ET BRANDEQUITY

LoanTap is an online platform committed to delivering customized loan products to millennials. They innovate in an otherwise dull loan segment, to deliver instant, flexible loans on consumer friendly terms to salaried professionals and businessmen.

The data science team at LoanTap is building an underwriting layer to determine the creditworthiness of MSMEs as well as individuals.

LoanTap deploys formal credit to salaried individuals and businesses 4 main financial instruments:

- Personal Loan
- EMI Free Loan
- Personal Overdraft
- Advance Salary Loan

This case study will focus on the underwriting process behind Personal Loan only

Column Profiling :

- loan_amnt: Amount borrower applied for.
- term: Loan duration (36 or 60 months).
- int_rate: Interest rate on loan.
- installment: Monthly repayment amount.
- grade: LoanTap assigned loan grade (Risk ratings by LoanTap.)
- sub_grade: LoanTap assigned loan grade (Risk ratings by LoanTap.)
- emp_title: Borrower's job title.
- emp_length: Duration of borrower's employment (0-10 years).
- home_ownership: Borrower's housing situation (own, rent, etc.).
- annual_inc: Borrower's yearly income.
- verification_status: Whether borrower's income was verified.
- issue_d: Loan issuance month.
- loan_status: Current status of the loan.
- purpose: Borrower's reason for the loan.
- title: The loan's title provided by the borrower.
- dti (Debt-to-Income ratio): Monthly debt vs. monthly income ratio.
- earliest_cr_line: Date of borrower's oldest credit account.
- open_acc: Number of borrower's active credit lines.
- pub_rec: Negative records on borrower's public credit profile.
- revol_bal: Total credit balance.
- revol_util: Usage percentage of 'revolving' accounts like credit cards.
- total_acc: Total number of borrower's credit lines.

- initial_list_status: Loan's first category ('W' or 'F').
- application_type: Individual or joint application.
- mort_acc: Number of borrower's mortgages.
- pub_rec_bankruptcies: Bankruptcy records for borrower.
- Address: Borrower's location.

In [346…
```
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")
```

In [347…
```
!gdown 1ZPYj7CZCfxntE8p2Lze_4QO4MyEOy6_d -O "loan.csv"
```

```
Downloading...
From: https://drive.google.com/uc?id=1ZPYj7CZCfxntE8p2Lze_4QO4MyEOy6_d
To: /content/loan.csv
100% 100M/100M [00:00<00:00, 167MB/s]
```

In [348…
```
data = pd.read_csv("loan.csv") # Reading the data
```

In [349…
```
data.head(3)
```

Out[349…

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status	i
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified	

In [350…
```
data.shape # Shape of the data
```

Out[350… (396030, 27)

In [351…
```
data.info()  # Exploring the data
```

```
<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
```

In [352…
```
pd.set_option('display.max_columns', None) # View all the columns
```

In [353…
```
data.head() # Top 5 rows from the data
```

Out[353…

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	Not Verified
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	Verified

Null Detection

In [354…
```
data.isnull().sum().sort_values(ascending = False)
```

Out[354...

	0
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

In [355...

```
missing_data_per = ((data.isnull().sum().sort_values(ascending = False) / data.shape[0])*100).reset_index()
missing_data_per.columns = ["Column_name" , "Missing_data_percentage"]
missing_data_per[missing_data_per["Missing_data_percentage"] > 0]
```

Out[355...

	Column_name	Missing_data_percentage
0	mort_acc	9.543469
1	emp_title	5.789208
2	emp_length	4.621115
3	title	0.443401
4	pub_rec_bankruptcies	0.135091
5	revol_util	0.069692

Insights :

- mort_acc (9.54%): Significant missing data, may require imputation or handling strategies.
- emp_title (5.79%): Moderate missing data, consider imputation based on job roles or titles.
- emp_length (4.62%): Missing data can be imputed or marked as unknown for employment length.
- title (0.44%): Low missing data, minimal impact on analysis.
- pub_rec_bankruptcies (0.14%): Very low missing data, likely can be ignored or imputed.
- revol_util (0.07%): Negligible missing data, can be easily handled with imputation if needed.

Duplicate Detection

In [356…
```
data.duplicated().sum()
```

Out[356…
```
0
```

Insights :

- This dataset does not contain any duplicate value

Unique Values and Counts

In [357…
```
for i in data.columns:
  print(f"Column {i} Unique Value Count :  {data[i].nunique()}\n")
  print(f"Column {i} Unique Values :  {data[i].unique()}\n")
  print(f"Column {i} Value Counts :  {data[i].value_counts()}\n")
  print("-"*80)
```

```
Column loan_amnt Unique Value Count :  1397

Column loan_amnt Unique Values :  [10000.  8000. 15600. ... 36275. 36475.   725.]

Column loan_amnt Value Counts :  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

--------------------------------------------------------------------------------
Column term Unique Value Count :  2

Column term Unique Values :  [' 36 months' ' 60 months']

Column term Value Counts :  term
36 months    302005
60 months     94025
Name: count, dtype: int64

--------------------------------------------------------------------------------
Column int_rate Unique Value Count :  566

Column int_rate Unique Values :  [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]

Column int_rate Value Counts :  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

--------------------------------------------------------------------------------
Column installment Unique Value Count :  55706

Column installment Unique Values :  [329.48 265.68 506.97 ... 343.14 118.13 572.44]

Column installment Value Counts :  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

--------------------------------------------------------------------------------
Column grade Unique Value Count :  7

Column grade Unique Values :  ['B' 'A' 'C' 'E' 'D' 'F' 'G']

Column grade Value Counts :  grade
B    116018
C    105987
A     64187
D     63524
E     31488
F     11772
G      3054
Name: count, dtype: int64

--------------------------------------------------------------------------------
Column sub_grade Unique Value Count :  35

Column sub_grade Unique Values :  ['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']

Column sub_grade Value Counts :  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

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

```
Column emp_title Unique Value Count :  173105

Column emp_title Unique Values :  ['Marketing' 'Credit analyst ' 'Statistician' ...
 "Michael's Arts & Crafts" 'licensed bankere' 'Gracon Services, Inc']

Column emp_title Value Counts :  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

--------------------------------------------------------------------------------
Column emp_length Unique Value Count :  11

Column emp_length Unique Values :  ['10+ years' '4 years' '< 1 year' '6 years' '9 years' '2 years' '3 years'
 '8 years' '7 years' '5 years' '1 year' nan]

Column emp_length Value Counts :  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

--------------------------------------------------------------------------------
Column home_ownership Unique Value Count :  6

