting years and months from the string and convert to total months
    years = int(column_value.split()[0].replace('yrs', ''))
    months = int(column_value.split()[1].replace('mon', ''))
    return years * 12 + months

def transform_duration_columns(data):
    # Transforming specific columns to total months

    data['AVERAGE_ACCT_AGE'] = data['AVERAGE_ACCT_AGE'].apply(convert_to_months)
    data['CREDIT_HISTORY_LENGTH'] = data['CREDIT_HISTORY_LENGTH'].
↪apply(convert_to_months)
    return data
```

[10]:

```
transform_duration_columns(data)
```

[10]:

```
        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
…            …                 …      …     …      …          …
233149    626432             63213      105405  60.72         34        20700
233150    606141             73651      100600  74.95         34        23775
```

```
233151    613658             33484       71212  48.45          77         22186
233152    548084             34259       73286  49.10          77         22186
233153    630213             75751      116009  66.81          77         22186

        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
…                   …                  …            …            …
233149               48                1050      01-08-88        Salaried
233150               51                 990      05-12-88   Self employed
233151               86                2299      01-06-76        Salaried
233152               86                2299      26-03-94        Salaried
233153               86                2299      18-02-84        Salaried

        … 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
…     …                    …                     …                   …
233149  …                     0                     0                4084
233150  …                     0                     0                1565
233151  …                     0                     0                   0
233152  …                     0                     0                   0
233153  …                     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
…                  …                           …
233149               0                             0
233150               0                             0
233151               0                             0
233152               0                             0
233153               0                             0

        DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS  AVERAGE_ACCT_AGE  \
0                                         0                 0
1                                         1                23
2                                         0                 0
```

```
3                                        0                 8
4                                        0                 0
…                                    …               …
233149                                   0                21
233150                                   0                 6
233151                                   0                 0
233152                                   0                 0
233153                                   0                 0

        CREDIT_HISTORY_LENGTH  NO_OF_INQUIRIES  loan_default
0                           0                0             0
1                          23                0             1
2                           0                0             0
3                          15                1             1
4                           0                1             1
…                        …              …            …
233149                     39                0             0
233150                      6                0             0
233151                      0                0             0
233152                      0                0             0
233153                      0                0             0

[225493 rows x 41 columns]
```

[11]:
```python
data.PERFORM_CNS_SCORE_DESCRIPTION.value_counts()
```

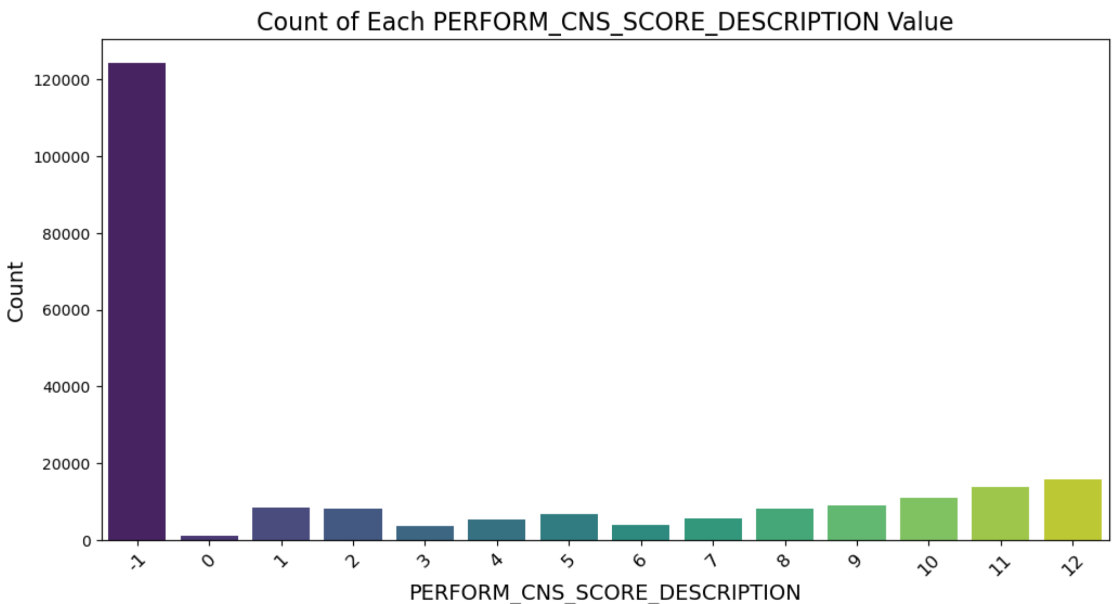
[11]:
```
No Bureau History Available                                111773
C-Very Low Risk                                             15715
A-Very Low Risk                                             13790
D-Very Low Risk                                             11134
B-Very Low Risk                                              9032
M-Very High Risk                                             8632
F-Low Risk                                                   8309
K-High Risk                                                  8107
H-Medium Risk                                                6695
E-Low Risk                                                   5695
I-Medium Risk                                                5440
G-Low Risk                                                   3902
Not Scored: Sufficient History Not Available                 3671
J-High Risk                                                  3667
Not Scored: Not Enough Info available on the customer        3557
Not Scored: No Activity seen on the customer (Inactive)      2815
Not Scored: No Updates available in last 36 months           1477
L-Very High Risk                                             1122
Not Scored: Only a Guarantor                                  957
Not Scored: More than 50 active Accounts found                  3
Name: PERFORM_CNS_SCORE_DESCRIPTION, dtype: int64
```

[12]:
```python
import numpy as np

