LoanTap LogisticRegression

November 30, 2024

1 Introduction

Based in India, Loantap is a well-known financial technology company that specializes in offering both consumers and corporations creative and adaptable loan options. Loantap uses technology to provide hassle-free borrowing experiences, such as flexible EMI alternatives, salary advances, and personal loans, with an emphasis on customer-centric solutions. They have been a reliable partner for borrowers looking for effective financial solutions because of their dedication to openness, quickness, and convenience.

Here, the Personal Loan section is the main focus. Patterns in borrower behavior and creditworthiness can be found by closely examining the dataset.

Analyzing this dataset might yield important information about each borrower's spending patterns, financial behaviors, and possible danger.

The knowledge acquired can balance risk management and consumer outreach to maximize loan disbursement.

2 Task

Examining the data to assess possible borrowers' creditworthiness. Developing a logistic regression model, assessing its effectiveness, and offering useful information for the underwriting procedure are your ultimate goals.

3 Features

loan_amnt : The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value

term : The number of payments on the loan. Values are in months and can be either 36 or 60

int_rate : Interest Rate on the loan

installment : The monthly payment owed by the borrower if the loan originates. grade : LoanTap assigned loan grade

sub_grade : LoanTap assigned loan subgrade

emp_title :The job title supplied by the Borrower when applying for the loan.* emp_length : Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years home_ownership : The home ownership status provided by the borrower during registration or obtained from the credit report

annual_inc : The self-reported annual income provided by the borrower during registration

verification_status : Indicates if income was verified by LoanTap, not verified, or if the income source was verified

issue_d : The month which the loan was funded

loan_status : Current status of the loan - Target Variable

purpose : A category provided by the borrower for the loan request

title : The loan title provided by the borrower

dti : A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LoanTap loan, divided by the borrower's self-reported monthly income

earliest_cr_line :The month the borrower's earliest reported credit line was opened

open_acc : The number of open credit lines in the borrower's credit file

pub_rec : Number of derogatory public records

revol_bal : Total credit revolving balance

revol_util : Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit

total_acc : The total number of credit lines currently in the borrower's credit file

initial_list_status : The initial listing status of the loan. Possible values are – W, F

application_type : Indicates whether the loan is an individual application or a joint application with two co-borrowers

mort_acc : Number of mortgage accounts

pub_rec_bankruptcies : Number of public record bankruptcies

Address: Address of the individual

4 Concept Used:

Exploratory Data Analysis

Feature Engineering

Logistic Regression

Precision Vs Recall Tradeoff

5 Exploratory Data Analysis

[68]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from scipy.stats import ttest_ind,chi2_contingency

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.preprocessing import MinMaxScaler, LabelEncoder, StandardScaler
from sklearn.metrics import (
    accuracy_score, confusion_matrix, classification_report,
    roc_auc_score, roc_curve, auc, precision_recall_curve,␣
 ↪average_precision_score,
    ConfusionMatrixDisplay,␣
 ↪RocCurveDisplay,f1_score,recall_score,precision_score
)

from statsmodels.stats.outliers_influence import variance_inflation_factor
from imblearn.over_sampling import SMOTE

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

[69]:
```python
lt_data =pd.read_csv('/content/logistic_regression.csv')
df = lt_data.copy()
df.head()
```

[69]:
```
   loan_amnt        term  int_rate  installment grade sub_grade  \
0    10000.0   36 months     11.44       329.48     B        B4
1     8000.0   36 months     11.99       265.68     B        B5
2    15600.0   36 months     10.49       506.97     B        B3
3     7200.0   36 months      6.49       220.65     A        A2
4    24375.0   60 months     17.27       609.33     C        C5

                 emp_title emp_length home_ownership  annual_inc  \
0                Marketing  10+ years           RENT    117000.0
1          Credit analyst     4 years       MORTGAGE     65000.0
2             Statistician   < 1 year           RENT     43057.0
3          Client Advocate    6 years           RENT     54000.0
4  Destiny Management Inc.    9 years       MORTGAGE     55000.0

  verification_status   issue_d  loan_status             purpose  \
0        Not Verified  Jan-2015   Fully Paid            vacation
1        Not Verified  Jan-2015   Fully Paid  debt_consolidation
```

```
2     Source Verified  Jan-2015   Fully Paid         credit_card
3         Not Verified  Nov-2014   Fully Paid         credit_card
4             Verified  Apr-2013  Charged Off         credit_card

                     title    dti earliest_cr_line  open_acc  pub_rec  \
0                 Vacation  26.24         Jun-1990      16.0      0.0
1       Debt consolidation  22.05         Jul-2004      17.0      0.0
2  Credit card refinancing  12.79         Aug-2007      13.0      0.0
3  Credit card refinancing   2.60         Sep-2006       6.0      0.0
4    Credit Card Refinance  33.95         Mar-1999      13.0      0.0

   revol_bal  revol_util  total_acc initial_list_status application_type  \
0    36369.0        41.8       25.0                   w       INDIVIDUAL
1    20131.0        53.3       27.0                   f       INDIVIDUAL
2    11987.0        92.2       26.0                   f       INDIVIDUAL
3     5472.0        21.5       13.0                   f       INDIVIDUAL
4    24584.0        69.8       43.0                   f       INDIVIDUAL

   mort_acc  pub_rec_bankruptcies  \
0       0.0                   0.0
1       3.0                   0.0
2       0.0                   0.0
3       0.0                   0.0
4       1.0                   0.0

                                              address
0     0174 Michelle Gateway\r\nMendozaberg, OK 22690
1  1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113
2  87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113
3            823 Reid Ford\r\nDelacruzside, MA 00813
4             679 Luna Roads\r\nGreggshire, VA 11650
```

[70]:
```python
pd.set_option('display.max_columns', None)
```

6 Data Exploration

[71]:
```python
df.shape
```

[71]: (396030, 27)

[72]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
```

```
---  ------                 --------------  -----
 0   loan_amnt              396030 non-null  float64
 1   term                   396030 non-null  object
 2   int_rate               396030 non-null  float64
 3   installment            396030 non-null  float64
 4   grade                  396030 non-null  object
 5   sub_grade              396030 non-null  object
 6   emp_title              373103 non-null  object
 7   emp_length             377729 non-null  object
 8   home_ownership         396030 non-null  object
 9   annual_inc             396030 non-null  float64
 10  verification_status    396030 non-null  object
 11  issue_d                396030 non-null  object
 12  loan_status            396030 non-null  object
 13  purpose                396030 non-null  object
 14  title                  394274 non-null  object
 15  dti                    396030 non-null  float64
 16  earliest_cr_line       396030 non-null  object
 17  open_acc               396030 non-null  float64
 18  pub_rec                396030 non-null  float64
 19  revol_bal              396030 non-null  float64
 20  revol_util             395754 non-null  float64
 21  total_acc              396030 non-null  float64
 22  initial_list_status    396030 non-null  object
 23  application_type       396030 non-null  object
 24  mort_acc               358235 non-null  float64
 25  pub_rec_bankruptcies   395495 non-null  float64
 26  address                396030 non-null  object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

[73]:
```
df.columns
```

[73]:
```
Index(['loan_amnt', 'term', 'int_rate', 'installment', 'grade', 'sub_grade',
       'emp_title', 'emp_length', 'home_ownership', 'annual_inc',
       'verification_status', 'issue_d', 'loan_status', 'purpose', 'title',
       'dti', 'earliest_cr_line', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status', 'application_type',
       'mort_acc', 'pub_rec_bankruptcies', 'address'],
      dtype='object')
```

[74]:
```
df.describe().T
```

[74]:
```
                         count          mean          std     min      25%  \
loan_amnt             396030.0  14113.888089  8357.441341  500.00  8000.00
int_rate              396030.0     13.639400     4.472157    5.32    10.49
installment           396030.0    431.849698   250.727790   16.08   250.33
```

```
annual_inc           396030.0  74203.175798  61637.621158    0.00  45000.00
dti                  396030.0     17.379514     18.019092    0.00     11.28
open_acc             396030.0     11.311153      5.137649    0.00      8.00
pub_rec              396030.0      0.178191      0.530671    0.00      0.00
revol_bal            396030.0  15844.539853  20591.836109    0.00   6025.00
revol_util           395754.0     53.791749     24.452193    0.00     35.80
total_acc            396030.0     25.414744     11.886991    2.00     17.00
mort_acc             358235.0      1.813991      2.147930    0.00      0.00
pub_rec_bankruptcies 395495.0      0.121648      0.356174    0.00      0.00

                          50%       75%         max
loan_amnt            12000.00  20000.00    40000.00
int_rate                13.33     16.49       30.99
installment            375.43    567.30     1533.81
annual_inc           64000.00  90000.00  8706582.00
dti                     16.91     22.98     9999.00
open_acc                10.00     14.00       90.00
pub_rec                  0.00      0.00       86.00
revol_bal            11181.00  19620.00  1743266.00
revol_util              54.80     72.90      892.30
total_acc               24.00     32.00      151.00
mort_acc                 1.00      3.00       34.00
pub_rec_bankruptcies     0.00      0.00        8.00
```

7 Duplicate and Null Value Detection

[75]:
```
df[df.duplicated()]
```

[75]:
```
Empty DataFrame
Columns: [loan_amnt, term, int_rate, installment, grade, sub_grade, emp_title,
emp_length, home_ownership, annual_inc, verification_status, issue_d,
loan_status, purpose, title, dti, earliest_cr_line, open_acc, pub_rec,
revol_bal, revol_util, total_acc, initial_list_status, application_type,
mort_acc, pub_rec_bankruptcies, address]
Index: []
```

[76]:
```
df.isna().any()[df.isna().any()]
```

[76]:
```
emp_title               True
emp_length              True
title                   True
revol_util              True
mort_acc                True
pub_rec_bankruptcies    True
dtype: bool
```

[77]:
```python
df.isna().sum().sort_values(ascending=False)
```

[77]:
```
mort_acc                37795
emp_title               22927
emp_length              18301
title                    1756
pub_rec_bankruptcies      535
revol_util                276
loan_amnt                   0
dti                         0
application_type            0
initial_list_status         0
total_acc                   0
revol_bal                   0
pub_rec                     0
open_acc                    0
earliest_cr_line            0
purpose                     0
term                        0
loan_status                 0
issue_d                     0
verification_status         0
annual_inc                  0
home_ownership              0
sub_grade                   0
grade                       0
installment                 0
int_rate                    0
address                     0
dtype: int64
```

[78]:
```python
def missing_data(df):
    total_missing_df = df.isnull().sum().sort_values(ascending =False)
    percent_missing_df = (df.isnull().sum()/df.isna().count()*100).
 ↪sort_values(ascending=False)
    missing_data_df = pd.concat([total_missing_df, percent_missing_df], axis=1,␣
 ↪keys=['Total', 'Percent'])
    return missing_data_df

missing_pct = missing_data(df)
missing_pct[missing_pct['Total']>0]
```

[78]:

	Total	Percent
mort_acc	37795	9.543469
emp_title	22927	5.789208
emp_length	18301	4.621115
title	1756	0.443401

```
pub_rec_bankruptcies    535   0.135091
revol_util              276   0.069692
```

7.1 Observation

5.78% of the values in insight_emp_title are missing.

4.62% of the values in emp_length are missing.

0.44% of the title's values are missing.

0.06% of revol_until's values are missing.

9.54% of mort_acc's values are missing.

0.13% of pub_rec_bankruptcies' values are missing.