Column home_ownership Unique Values :  ['RENT' 'MORTGAGE' 'OWN' 'OTHER' 'NONE' 'ANY']

Column home_ownership Value Counts :  home_ownership
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: count, dtype: int64

--------------------------------------------------------------------------------
Column annual_inc Unique Value Count :  27197

Column annual_inc Unique Values :  [117000.    65000.    43057.   ...  36111.    47212.    31789.88]

Column annual_inc Value Counts :  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

--------------------------------------------------------------------------------
Column verification_status Unique Value Count :  3

Column verification_status Unique Values :  ['Not Verified' 'Source Verified' 'Verified']

Column verification_status Value Counts :  verification_status
Verified           139563
Source Verified    131385
Not Verified       125082
Name: count, dtype: int64

--------------------------------------------------------------------------------
Column issue_d Unique Value Count :  115

Column issue_d Unique Values :  ['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']

Column issue_d Value Counts :  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

----------------------------------------------------------------------------
Column loan_status Unique Value Count :  2

Column loan_status Unique Values :  ['Fully Paid' 'Charged Off']

Column loan_status Value Counts :  loan_status
Fully Paid     318357
Charged Off     77673
Name: count, dtype: int64

----------------------------------------------------------------------------
Column purpose Unique Value Count :  14

Column purpose Unique Values :  ['vacation' 'debt_consolidation' 'credit_card' 'home_improvement'
 'small_business' 'major_purchase' 'other' 'medical' 'wedding' 'car'
 'moving' 'house' 'educational' 'renewable_energy']

Column purpose Value Counts :  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

----------------------------------------------------------------------------
Column title Unique Value Count :  48816

Column title Unique Values :  ['Vacation' 'Debt consolidation' 'Credit card refinancing' ...
 'Credit buster ' 'Loanforpayoff' 'Toxic Debt Payoff']

Column title Value Counts :  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
```

```
--------------------------------------------------------------------------------
Column dti Unique Value Count :  4262

Column dti Unique Values :  [26.24 22.05 12.79 ... 40.56 47.09 55.53]

Column dti Value Counts :  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

--------------------------------------------------------------------------------
Column earliest_cr_line Unique Value Count :  684

Column earliest_cr_line Unique Values :  ['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']

Column earliest_cr_line Value Counts :  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

--------------------------------------------------------------------------------
Column open_acc Unique Value Count :  61

Column open_acc Unique Values :  [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.]

Column open_acc Value Counts :  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
```

```
--------------------------------------------------------------------------------
Column pub_rec Unique Value Count :  20

Column pub_rec Unique Values :  [ 0.  1.  2.  3.  4.  6.  5.  8.  9. 10. 11.  7. 19. 13. 40. 17. 86. 12.
 24. 15.]

Column pub_rec Value Counts :  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

--------------------------------------------------------------------------------
Column revol_bal Unique Value Count :  55622

Column revol_bal Unique Values :  [ 36369.  20131.  11987. ...  34531. 151912.  29244.]

Column revol_bal Value Counts :  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

--------------------------------------------------------------------------------
Column revol_util Unique Value Count :  1226

Column revol_util Unique Values :  [ 41.8   53.3   92.2  ...  56.26 111.4  128.1 ]

Column revol_util Value Counts :  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

--------------------------------------------------------------------------------
Column total_acc Unique Value Count :  118

Column total_acc Unique Values :  [ 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.]

Column total_acc Value Counts :  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

--------------------------------------------------------------------------------
Column initial_list_status Unique Value Count :  2

Column initial_list_status Unique Values :  ['w' 'f']

Column initial_list_status Value Counts :  initial_list_status
f    238066
w    157964
Name: count, dtype: int64

--------------------------------------------------------------------------------
Column application_type Unique Value Count :  3

Column application_type Unique Values :  ['INDIVIDUAL' 'JOINT' 'DIRECT_PAY']

Column application_type Value Counts :  application_type
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: count, dtype: int64

--------------------------------------------------------------------------------
Column mort_acc Unique Value Count :  33

Column mort_acc Unique Values :  [ 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.]

Column mort_acc Value Counts :  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

--------------------------------------------------------------------------------
Column pub_rec_bankruptcies Unique Value Count :  9

Column pub_rec_bankruptcies Unique Values :  [ 0.  1.  2.  3. nan  4.  5.  6.  7.  8.]

Column pub_rec_bankruptcies Value Counts :  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
```

```
--------------------------------------------------------------------------------
Column address Unique Value Count :  393700

Column address Unique Values :  ['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']

Column address Value Counts :  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

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

Converstion Of Date Time Column

In [358…
```
data["issue_d"] = pd.to_datetime(data["issue_d"])
data["earliest_cr_line"] = pd.to_datetime(data["earliest_cr_line"])
```

In [359…
```
data.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  datetime64[ns]
 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  datetime64[ns]
 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: datetime64[ns](2), float64(12), object(13)
memory usage: 81.6+ MB
```

Imputing Null Values

In [360…
```
missing_columns = missing_data_per[missing_data_per["Missing_data_percentage"] > 0]["Column_name"].values
```

In [361…
```
num_col = []
cat_col = []
for i in missing_columns:
  if data[i].dtype == "object":
    cat_col.append(i)
  elif data[i].dtype == "float64":
    num_col.append(i)

cat_col.remove("emp_title")
```

In [362…
```
data['emp_title'] = data['emp_title'].fillna("Not Available")
```

In [363…
```
from sklearn.impute import SimpleImputer
num_imputer = SimpleImputer(strategy="median")
cat_imputer = SimpleImputer(strategy = "most_frequent")
```

In [364…
```
data[num_col] = num_imputer.fit_transform(data[num_col])
data[cat_col] = cat_imputer.fit_transform(data[cat_col])
```

In [365…
```
data.isnull().sum()
```

Out[365…

	0
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

Data Statistics

In [366…
```
data.describe().T
```

Out[366...