def extract_score(description):
    """Extracts the score letter before the dash or returns 'N' if no dash␣
 ↪exists."""
    return description.split("-")[0] if "-" in description else 'N'

def transform_cns_score_description(data):
    """Transforms the CNS score description into numerical values."""
    # Map for transforming CNS descriptions into numerical values
    score_mapping = {'N': -1,'L': 0, 'M': 1, 'K': 2, 'J': 3, 'I': 4, 'H': 5,␣
 ↪'G': 6, 'E': 7,
                     'F': 8,  'B': 9, 'D': 10, 'A': 11, 'C': 12}

    # Extracting score description and map it to numerical values
    data['PERFORM_CNS_SCORE_DESCRIPTION'] = (
        data['PERFORM_CNS_SCORE_DESCRIPTION']
        .apply(extract_score)
        .map(score_mapping)
        .astype(np.int32)
    )

transform_cns_score_description(data)
```

[13]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Counting occurrences of each unique value
value_counts = data['PERFORM_CNS_SCORE_DESCRIPTION'].value_counts()

print(value_counts)

# Plotting the counts using Seaborn
plt.figure(figsize=(12, 6))
sns.barplot(x=value_counts.index, y=value_counts.values, palette="viridis")
plt.title('Count of Each PERFORM_CNS_SCORE_DESCRIPTION Value', fontsize=16)
plt.xlabel('PERFORM_CNS_SCORE_DESCRIPTION', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.show()
```

```
-1     124253
 12     15715
 11     13790
 10     11134
 9       9032
 1       8632
```

```
8       8309
2       8107
5       6695
7       5695
4       5440
6       3902
3       3667
0       1122
Name: PERFORM_CNS_SCORE_DESCRIPTION, dtype: int64
```

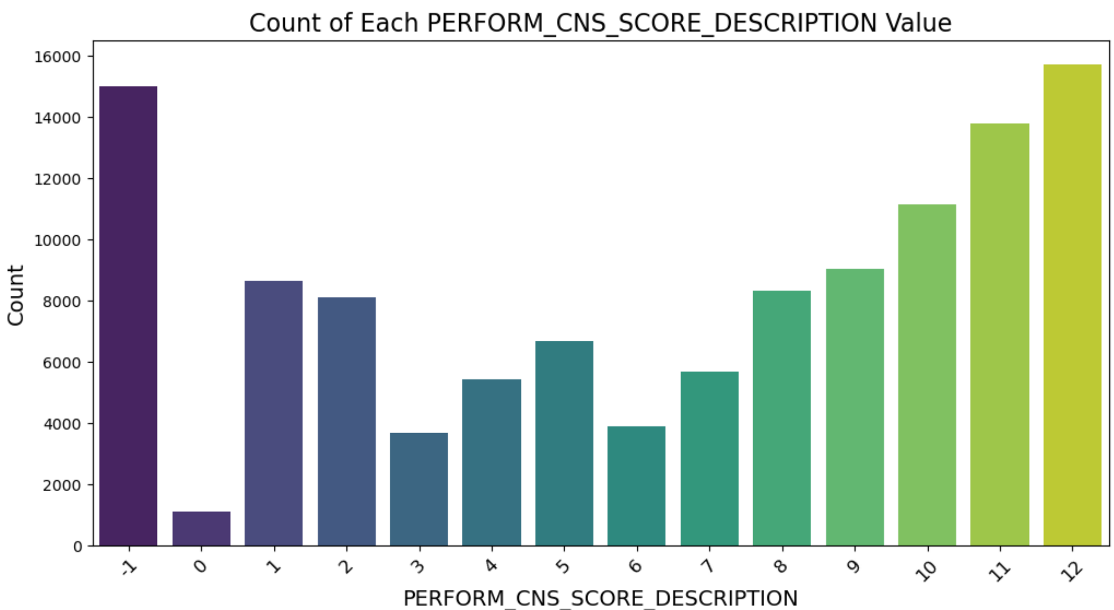


[14]:
```
import pandas as pd

data_non_minus1 = data[data['PERFORM_CNS_SCORE_DESCRIPTION'] != -1]
data_minus1_sampled = data[data['PERFORM_CNS_SCORE_DESCRIPTION'] == -1].
↪sample(n=15000, random_state=1)

data = pd.concat([data_non_minus1, data_minus1_sampled])
```

[15]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Counting occurrences of each unique value
value_counts = data['PERFORM_CNS_SCORE_DESCRIPTION'].value_counts()

print(value_counts)
```

```
# Plotting the counts using Seaborn
plt.figure(figsize=(12, 6))
sns.barplot(x=value_counts.index, y=value_counts.values, palette="viridis")
plt.title('Count of Each PERFORM_CNS_SCORE_DESCRIPTION Value', fontsize=16)
plt.xlabel('PERFORM_CNS_SCORE_DESCRIPTION', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.xticks(rotation=45, fontsize=12)
plt.show()
```