Checking for Unique Values

[79]:
```python
for _ in df.columns:
    print()
    print(f'Total Unique Values in {_} column are :- {df[_].nunique()}')
    print(f'Unique Values in {_} column are :-\n {df[_].unique()}')
    print(f'Value_counts of {_} column :-\n {df[_].value_counts()}')
    print()
    print('-'*120)
```

```
Total Unique Values in loan_amnt column are :- 1397
Unique Values in loan_amnt column are :-
 [10000.  8000. 15600. … 36275. 36475.   725.]
Value_counts of loan_amnt column :-
 loan_amnt
10000.0    27668
12000.0    21366
15000.0    19903
20000.0    18969
35000.0    14576
           …
36225.0        1
950.0          1
37800.0        1
30050.0        1
725.0          1
Name: count, Length: 1397, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in term column are :- 2
Unique Values in term column are :-
```

```
 [' 36 months' ' 60 months']
Value_counts of term column :-
 term
36 months    302005
60 months     94025
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in int_rate column are :- 566
Unique Values in int_rate column are :-
 [11.44 11.99 10.49  6.49 17.27 13.33  5.32 11.14 10.99 16.29 13.11 14.64
  9.17 12.29  6.62  8.39 21.98  7.9   6.97  6.99 15.61 11.36 13.35 12.12
  9.99  8.19 18.75  6.03 14.99 16.78 13.67 13.98 16.99 19.91 17.86 21.49
 12.99 18.54  7.89 17.1  18.25 11.67  6.24  8.18 12.35 14.16 17.56 18.55
 22.15 10.39 15.99 16.07 24.99  9.67 19.19 21.   12.69 10.74  6.68 19.22
 11.49 16.55 19.97 24.7  13.49 18.24 16.49 25.78 25.83 18.64  7.51 13.99
 15.22 15.31  7.69 19.53 10.16  7.62  9.75 13.68 15.88 14.65  6.92 23.83
 10.75 18.49 20.31 17.57 27.31 19.99 22.99 12.59 10.37 14.33 13.53 22.45
 24.5  17.99  9.16 12.49 11.55 17.76 28.99 23.1  20.49 22.7  10.15  6.89
 19.52  8.9  14.3   9.49 25.99 24.08 13.05 14.98 16.59 11.26 25.89 14.48
 21.99 23.99  5.99 14.47 11.53  8.67  8.59 10.64 23.28 25.44  9.71 16.2
 19.24 24.11 15.8  15.96 14.49 18.99  5.79 19.29 14.54 14.09  9.25 19.05
 17.77 18.92 20.75 10.65 18.85 10.59 12.85 11.39 13.65 13.06  7.12 20.99
 13.61 12.73 14.46 16.24 25.49  7.39 10.78 20.8   7.88 15.95 12.39 21.18
 21.97 15.77  6.39 10.   12.53 13.43  7.49 25.57 21.48 18.39 11.47  7.26
 15.68 19.04 14.31 24.24  5.42 23.43 19.47  6.54 23.32 17.58 14.72  7.66
  9.76 13.23 13.48 12.42  9.8  11.71 14.27 21.15 22.95  8.49 17.74 15.59
 13.72  9.45  7.29 15.1  11.86 19.72 14.35 11.22 15.62 15.81 12.41 28.67
 11.48 13.66  9.91 23.76 17.14 18.84 12.23  6.17  8.94 14.22 19.03 25.29
  8.99  9.88 15.58 27.49  8.07 22.47 19.2  13.44 22.4  12.79 18.2  13.18
  7.24 14.84  5.93 15.28 13.85 25.28  8.    9.62 12.05 15.7  20.2  13.57
 21.67  7.4  25.8  12.68 11.83  7.37 11.11 14.85 16.   11.12 23.63  6.
  7.99  7.91 14.83 21.7  26.06 16.77 27.34 12.21  7.68 15.27 19.69  9.63
  7.14 20.5  16.02 12.84  7.74 15.33 19.79 22.2  18.62 17.49 16.89 15.21
 14.79 18.67  9.32 15.41 15.65 23.5  22.9  11.34 22.11 19.48 14.75 28.14
 13.22 23.4  23.13 28.18 12.88 22.06 24.49 16.45 21.6  28.49  8.38  6.76
 10.83 13.79  8.88 17.88 17.97 14.26  6.91 13.47  8.6  27.88  8.63 10.25
 14.91 12.74 10.96 25.88  7.43 16.4  20.25 24.89 12.87 20.16 14.17 12.18
 17.51 13.92 20.53 26.77 10.62 26.49 16.32 12.61 21.36 14.61 15.37 20.3
 14.59 16.7  19.89 10.95 18.17 18.21 17.93 22.39 24.83 13.8  19.42 23.7
  7.59 13.17 18.09 13.04 25.69  9.07 15.23 14.42 23.33 16.69 10.36 14.96
 10.38 26.24 24.2  12.98 20.85 13.36 26.57 23.52 22.78 13.16 15.13 25.11
 13.55 10.51 11.78  7.05 11.46 21.28 12.09 16.35  8.7  26.99 14.11 26.14
 16.82 23.26 18.79 10.28 19.36 18.3  17.06 17.19  7.75 17.34 20.89 22.35
 19.66 13.62 22.74 11.89 23.59  8.24 20.62 11.97 15.2  20.48 12.36 10.71
 25.09 20.11 27.79 29.49 11.58 19.13 11.66 13.75 30.74  9.38 27.99 11.59
```

```
  9.64 25.65  9.96 19.41 14.18 10.08 17.43 24.74 14.74 17.04 15.57 30.49
 17.8  10.91 14.82 29.96 12.92 12.22 15.45 11.72 10.2  14.7  20.69 15.05
 24.33 14.93 10.33 16.95 28.88 11.03 28.34 21.22 18.07  9.33 12.17 19.74
 20.9  20.03 17.39 29.67 12.04 23.22 10.01 22.48 24.76 13.3  20.77 10.14
 14.5  30.94  8.32 13.24 21.59 21.27 24.52 11.54 10.46 13.87 30.99  9.51
  9.83 19.39 12.86 30.79 21.74 11.09 16.11 17.26 22.85 18.91 18.43  9.2
 21.14 12.62 21.21 29.99 14.88 13.12 30.89 16.08 12.54 28.69 12.8  11.28
 23.91 22.94 19.16 20.86 11.63 19.82 11.41 21.82 12.72 20.4   9.7  18.72
 18.36 14.25 13.84 18.78 17.15 15.25 16.63 16.15 11.91 14.07  9.01 15.01
 21.64 15.83 18.53  7.42 12.67 15.76 16.33 30.84 13.93 14.12 14.28 20.17
 24.59 20.52 17.03 17.9  14.67 15.38 17.46 14.62 14.38 24.4  22.64 17.54
 17.44 15.07]
Value_counts of int_rate column :-
 int_rate
10.99    12411
12.99     9632
15.61     9350
11.99     8582
8.90      8019
         …
14.28        1
18.72        1
18.36        1
30.84        1
24.59        1
Name: count, Length: 566, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in installment column are :- 55706
Unique Values in installment column are :-
 [329.48 265.68 506.97 … 343.14 118.13 572.44]
Value_counts of installment column :-
 installment
327.34     968
332.10     791
491.01     736
336.90     686
392.81     683
          …
364.37       1
1015.29      1
398.04       1
544.94       1
572.44       1
Name: count, Length: 55706, dtype: int64
```

```
--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in grade column are :- 7
Unique Values in grade column are :-
 ['B' 'A' 'C' 'E' 'D' 'F' 'G']
Value_counts of grade column :-
 grade
B    116018
C    105987
A     64187
D     63524
E     31488
F     11772
G      3054
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in sub_grade column are :- 35
Unique Values in sub_grade column are :-
 ['B4' 'B5' 'B3' 'A2' 'C5' 'C3' 'A1' 'B2' 'C1' 'A5' 'E4' 'A4' 'A3' 'D1'
 'C2' 'B1' 'D3' 'D5' 'D2' 'E1' 'E2' 'E5' 'F4' 'E3' 'D4' 'G1' 'F5' 'G2'
 'C4' 'F1' 'F3' 'G5' 'G4' 'F2' 'G3']
Value_counts of sub_grade column :-
 sub_grade
B3    26655
B4    25601
C1    23662
C2    22580
B2    22495
B5    22085
C3    21221
C4    20280
B1    19182
A5    18526
C5    18244
D1    15993
A4    15789
D2    13951
D3    12223
D4    11657
A3    10576
A1     9729
D5     9700
A2     9567
E1     7917
```

```
E2     7431
E3     6207
E4     5361
E5     4572
F1     3536
F2     2766
F3     2286
F4     1787
F5     1397
G1     1058
G2      754
G3      552
G4      374
G5      316
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in emp_title column are :- 173105
Unique Values in emp_title column are :-
 ['Marketing' 'Credit analyst ' 'Statistician' …
 "Michael's Arts & Crafts" 'licensed bankere' 'Gracon Services, Inc']
Value_counts of emp_title column :-
 emp_title
Teacher                    4389
Manager                    4250
Registered Nurse           1856
RN                         1846
Supervisor                 1830
                           …
Postman                       1
McCarthy & Holthus, LLC       1
jp flooring                   1
Histology Technologist        1
Gracon Services, Inc          1
Name: count, Length: 173105, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in emp_length column are :- 11
Unique Values in emp_length column are :-
 ['10+ years' '4 years' '< 1 year' '6 years' '9 years' '2 years' '3 years'
 '8 years' '7 years' '5 years' '1 year' nan]
Value_counts of emp_length column :-
 emp_length
10+ years    126041
```

```
2 years       35827
< 1 year      31725
3 years       31665
5 years       26495
1 year        25882
4 years       23952
6 years       20841
7 years       20819
8 years       19168
9 years       15314
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in home_ownership column are :- 6
Unique Values in home_ownership column are :-
 ['RENT' 'MORTGAGE' 'OWN' 'OTHER' 'NONE' 'ANY']
Value_counts of home_ownership column :-
 home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in annual_inc column are :- 27197
Unique Values in annual_inc column are :-
 [117000.    65000.    43057.   …  36111.    47212.    31789.88]
Value_counts of annual_inc column :-
 annual_inc
60000.00    15313
50000.00    13303
65000.00    11333
70000.00    10674
40000.00    10629
            …
72179.00        1
50416.00        1
46820.80        1
10368.00        1
31789.88        1
Name: count, Length: 27197, dtype: int64
```

```
--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in verification_status column are :- 3
Unique Values in verification_status column are :-
 ['Not Verified' 'Source Verified' 'Verified']
Value_counts of verification_status column :-
 verification_status
Verified           139563
Source Verified    131385
Not Verified       125082
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in issue_d column are :- 115
Unique Values in issue_d column are :-
 ['Jan-2015' 'Nov-2014' 'Apr-2013' 'Sep-2015' 'Sep-2012' 'Oct-2014'
 'Apr-2012' 'Jun-2013' 'May-2014' 'Dec-2015' 'Apr-2015' 'Oct-2012'
 'Jul-2014' 'Feb-2013' 'Oct-2015' 'Jan-2014' 'Mar-2016' 'Apr-2014'
 'Jun-2011' 'Apr-2010' 'Jun-2014' 'Oct-2013' 'May-2013' 'Feb-2015'
 'Oct-2011' 'Jun-2015' 'Aug-2013' 'Feb-2014' 'Dec-2011' 'Mar-2013'
 'Jun-2016' 'Mar-2014' 'Nov-2013' 'Dec-2014' 'Apr-2016' 'Sep-2013'
 'May-2016' 'Jul-2015' 'Jul-2013' 'Aug-2014' 'May-2008' 'Mar-2010'
 'Dec-2013' 'Mar-2012' 'Mar-2015' 'Sep-2011' 'Jul-2012' 'Dec-2012'
 'Sep-2014' 'Nov-2012' 'Nov-2015' 'Jan-2011' 'May-2012' 'Feb-2016'
 'Jun-2012' 'Aug-2012' 'Jan-2016' 'May-2015' 'Oct-2016' 'Aug-2015'
 'Jul-2016' 'May-2009' 'Aug-2016' 'Jan-2012' 'Jan-2013' 'Nov-2010'
 'Jul-2011' 'Mar-2011' 'Feb-2012' 'May-2011' 'Aug-2010' 'Nov-2016'
 'Jul-2010' 'Sep-2010' 'Dec-2010' 'Feb-2011' 'Jun-2009' 'Aug-2011'
 'Dec-2016' 'Mar-2009' 'Jun-2010' 'May-2010' 'Nov-2011' 'Sep-2016'
 'Oct-2009' 'Mar-2008' 'Nov-2008' 'Dec-2009' 'Oct-2010' 'Sep-2009'
 'Oct-2007' 'Aug-2009' 'Jul-2009' 'Nov-2009' 'Jan-2010' 'Dec-2008'
 'Feb-2009' 'Oct-2008' 'Apr-2009' 'Feb-2010' 'Apr-2011' 'Apr-2008'
 'Aug-2008' 'Jan-2009' 'Feb-2008' 'Aug-2007' 'Sep-2008' 'Dec-2007'
 'Jan-2008' 'Sep-2007' 'Jun-2008' 'Jul-2008' 'Jun-2007' 'Nov-2007'
 'Jul-2007']
Value_counts of issue_d column :-
 issue_d
Oct-2014    14846
Jul-2014    12609
Jan-2015    11705
Dec-2013    10618
Nov-2013    10496
            …
Jul-2007       26
```

```
Sep-2008       25
Nov-2007       22
Sep-2007       15
Jun-2007        1
Name: count, Length: 115, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in loan_status column are :- 2
Unique Values in loan_status column are :-
 ['Fully Paid' 'Charged Off']
Value_counts of loan_status column :-
 loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in purpose column are :- 14
Unique Values in purpose column are :-
 ['vacation' 'debt_consolidation' 'credit_card' 'home_improvement'
 'small_business' 'major_purchase' 'other' 'medical' 'wedding' 'car'
 'moving' 'house' 'educational' 'renewable_energy']
Value_counts of purpose column :-
 purpose
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
house                   2201
wedding                 1812
renewable_energy         329
educational              257
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in title column are :- 48816
```

```
Unique Values in title column are :-
 ['Vacation' 'Debt consolidation' 'Credit card refinancing' …
 'Credit buster ' 'Loanforpayoff' 'Toxic Debt Payoff']
Value_counts of title column :-
 title
Debt consolidation            152472
Credit card refinancing        51487
Home improvement               15264
Other                          12930
Debt Consolidation             11608
                                …
Graduation/Travel Expenses         1
Daughter's Wedding Bill            1
gotta move                         1
creditcardrefi                     1
Toxic Debt Payoff                  1
Name: count, Length: 48816, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in dti column are :- 4262
Unique Values in dti column are :-
 [26.24 22.05 12.79 … 40.56 47.09 55.53]
Value_counts of dti column :-
 dti
0.00     313
14.40    310
19.20    302
16.80    301
18.00    300
        …
59.18      1
48.37      1
45.71      1
42.38      1
55.53      1
Name: count, Length: 4262, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in earliest_cr_line column are :- 684
Unique Values in earliest_cr_line column are :-
 ['Jun-1990' 'Jul-2004' 'Aug-2007' 'Sep-2006' 'Mar-1999' 'Jan-2005'
 'Aug-2005' 'Sep-1994' 'Jun-1994' 'Dec-1997' 'Dec-1990' 'May-1984'
 'Apr-1995' 'Jan-1997' 'May-2001' 'Mar-1982' 'Sep-1996' 'Jan-1990'
 'Mar-2000' 'Jan-2006' 'Oct-2006' 'Jan-2003' 'May-2008' 'Oct-2003'
```