	count	mean	min	25%	50%	75%	max	std
loan_amnt	396030.0	14113.888089	500.0	8000.0	12000.0	20000.0	40000.0	8357.441341
int_rate	396030.0	13.6394	5.32	10.49	13.33	16.49	30.99	4.472157
installment	396030.0	431.849698	16.08	250.33	375.43	567.3	1533.81	250.72779
annual_inc	396030.0	74203.175798	0.0	45000.0	64000.0	90000.0	8706582.0	61637.621158
issue_d	396030	2014-02-02 15:57:58.045602560	2007-06-01 00:00:00	2013-05-01 00:00:00	2014-04-01 00:00:00	2015-03-01 00:00:00	2016-12-01 00:00:00	NaN
dti	396030.0	17.379514	0.0	11.28	16.91	22.98	9999.0	18.019092
earliest_cr_line	396030	1998-05-03 09:34:15.062495488	1944-01-01 00:00:00	1994-10-01 00:00:00	1999-09-01 00:00:00	2003-04-01 00:00:00	2013-10-01 00:00:00	NaN
open_acc	396030.0	11.311153	0.0	8.0	10.0	14.0	90.0	5.137649
pub_rec	396030.0	0.178191	0.0	0.0	0.0	0.0	86.0	0.530671
revol_bal	396030.0	15844.539853	0.0	6025.0	11181.0	19620.0	1743266.0	20591.836109
revol_util	396030.0	53.792451	0.0	35.9	54.8	72.9	892.3	24.443685
total_acc	396030.0	25.414744	2.0	17.0	24.0	32.0	151.0	11.886991
mort_acc	396030.0	1.736308	0.0	0.0	1.0	3.0	34.0	2.056819
pub_rec_bankruptcies	396030.0	0.121483	0.0	0.0	0.0	0.0	8.0	0.355962

In [367...
```
data.describe(include = "object").T
```

Out[367...

	count	unique	top	freq
term	396030	2	36 months	302005
grade	396030	7	B	116018
sub_grade	396030	35	B3	26655
emp_title	396030	173106	Not Available	22927
emp_length	396030	11	10+ years	144342
home_ownership	396030	6	MORTGAGE	198348
verification_status	396030	3	Verified	139563
loan_status	396030	2	Fully Paid	318357
purpose	396030	14	debt_consolidation	234507
title	396030	48816	Debt consolidation	154228
initial_list_status	396030	2	f	238066
application_type	396030	3	INDIVIDUAL	395319
address	396030	393700	USCGC Smith\r\nFPO AE 70466	8

Insights :

- Loan Amount and Terms: Average loan amount is 14,114 with most loans ranging between 8,000 and 20,000. Most loans have a 36-month term.
- Interest Rate and Installments: Average interest rate is 13.64%, with monthly installments averaging $431.85, showing significant variation based on loan size.
- Income and DTI: Average annual income is $74,203, but with high variance. The average debt-to-income ratio is 17.38%, with some extreme outliers.
- Credit History: On average, borrowers have 11 open accounts. Public records (e.g., bankruptcies) are rare, with an average of 0.18.
- Loan Purpose and Employment: Most loans are for debt consolidation. The most common job title is Teacher, with many borrowers having 10+ years of employment.

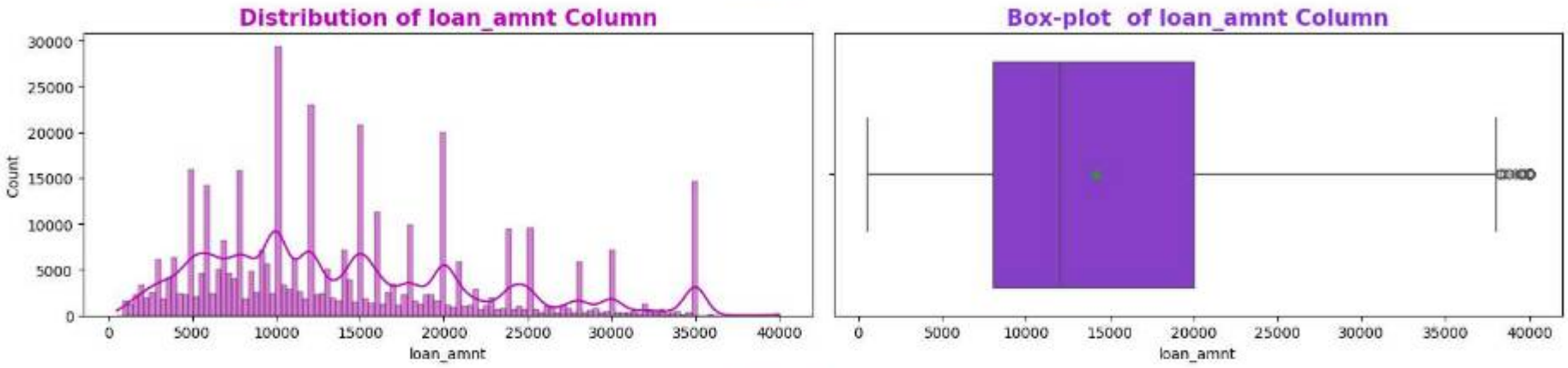
Data Visualization

In [368...
```
num_col2 = data.select_dtypes(include=np.number).columns
num_col2
```

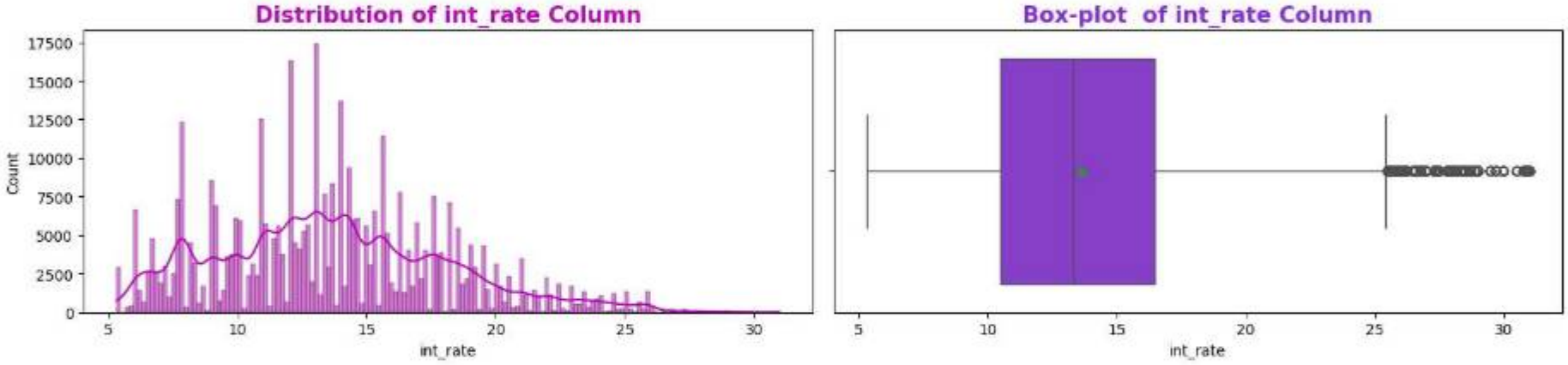
Out[368…
```
Index(['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'mort_acc',
       'pub_rec_bankruptcies'],
      dtype='object')
```