```
 12    15715
-1     15000
 11    13790
 10    11134
 9      9032
 1      8632
 8      8309
 2      8107
 5      6695
 7      5695
 4      5440
 6      3902
 3      3667
 0      1122
Name: PERFORM_CNS_SCORE_DESCRIPTION, dtype: int64
```



[16]:
```
data.shape
```

[16]: (116240, 41)

[17]:
```
# Defining columns to work with
primary_secondary_columns = [
    'PRI_NO_OF_ACCTS', 'SEC_NO_OF_ACCTS',
    'PRI_ACTIVE_ACCTS', 'SEC_ACTIVE_ACCTS',
    'PRI_OVERDUE_ACCTS', 'SEC_OVERDUE_ACCTS',
    'PRI_CURRENT_BALANCE', 'SEC_CURRENT_BALANCE',
    'PRI_SANCTIONED_AMOUNT', 'SEC_SANCTIONED_AMOUNT',
    'PRI_DISBURSED_AMOUNT', 'SEC_DISBURSED_AMOUNT',
    'PRIMARY_INSTAL_AMT', 'SEC_INSTAL_AMT'
]

# Function for creating new aggregated columns
def create_new_columns(data):
    # Creating new columns by summing primary and secondary account values
    data['NO_OF_ACCTS'] = data['PRI_NO_OF_ACCTS'] + data['SEC_NO_OF_ACCTS']
    data['ACTIVE_ACCTS'] = data['PRI_ACTIVE_ACCTS'] + data['SEC_ACTIVE_ACCTS']
    data['OVERDUE_ACCTS'] = data['PRI_OVERDUE_ACCTS'] +␣
 ↪data['SEC_OVERDUE_ACCTS']
    data['CURRENT_BALANCE'] = data['PRI_CURRENT_BALANCE'] +␣
 ↪data['SEC_CURRENT_BALANCE']
    data['SANCTIONED_AMOUNT'] = data['PRI_SANCTIONED_AMOUNT'] +␣
 ↪data['SEC_SANCTIONED_AMOUNT']
    data['DISBURSED_AMOUNT'] = data['PRI_DISBURSED_AMOUNT'] +␣
 ↪data['SEC_DISBURSED_AMOUNT']
    data['INSTAL_AMT'] = data['PRIMARY_INSTAL_AMT'] + data['SEC_INSTAL_AMT']

    # Dropping the original primary and secondary columns
    data.drop(columns=primary_secondary_columns, inplace=True)

# Applying the transformation
create_new_columns(data)

# Listting new columns
new_columns = [
    'NO_OF_ACCTS', 'ACTIVE_ACCTS', 'OVERDUE_ACCTS', 'CURRENT_BALANCE',
    'SANCTIONED_AMOUNT', 'DISBURSED_AMOUNT', 'INSTAL_AMT'
]

# Function for printing unique values for each new column
def print_column_info(data, columns):
    for col in columns:
        print(f"{col} : distinct values")
        print(f"{data[col].nunique()} : No. of unique items")
        print("-" * 30)
```

```
# Printing the distinct values and counts
print_column_info(data, new_columns)
```

```
NO_OF_ACCTS : distinct values
106 : No. of unique items
------------------------------
ACTIVE_ACCTS : distinct values
37 : No. of unique items
------------------------------
OVERDUE_ACCTS : distinct values
22 : No. of unique items
------------------------------
CURRENT_BALANCE : distinct values
67705 : No. of unique items
------------------------------
SANCTIONED_AMOUNT : distinct values
43126 : No. of unique items
------------------------------
DISBURSED_AMOUNT : distinct values
46438 : No. of unique items
------------------------------
INSTAL_AMT : distinct values
27540 : No. of unique items
------------------------------
```

[18]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['Employment_Type'] = label_encoder.fit_transform(data['Employment_Type'])
```