'Jun-2004'	'Jan-1999'	'Apr-1994'	'Apr-1998'	'Jul-2007'	'Apr-2002'
'Oct-2007'	'Jun-2009'	'May-1997'	'Jul-2006'	'Sep-2003'	'Aug-1992'
'Dec-1988'	'Feb-2002'	'Jan-1992'	'Aug-2001'	'Dec-2010'	'Oct-1999'
'Sep-2004'	'Aug-1994'	'Jul-2003'	'Apr-2000'	'Dec-2004'	'Jun-1995'
'Dec-2003'	'Jul-1994'	'Oct-1990'	'Dec-2001'	'Apr-1999'	'Feb-1995'
'May-2003'	'Oct-2002'	'Mar-2004'	'Aug-2003'	'Oct-2000'	'Nov-2004'
'Mar-2010'	'Mar-1996'	'May-1994'	'Jun-1996'	'Nov-1986'	'Jan-2001'
'Jan-2002'	'Mar-2001'	'Sep-2012'	'Apr-2006'	'May-1998'	'Dec-2002'
'Nov-2003'	'Oct-2005'	'May-1990'	'Jun-2003'	'Jun-2001'	'Jan-1998'
'Oct-1978'	'Feb-2001'	'Jun-2006'	'Aug-1993'	'Apr-2001'	'Nov-2001'
'Feb-2003'	'Jun-1993'	'Sep-1992'	'Nov-1992'	'Jun-1983'	'Oct-2001'
'Jul-1999'	'Sep-1997'	'Nov-1993'	'Feb-1993'	'Apr-2007'	'Nov-1999'
'Nov-2005'	'Dec-1992'	'Mar-1986'	'May-1989'	'Dec-2000'	'Mar-1991'
'Mar-2005'	'Jun-2010'	'Dec-1998'	'Sep-2001'	'Nov-2000'	'Jan-1994'
'Aug-2002'	'Jan-2011'	'Aug-2008'	'Jun-2005'	'Nov-1997'	'May-1996'
'Apr-2010'	'May-1993'	'Sep-2005'	'Jun-1992'	'Apr-1986'	'Aug-1996'
'Aug-1997'	'Jul-2005'	'May-2011'	'Sep-2002'	'Jan-1989'	'Aug-1999'
'Feb-1992'	'Sep-1999'	'Jul-2001'	'May-1980'	'Oct-2008'	'Nov-2007'
'Apr-1997'	'Jun-1986'	'Sep-1998'	'Jun-1982'	'Oct-1981'	'Feb-1994'
'Dec-1984'	'Nov-1991'	'Nov-2006'	'Aug-2000'	'Oct-2004'	'Jun-2011'
'Apr-1988'	'May-2004'	'Aug-1988'	'Mar-1994'	'Aug-2004'	'Dec-2006'
'Nov-1998'	'Oct-1997'	'Mar-1989'	'Feb-1988'	'Jul-1982'	'Nov-1995'
'Mar-1997'	'Oct-1994'	'Jul-1998'	'Jun-2002'	'May-1991'	'Oct-2011'
'Sep-2007'	'Jan-2007'	'Jan-2010'	'Mar-1987'	'Feb-1997'	'Oct-1986'
'Mar-2002'	'Jul-1993'	'Mar-2007'	'Aug-1989'	'Oct-1995'	'May-2007'
'Dec-1993'	'Jun-1989'	'Apr-2004'	'Jun-1997'	'Apr-1996'	'Apr-1992'
'Oct-1998'	'Mar-1983'	'Mar-1985'	'Oct-1993'	'Feb-2000'	'Apr-2003'
'Oct-1985'	'Jul-1985'	'May-1978'	'Sep-2010'	'Oct-1996'	'Sep-2009'
'Jun-1999'	'Jan-2000'	'Sep-1987'	'Aug-1998'	'Jan-1995'	'Jul-1988'
'May-2000'	'Jun-1981'	'Feb-1998'	'Nov-1996'	'Aug-1967'	'Dec-1999'
'Aug-2006'	'Nov-2009'	'Jul-2000'	'Mar-1988'	'Jul-1992'	'Jul-1991'
'Mar-1990'	'May-1986'	'Jun-1991'	'Dec-1987'	'Jul-1996'	'Jul-1997'
'Aug-1990'	'Jan-1988'	'Dec-2005'	'Mar-2003'	'Feb-1999'	'Nov-1990'
'Jun-2000'	'Dec-1996'	'Jan-2004'	'May-1999'	'Sep-1972'	'Jul-1981'
'Sep-1993'	'Feb-2009'	'Nov-2002'	'Nov-1969'	'Jan-1993'	'May-2005'
'Sep-1982'	'Apr-1990'	'Feb-1996'	'Mar-1993'	'Apr-1978'	'Jul-1995'
'May-1995'	'Apr-1991'	'Mar-1998'	'Aug-1991'	'Jul-2002'	'Oct-1989'
'Apr-1984'	'Dec-2009'	'Sep-2000'	'Jan-1982'	'Jun-1998'	'Jan-1996'
'Nov-1987'	'May-2010'	'Jul-1989'	'Jun-1987'	'Oct-1987'	'Aug-1995'
'Feb-2004'	'Oct-1991'	'Dec-1989'	'Oct-1992'	'Feb-2005'	'Apr-1993'
'Dec-1985'	'Sep-1979'	'Feb-2007'	'Nov-1989'	'Apr-2005'	'Mar-1978'
'Sep-1985'	'Nov-1994'	'Jun-2008'	'Apr-1987'	'Dec-1983'	'Dec-2007'
'May-1979'	'May-1992'	'Jul-1990'	'Mar-1995'	'Feb-2006'	'Feb-1985'
'Sep-1989'	'Aug-2009'	'Nov-2008'	'Nov-1981'	'Jan-2008'	'Aug-1987'
'Nov-1985'	'Dec-1965'	'Sep-1995'	'Jan-1986'	'Oct-2009'	'May-2002'
'Aug-1980'	'Sep-1977'	'Sep-1988'	'Oct-1984'	'May-1988'	'Aug-1984'
'Nov-1988'	'May-1974'	'Nov-1982'	'Oct-1983'	'Sep-1991'	'Feb-1984'
'Feb-1991'	'Jan-1981'	'Jun-1985'	'Dec-1976'	'Dec-1994'	'Dec-1980'

```
'Sep-1984' 'Jun-2007' 'Aug-1979' 'Sep-2008' 'Apr-1983' 'Mar-2006'
'Jun-1984' 'Jul-1984' 'Jan-1985' 'Dec-1995' 'Apr-2008' 'Mar-2008'
'Jan-1983' 'Dec-1986' 'Jun-1979' 'Dec-1975' 'Nov-1983' 'Jul-1986'
'Nov-1977' 'Dec-1982' 'May-1985' 'Feb-1983' 'Aug-1982' 'Oct-1980'
'Mar-1979' 'Jan-1978' 'Mar-1984' 'May-1983' 'Jul-2008' 'Apr-1982'
'Jul-1983' 'Feb-1990' 'Dec-2008' 'Jul-1975' 'Dec-1971' 'Feb-2008'
'Mar-2011' 'Feb-1987' 'Feb-1989' 'Aug-1985' 'Jul-2010' 'Apr-1989'
'Feb-1980' 'May-2006' 'Nov-2010' 'Apr-2009' 'Feb-2010' 'May-1976'
'Feb-1981' 'Jan-2012' 'Oct-1988' 'Nov-1984' 'May-1982' 'Oct-1975'
'Jun-1988' 'May-1972' 'Apr-2013' 'Sep-1990' 'Oct-1982' 'Feb-2013'
'Mar-1992' 'Aug-1981' 'Feb-2011' 'Nov-1974' 'Feb-1978' 'Sep-1983'
'Jul-2011' 'Nov-1979' 'Aug-1983' 'Apr-1985' 'Jul-2009' 'Jan-1971'
'Jul-1987' 'Aug-1978' 'Aug-2010' 'Oct-1976' 'Aug-1986' 'Jan-1991'
'Dec-1991' 'May-2009' 'Aug-2011' 'Jun-1964' 'Jan-1974' 'May-1981'
'Jun-1972' 'Jun-1978' 'Sep-1986' 'Jan-1987' 'Jan-1975' 'Feb-1982'
'Jan-1980' 'Feb-1977' 'Sep-1980' 'Nov-1978' 'Jul-1974' 'Jun-1970'
'Jan-1984' 'Nov-1980' 'May-1987' 'Sep-1970' 'Jan-1976' 'Feb-1986'
'Oct-2010' 'Apr-1979' 'Oct-1979' 'Jan-1979' 'Sep-2011' 'Jul-1979'
'Sep-1975' 'Mar-1981' 'Aug-1971' 'Apr-1980' 'Apr-1977' 'Jan-1965'
'Nov-1976' 'Nov-1970' 'Nov-2011' 'Nov-1973' 'Sep-1981' 'Jul-1980'
'Mar-2012' 'Dec-1974' 'Mar-1977' 'Dec-1977' 'May-2012' 'Dec-1979'
'Jan-2009' 'Jan-1970' 'Dec-2011' 'Feb-1979' 'Mar-1976' 'Jan-1973'
'Oct-1973' 'Mar-1969' 'Oct-1977' 'Mar-1975' 'Aug-1977' 'Jun-1969'
'Oct-1963' 'Nov-1960' 'Aug-1970' 'Feb-1975' 'Sep-1974' 'May-1966'
'Apr-1972' 'Apr-1973' 'Apr-2012' 'May-1975' 'Sep-1966' 'Feb-1969'
'Feb-2012' 'Jan-1961' 'Aug-1973' 'Feb-1972' 'Apr-1975' 'Jul-1978'
'Oct-1970' 'Mar-1980' 'Sep-1976' 'Apr-2011' 'Nov-2012' 'Aug-1976'
'Jun-1975' 'Apr-1981' 'Mar-2009' 'Jun-1977' 'Apr-1971' 'Sep-1969'
'Jun-2012' 'Apr-1976' 'Feb-1965' 'Jul-1977' 'Jun-1976' 'Mar-1973'
'Oct-1972' 'Dec-1978' 'Nov-1967' 'Sep-1967' 'Nov-1971' 'Jun-1980'
'May-1964' 'Feb-1971' 'May-1970' 'Apr-1970' 'Mar-1971' 'Apr-1969'
'Jan-1963' 'Jun-1974' 'Oct-1974' 'May-1977' 'Dec-1981' 'Jan-1969'
'Feb-1976' 'Mar-1970' 'Aug-1968' 'Feb-1970' 'Jun-1971' 'Jun-1963'
'Jun-2013' 'Mar-1972' 'Aug-2012' 'Jan-1967' 'Feb-1968' 'Dec-1969'
'Jan-1977' 'Jul-1970' 'Feb-1973' 'Mar-1974' 'Feb-1974' 'Dec-1960'
'Jul-1972' 'Jul-1973' 'Sep-1964' 'Jul-1965' 'Oct-1958' 'Jul-2012'
'Jun-1973' 'Sep-1978' 'Nov-1975' 'Jul-1963' 'Jan-1964' 'Dec-1968'
'May-1958' 'Sep-1973' 'May-1971' 'Dec-1972' 'Aug-1965' 'Jul-1976'
'Oct-2012' 'May-1973' 'Apr-1955' 'Apr-1966' 'Jan-1968' 'Nov-1968'
'Oct-1969' 'Mar-2013' 'Jan-2013' 'Jul-1967' 'Oct-1965' 'Jan-1966'
'Aug-1972' 'Jul-1969' 'May-1965' 'Jan-1953' 'Aug-1974' 'May-1968'
'Aug-1969' 'May-2013' 'Oct-1967' 'Aug-1975' 'Apr-1974' 'Sep-1971'
'Apr-1968' 'Jul-1971' 'Jan-1972' 'Nov-1965' 'Dec-1970' 'Dec-1973'
'Nov-1972' 'Oct-1959' 'Oct-1962' 'Apr-1967' 'Oct-1971' 'Nov-1963'
'Oct-1968' 'Dec-1962' 'Jun-1960' 'Jan-1960' 'Sep-2013' 'May-1969'
'Dec-1966' 'Feb-1967' 'Dec-1967' 'Aug-1961' 'Sep-1968' 'Oct-1964'
'Aug-1966' 'Jul-1966' 'Apr-1964' 'Sep-1962' 'Jul-2013' 'Jun-1967'
'Apr-1965' 'Jun-1966' 'Jan-1955' 'Jan-1962' 'Feb-1964' 'Aug-1958'
```

```
 'Jul-1968' 'May-1967' 'Dec-1959' 'Sep-1963' 'Dec-2012' 'Dec-1963'
 'Jan-1944' 'Jun-1965' 'May-1962' 'Mar-1967' 'Mar-1968' 'Jan-1956'
 'Sep-1965' 'Dec-1951' 'Aug-2013' 'Jun-1968' 'Mar-1965' 'Oct-1957'
 'Nov-1966' 'Dec-1958' 'Feb-1957' 'Feb-1963' 'Mar-1963' 'Jan-1959'
 'May-1955' 'Feb-1966' 'Nov-1950' 'Mar-1964' 'Jan-1958' 'Nov-1964'
 'Sep-1961' 'Apr-1963' 'Jul-1964' 'Nov-1955' 'Jun-1957' 'Dec-1964'
 'Nov-1953' 'Apr-1961' 'Mar-1966' 'Oct-1960' 'Jul-1959' 'Jul-1961'
 'Jan-1954' 'Dec-1956' 'Mar-1962' 'Jul-1960' 'Sep-1959' 'Dec-1950'
 'Oct-1966' 'Apr-1960' 'Jul-1958' 'Nov-1954' 'Nov-1957' 'Jun-1962'
 'May-1963' 'Jul-1955' 'Oct-1950' 'Dec-1961' 'Aug-1951' 'Oct-2013'
 'Aug-1964' 'Apr-1962' 'Jun-1955' 'Jul-1962' 'Jan-1957' 'Nov-1958'
 'Jul-1951' 'Nov-1959' 'Apr-1958' 'Mar-1960' 'Sep-1957' 'Nov-1961'
 'Sep-1960' 'May-1959' 'Jun-1959' 'Feb-1962' 'Sep-1956' 'Aug-1960'
 'Feb-1961' 'Jan-1948' 'Aug-1963' 'Oct-1961' 'Aug-1962' 'Aug-1959']
Value_counts of earliest_cr_line column :-
 earliest_cr_line
Oct-2000    3017
Aug-2000    2935
Oct-2001    2896
Aug-2001    2884
Nov-2000    2736
            …
Jul-1958       1
Nov-1957       1
Jan-1953       1
Jul-1955       1
Aug-1959       1
Name: count, Length: 684, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in open_acc column are :- 61
Unique Values in open_acc column are :-
 [16. 17. 13.  6.  8. 11.  5. 30.  9. 15. 12. 10. 18.  7.  4. 14. 20. 19.
 21. 23.  3. 26. 42. 22. 25. 28.  2. 34. 24. 27. 31. 32. 33.  1. 29. 36.
 40. 35. 37. 41. 44. 39. 49. 48. 38. 51. 50. 43. 46.  0. 47. 57. 53. 58.
 52. 54. 45. 90. 56. 55. 76.]
Value_counts of open_acc column :-
 open_acc
9.0     36779
10.0    35441
8.0     35137
11.0    32695
7.0     31328
        …
55.0        2
76.0        2
```

```
58.0        1
57.0        1
90.0        1
Name: count, Length: 61, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in pub_rec column are :- 20
Unique Values in pub_rec column are :-
 [ 0.  1.  2.  3.  4.  6.  5.  8.  9. 10. 11.  7. 19. 13. 40. 17. 86. 12.
 24. 15.]
Value_counts of pub_rec column :-
 pub_rec
0.0     338272
1.0      49739
2.0       5476
3.0       1521
4.0        527
5.0        237
6.0        122
7.0         56
8.0         34
9.0         12
10.0        11
11.0         8
13.0         4
12.0         4
19.0         2
40.0         1
17.0         1
86.0         1
24.0         1
15.0         1
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in revol_bal column are :- 55622
Unique Values in revol_bal column are :-
 [ 36369.  20131.  11987. …  34531. 151912.  29244.]
Value_counts of revol_bal column :-
 revol_bal
0.0         2128
5655.0        41
6095.0        38
7792.0        38
```

```
3953.0        37
            …
42573.0        1
72966.0        1
105342.0       1
37076.0        1
29244.0        1
Name: count, Length: 55622, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in revol_util column are :- 1226
Unique Values in revol_util column are :-
 [ 41.8   53.3   92.2  …  56.26 111.4  128.1 ]
Value_counts of revol_util column :-
 revol_util
0.00      2213
53.00      752
60.00      739
61.00      734
55.00      730
          …
892.30       1
110.10       1
123.00       1
49.63        1
128.10       1
Name: count, Length: 1226, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in total_acc column are :- 118
Unique Values in total_acc column are :-
 [ 25.  27.  26.  13.  43.  23.  15.  40.  37.  61.  35.  22.  20.  36.
  38.   7.  18.  10.  17.  29.  16.  21.  34.   9.  14.  59.  41.  19.
  12.  30.  56.  24.  28.   8.  52.  31.  44.  39.  50.  11.  62.  32.
   5.  33.  46.  42.   6.  49.  45.  57.  48.  67.  47.  51.  58.   3.
  55.  63.  53.   4.  71.  69.  54.  64.  81.  72.  60.  68.  65.  73.
  78.  84.   2.  76.  75.  79.  87.  77. 104.  89.  70. 105.  97.  66.
 108.  74.  80.  82.  91.  93. 106.  90.  85.  88.  83. 111.  86. 101.
 135.  92.  94.  95.  99. 102. 129. 110. 124. 151. 107. 118. 150. 115.
 117.  96.  98. 100. 116. 103.]
Value_counts of total_acc column :-
 total_acc
21.0     14280
22.0     14260
```

```
20.0     14228
23.0     13923
24.0     13878
         …
110.0        1
129.0        1
135.0        1
104.0        1
103.0        1
Name: count, Length: 118, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in initial_list_status column are :- 2
Unique Values in initial_list_status column are :-
 ['w' 'f']
Value_counts of initial_list_status column :-
 initial_list_status
f    238066
w    157964
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in application_type column are :- 3
Unique Values in application_type column are :-
 ['INDIVIDUAL' 'JOINT' 'DIRECT_PAY']
Value_counts of application_type column :-
 application_type
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in mort_acc column are :- 33
Unique Values in mort_acc column are :-
 [ 0.  3.  1.  4.  2.  6.  5. nan 10.  7. 12. 11.  8.  9. 13. 14. 22. 34.
 15. 25. 19. 16. 17. 32. 18. 24. 21. 20. 31. 28. 30. 23. 26. 27.]
Value_counts of mort_acc column :-
 mort_acc
0.0     139777
1.0      60416
2.0      49948
```

```
3.0      38049
4.0      27887
5.0      18194
6.0      11069
7.0       6052
8.0       3121
9.0       1656
10.0       865
11.0       479
12.0       264
13.0       146
14.0       107
15.0        61
16.0        37
17.0        22
18.0        18
19.0        15
20.0        13
24.0        10
22.0         7
21.0         4
25.0         4
27.0         3
32.0         2
31.0         2
23.0         2
26.0         2
28.0         1
30.0         1
34.0         1
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in pub_rec_bankruptcies column are :- 9
Unique Values in pub_rec_bankruptcies column are :-
 [ 0.  1.  2.  3. nan  4.  5.  6.  7.  8.]
Value_counts of pub_rec_bankruptcies column :-
 pub_rec_bankruptcies
0.0    350380
1.0     42790
2.0      1847
3.0       351
4.0        82
5.0        32
6.0         7
7.0         4
```

```
8.0         2
Name: count, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------

Total Unique Values in address column are :- 393700
Unique Values in address column are :-
 ['0174 Michelle Gateway\r\nMendozaberg, OK 22690'
 '1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113'
 '87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113' …
 '953 Matthew Points Suite 414\r\nReedfort, NY 70466'
 '7843 Blake Freeway Apt. 229\r\nNew Michael, FL 29597'
 '787 Michelle Causeway\r\nBriannaton, AR 48052']
Value_counts of address column :-
 address
USCGC Smith\r\nFPO AE 70466                           8
USS Johnson\r\nFPO AE 48052                           8
USNS Johnson\r\nFPO AE 05113                          8
USS Smith\r\nFPO AP 70466                             8
USNS Johnson\r\nFPO AP 48052                          7
                                                     ..
455 Tricia Cove\r\nAustinbury, FL 00813               1
7776 Flores Fall\r\nFernandezshire, UT 05113          1
6577 Mia Harbors Apt. 171\r\nRobertshire, OK 22690    1
8141 Cox Greens Suite 186\r\nMadisonstad, VT 05113    1
787 Michelle Causeway\r\nBriannaton, AR 48052         1
Name: count, Length: 393700, dtype: int64

--------------------------------------------------------------------------------
----------------------------------------
```

[80]:
```
df.loc[df['revol_util'].isna(),'revol_util'] = 0.0
df.loc[df['mort_acc'].isna(),'mort_acc'] = 0.0
df.loc[df['pub_rec_bankruptcies'].isna(),'pub_rec_bankruptcies'] = 0.0
df.loc[df['emp_title'].isna(),'emp_title'] = 'No Employee Title'
df.loc[df['title'].isna(),'title'] = 'Unavailable'
df['emp_length'] = df['emp_length'].fillna('< 1 year')
```

[81]:
```
df.isna().sum()
```

[81]:
```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
```

```
emp_title               0
emp_length              0
home_ownership          0
annual_inc              0
verification_status     0
issue_d                 0
loan_status             0
purpose                 0
title                   0
dti                     0
earliest_cr_line        0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util              0
total_acc               0
initial_list_status     0
application_type        0
mort_acc                0
pub_rec_bankruptcies    0
address                 0
dtype: int64
```

8 Feature Engineering

[82]:

```
df['pub_rec'] = [1 if i > 1 else 0 for i in df['pub_rec']]
df['mort_acc'] = [1 if i > 1 else 0 for i in df['mort_acc']]
df['pub_rec_bankruptcies'] = [1 if i > 1 else 0 for i in␣
 ↪df['pub_rec_bankruptcies']]
```

[83]:

```
df.sample()
```

[83]:

```
        loan_amnt        term  int_rate  installment grade sub_grade  \
373884    13000.0   60 months     14.65       306.89     C        C5

                emp_title emp_length home_ownership  annual_inc  \
373884  Medical assistant    8 years           RENT     33000.0

       verification_status   issue_d  loan_status             purpose  \
373884     Source Verified  Jun-2015  Charged Off  debt_consolidation

                     title   dti earliest_cr_line  open_acc  pub_rec  \
373884  Debt consolidation  26.0         Dec-2005      21.0        0

        revol_bal  revol_util  total_acc initial_list_status application_type  \
373884    14599.0        36.5       31.0                   w       INDIVIDUAL
```

```
        mort_acc  pub_rec_bankruptcies  \
373884         0                     0

                                                 address
373884  628 Leah Passage Suite 506\r\nNew Zoeberg, MA …
```

[84]:
```python
# issue_date into month and year
df[['issue_month', 'issue_year']] = df['issue_d'].str.split('-', expand=True)
df.drop(['issue_d'], axis=1, inplace=True)
```

[85]:
```python
# er_cr_line date into month and year
df[['er_cr_line_m', 'er_cr_line_y']] = df['earliest_cr_line'].str.split('-',␣
↪expand=True)
df.drop(['earliest_cr_line'], axis=1, inplace=True)
```

[86]:
```python
df['address']
```

[86]:
```
0             0174 Michelle Gateway\r\nMendozaberg, OK 22690
1          1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113
2          87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113
3                    823 Reid Ford\r\nDelacruzside, MA 00813
4                     679 Luna Roads\r\nGreggshire, VA 11650
                                …
396025     12951 Williams Crossing\r\nJohnnyville, DC 30723
396026     0114 Fowler Field Suite 028\r\nRachelborough, …
396027     953 Matthew Points Suite 414\r\nReedfort, NY 7…
396028     7843 Blake Freeway Apt. 229\r\nNew Michael, FL…
396029        787 Michelle Causeway\r\nBriannaton, AR 48052
Name: address, Length: 396030, dtype: object
```

[87]:
```python
# address into State and Zip code
import re
df[['state','zipcode']] = df['address'].str.extract(r'([A-Z]{2}) (\d{5})')
df.drop(['address'], axis=1, inplace=True)
```

[88]:
```python
df['emp_length_yrs'] = df['emp_length'].str.extract('(\d+)')
df.drop(['emp_length'], axis=1, inplace=True)
```

[89]:
```python
df['term'] = df['term'].str.split().str[0].astype('object')
```

[90]:
```python
df.sample()
```

[90]:
```
       loan_amnt term  int_rate  installment grade sub_grade  \
78624     7000.0   36      9.67       224.79     B        B1

                   emp_title home_ownership  annual_inc verification_status  \
```

```
78624  Operations Consultant          RENT     62000.0     Source Verified

      loan_status             purpose                title    dti  open_acc  \
78624  Fully Paid  debt_consolidation  Debt consolidation  11.91      21.0

       pub_rec  revol_bal  revol_util  total_acc initial_list_status  \
78624        0     5353.0         8.0       48.0                   f

      application_type  mort_acc  pub_rec_bankruptcies issue_month issue_year  \
78624       INDIVIDUAL         1                     0         Mar       2014

      er_cr_line_m er_cr_line_y state zipcode emp_length_yrs
78624          Jul         1990    NH   29597             10
```

[91]:
```python
df.shape
```

[91]: (396030, 30)

[92]:
```python
cat_cols = df.select_dtypes(include='object')

num_cols = df.select_dtypes(exclude='object')
```

[93]:
```python
cat_cols.sample(3)
```

[93]:
```
       term grade sub_grade                    emp_title home_ownership  \
359637   60     D        D4              Project Manager       MORTGAGE
82156    36     D        D1  Business Services Consultant           RENT
274165   36     E        E4   Core Business Distribution           RENT

       verification_status  loan_status             purpose  \
359637     Source Verified  Charged Off    home_improvement
82156      Source Verified   Fully Paid  debt_consolidation
274165        Not Verified   Fully Paid  debt_consolidation

                     title initial_list_status application_type issue_month  \
359637    Home improvement                   f       INDIVIDUAL         Apr
82156   Debt consolidation                   f       INDIVIDUAL         Mar
274165  Debt consolidation                   f       INDIVIDUAL         Jul

       issue_year er_cr_line_m er_cr_line_y state zipcode emp_length_yrs
359637       2014          Dec         2005    SC   86630              2
82156        2014          May         2001    AP   29597              1
274165       2014          Jan         1992    DE   05113              1
```