In [369…
```
for i in num_col2:
  plt.figure(figsize=(15,4))
  plt.subplot(1,2,1)
  sns.histplot(x = data[i],kde = True,color = "m")
  plt.title(f'Distribution of {i} Column',fontsize=15,fontweight='bold',color='m')
  plt.subplot(1,2,2)
  sns.boxplot(x = data[i],color ="blueviolet",showmeans = True)
  plt.title(f'Box-plot  of {i} Column',fontsize=15,fontweight='bold',color='blueviolet')
  plt.suptitle(f'{i}',fontsize=15,fontweight='bold',color='r')
  plt.tight_layout()
  plt.show()
```

loan_amnt

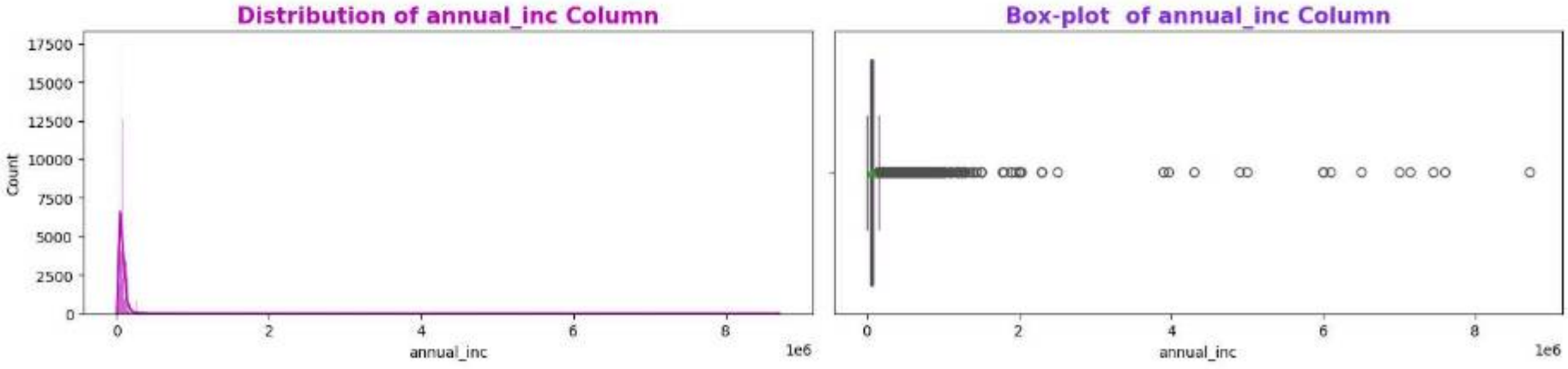


int_rate

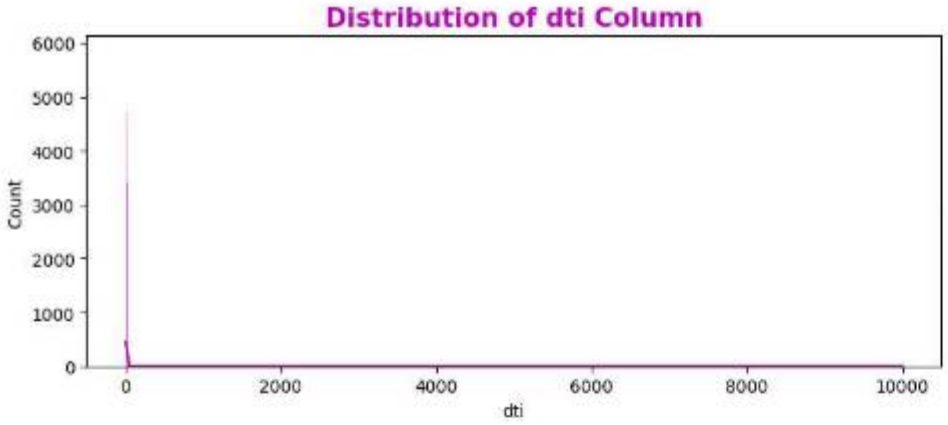


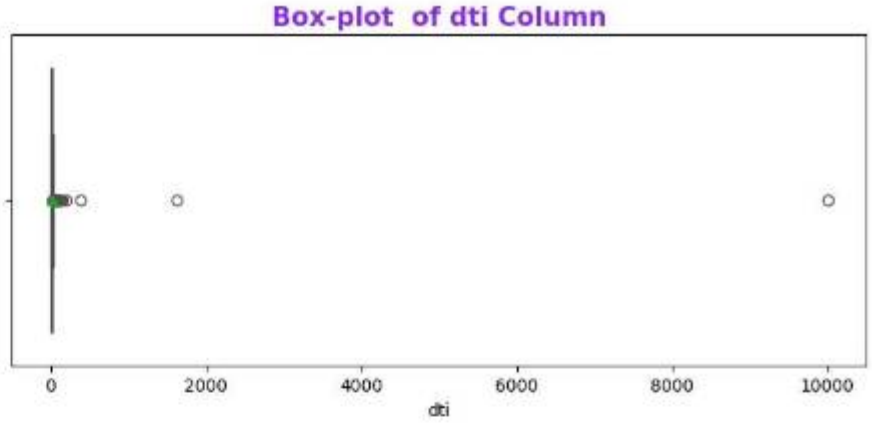
installment

annual_inc

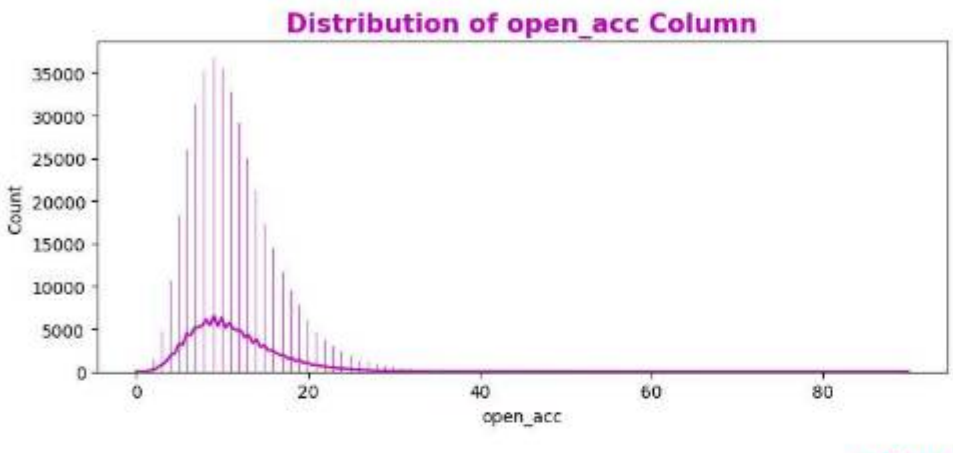


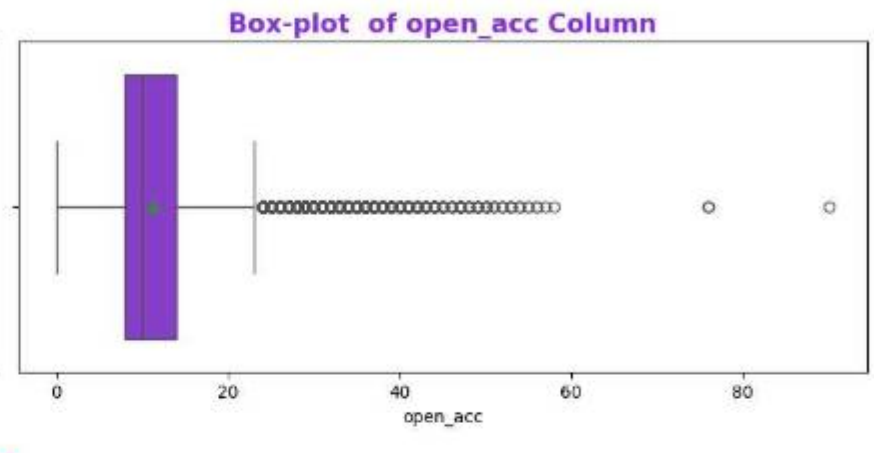