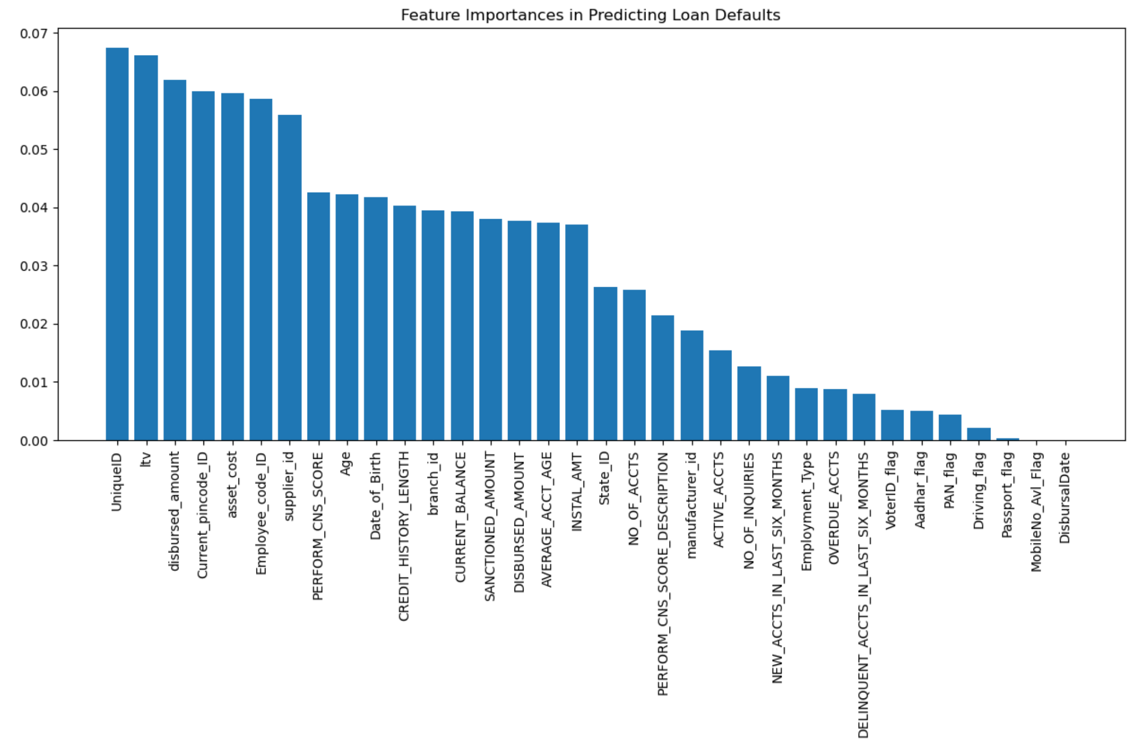
[19]:
```
def convert_year(year):
    """
    Converts two-digit year to four-digit year.
    Assumes years < 20 are in 2000s, otherwise in 1900s.
    """
    year = int(year)
    if 0 <= year < 20:
        return year + 2000
    else:
        return year + 1900

def age(dob):
    """
    Converts a date string (format: DD-MM-YY) to the year of birth as an␣
 ↪integer.
    """
    return convert_year(dob[-2:])
```

```python
def calculate_age(data):
    """
    Adds an 'Age' column to the dataset by calculating the age of applicants at␣
 ↪loan disbursal.
    Drops the original 'Date_of_Birth' and 'DisbursalDate' columns.
    """
    # Converting Date of Birth and Disbursal Date to years
    data['Date_of_Birth'] = data['Date_of_Birth'].apply(age)
    data['DisbursalDate'] = data['DisbursalDate'].apply(age)

    # Calculating age
    data['Age'] = data['DisbursalDate'] - data['Date_of_Birth']


# Applying the function to DataFrame
calculate_age(data)
```

[20]:

```python
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt


# Preparing data for training
X = data.drop('loan_default', axis=1)
y = data['loan_default']

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initializing and training the RandomForest Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

importances = model.feature_importances_
features = X.columns

# Sorting the feature importances in descending order and plotting them
sorted_indices = np.argsort(importances)[::-1]
plt.figure(figsize=(12, 8))
plt.title('Feature Importances in Predicting Loan Defaults')
plt.bar(range(len(importances)), importances[sorted_indices], align='center')
```

```
plt.xticks(range(len(importances)), [features[i] for i in sorted_indices],␣
 ↪rotation=90)
plt.tight_layout()
plt.show()
```



This analysis using a RandomForestClassifier highlights the key predictors of loan defaults, identifying features like UniqueID, loan to value ratio, and disbursed amount as significant. Insights from this feature importance analysis are crucial for refining models and improving decision-making in loan approvals.

[21]:
```
# dropping least important ones
data = data.drop(columns=['MobileNo_Avl_Flag', 'Driving_flag', 'Aadhar_flag',␣
 ↪'PAN_flag', 'VoterID_flag', 'Passport_flag'])
```

[22]:
```
data.head().T
```

[22]:
```
                                          1          3          5  \
UniqueID                          537409.00  624493.00  518279.00
disbursed_amount                   47145.00   57513.00   54513.00
asset_cost                         65550.00   66113.00   61900.00
ltv                                   73.23      88.48      89.66
branch_id                             67.00      67.00      67.00
supplier_id                        22807.00   22807.00   22807.00
manufacturer_id                       45.00      45.00      45.00
```

Current_pincode_ID	1502.00	1501.00	1501.00
Date_of_Birth	1985.00	1993.00	1990.00
Employment_Type	1.00	1.00	1.00
DisbursalDate	2018.00	2018.00	2018.00
State_ID	6.00	6.00	6.00
Employee_code_ID	1998.00	1998.00	1998.00
PERFORM_CNS_SCORE	598.00	305.00	825.00
PERFORM_CNS_SCORE_DESCRIPTION	4.00	0.00	11.00
NEW_ACCTS_IN_LAST_SIX_MONTHS	0.00	0.00	0.00
DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS	1.00	0.00	0.00
AVERAGE_ACCT_AGE	23.00	8.00	21.00
CREDIT_HISTORY_LENGTH	23.00	15.00	24.00
NO_OF_INQUIRIES	0.00	1.00	0.00
loan_default	1.00	1.00	0.00
NO_OF_ACCTS	1.00	3.00	2.00
ACTIVE_ACCTS	1.00	0.00	0.00
OVERDUE_ACCTS	1.00	0.00	0.00
CURRENT_BALANCE	27600.00	0.00	0.00
SANCTIONED_AMOUNT	50200.00	0.00	0.00
DISBURSED_AMOUNT	50200.00	0.00	0.00
INSTAL_AMT	1991.00	31.00	1347.00
Age	33.00	25.00	28.00