[94]:
```python
num_cols.skew()
```

[94]:
```
loan_amnt                0.777285
int_rate                 0.420669
installment              0.983598
annual_inc              41.042725
dti                    431.051225
open_acc                 1.213019
pub_rec                  6.812303
revol_bal               11.727515
revol_util              -0.074238
total_acc                0.864328
mort_acc                 0.412225
pub_rec_bankruptcies    12.936099
dtype: float64
```

8.1 Observation

Features are Right skewed

[95]:
```
df1 = df.copy()
```

9 What percentage of customers have fully paid their Loan Amount?

[96]:
```
df['loan_status'].value_counts(normalize=True)*100
```

[96]:
```
loan_status
Fully Paid     80.387092
Charged Off    19.612908
Name: proportion, dtype: float64
```

9.1 Observation

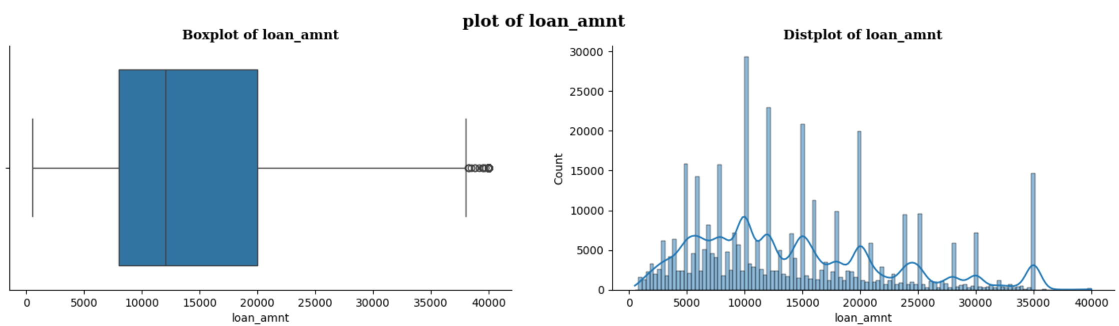
Data is significantly imbalanced

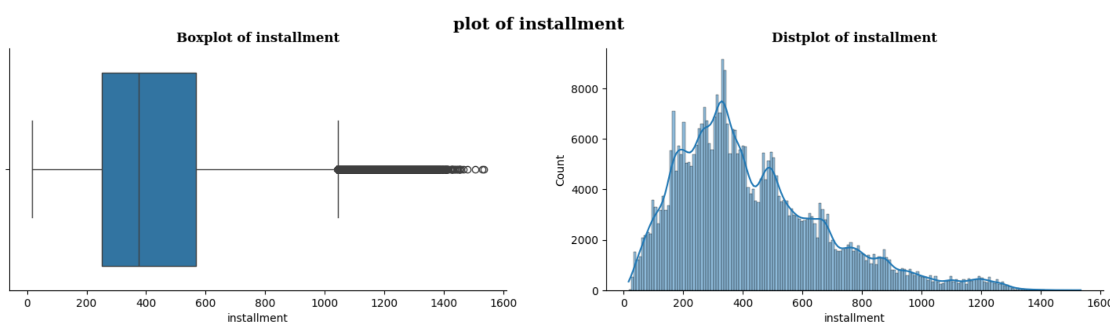
[97]:
```
cp =␣
↪['indigo','m','darkviolet','magenta','mediumorchid','violet','purple','orchid','mediumpurpl
```

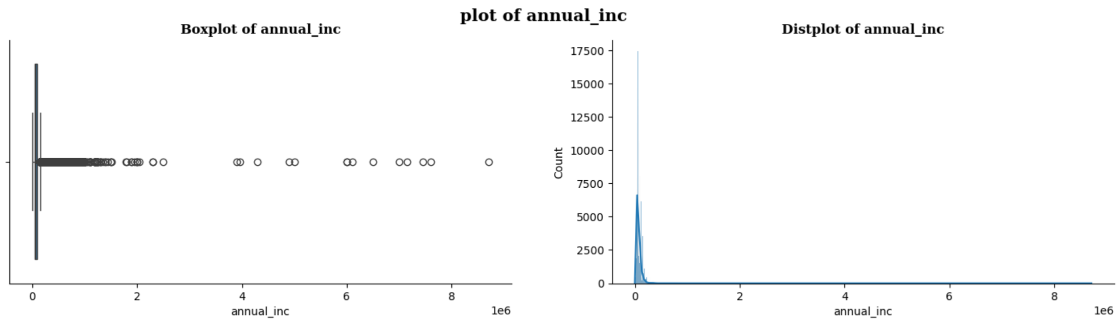
[98]:
```
num_cols.iloc[:,[0,2,3,4,5,6,8,9,10]].sample()
```

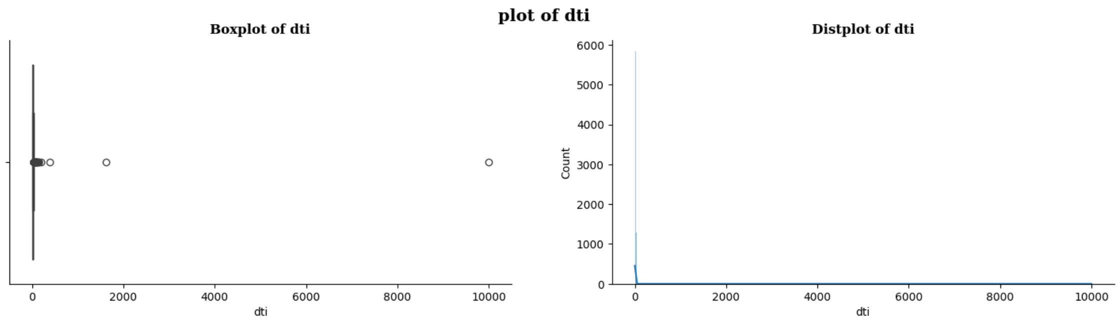
[98]:
```
        loan_amnt  installment  annual_inc    dti  open_acc  pub_rec  \
221181     8500.0       280.42     57000.0  19.58       9.0        0

        revol_util  total_acc  mort_acc
221181        60.9       17.0         0
```

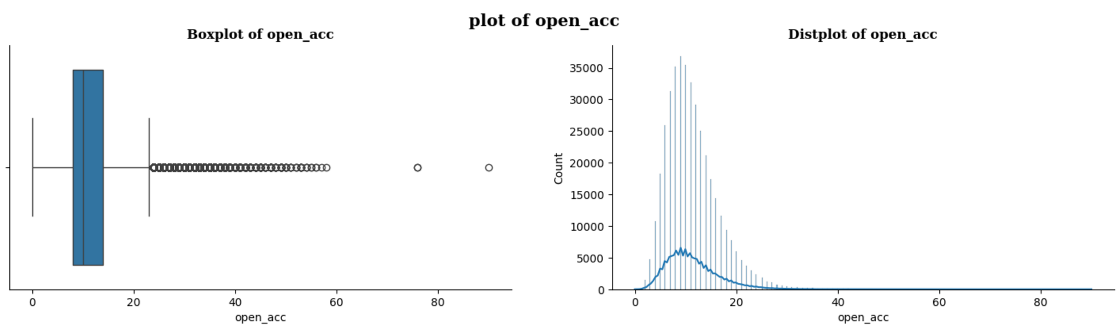
[99]:
```python
outlier_graphical_cols = num_cols.iloc[:,[0,2,3,4,5,6,8,9,10]]
for _,col in enumerate(outlier_graphical_cols.columns):
    plt.figure(figsize=(18,4))
    plt.suptitle(f'plot of {col}',fontsize=15,fontfamily='serif',fontweight='bold')
    plt.subplot(121)
    sns.boxplot(x=df[col])
    plt.title(f'Boxplot of {col}',fontsize=12,fontfamily='serif',fontweight='bold')
    plt.subplot(122)
    sns.histplot(x=df[col], kde=True)
    plt.title(f'Distplot of {col}',fontsize=12,fontfamily='serif',fontweight='bold')
    sns.despine()
    plt.show()
```

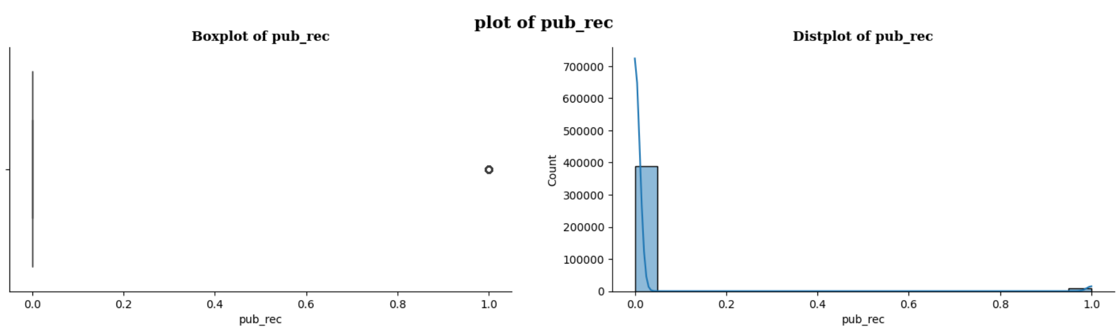


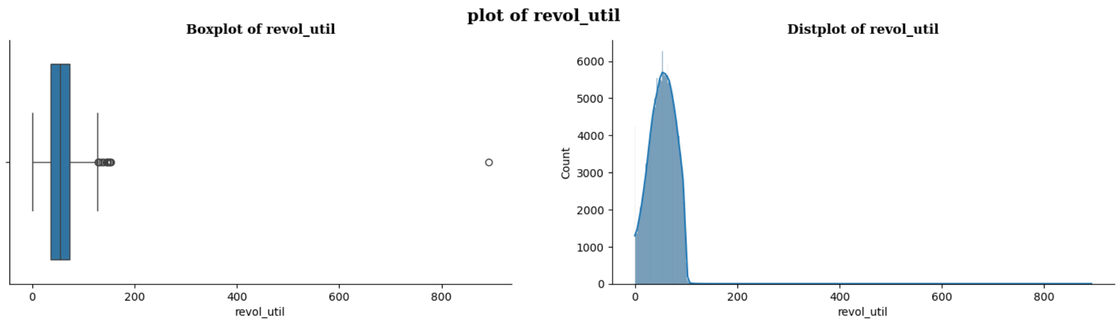


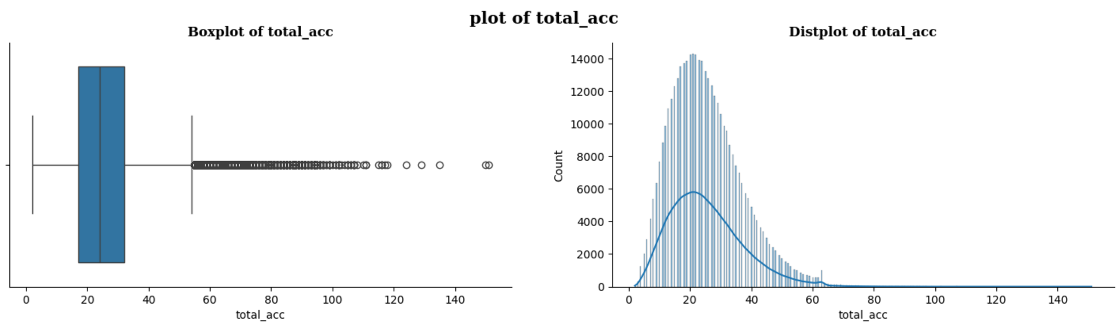

plot of annual_inc
Boxplot of annual_inc
Distplot of annual_inc
annual_inc
1e6
Count

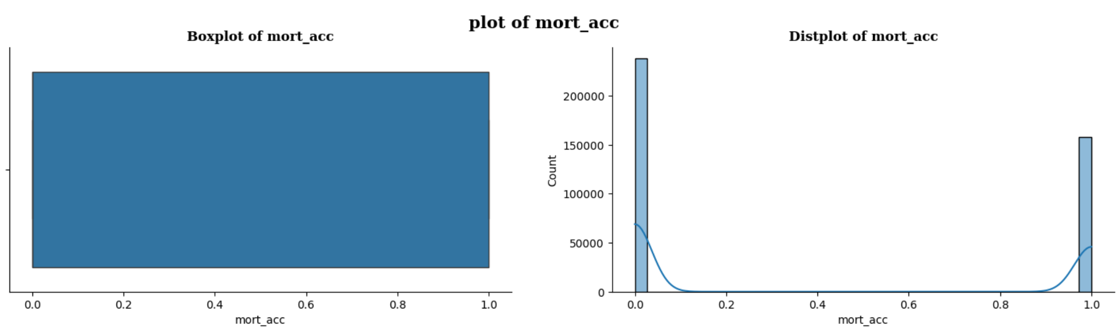

plot of dti
Boxplot of dti
Distplot of dti
dti
Count


plot of open_acc
Boxplot of open_acc
Distplot of open_acc
open_acc
Count


plot of pub_rec
Boxplot of pub_rec
Distplot of pub_rec
pub_rec
Count







9.2 Observation

The data points to a high frequency of outliers, which calls for more research into outlier detection methods.

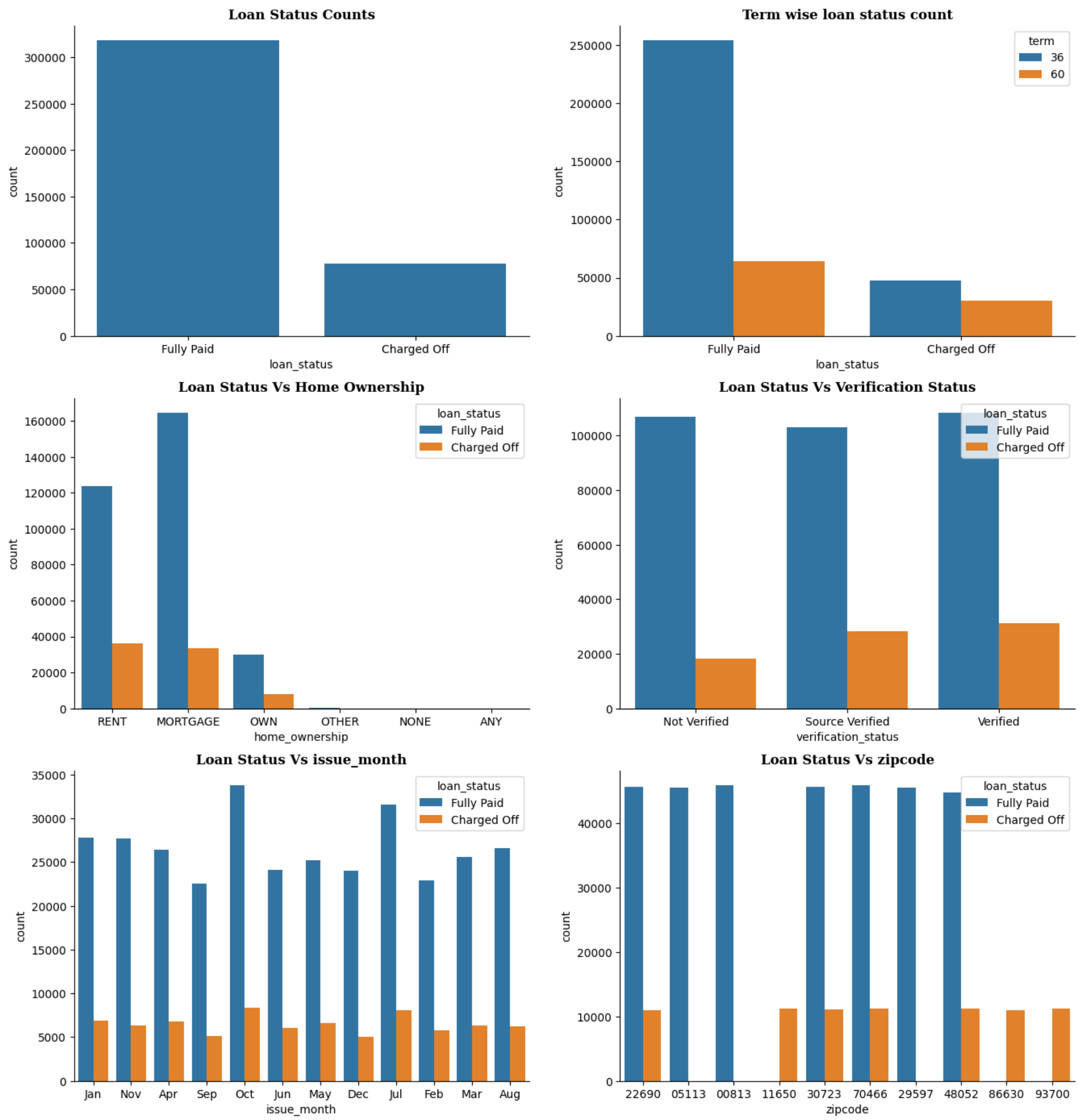
Potential outliers might still exist among the numerical features.

The potential advantage of creating binary features from these variables is demonstrated by the notable sparse distribution of unique values displayed by features like Pub_rec, Mort_acc, and Pub_rec_bankruptcies.