dti



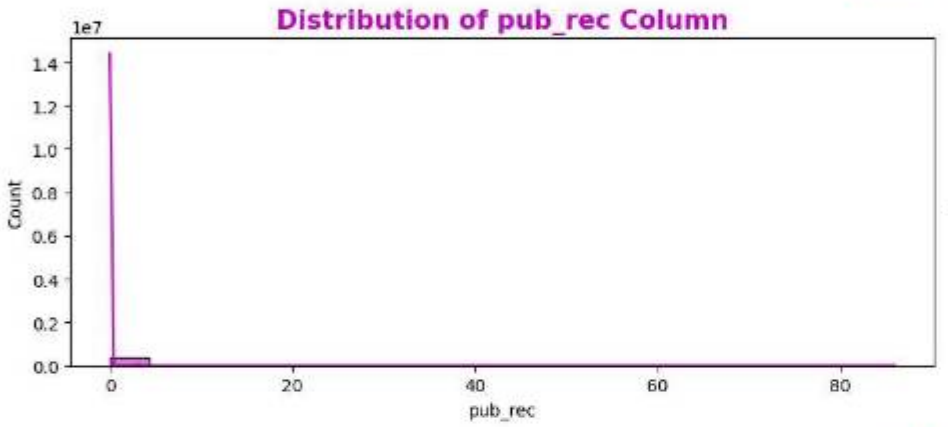


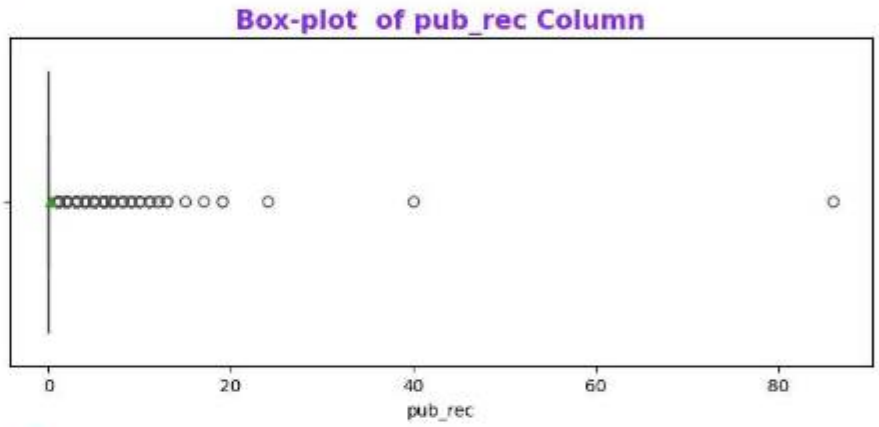
open_acc



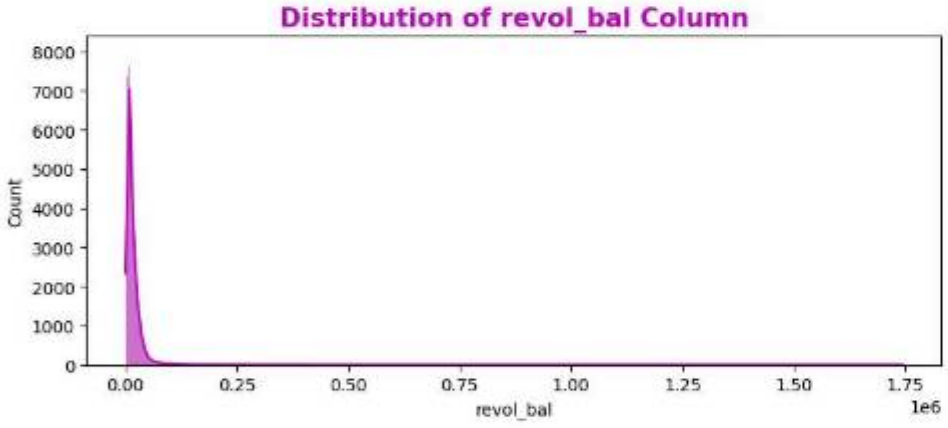


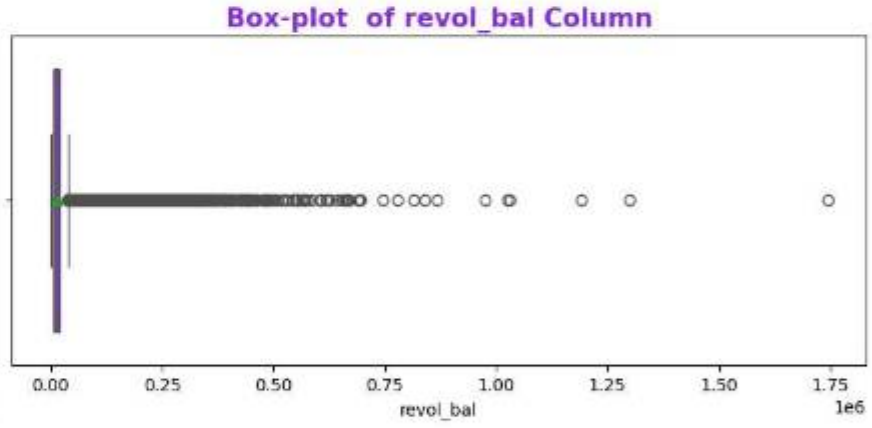
pub_rec



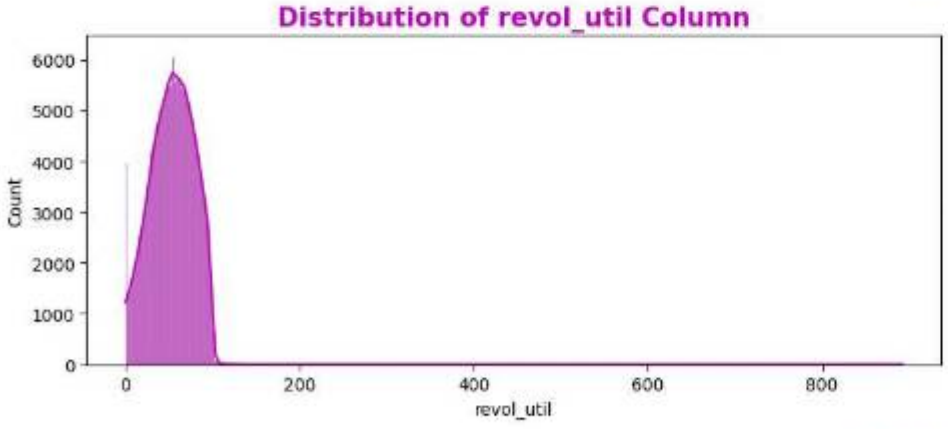


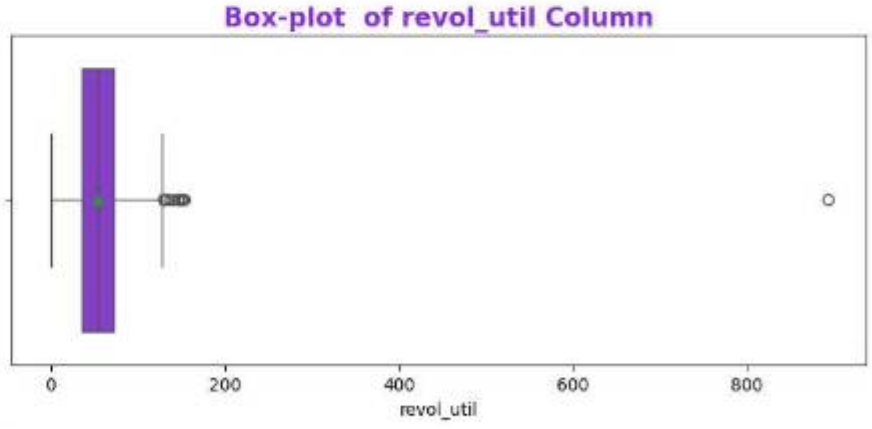
revol_bal



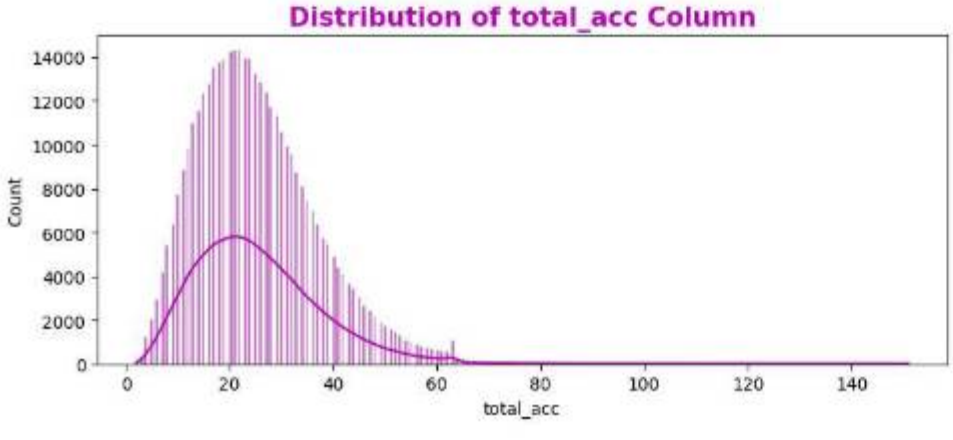


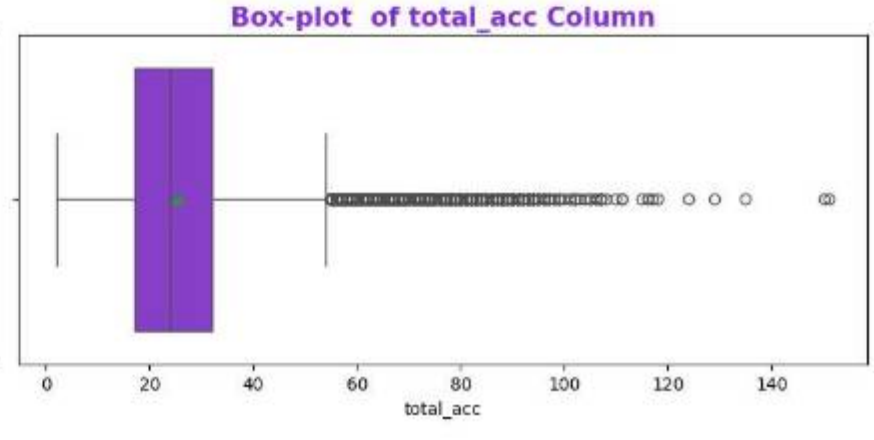
revol_util

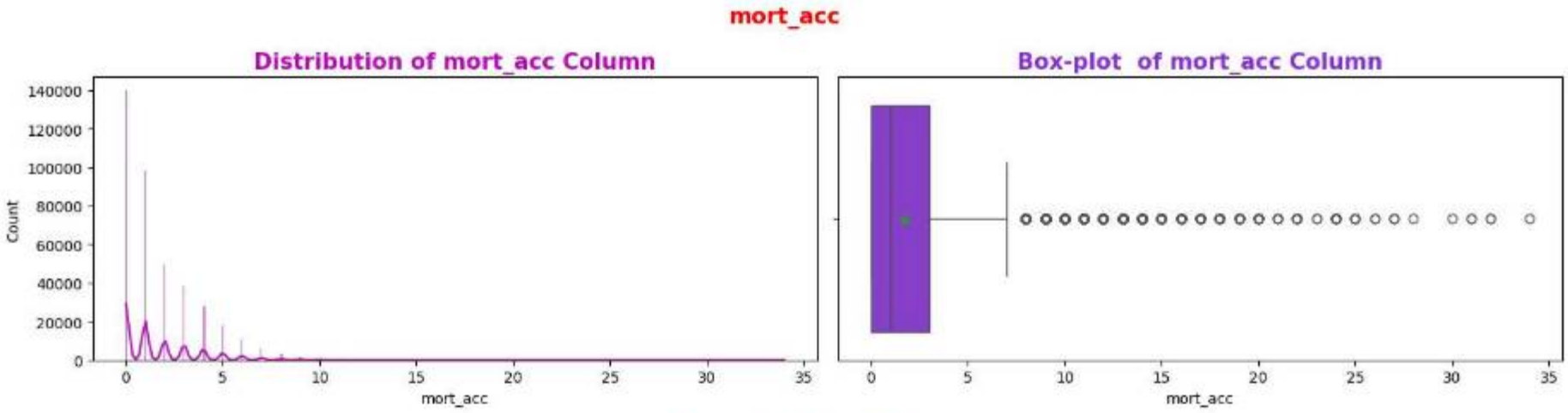




total_acc







Insights :

- The analysis indicates the presence of outliers, which requires further exploration using outlier detection methods.
- Several numerical features may still contain potential outliers that need closer examination.
- Features like pub_rec, mort_acc, and pub_rec_bankruptcies have a sparse distribution of unique values.