	8	9
UniqueID	490213.00	510980.00
disbursed_amount	53713.00	52603.00
asset_cost	61973.00	61300.00
ltv	89.56	86.95
branch_id	67.00	67.00
supplier_id	22807.00	22807.00
manufacturer_id	45.00	45.00
Current_pincode_ID	1497.00	1492.00
Date_of_Birth	1991.00	1968.00
Employment_Type	1.00	0.00
DisbursalDate	2018.00	2018.00
State_ID	6.00	6.00
Employee_code_ID	1998.00	1998.00
PERFORM_CNS_SCORE	718.00	818.00
PERFORM_CNS_SCORE_DESCRIPTION	10.00	11.00
NEW_ACCTS_IN_LAST_SIX_MONTHS	0.00	0.00
DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS	0.00	0.00
AVERAGE_ACCT_AGE	56.00	19.00
CREDIT_HISTORY_LENGTH	56.00	19.00
NO_OF_INQUIRIES	1.00	0.00
loan_default	0.00	0.00
NO_OF_ACCTS	1.00	1.00
ACTIVE_ACCTS	1.00	0.00

OVERDUE_ACCTS	0.00	0.00
CURRENT_BALANCE	-41.00	0.00
SANCTIONED_AMOUNT	365384.00	0.00
DISBURSED_AMOUNT	365384.00	0.00
INSTAL_AMT	0.00	2608.00
Age	27.00	50.00

[23]:

```
import plotly.express as px
fig = px.scatter(
    data,
    x='disbursed_amount',
    y='asset_cost',
    color='loan_default',
    facet_col='loan_default',  # Creating a separate plot for each loan_default␣
 ↪category
    title='Disbursed Amount vs. Asset Cost by Loan Default Status',
    labels={'loan_default': 'Loan Default Status'}
)

fig.update_layout(
    xaxis_title='Disbursed Amount',
    yaxis_title='Asset Cost'
)

fig.show()
```

The visualization segregates the data on disbursed loan amounts versus the asset costs into two groups based on the loan default status, with one panel for non-defaulters and another for defaulters.

[24]:

```
import plotly.express as px
import pandas as pd

# Calculating the correlation matrix for all numeric columns
corr = data.corr()

# Converting the correlation matrix to a long format for easier use with Plotly
corr_long = corr.reset_index().melt(id_vars='index')
corr_long.columns = ['Variable 1', 'Variable 2', 'Correlation']

# Creating an interactive heatmap using Plotly Express
fig = px.density_heatmap(
    corr_long,
    x='Variable 1',
    y='Variable 2',
    z='Correlation',
    nbinsx=len(corr.columns),
    nbinsy=len(corr.columns),
```

```
    color_continuous_scale='RdBu_r',  # Color scale that centers around zero
    range_color=[-1,1]  # Ensuring color scale ranges from -1 to 1
)

# Improving layout
fig.update_layout(
    title='Interactive Correlation Matrix of Loan Data',
    xaxis=dict(title='Variables', tickangle=-45),
    yaxis=dict(title='Variables'),
    coloraxis_colorbar=dict(title='Correlation'),
)

# Adding hover text for making it easier to read exact correlation values
fig.update_traces(hovertemplate='Variable 1: %{x}<br>Variable 2:␣
 ↪%{y}<br>Correlation: %{z:.2f}')

fig.show()
```

Interactive correlation matrix visualizes the relationships between various loan-related variables, enabling an strong exploration of how features interact within the dataset. The heatmap uses shades of red and blue to depict the strength and direction of correlations, where red represents positive correlations and blue represents negative ones.

[25]:
```
import plotly.graph_objects as go
import pandas as pd

# Ensuring the transaction count is calculated
if 'count_by_age_who_took_loan' not in data.columns:
    data['count_by_age_who_took_loan'] = data.
 ↪groupby('Age')['disbursed_amount'].transform('count')

# Preparing age_data with aggregated value
if 'mean_disbursed_amount' not in data.columns:
    age_data = data.groupby('Age').agg({
        'disbursed_amount': 'mean',
        'count_by_age_who_took_loan': 'mean'
    }).reset_index()
else:
    age_data = data

# Creating a figure with secondary y-axis
fig = go.Figure()

# Adding a connected scatter plot for the average disbursed amount
fig.add_trace(
    go.Scatter(
        x=age_data['Age'],
```

```
        y=age_data['disbursed_amount'],
        name='Average Disbursed Amount',
        mode='lines+markers',  # Using both lines and markers
        marker=dict(color='RoyalBlue', size=8),  # Enhanced marker styling
        line=dict(color='RoyalBlue', width=3)  # Thicker line for better␣
↪visibility
    )
)

# Adding a bar chart for the transaction count on secondary y-axis
fig.add_trace(
    go.Bar(
        x=age_data['Age'],
        y=age_data['count_by_age_who_took_loan'],
        name='count of people who took loan w.r.t Age',
        yaxis='y2',  # Assigning to secondary y-axis
        marker=dict(color='Crimson'),  # Brighter red color for better contrast
    )
)

# Enhancing the plot with a secondary axis for the transaction count
fig.update_layout(
    title={
        'text': 'Average Disbursed Amount and Transaction Count by Age',
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    },
    xaxis_title='Age',
    yaxis=dict(
        title='Average Disbursed Amount',
        side='left',  # Primary y-axis on the left
        color='RoyalBlue'
    ),
    yaxis2=dict(
        title='count of people who took loan w.r.t Age',
        overlaying='y',
        side='right',  # Secondary y-axis on the right
        color='Crimson'
    ),
    template='plotly_white',

    legend=dict(
        x=0.5,
        y=-0.3,
        xanchor='center',
```

```
        orientation='h'  # Horizontal orientation to spread it out
    )

)

# Adding hover functionality for better interactivity
fig.update_traces(hoverinfo="all", hovertemplate="Age: %{x}<br>Value: %{y}")

fig.show()
```