[100]:

```
plt.figure(figsize=(16,17))
plt.suptitle('Countplots categorical features w.r.t. to target variable␣
↪loan_status',
             fontsize=14,fontfamily='serif',fontweight='bold')
plt.subplot(321)
sns.countplot(data=df, x='loan_status')
plt.title('Loan Status Counts',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(322)
sns.countplot(data=df, x='loan_status', hue='term')
plt.title('Term wise loan status␣
↪count',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(323)
sns.countplot(data=df, x='home_ownership', hue='loan_status')
plt.title('Loan Status Vs Home␣
↪Ownership',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(324)
sns.countplot(data=df, x='verification_status', hue='loan_status')
plt.title('Loan Status Vs Verification␣
↪Status',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(325)
sns.countplot(data=df, x='issue_month', hue='loan_status')
plt.title('Loan Status Vs␣
↪issue_month',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(326)
sns.countplot(data=df, x='zipcode', hue='loan_status')
plt.title('Loan Status Vs␣
↪zipcode',fontsize=12,fontfamily='serif',fontweight='bold')
sns.despine()
plt.show()
```

Countplots categorical features w.r.t. to target variable loan_status



[101]:
```python
zip_codes = ["11650", "86630", "93700"]
states = df[df['zipcode'].isin(zip_codes)]['state']

for zip_code, state in zip(zip_codes, states):
    print(f"Zip code: {zip_code}, State: {state}")
```

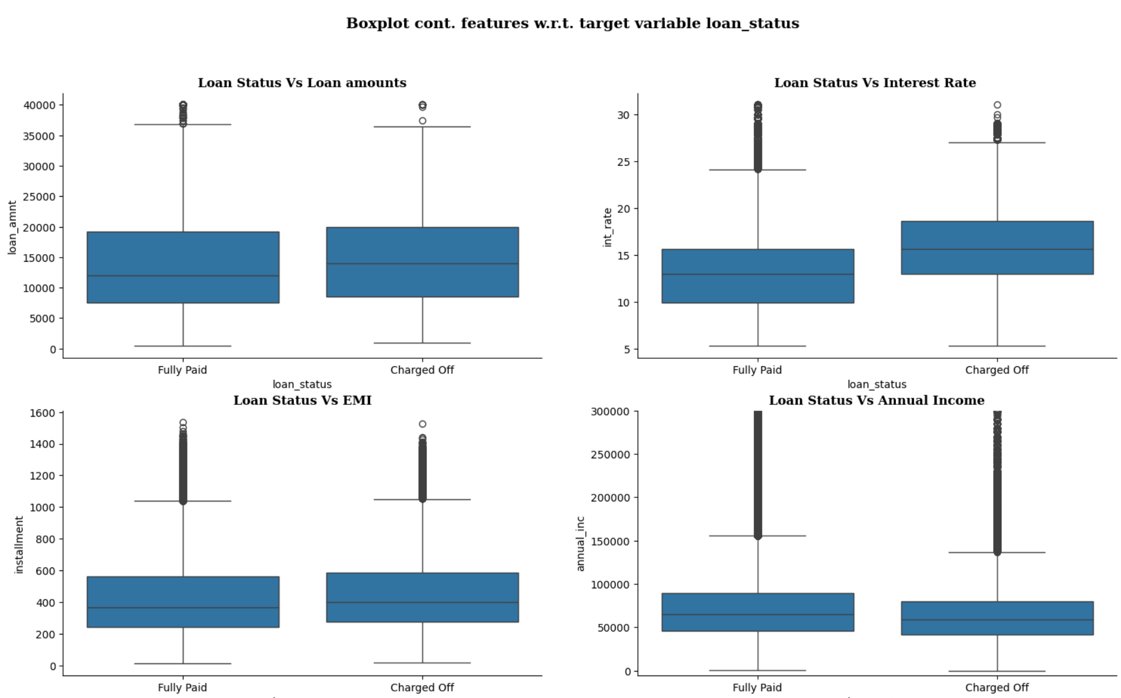
```
Zip code: 11650, State: VA
Zip code: 86630, State: MI
Zip code: 93700, State: MD
```

9.3 Observation

Loans in zip codes 11650, 86630, and 93700 have not been fully repaid, according to observations.

Borrowers who live in “VA,” “MI,” and “MD” have not paid back their loans.

[102]:
```
plt.figure(figsize=(18,10))
plt.suptitle('Boxplot cont. features w.r.t. target variable loan_status',
             fontsize=14,fontfamily='serif',fontweight='bold')
plt.subplot(221)
sns.boxplot(data=df, x='loan_status', y='loan_amnt')
plt.title('Loan Status Vs Loan␣
 ↪amounts',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(222)
sns.boxplot(data=df, x='loan_status', y='int_rate')
plt.title('Loan Status Vs Interest Rate␣
 ↪',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(223)
sns.boxplot(data=df, x='loan_status', y='installment')
plt.title('Loan Status Vs EMI',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(224)
sns.boxplot(data=df, x='loan_status', y='annual_inc')
plt.ylim(bottom=-5000, top=300000)
plt.title('Loan Status Vs Annual␣
 ↪Income',fontsize=12,fontfamily='serif',fontweight='bold')
sns.despine()
plt.show()
```



9.4 Observation

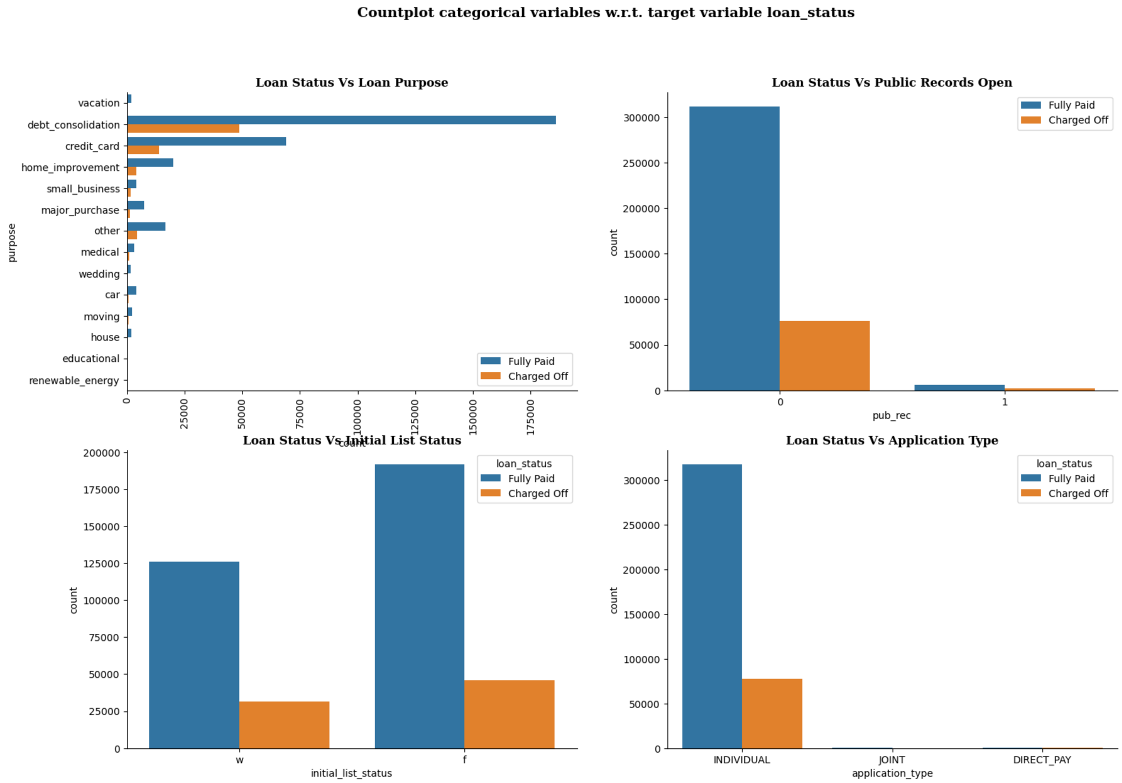
The median interest rate for Charged Off clients is significantly greater than that of Fully Paid customers.

Compared to fully paid clients, charged off customers have a lower median annual income.

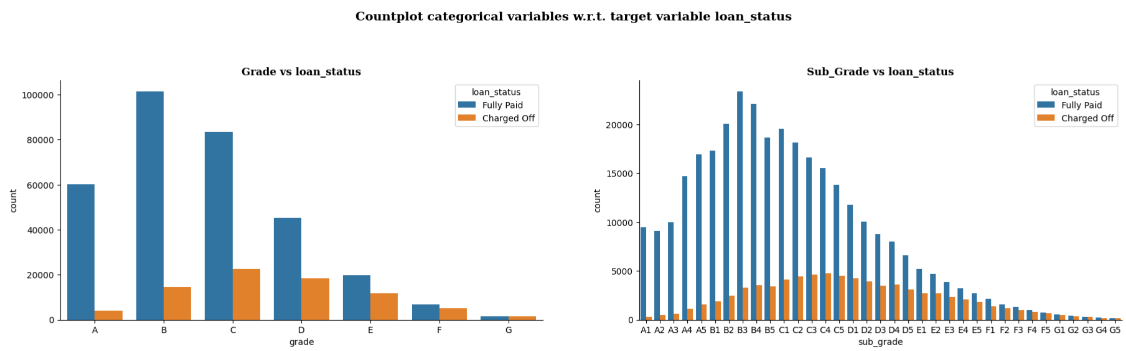
Customers who are charged off typically have a higher median EMI than those who are fully paid.

Charged Off clients' median loan amounts are higher than those of fully paid-off customers.

[103]:
```
plt.figure(figsize=(18,12))
plt.suptitle('Countplot categorical variables w.r.t. target variable␣
 ↪loan_status',
             fontsize=14,fontfamily='serif',fontweight='bold')
plt.subplot(221)
sns.countplot(data=df, y='purpose', hue='loan_status')
plt.xticks(rotation=90)
plt.title('Loan Status Vs Loan␣
 ↪Purpose',fontsize=12,fontfamily='serif',fontweight='bold')
plt.legend(loc=4)
plt.subplot(222)
sns.countplot(data=df, x='pub_rec',hue='loan_status')
plt.title('Loan Status Vs Public Records␣
 ↪Open',fontsize=12,fontfamily='serif',fontweight='bold')
plt.legend(loc=1)
plt.subplot(223)
sns.countplot(data=df, x='initial_list_status', hue='loan_status')
plt.title('Loan Status Vs Initial List␣
 ↪Status',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(224)
sns.countplot(data=df, x='application_type',hue='loan_status')
plt.title('Loan Status Vs Application␣
 ↪Type',fontsize=12,fontfamily='serif',fontweight='bold')
sns.despine()
plt.show()
```



[104]:
```
plt.figure(figsize=(22,11))
plt.suptitle('Countplot categorical variables w.r.t. target variable␣
↪loan_status',
             fontsize=14,fontfamily='serif',fontweight='bold')
plt.subplot(221)
grade = sorted(df.grade.unique().tolist())
sns.countplot(x='grade', data=df, hue='loan_status', order=grade)
plt.title('Grade vs␣
↪loan_status',fontsize=12,fontfamily='serif',fontweight='bold')
plt.subplot(222)
sub_grade = sorted(df.sub_grade.unique().tolist())
sns.countplot(x='sub_grade', data=df, hue='loan_status', order=sub_grade,)
plt.title('Sub_Grade vs␣
↪loan_status',fontsize=12,fontfamily='serif',fontweight='bold')
sns.despine()
plt.show()
```



9.5 Observation

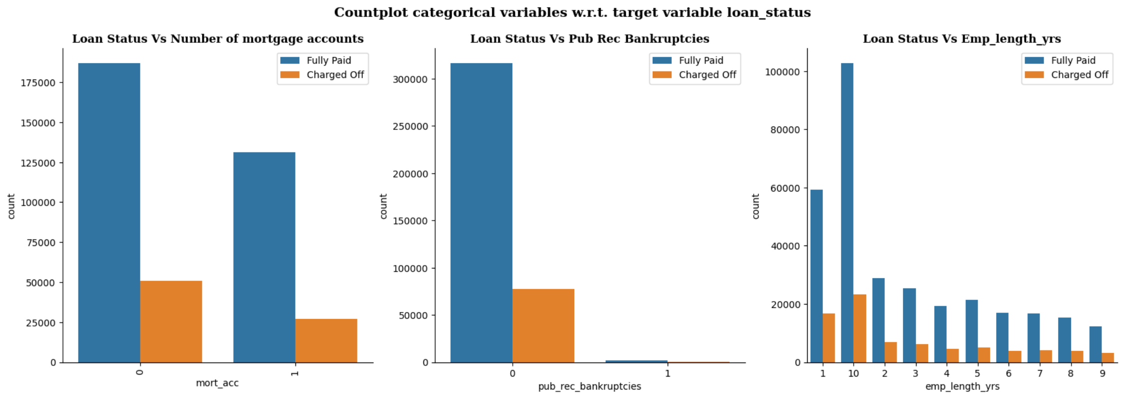
Credit card and debit consolidation are the two most common lending purposes.

Individual loan applications are the most common type.

When seen graphically, the distribution of open__acc seems to be very regular.

The distributions of the Charged Off and Fully Paid categories are comparable.

[105]:
```
plt.figure(figsize=(20,6))
plt.suptitle('Countplot categorical variables w.r.t. target variable loan_status',
             fontsize=14,fontfamily='serif',fontweight='bold')
plt.subplot(131)
sns.countplot(data=df, x='mort_acc',hue='loan_status')
plt.xticks(rotation=90)
plt.title('Loan Status Vs Number of mortgage accounts',fontsize=12,fontfamily='serif',fontweight='bold')
plt.legend(loc=1)
plt.subplot(132)
sns.countplot(data=df, x='pub_rec_bankruptcies',hue='loan_status')
plt.title('Loan Status Vs Pub Rec Bankruptcies',fontsize=12,fontfamily='serif',fontweight='bold')
plt.legend(loc=1)
plt.subplot(133)
order = sorted(df.emp_length_yrs.unique().tolist())
sns.countplot(data=df, x='emp_length_yrs',hue='loan_status',order=order,)
plt.title('Loan Status Vs Emp_length_yrs',fontsize=12,fontfamily='serif',fontweight='bold')
plt.legend(loc=1)
sns.despine()
plt.show()
```



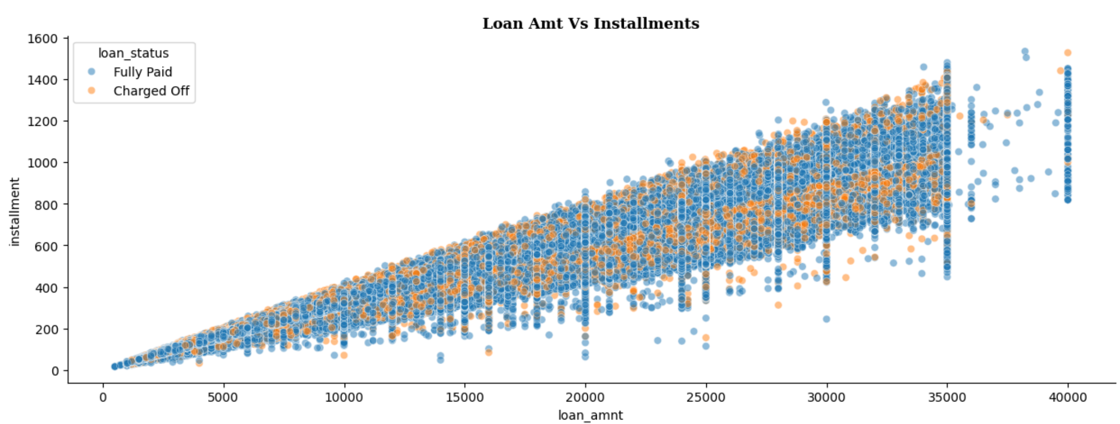
10 Comment about the correlation between Loan Amount and Installment features.

[106]:
```
df[['loan_amnt', 'installment']].corr()
```

[106]:

	loan_amnt	installment
loan_amnt	1.000000	0.953929
installment	0.953929	1.000000

[107]:
```
plt.figure(figsize = (15,5))
sns.scatterplot(data = df, x = 'loan_amnt', y = 'installment', alpha = 0.5, hue␣
 ↪= 'loan_status')
plt.title('Loan Amt Vs␣
 ↪Installments',fontsize=12,fontfamily='serif',fontweight='bold')
sns.despine()
plt.show()
```



10.1 Observation

The degree and direction of the linear link between two variables are measured by the correlation coefficient. In this instance, there is a strong positive linear link between "loan_amnt" and "installment," as evidenced by the high correlation coefficient between the two variables (around 0.95).

Establishing suitable loan terms requires an understanding of the connection between the loan amount and monthly payments. Depending on the borrower's capacity to make installment payments for varying loan amounts, lenders may modify loan parameters including interest rates and payback schedules.

Multicollinearity between strongly correlated predictor variables must be carefully considered when developing predictive models. Unstable estimates and trouble comprehending the model coefficients might result from multicollinearity. Consequently, multicollinearity may need to be addressed using strategies like variable selection or regularization.

11 The majority of people have home ownership as ________.

[108]:
```
(df['home_ownership'].value_counts(normalize=True)*100).to_frame()
```

[108]:

	proportion
home_ownership	
MORTGAGE	50.084085
RENT	40.347953
OWN	9.531096
OTHER	0.028281
NONE	0.007828
ANY	0.000758

11.1 Observation

Roughly 50.08%, are mortgage holders, suggesting that a sizable percentage of people own homes thanks to mortgage arrangements.

12 People with grades 'A' are more likely to fully pay their loan. (T/F)

[109]:
```
pd.crosstab(df['grade'],df['loan_status'], normalize = 'index')
```

[109]:

loan_status	Charged Off	Fully Paid
grade		
A	0.062879	0.937121
B	0.125730	0.874270
C	0.211809	0.788191
D	0.288678	0.711322
E	0.373634	0.626366

```
F                0.427880    0.572120
G                0.478389    0.521611
```

12.1 Observation

It's true. With over 93.71% of loans being fully repaid, borrowers with grade "A" credit have a remarkably high chance of doing so. This implies that borrowers who have the best credit scores are more likely to successfully complete their loan obligations.