- It may be beneficial to create binary features from variables with sparse distributions, such as pub_rec, mort_acc, and pub_rec_bankruptcies.

In [370…
```
data.head(3)
```

Out[370…

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status	i
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	Not Verified	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	Not Verified	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	Source Verified	

Splitting Address Column

In [371…
```
data["state"]  = data["address"].str.split().str[-2]
data["pincode"] =  data["address"].str.split().str[-1]
```

Extracting Month Name From Issued Date Column

In [372…
```
data["issued_d_month"] = data["issue_d"].dt.month_name() # Extracting Month Name from issued_date
```

Q1. What percentage of customers have fully paid their Loan Amount?

In [373…
```
round(data['loan_status'].value_counts(normalize=True)*100,2)
```

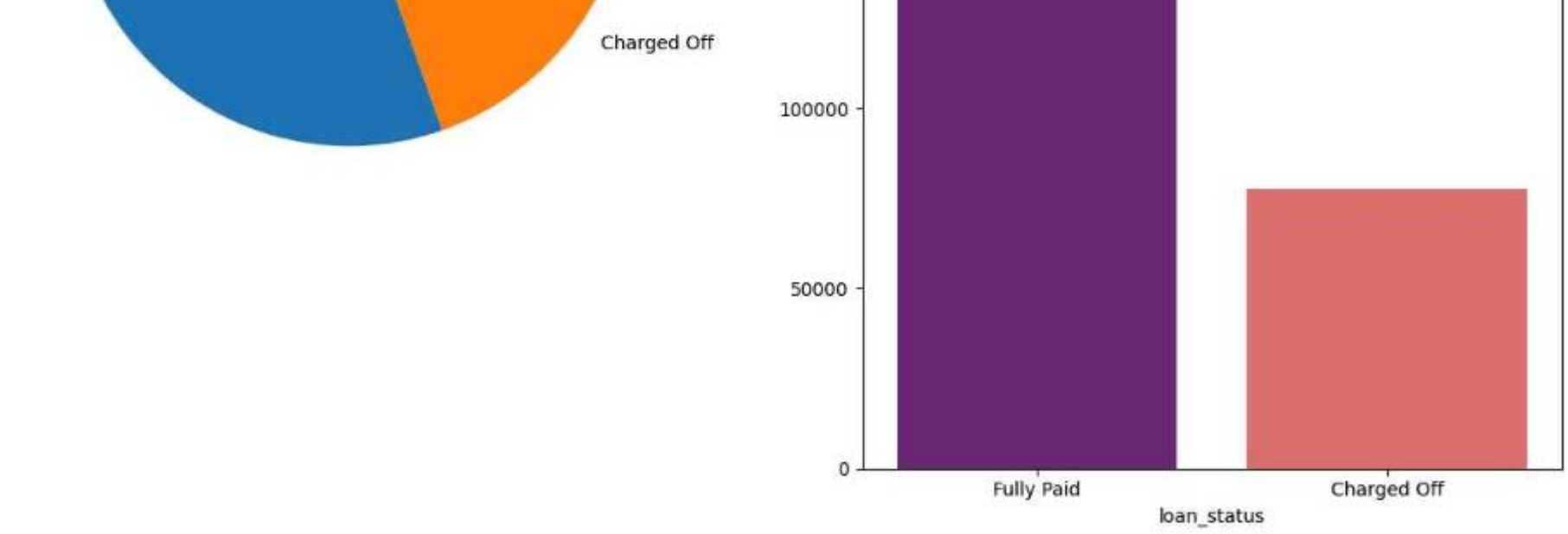
Out[373…

loan_status	proportion
Fully Paid	80.39
Charged Off	19.61

dtype: float64

In [374…

```
plt.figure(figsize = (12,6))
plt.subplot(1,2,1)
plt.pie(x=data['loan_status'].value_counts().values, labels=data['loan_status'].value_counts().index,autopct='%1.1f%%')
 #explode=[0.0,0.1,0.1,0.1])
plt.suptitle('Distribution of Loan Status', fontsize=10, fontweight='bold',color='blueviolet')
plt.subplot(1,2,2)
sns.countplot(data = data , x = "loan_status",palette = "magma")
plt.tight_layout()
plt.show()
```