This graph provides a dual perspective on the average disbursed loan amount and the count of loans taken by people across different ages, depicted through a combined bar chart and scatter plot. The scatter plot represents the average loan amount disbursed, showing a relatively stable trend across mid ages with a slight peak around 25 years old, while the bar chart displays a decrease in loan counts as age increases, particularly after 30 years. This visualization not only helps in identifying key trends in loan distribution and acceptance among different age groups.

[26]:

```
import pandas as pd
import plotly.express as px

data1 = data.copy()

# Creating a dictionary for State IDs and their corresponding names
state_names = {
    1: "Andhra Pradesh", 2: "Arunachal Pradesh", 3: "Assam", 4: "Bihar", 5:␣
 ↪"Chhattisgarh",
    6: "Goa", 7: "Gujarat", 8: "Haryana", 9: "Himachal Pradesh", 10:␣
 ↪"Jharkhand",
    11: "Karnataka", 12: "Kerala", 13: "Madhya Pradesh", 14: "Maharashtra", 15:␣
 ↪"Manipur",
    16: "Meghalaya", 17: "Mizoram", 18: "Nagaland", 19: "Odisha", 20: "Punjab",
    21: "Rajasthan", 22: "Sikkim"
}

# Mapping State_ID to state names
data1['State_ID'] = data1['State_ID'].map(state_names)

# Grouping data to sum up disbursed amounts by State and Branch
grouped_data = data1.groupby(['State_ID', 'branch_id'])['disbursed_amount'].
 ↪sum().reset_index()

# Determining the min and max values for setting a better color scale
min_value = grouped_data['disbursed_amount'].min()
max_value = grouped_data['disbursed_amount'].max()

# Creating a treemap
fig = px.treemap(
```

```
    grouped_data,
    path=['State_ID', 'branch_id'],  # Defines hierarchy: first by state, then␣
 ↪by branch
    values='disbursed_amount',  # Size of each block based on the disbursed␣
 ↪amount
    color='disbursed_amount',  # Color scale based on the disbursed amount
    color_continuous_scale=[
        (0, 'red'),    # Blue for the lowest values
        (0.5, 'blue'), # White for the middle values
        (1, 'green')      # Red for the highest values
    ],  # Blue color scale for visual appeal
    title="<b>               Disbursement Amount by State and Branch</b>"
)

# Customizing text labels to show both branch ID and disbursed amount in each␣
 ↪block
fig.update_traces(
    texttemplate="<b>Branch %{label}</b><br>%{value:,.0f}",  # Shows branch ID␣
 ↪and formatted disbursed amount
    textposition="middle center"  # Center align text within each block
)

# Adjusting layout and color bar settings for better visibility and readability
fig.update_layout(
    coloraxis_colorbar=dict(
        title="Disbursed Amount",  # Title for the color bar
        titleside="right",
        tickprefix="",  # No prefix, clean numeric values
        tickvals=[min_value, (min_value + max_value) / 2, max_value],  # Custom␣
 ↪tick values
        ticktext=[f"{min_value:,.0f}", f"{(min_value + max_value) / 2:,.0f}",␣
 ↪f"{max_value:,.0f}"]  # Custom tick labels
    ),
    font=dict(size=14),  # Font size for readability
    width=1100,  # Width of the figure
    height=1050   # Height of the figure
)

fig.show()
```

This visual treemap showcases the disbursed loan amounts by state and branch across India, using a color gradient to indicate the volume of disbursements. Each rectangle represents a branch within a state, with its size proportional to the disbursed amount, allowing quick visual comparisons. The color scale from red to green effectively highlights variations in disbursed amounts, where red indicates lower and green indicates higher values, helping the banks to get to know about how the financial activities are going on in branches.

[27]:
```
import plotly.express as px

data['loan_default'] = data['loan_default'].astype(str)

fig = px.scatter(
    data,
    x='NEW_ACCTS_IN_LAST_SIX_MONTHS',
    y='DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS',
    color='loan_default',
    title='New vs. Delinquent Accounts in Last Six Months',
    labels={
        'NEW_ACCTS_IN_LAST_SIX_MONTHS': 'New Accounts',
        'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS': 'Delinquent Accounts',
        'loan_default': 'Loan Default Status'
    },
    hover_data=['PERFORM_CNS_SCORE', 'NO_OF_ACCTS'],  # Adding more data to␣
 ↪hover tooltips
    size='PERFORM_CNS_SCORE',  # Size markers by CNS score, adjust as necessary
    color_continuous_scale=px.colors.diverging.Tealrose,  # Adjusts color scale␣
 ↪for visual appeal
    trendline='ols',  # Adds a trend line using ordinary least squares␣
 ↪regression
    size_max=15,  # Max size of markers
    category_orders={'loan_default': ['0', '1']}  # Ensuring consistent order␣
 ↪for categorical color
)

# Updating axes titles for clarity
fig.update_layout(
    xaxis_title='New Accounts in the Last Six Months',
    yaxis_title='Delinquent Accounts in the Last Six Months',
    legend_title='Loan Default Status'
)

# Enhancing marker appearance
fig.update_traces(marker_line_width=1, marker_line_color='black')

fig.show()
```

This visualization plots the relationship between new accounts and delinquent accounts in the last six months, differentiated by loan default status. It uses a scatter plot to show individual data points, where the color and size of each point are indicative of the loan default status and the borrower's credit score. The plot also includes a trend line to show the general pattern, providing good insights into how recent financial activities correlate.