13 Name the top 2 afforded job titles.

[110]:
```
df[df['emp_title'] != 'No Employee Title']['emp_title'].value_counts().
 ↪to_frame().head()
```

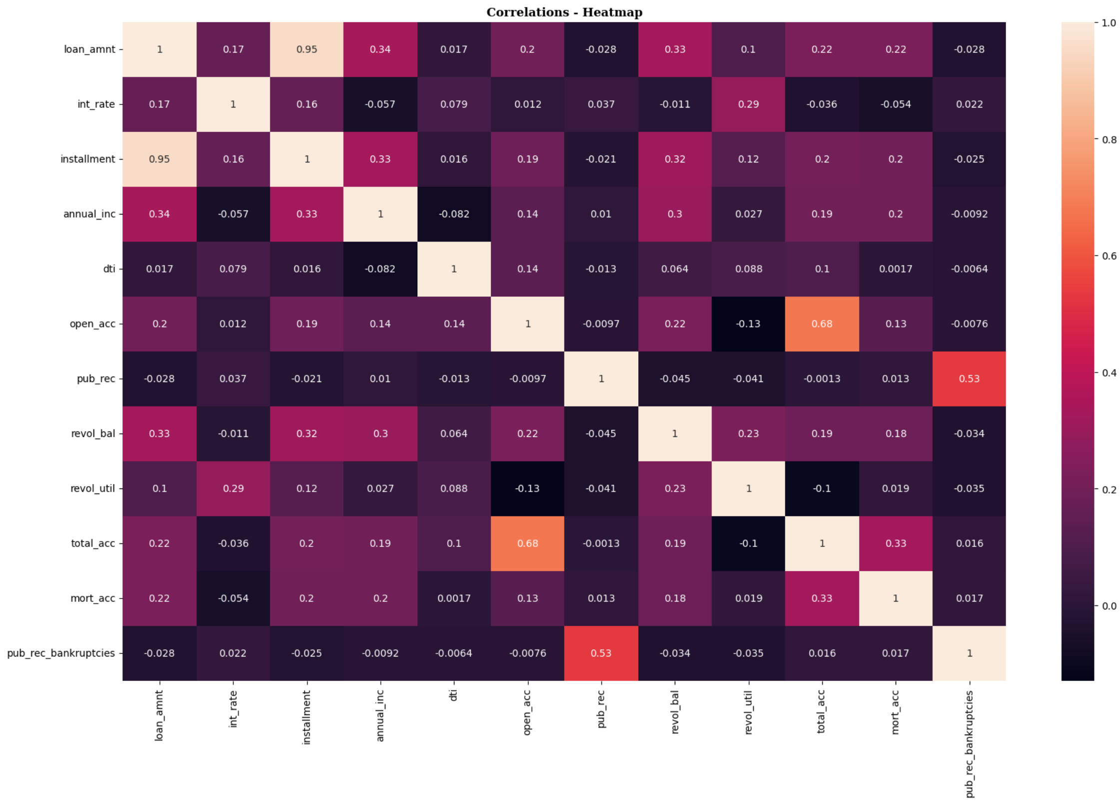
[110]:
```
                   count
emp_title
Teacher             4389
Manager             4250
Registered Nurse    1856
RN                  1846
Supervisor          1830
```

[111]:
```
df.groupby('emp_title')['loan_status'].count().sort_values(ascending=False).
 ↪to_frame()[1:6]
```

[111]:
```
                  loan_status
emp_title
Teacher                  4389
Manager                  4250
Registered Nurse         1856
RN                       1846
Supervisor               1830
```

The Most afforded job titles are Teachers & Managers.

[112]:
```
plt.figure(figsize=(20,12))
sns.heatmap(num_cols.corr(), annot=True)
plt.title('Correlations -␣
 ↪Heatmap',fontsize=12,fontfamily='serif',fontweight='bold')
plt.show()
```



13.1 Observation

Loan_amnt and installment have a strong association, meaning that greater loan amounts translate into larger installment payments.

There is a strong association between the variables total_acc and open_acc.

A significant relationship exists between pub_rec and pub_rec_bankruptcies.

14 Handling Outliers

[113]:
```
numerical_cols = df.select_dtypes(include=np.number).columns
numerical_cols
```

[113]:
```
Index(['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'mort_acc',
       'pub_rec_bankruptcies'],
      dtype='object')
```

[114]:
```
def remove_outliers_zscore(df, threshold=2): # approx 95% of data
    """
    Remove outliers from a DataFrame using the Z-score method.
```

```
    Parameters:
        df (DataFrame): The input DataFrame.
        threshold (float): The Z-score threshold for identifying outliers.
                           Observations with a Z-score greater than this␣
↪threshold will be considered as outliers.
    Returns:
        DataFrame: The DataFrame with outliers removed.
    """
    z_scores = (df[numerical_cols] - df[numerical_cols].mean()) /␣
↪df[numerical_cols].std()

    outliers = np.abs(z_scores) > threshold

    df_cleaned = df[~outliers.any(axis=1)]

    return df_cleaned

cleaned_df = remove_outliers_zscore(df1)
print(cleaned_df.shape)
```

```
(311392, 30)
```

[115]:
```
def clip_outliers_zscore(df, threshold=2):
    """
    Clip outliers in a DataFrame using the Z-score method.

    Parameters:
        df (DataFrame): The input DataFrame.
        threshold (float): The Z-score threshold for identifying outliers.
                           Observations with a Z-score greater than this␣
↪threshold will be considered as outliers.

    Returns:
        DataFrame: The DataFrame with outliers clipped.
    """
    z_scores = (df[numerical_cols] - df[numerical_cols].mean()) /␣
↪df[numerical_cols].std()

    clipped_values = df[numerical_cols].clip(df[numerical_cols].mean() -␣
↪threshold * df[numerical_cols].std(),
                                             df[numerical_cols].mean() +␣
↪threshold * df[numerical_cols].std(),
                                             axis=1)

    df_clipped = df.copy()
    df_clipped[numerical_cols] = clipped_values
```

```
    return df_clipped

clipped_df = clip_outliers_zscore(df1)
print(clipped_df.shape)
```

```
(396030, 30)
```

[116]:
```
data = cleaned_df.copy()
cp_data = clipped_df.copy()
data.sample()
```

[116]:
```
        loan_amnt term  int_rate  installment grade sub_grade  \
121377    16000.0   36     13.67       544.29     B        B5

                      emp_title home_ownership  annual_inc  \
121377  Alcatel-Lucent USA, Inc.           RENT     41500.0

       verification_status loan_status             purpose              title  \
121377            Verified  Fully Paid  debt_consolidation  Consolidation 001

         dti  open_acc  pub_rec  revol_bal  revol_util  total_acc  \
121377  19.2       7.0        0    14830.0        56.4        8.0

       initial_list_status application_type  mort_acc  pub_rec_bankruptcies  \
121377                   f       INDIVIDUAL         0                     0

       issue_month issue_year er_cr_line_m er_cr_line_y state zipcode  \
121377         Apr       2012          Jun         2003    ID   22690

       emp_length_yrs
121377              2
```

[117]:
```
data['pub_rec_bankruptcies'].value_counts() , data['pub_rec'].value_counts()
```

[117]:
```
(pub_rec_bankruptcies
 0    311392
 Name: count, dtype: int64,
 pub_rec
 0    311392
 Name: count, dtype: int64)
```

[118]:
```
cp_data['pub_rec_bankruptcies'].value_counts() , cp_data['pub_rec'].
 ↪value_counts()
```

[118]:
```
(pub_rec_bankruptcies
 0.000000    393705
```

```
0.158662      2325
Name: count, dtype: int64,
pub_rec
0.000000    388011
0.301947      8019
Name: count, dtype: int64)
```

[119]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 311392 entries, 0 to 396029
Data columns (total 30 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             311392 non-null  float64
 1   term                  311392 non-null  object
 2   int_rate              311392 non-null  float64
 3   installment           311392 non-null  float64
 4   grade                 311392 non-null  object
 5   sub_grade             311392 non-null  object
 6   emp_title             311392 non-null  object
 7   home_ownership        311392 non-null  object
 8   annual_inc            311392 non-null  float64
 9   verification_status   311392 non-null  object
 10  loan_status           311392 non-null  object
 11  purpose               311392 non-null  object
 12  title                 311392 non-null  object
 13  dti                   311392 non-null  float64
 14  open_acc              311392 non-null  float64
 15  pub_rec               311392 non-null  int64
 16  revol_bal             311392 non-null  float64
 17  revol_util            311392 non-null  float64
 18  total_acc             311392 non-null  float64
 19  initial_list_status   311392 non-null  object
 20  application_type      311392 non-null  object
 21  mort_acc              311392 non-null  int64
 22  pub_rec_bankruptcies  311392 non-null  int64
 23  issue_month           311392 non-null  object
 24  issue_year            311392 non-null  object
 25  er_cr_line_m          311392 non-null  object
 26  er_cr_line_y          311392 non-null  object
 27  state                 311392 non-null  object
 28  zipcode               311392 non-null  object
 29  emp_length_yrs        311392 non-null  object
dtypes: float64(9), int64(3), object(18)
memory usage: 73.6+ MB
```

[120]:
```
data['loan_status']=data.loan_status.map({'Fully Paid':1, 'Charged Off':0})

data['initial_list_status']=data.initial_list_status.map({'w':0, 'f':1})

data.head()
```

[120]:
```
   loan_amnt term  int_rate  installment grade sub_grade  \
0    10000.0   36     11.44       329.48     B        B4
1     8000.0   36     11.99       265.68     B        B5
2    15600.0   36     10.49       506.97     B        B3
3     7200.0   36      6.49       220.65     A        A2
4    24375.0   60     17.27       609.33     C        C5

                 emp_title home_ownership  annual_inc verification_status  \
0                Marketing           RENT    117000.0        Not Verified
1          Credit analyst        MORTGAGE     65000.0        Not Verified
2             Statistician           RENT     43057.0     Source Verified
3          Client Advocate           RENT     54000.0        Not Verified
4  Destiny Management Inc.       MORTGAGE     55000.0            Verified

   loan_status             purpose                    title    dti  open_acc  \
0            1            vacation                 Vacation  26.24      16.0
1            1  debt_consolidation       Debt consolidation  22.05      17.0
2            1         credit_card  Credit card refinancing  12.79      13.0
3            1         credit_card  Credit card refinancing   2.60       6.0
4            0         credit_card    Credit Card Refinance  33.95      13.0

   pub_rec  revol_bal  revol_util  total_acc  initial_list_status  \
0        0    36369.0        41.8       25.0                    0
1        0    20131.0        53.3       27.0                    1
2        0    11987.0        92.2       26.0                    1
3        0     5472.0        21.5       13.0                    1
4        0    24584.0        69.8       43.0                    1

  application_type  mort_acc  pub_rec_bankruptcies issue_month issue_year  \
0       INDIVIDUAL         0                     0         Jan       2015
1       INDIVIDUAL         1                     0         Jan       2015
2       INDIVIDUAL         0                     0         Jan       2015
3       INDIVIDUAL         0                     0         Nov       2014
4       INDIVIDUAL         0                     0         Apr       2013

  er_cr_line_m er_cr_line_y state zipcode emp_length_yrs
0          Jun         1990    OK   22690             10
1          Jul         2004    SD   05113              4
2          Aug         2007    WV   05113              1
3          Sep         2006    MA   00813              6
4          Mar         1999    VA   11650              9
```

15 Feature selection - Hypothesis testing & VIF(multicolinearity)

[121]:
```
lt = data.
↪drop(columns=['emp_title','title','sub_grade','er_cr_line_m','er_cr_line_y','initial_list_s
                                        ␣
↪'state','issue_month','issue_year','pub_rec','pub_rec_bankruptcies'],axis=1)
lt.shape
```

[121]: (311392, 19)

15.1 OneHotEncoding on feature having multiple variable

[122]:
```
dummies=['zipcode',␣
↪'grade','purpose','home_ownership','verification_status','application_type']
ltd = pd.get_dummies(lt, columns=dummies, drop_first=True)*1

ltd.shape
```

[122]: (311392, 50)

[123]:
```
X = ltd.drop(['loan_status'], axis=1)
y = ltd['loan_status']
```

[124]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
↪2,stratify=y,random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(249113, 49)
(62279, 49)
(249113,)
(62279,)
```

16 Minmax scaling

[126]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train = pd.DataFrame(X_train, columns=X.columns)
X_test = pd.DataFrame(X_test, columns=X.columns)

X_train.head()
```

[126]:
```
   loan_amnt  term  int_rate  installment  annual_inc       dti  open_acc  \
0   0.379538   0.0  0.339161     0.411590    0.207250  0.465341  0.368421
1   0.643564   1.0  0.680070     0.524221    0.367868  0.252652  0.473684
2   0.168317   0.0  0.208625     0.176198    0.134712  0.357576  0.368421
3   0.379538   1.0  0.680070     0.307444    0.367868  0.449242  0.315789
4   0.368812   0.0  0.543706     0.421460    0.246109  0.315530  0.263158

   revol_bal  revol_util  total_acc  mort_acc  emp_length_yrs  zipcode_05113  \
0   0.171897    0.419816   0.276596       0.0        0.111111            0.0
1   0.221905    0.590398   0.340426       0.0        1.000000            0.0
2   0.052236    0.304392   0.212766       0.0        0.000000            0.0
3   0.255109    0.767109   0.297872       1.0        1.000000            0.0
4   0.090649    0.614913   0.361702       0.0        0.000000            1.0

   zipcode_11650  zipcode_22690  zipcode_29597  zipcode_30723  zipcode_48052  \
0            0.0            0.0            0.0            0.0            0.0
1            0.0            0.0            1.0            0.0            0.0
2            1.0            0.0            0.0            0.0            0.0
3            0.0            0.0            0.0            1.0            0.0
4            0.0            0.0            0.0            0.0            0.0

   zipcode_70466  zipcode_86630  zipcode_93700  grade_B  grade_C  grade_D  \
0            0.0            0.0            0.0      1.0      0.0      0.0
1            0.0            0.0            0.0      0.0      0.0      1.0
2            0.0            0.0            0.0      0.0      0.0      0.0
3            0.0            0.0            0.0      0.0      0.0      1.0
4            0.0            0.0            0.0      0.0      1.0      0.0

   grade_E  grade_F  grade_G  purpose_credit_card  purpose_debt_consolidation  \
0      0.0      0.0      0.0                  0.0                         1.0
1      0.0      0.0      0.0                  0.0                         1.0
2      0.0      0.0      0.0                  1.0                         0.0
3      0.0      0.0      0.0                  0.0                         0.0
4      0.0      0.0      0.0                  0.0                         1.0

   purpose_educational  purpose_home_improvement  purpose_house  \
0                  0.0                       0.0            0.0
1                  0.0                       0.0            0.0
2                  0.0                       0.0            0.0
3                  0.0                       0.0            0.0
4                  0.0                       0.0            0.0

   purpose_major_purchase  purpose_medical  purpose_moving  purpose_other  \
0                     0.0              0.0             0.0            0.0
1                     0.0              0.0             0.0            0.0
2                     0.0              0.0             0.0            0.0
3                     0.0              0.0             0.0            0.0
```

```
4                       0.0                0.0               0.0               0.0

   purpose_renewable_energy  purpose_small_business  purpose_vacation  \
0                       0.0                     0.0               0.0
1                       0.0                     0.0               0.0
2                       0.0                     0.0               0.0
3                       0.0                     1.0               0.0
4                       0.0                     0.0               0.0

   purpose_wedding  home_ownership_MORTGAGE  home_ownership_NONE  \
0              0.0                      0.0                  0.0
1              0.0                      1.0                  0.0
2              0.0                      0.0                  0.0
3              0.0                      0.0                  0.0
4              0.0                      0.0                  0.0

   home_ownership_OTHER  home_ownership_OWN  home_ownership_RENT  \
0                   0.0                 0.0                  1.0
1                   0.0                 0.0                  0.0
2                   0.0                 0.0                  1.0
3                   0.0                 0.0                  1.0
4                   0.0                 0.0                  1.0

   verification_status_Source Verified  verification_status_Verified  \
0                                  0.0                           0.0
1                                  1.0                           0.0
2                                  0.0                           0.0
3                                  0.0                           0.0
4                                  0.0                           0.0

   application_type_INDIVIDUAL  application_type_JOINT
0                          1.0                     0.0
1                          1.0                     0.0
2                          1.0                     0.0
3                          1.0                     0.0
4                          1.0                     0.0
```

[127]:
```
logreg_model = LogisticRegression()
logreg_model.fit(X_train, y_train)
```

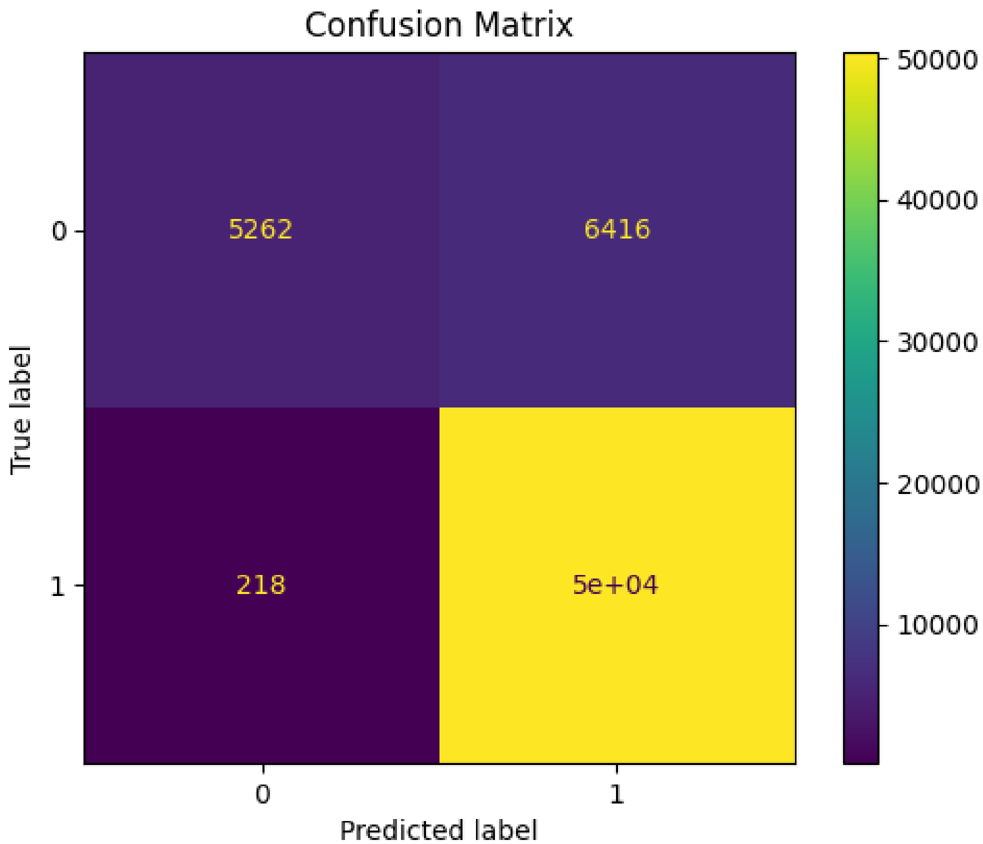
[127]: LogisticRegression()

[129]:
```
y_train_pred = logreg_model.predict(X_train)
y_test_pred = logreg_model.predict(X_test)

logreg_model.score(X_test, y_test) , logreg_model.score(X_test, y_test_pred)
```

[129]: (0.8934793429566948, 1.0)

[131]:
```
print('Train Accuracy :', logreg_model.score(X_train, y_train))
print('Train F1 Score:',f1_score(y_train,y_train_pred))
print('Train Recall Score:',recall_score(y_train,y_train_pred))
print('Train Precision Score:',precision_score(y_train,y_train_pred))

print('\nTest Accuracy :',logreg_model.score(X_test,y_test))
print('Test F1 Score:',f1_score(y_test,y_test_pred))
print('Test Recall Score:',recall_score(y_test,y_test_pred))
print('Test Precision Score:',precision_score(y_test,y_test_pred))

# Confusion Matrix
cm = confusion_matrix(y_test, y_test_pred)
disp = ConfusionMatrixDisplay(cm)
disp.plot()
plt.title('Confusion Matrix')
plt.show()
```

```
Train Accuracy : 0.8934339034895811
Train F1 Score: 0.9382212696440165
Train Recall Score: 0.995953577304 46
Train Precision Score: 0.886815362280586

Test Accuracy : 0.8934793429566948
Test F1 Score: 0.9382309124767225
Test Recall Score: 0.995691784747337
Test Precision Score: 0.8870402647933943
```



[132]:
```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.45      0.61     11678
           1       0.89      1.00      0.94     50601

    accuracy                           0.89     62279
   macro avg       0.92      0.72      0.78     62279
weighted avg       0.90      0.89      0.88     62279
```

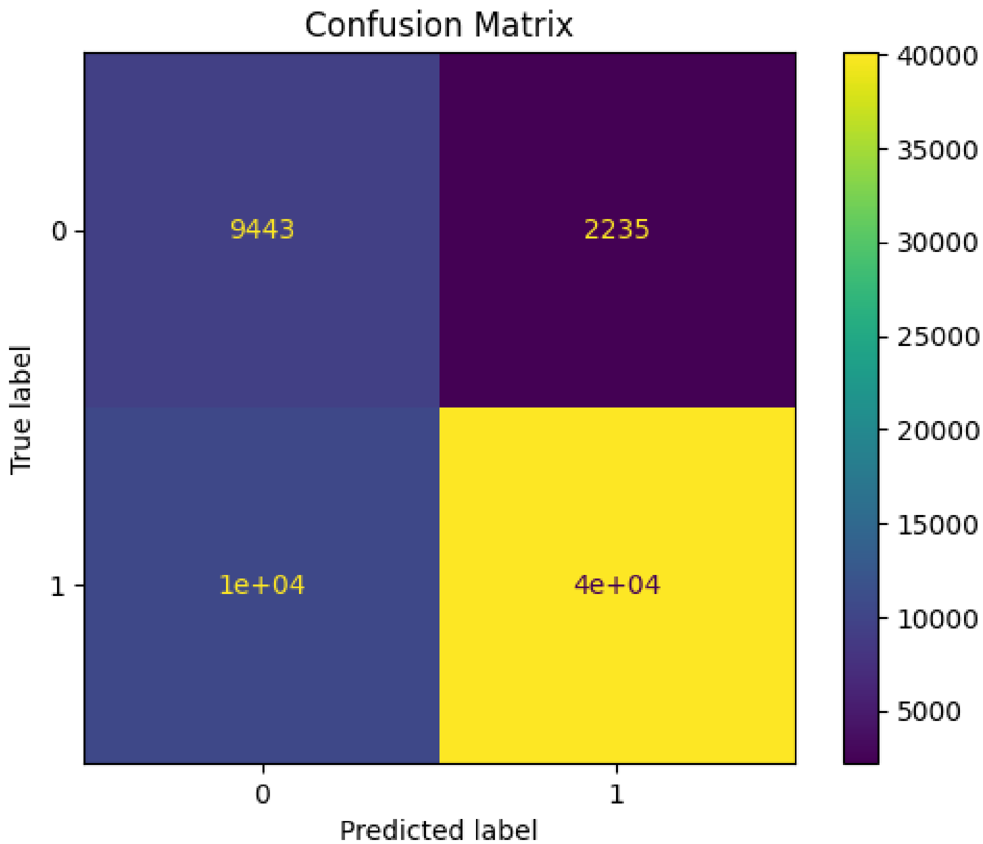
[133]:
```
sm=SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train,y_train.ravel())

print(f"Before OverSampling, count of label 1: {sum(y_train == 1)}")
print(f"Before OverSampling, count of label 0: {sum(y_train == 0)}")
print(f"After OverSampling, count of label 1: {sum(y_train_res == 1)}")
print(f"After OverSampling, count of label 0: {sum(y_train_res == 0)}")
```

```
Before OverSampling, count of label 1: 202401
Before OverSampling, count of label 0: 46712
After OverSampling, count of label 1: 202401
After OverSampling, count of label 0: 202401
```

[135]:

```python
model = LogisticRegression()
model.fit(X_train_res, y_train_res)
train_preds = model.predict(X_train)
test_preds = model.predict(X_test)

print('Train Accuracy :', model.score(X_train, y_train))
print('Train F1 Score:',f1_score(y_train,train_preds))
print('Train Recall Score:',recall_score(y_train,train_preds))
print('Train Precision Score:',precision_score(y_train,train_preds))

print('\nTest Accuracy :',model.score(X_test,y_test))
print('Test F1 Score:',f1_score(y_test,test_preds))
print('Test Recall Score:',recall_score(y_test,test_preds))
print('Test Precision Score:',precision_score(y_test,test_preds))

# Confusion Matrix
cm = confusion_matrix(y_test, test_preds)
disp = ConfusionMatrixDisplay(cm)
disp.plot()
plt.title('Confusion Matrix')
plt.show()
```

```
Train Accuracy : 0.7935033498853934
Train F1 Score: 0.8614290524614452
Train Recall Score: 0.7899763341090212
Train Precision Score: 0.9470928304032602

Test Accuracy : 0.7955330047046356
Test F1 Score: 0.862983924767049
Test Recall Score: 0.7925139819371159
Test Precision Score: 0.9472092968325578
```



[136]:
```
y_pred = test_preds
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.47      0.81      0.60     11678
           1       0.95      0.79      0.86     50601

    accuracy                           0.80     62279
   macro avg       0.71      0.80      0.73     62279
weighted avg       0.86      0.80      0.81     62279
```

16.1 Observation

By correctly identifying 80% of real defaulters, the model exhibits a high recall score.

Only 47% of anticipated defaulters actually become defaulters, indicating a low precision for the positive class (defaulters).

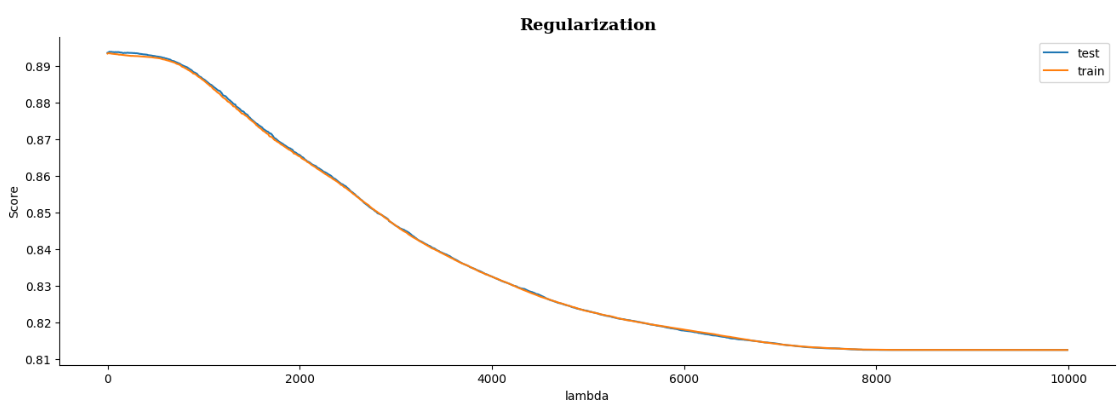
Although the model is successful in identifying the majority of defaulters, it also produces a large number of false positives, as evidenced by its high recall and low precision. As a result, many worthy clients might not be granted loans.

Despite an overall accuracy of 80%, the low precision negatively impacts the F1 score, lowering it to 60%. This demonstrates how the model's performance involves a trade-off between recall and precision.

17 Reguralization Model

[138]:
```
lamb = np.arange(0.01, 10000, 10)

train_scores = []
test_scores = []

for lam in lamb:
    model = LogisticRegression(C = 1/lam)
    model.fit(X_train, y_train)

    tr_score = model.score(X_train, y_train)
    te_score = model.score(X_test, y_test)

    train_scores.append(tr_score)
    test_scores.append(te_score)
```

[140]:
```
ran = np.arange(0.01, 10000, 10)
plt.figure(figsize=(16,5))
sns.lineplot(x=ran,y=test_scores,label='test')
sns.lineplot(x=ran,y=train_scores,label='train')
plt.title('Regularization',fontsize=14,fontfamily='serif',fontweight='bold')
plt.xlabel("lambda")
plt.ylabel("Score")
sns.despine()
plt.show()
```



[141]:
```
best_lamb = 0.01 + (10*2)
best_lamb
```

[141]: 20.01

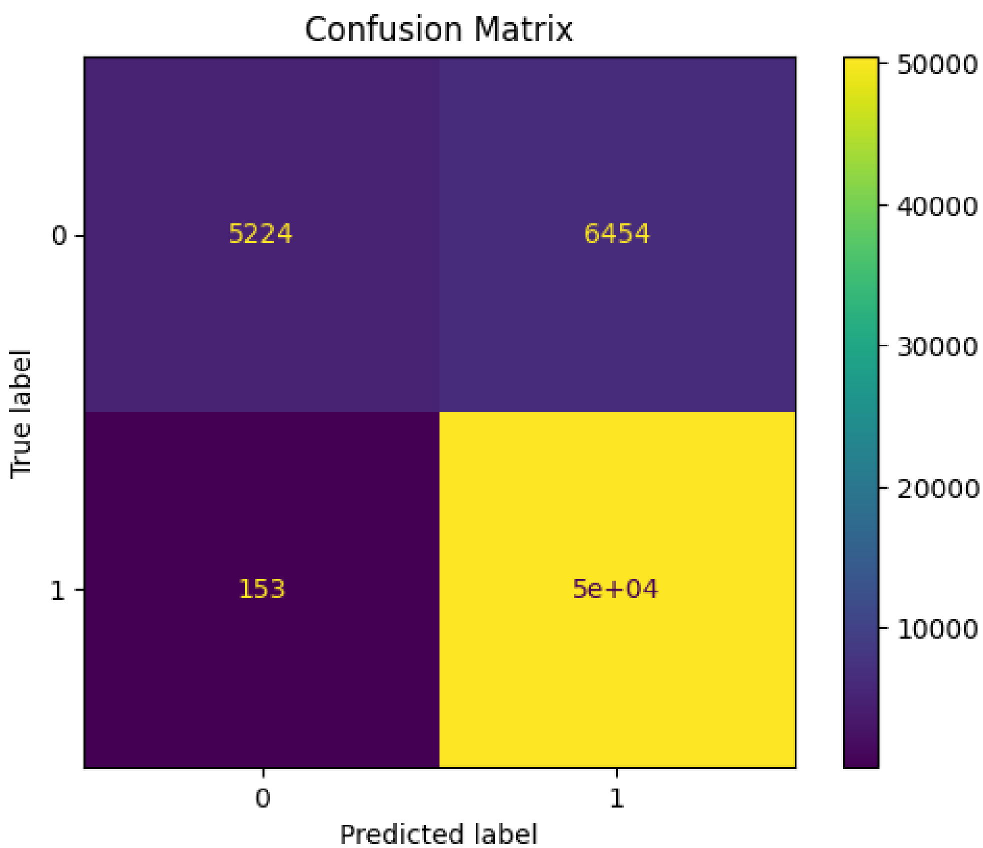
[142]:
```
reg_model = LogisticRegression(C=1/best_lamb)
reg_model.fit(X_train, y_train)
```

[142]: LogisticRegression(C=0.04997501249375312)

[144]:
```
y_reg_pred = reg_model.predict(X_test)
y_reg_pred_proba = reg_model.predict_proba(X_test)

print(f'Logistic Regression Model Score with best lambda: ',end='')
print(round(model.score(X_test, y_test)*100,2),'%')
```

```
Logistic Regression Model Score with best lambda: 81.25 %
```

[145]:
```
cm = confusion_matrix(y_test, y_reg_pred)
disp = ConfusionMatrixDisplay(cm)
disp.plot()
plt.title('Confusion Matrix')
plt.show()
```



[146]:
```
print(classification_report(y_test, y_reg_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.45      0.61     11678
           1       0.89      1.00      0.94     50601

    accuracy                           0.89     62279
   macro avg       0.93      0.72      0.78     62279
weighted avg       0.90      0.89      0.88     62279
```

18 K-fold - Cross validation

[147]:
```
x=scaler.fit_transform(X)

kfold = KFold(n_splits=10)
accuracy = np.mean(cross_val_score(reg_model,x,y,cv=kfold,scoring='accuracy'))
print("Cross Validation accuracy : {:.3f}".format(accuracy))
```

```
Cross Validation accuracy : 0.894
```

[148]:
```
cm = confusion_matrix(y_test, y_reg_pred)
cm_df = pd.DataFrame(cm, index=['Defaulter','Fully paid'],␣
↪columns=['Defaulter','Fully paid'])
cm_df
```

[148]:

	Defaulter	Fully paid
Defaulter	5224	6454
Fully paid	153	50448

18.1 Observation

TN = 5223 (True Negative: Charged Off was accurately predicted)

TP is 50450. (True Positive: Fully Paid and accurately forecasted)

FP is 6455. (False Positive: Charged off even if fully paid was predicted.)

FN = 151 (False Negative: Charged Off was predicted but paid in full)

Actual Charged Off Negative = 5223 + 6455 = 11678

151 + 50450 = 50601 is the actual positive (fully paid) amount.

5223 + 151 = 5374 is the predicted negative (charged off).

Fully Paid Predicted Positive = 6455 + 50450 = 56905

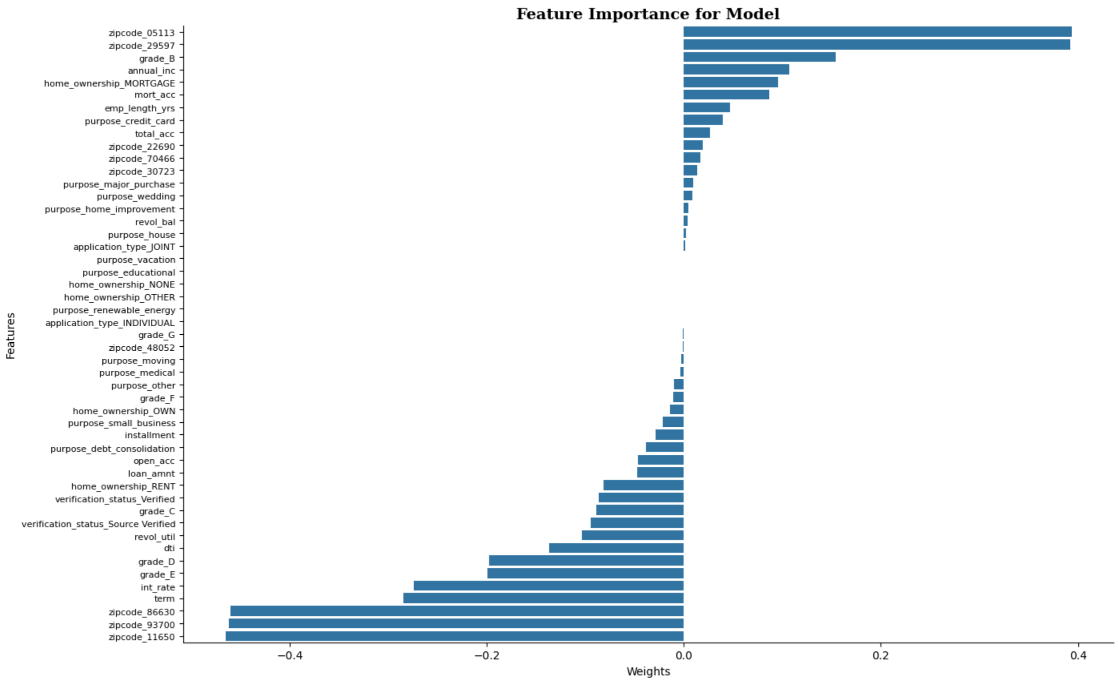
[149]:
```
coeff_df = pd.DataFrame()
coeff_df['Features'] = X_train_res.columns
coeff_df['Weights'] = model.coef_[0]
coeff_df['ABS_Weights'] = abs(coeff_df['Weights'])
coeff_df = coeff_df.sort_values(['ABS_Weights'], ascending=False)
coeff_df
```

[149]:

	Features	Weights	ABS_Weights
13	zipcode_11650	-0.465074	0.465074
20	zipcode_93700	-0.461597	0.461597
19	zipcode_86630	-0.460198	0.460198
12	zipcode_05113	0.393420	0.393420
15	zipcode_29597	0.392065	0.392065
1	term	-0.284876	0.284876
2	int_rate	-0.274231	0.274231
24	grade_E	-0.199183	0.199183
23	grade_D	-0.197492	0.197492
21	grade_B	0.153863	0.153863
5	dti	-0.136682	0.136682
4	annual_inc	0.107301	0.107301
8	revol_util	-0.103188	0.103188
40	home_ownership_MORTGAGE	0.095547	0.095547