[28]:
```
import plotly.express as px
```

```python
# Recreating the data mapping for colors based on the risk level
risk_levels = {
    0: 'High Risk',
    1: 'High Risk',
    2: 'High Risk',
    3: 'High Risk',
    4: 'Moderate Risk',
    5: 'Moderate Risk',
    6: 'Moderate Risk',
    7: 'Moderate Risk',
    8: 'Low Risk',
    9: 'Low Risk',
    10: 'Low Risk',
    11: 'Very Low Risk',
    12: 'Very Low Risk'
}

# Mapping scores to risk levels
data['Risk Level'] = data['PERFORM_CNS_SCORE_DESCRIPTION'].map(risk_levels)

# Defining a custom color scale to visually represent different risk levels
color_scale = {
    "High Risk": 'red',
    "Moderate Risk": 'orange',
    "Low Risk": 'green',
    "Very Low Risk": 'blue'
}

fig = px.scatter(
    data,
    x='PERFORM_CNS_SCORE',
    y='CREDIT_HISTORY_LENGTH',
    size='NO_OF_INQUIRIES',  # Varying size based on the number of inquiries
    color='Risk Level',  # Color by the mapped risk level
    color_discrete_map=color_scale,  # Applying the custom color scale
    hover_name='PERFORM_CNS_SCORE_DESCRIPTION',  # Shows score description on␣
 ↪hover
    title='CNS Score vs. Credit History Length',
    labels={'PERFORM_CNS_SCORE': 'CNS Score', 'CREDIT_HISTORY_LENGTH': 'Credit␣
 ↪History Length (Months)'}
)

fig.update_layout(
    xaxis_title='CNS Score',
    yaxis_title='Credit History Length in Months',
    legend_title='Risk Level'
)
```

```
# Adding annotations for risk levels directly in the plot
annotations = [
    dict(x=950, y=350, xref="x", yref="y",
         text="Risk Level:<br>0-3: High Risk<br>4-7: Moderate Risk<br>8-11: Low␣
↪Risk<br>12: Very Low Risk",
         showarrow=False, bgcolor="white", bordercolor="black", borderwidth=1)
]
fig.update_layout(annotations=annotations)

fig.show()
```

This scatter plot illustrates the relationship between the CNS Score and the length of credit history, categorized by risk level. It uses color to differentiate between high, moderate, low, and very low risk, providing an understanding of how credit score and credit history depth impact risk assessment. The size of each point reflects the number of inquiries, adding another layer of detail that highlights how frequent credit checks might correlate with credit history and risk.

[29]:
```
import pandas as pd
import plotly.graph_objects as go


# Creating a cross-tabulation
cross_tab = pd.crosstab(data['Employment_Type'], data['loan_default'],␣
↪margins=True)

# Replacing indices 0 and 1 with 'Employed' and 'Self-Employed'
cross_tab.index = ['Employed', 'Self-Employed', 'All']  # Replacing with␣
↪appropriate names

# Calculating the percentage of defaults
if '1' in cross_tab.columns:
    cross_tab['FG_PCT'] = round(cross_tab['1'] / cross_tab['All'], 3)
else:
    cross_tab['FG_PCT'] = 0

# Preparing data for plotting
main_data = cross_tab.drop(columns='All')[:-1]  # Exclude 'All' row
made = [f"{round(value * 100, 1)}%" for value in main_data['FG_PCT']]
missed = [f"{round((1 - value) * 100, 1)}%" for value in main_data['FG_PCT']]
x = main_data.index

# Defining aesthetic elements
color_1 = dict(color='#40826d')  # Color for defaults
color_2 = dict(color='#E85285')  # Color for non-defaults
scatter_line = dict(color='black', width=1.5)
font = dict(family='sans serif', size=12, color='white')
```

```python
# Creating the bar plots
made_bar = go.Bar(
    text=made,
    name='Default',
    x=x,
    y=main_data['1'],  # Uses string '1' for the column name
    textfont=font,
    marker=color_1
)

missed_bar = go.Bar(
    text=missed,
    name='No Default',
    x=x,
    y=main_data['0'],  # Uses string '0' for the column name
    textfont=font,
    marker=color_2
)

# Line for percentage made
line = go.Scatter(
    x=x,
    y=main_data['1'],  # Uses string '1' for the column name
    line=scatter_line,
    mode='lines+markers',
    name='Default Rate Line'
)

# Combining plots
fig_data = [made_bar, missed_bar, line]

fig = go.Figure(data=fig_data)

fig.update_layout(
    title='Loan Default Percentage by Employment Type',
    yaxis_title='Frequency of Loans',
    xaxis_title='Employment Type',
    showlegend=True,
    barmode='stack'
)

# Rendering the plot
fig.show()
```

This stacked bar chart visualizes the loan default percentages by employment type, contrasting employed and self-employed individuals. Employed borrowers show a lower default rate of 18.9%, whereas self-employed individuals have a higher rate at 21.8%. The visualization tells the higher financial reliability among employed borrowers and tells about the potential risks for lenders when dealing with self-employed individuals.

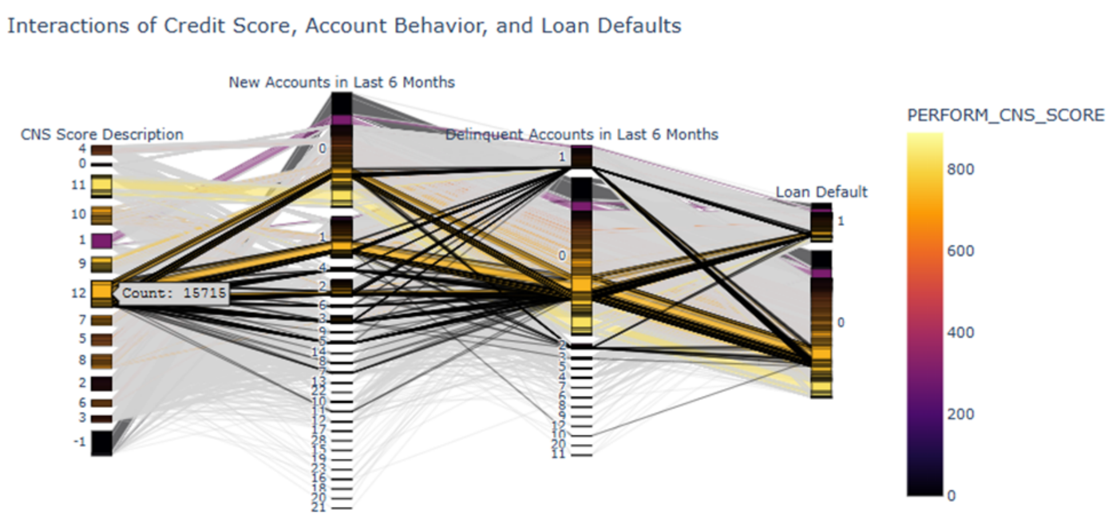
[30]:
```
import pandas as pd
import plotly.express as px


# Generating a scatter plot matrix
fig = px.scatter_matrix(data, dimensions=['CURRENT_BALANCE',␣
 ↪'SANCTIONED_AMOUNT', 'DISBURSED_AMOUNT', 'INSTAL_AMT'],
                        title='Relationship between Financial Attributes of␣
 ↪Loans',
                        labels={
                            'CURRENT_BALANCE': 'Current Balance',
                            'SANCTIONED_AMOUNT': 'Sanctioned Amount',
                            'DISBURSED_AMOUNT': 'Disbursed Amount',
                            'INSTAL_AMT': 'Installment Amount'
                        })

# Customizing aesthetic elements
fig.update_layout(
    dragmode='select',
    width=800,
    height=800,
    hovermode='closest'
)

# Rendering the plot
fig.show()
```

```
C:\Users\jmand\anaconda3\lib\site-packages\plotly\express\_core.py:279:
FutureWarning:

iteritems is deprecated and will be removed in a future version. Use .items
instead.
```

This scatter plot matrix provides a comprehensive visualization of the relationships between various financial attributes of loans, including Current Balance, Sanctioned Amount, Disbursed Amount, and Installment Amount. Each panel displays a scatter plot that explores the correlation between two distinct financial metrics, offering insights into how these attributes interact with each other across the loan dataset. This type of visualization quickly gives us a glance while identifying trends, potential outliers, and the strength of relationships between financial variables, which are very crucial for decision-making processes.

[31]:
```
# FAILED EXPERIMENT CODE for Parallel Categories Diagram

# import plotly.express as px
```

```
# import pandas as pd

# # Create the Parallel Categories Diagram
# fig = px.parallel_categories(
#     data,
#     dimensions=['PERFORM_CNS_SCORE_DESCRIPTION', 'NEW_ACCTS_IN_LAST_SIX_MONTHS', 'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS', 'loan_default'],
#     color='PERFORM_CNS_SCORE',  # Using CNS score as the color scale to see how score influences other categories
#     color_continuous_scale=px.colors.sequential.Inferno,
#     labels={
#         'PERFORM_CNS_SCORE_DESCRIPTION': 'CNS Score Description',
#         'NEW_ACCTS_IN_LAST_SIX_MONTHS': 'New Accounts in Last 6 Months',
#         'DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS': 'Delinquent Accounts in Last 6 Months',
#         'loan_default': 'Loan Default'
#     },
#     title='Interactions of Credit Score, Account Behavior, and Loan Defaults'
# )

# fig.show()
```



The Parallel Categories visualisation is made using Plotly Express for representing the relation between a borrower's credit score, new and delinquent accounts, and loan default status. The diagram uses color scales to highlight the influence of the CNS score on loan defaults and account behaviors. However, this visualization is a failed experiment because it produced an overly complex and difficult to interpret diagram. The connections and overlapping lines made it hard to draw clear conclusions, thus not effectively giving the proper insights about risk factors associated with loan defaults.

```
[ ]:
```