```
45  verification_status_Source Verified -0.094784     0.094784
22                              grade_C -0.088887     0.088887
10                             mort_acc  0.086515     0.086515
46         verification_status_Verified -0.086047     0.086047
44                  home_ownership_RENT -0.081123     0.081123
0                             loan_amnt -0.047541     0.047541
11                        emp_length_yrs  0.047248     0.047248
6                              open_acc -0.046230     0.046230
27                  purpose_credit_card  0.039204     0.039204
28           purpose_debt_consolidation -0.038272     0.038272
3                           installment -0.028448     0.028448
9                             total_acc  0.026258     0.026258
37               purpose_small_business -0.020956     0.020956
14                        zipcode_22690  0.019340     0.019340
18                        zipcode_70466  0.016947     0.016947
43                   home_ownership_OWN -0.014123     0.014123
16                        zipcode_30723  0.013786     0.013786
25                              grade_F -0.010375     0.010375
35                        purpose_other -0.009954     0.009954
32               purpose_major_purchase  0.009729     0.009729
39                      purpose_wedding  0.008596     0.008596
30             purpose_home_improvement  0.004803     0.004803
7                             revol_bal  0.003796     0.003796
33                      purpose_medical -0.003333     0.003333
34                       purpose_moving -0.002790     0.002790
31                        purpose_house  0.002104     0.002104
17                        zipcode_48052 -0.001237     0.001237
48               application_type_JOINT  0.001189     0.001189
26                              grade_G -0.000720     0.000720
47          application_type_INDIVIDUAL -0.000514     0.000514
36             purpose_renewable_energy -0.000335     0.000335
42                 home_ownership_OTHER -0.000185     0.000185
41                  home_ownership_NONE -0.000161     0.000161
29                  purpose_educational -0.000156     0.000156
38                     purpose_vacation -0.000053     0.000053
```

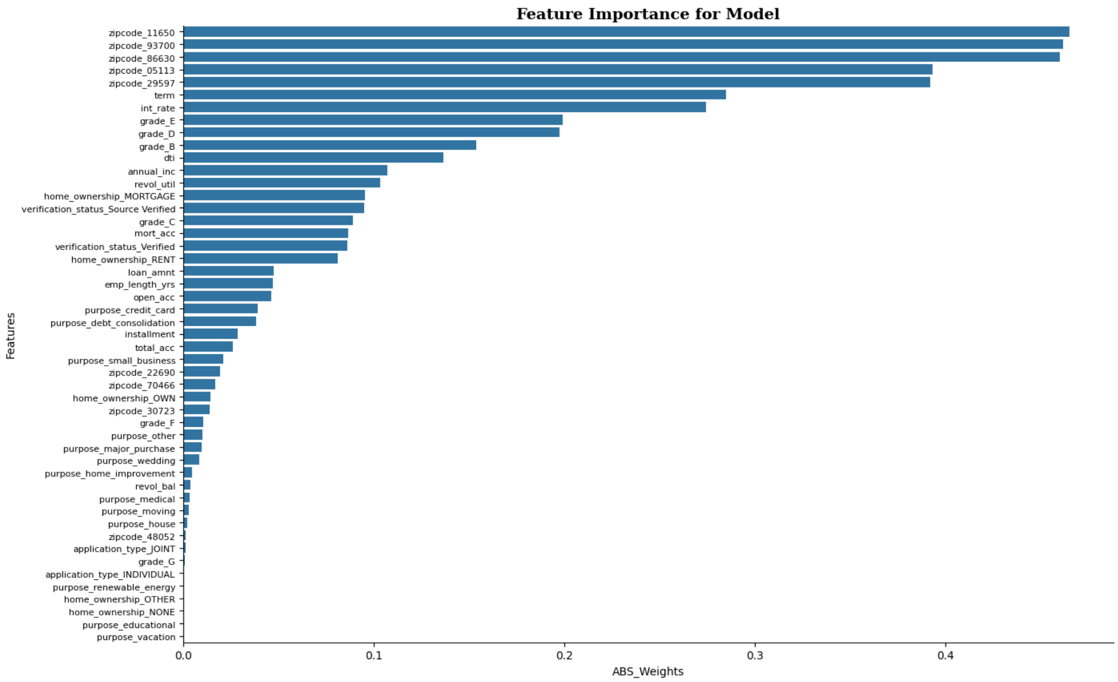
[150]:
```python
imp_feature = coeff_df.sort_values(by='Weights',ascending=False)
plt.figure(figsize=(15,10))
sns.barplot(y = imp_feature['Features'],
           x = imp_feature['Weights'])
plt.title("Feature Importance for Model",fontsize=14,fontfamily='serif',fontweight='bold')
plt.xlabel("Weights")
plt.yticks(fontsize=8)
plt.ylabel("Features")
sns.despine()
plt.show()
```



[151]:
```
model.intercept_
```

[151]: array([1.76790228])

[152]:
```
plt.figure(figsize=(15,10))
sns.barplot(y = coeff_df['Features'],x = coeff_df['ABS_Weights'])
plt.title("Feature Importance for␣
  ↪Model",fontsize=14,fontfamily='serif',fontweight='bold')
plt.xlabel("ABS_Weights")
plt.yticks(fontsize=8)
plt.ylabel("Features")
sns.despine()
plt.show()
```



18.2 Observation

Certain zip codes have a large influence on the prediction of defaulters, as evidenced by the model's significant weighting of the zip_code, annual income, and grade characteristics.

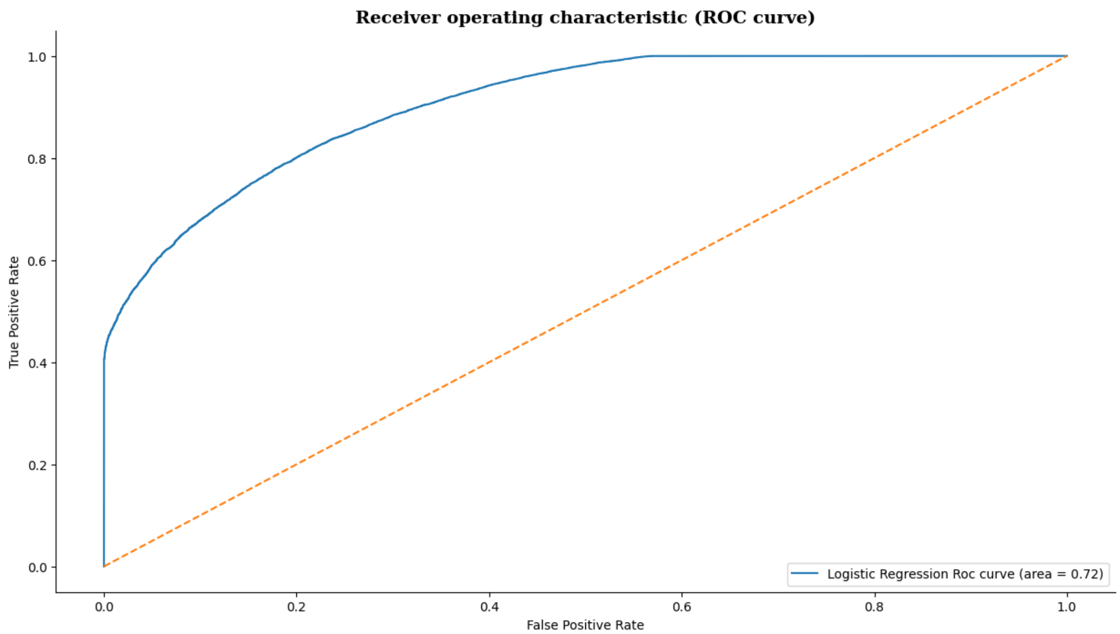
High positive coefficients also indicate the significance of features like loan_amnt (loan amount), open_acc (number of open accounts), and dti (debt-to-income ratio) in forecasting default risk.

However, a number of zip codes show significant negative coefficients, indicating that they are linked to a decreased default risk.

19 ROC AUC curve

[155]:
```
logit_roc_auc = roc_auc_score(y_test,y_reg_pred)

fpr,tpr,thresholds = roc_curve(y_test,y_reg_pred_proba[:,1])

roc_auc = auc(fpr, tpr)

# plot ROC curve
plt.figure(figsize=(15,8))
plt.plot(fpr,tpr,label='Logistic Regression Roc curve (area = %0.2f)'%
 ↪logit_roc_auc)
plt.plot([0,1],[0,1],'--')
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC␣
↪curve)',fontsize=14,fontfamily='serif',fontweight='bold')
plt.legend(loc="lower right")
sns.despine()
plt.show()
```



[156]:
```
logit_roc_auc
```

[156]: 0.7221566085466022

[157]:
```
roc_auc = auc(fpr, tpr)
roc_auc
```

[157]: 0.9036968327803755

19.1 Observation

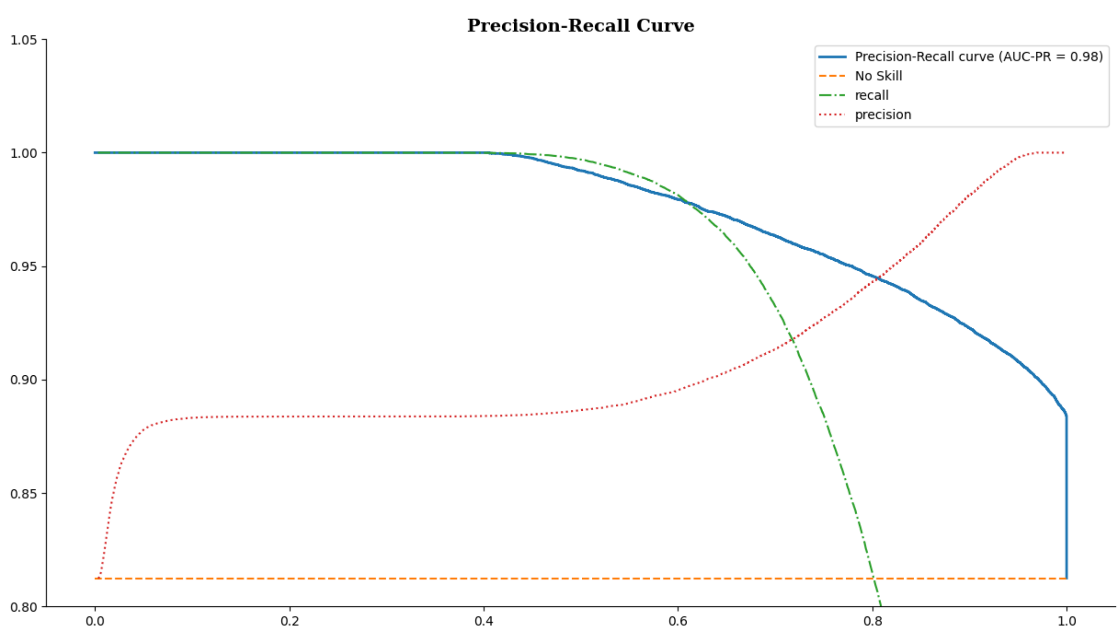
Model performance is represented by the ROC curve area, which is 72%. This shows that 72% of the time, the model successfully differentiates between classes.

To guarantee accurate forecasts, we should ideally strive for a greater True Positive Rate (TPR) and a lower False Positive Rate (FPR).

The ROC curve shows that False Positives rise in tandem with an increase in True Positives.

This trade-off suggests that there is a higher chance of incorrectly classifying Charged Off customers as Fully Paid, which could result in Non-Performing Assets (NPAs), even while there is a greater chance of finding more Fully Paid customers.

[159]:
```
precision, recall, thresholds = precision_recall_curve(y_test,␣
 ↪y_reg_pred_proba[:,1])

average_precision = average_precision_score(y_test, y_reg_pred_proba[:,1])

no_skill = len(y_test[y_test==1]) / len(y_test)

plt.figure(figsize=(15,8))
plt.plot(recall, precision, lw=2, label=f'Precision-Recall curve (AUC-PR =␣
 ↪{average_precision:.2f})')
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
plt.plot(thresholds, recall[0:thresholds.shape[0]], label='recall',linestyle='-.
 ↪')
plt.plot(thresholds, precision[0:thresholds.shape[0]],␣
 ↪label='precision',linestyle='dotted')
# plt.xlim([0.0, 1.0])
plt.ylim([0.8, 1.05])
plt.title('Precision-Recall␣
 ↪Curve',fontsize=14,fontfamily='serif',fontweight='bold')
plt.legend(loc='upper right')
sns.despine()
plt.show()
```



[161]:
```
auc(recall, precision)
```

[161]: 0.9750571212298236

19.2 Observation

Model performance is represented by the ROC curve area, which is 72%. This shows that 72% of the time, the model successfully differentiates between classes.

To guarantee accurate forecasts, we should ideally strive for a greater True Positive Rate (TPR) and a lower False Positive Rate (FPR).

The ROC curve shows that False Positives rise in tandem with an increase in True Positives.

This trade-off suggests that there is a higher chance of incorrectly classifying Charged Off customers as Fully Paid, which could result in Non-Performing Assets (NPAs), even while there is a greater chance of finding more Fully Paid customers.

The precision-recall curve's Area Under the Curve (AUC) is 0.975. This high AUC value indicates that the model has great precision-recall qualities and performs exceptionally well in differentiating between positive and negative classes.

Precision-recall curves, which concentrate on precise forecasts of the pertinent class (in this example, Class 1-Fully Paid), are essential, particularly in datasets that are unbalanced.

Recall and precision calculations ignore true negatives, concentrating only on accurately predicting fully paid clients.

A high AUC (97.5%) highlights the model's effectiveness and robustness in class distinction.

The goal of optimal model refining is to increase precision by reducing False Positives, which is essential for enhancing overall performance and reducing risks.

[162]:
```
lr = LogisticRegression(max_iter=1000, class_weight='balanced')

lr_model = lr.fit(X_train, y_train)

print(classification_report(y_test, lr_model.predict(X_test)))

cm_bal = confusion_matrix(y_test, lr_model.predict(X_test))
cm_bal_df = pd.DataFrame(cm_bal, index=['Defaulter','Fully paid'],
    columns=['Defaulter','Fully paid'])
cm_bal_df
```

```
              precision    recall  f1-score   support

           0       0.47      0.81      0.60     11678
           1       0.95      0.79      0.86     50601

    accuracy                           0.79     62279
   macro avg       0.71      0.80      0.73     62279
weighted avg       0.86      0.79      0.81     62279
```

[162]:
```
             Defaulter  Fully paid
Defaulter         9468        2210
Fully paid       10586       40015
```

[163]:
```
lr_model.intercept_
```

[163]:
```
array([6.35576272])
```

20 Thinking from a bank's perspective, which metric should our primary focus be on..

a. ROC AUC
b. Precision
c. Recall
d. F1 Score

Reducing risks and increasing profits are crucial from a bank's point of view. Because it includes both True Positive Rate (TPR) and False Positive Rate (FPR), ROC AUC (Receiver Operating Characteristic Area Under Curve) is in fact an important statistic.

21 How does the gap in precision and recall affect the bank?

Evaluating false positives and false negatives, which are measured by metrics like recall and precision, is essential to understanding the mistakes made by a model. A low recall presents a serious danger to the bank.

Thus, the bank will be impacted by the discrepancy between recall and precision. The number of inaccurate guesses will rise as the difference grows.

Reduced False Positives are the result of high precision. So, fewer non-performing loan accounts.

There are fewer False Negatives when recall is high. i.e., keeping loyal customers.

22 Which were the features that heavily affected the outcome?

In our situation, the most crucial features appear to be Address (Zipcode), Annual Income, and Grade.

23 Will the results be affected by geographical location? (Yes/No)

Yes, it is evident that zip_code (Address) is a crucial attribute, meaning that geographic location affects our outcome.

24 Recommendations

To properly control the precision-recall trade-off, concentrate on optimizing the F1 score and area under the Precision-Recall Curve. This improves risk management by lowering false positives and guaranteeing the identification of the majority of defaulters.

Hyperparameter adjustment and the use of more sophisticated classifiers, like as Random Forests or XGBoost, can improve model performance and capture complicated correlations in the data.

Stratified k-fold cross-validation was used to guarantee that the minority class was represented in each fold, yielding accurate model performance estimates.

Examine loans with lower grades more closely, and think about raising interest rates to offset the increased risk.

Use focused tactics, such extra verification procedures or higher interest rates, for high-risk zip codes.

To reduce the risk of default, evaluate small business loans with extra collateral requirements and financial health assessments.

LoanTap can improve loan approval procedures, reduce non-performing asset (NPA) risk, and guarantee long-term growth and financial stability by putting these suggestions into practice.