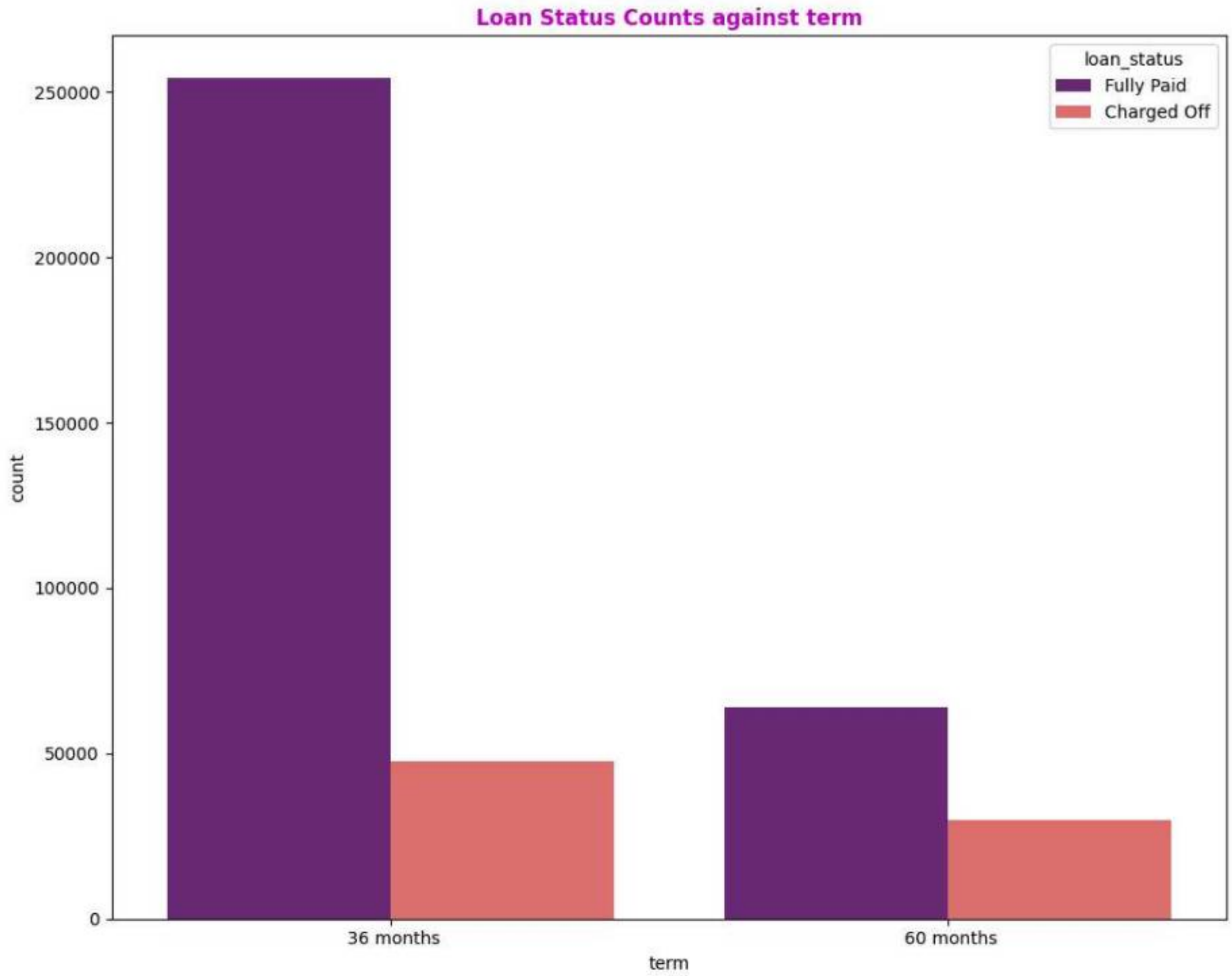


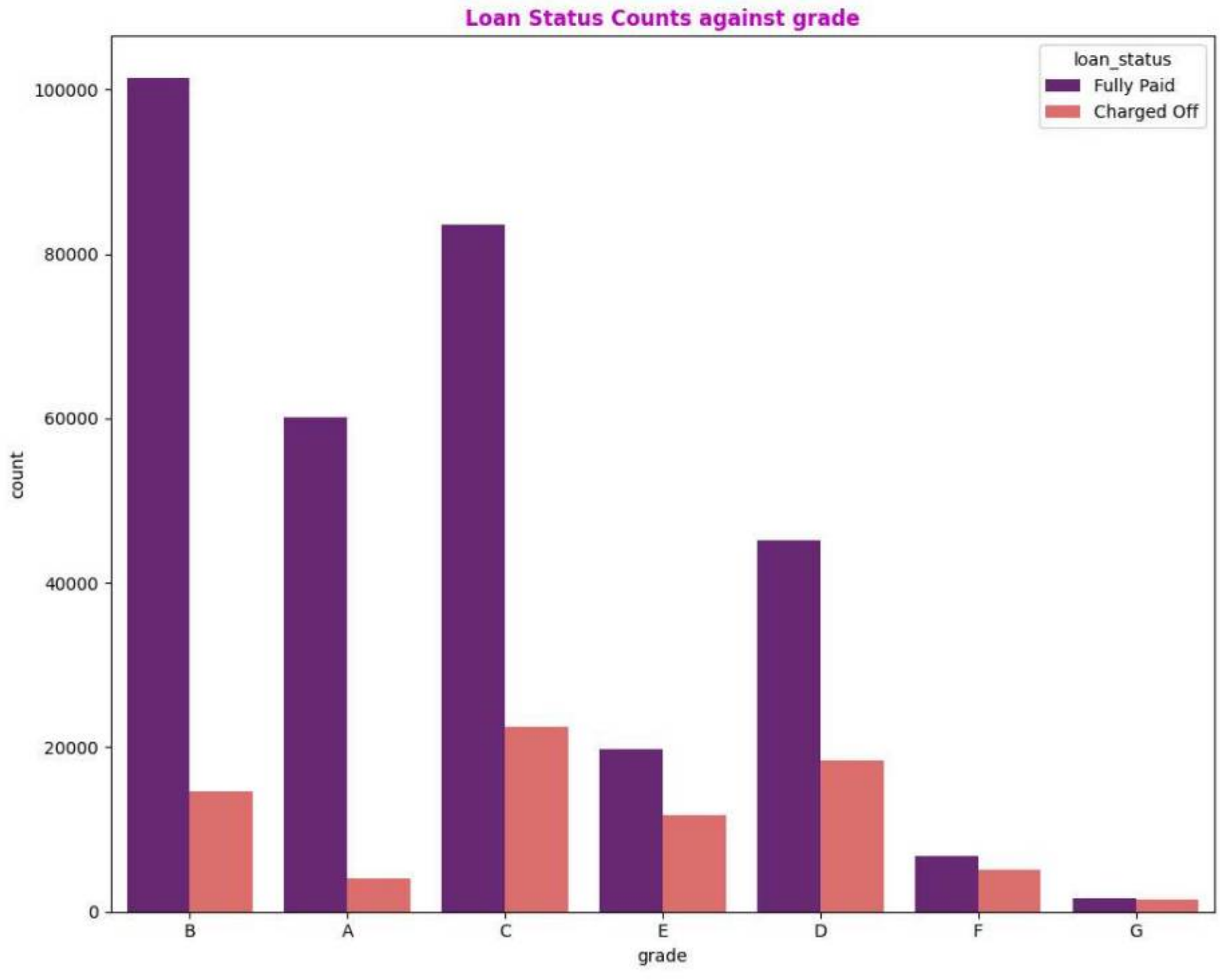
In [375…

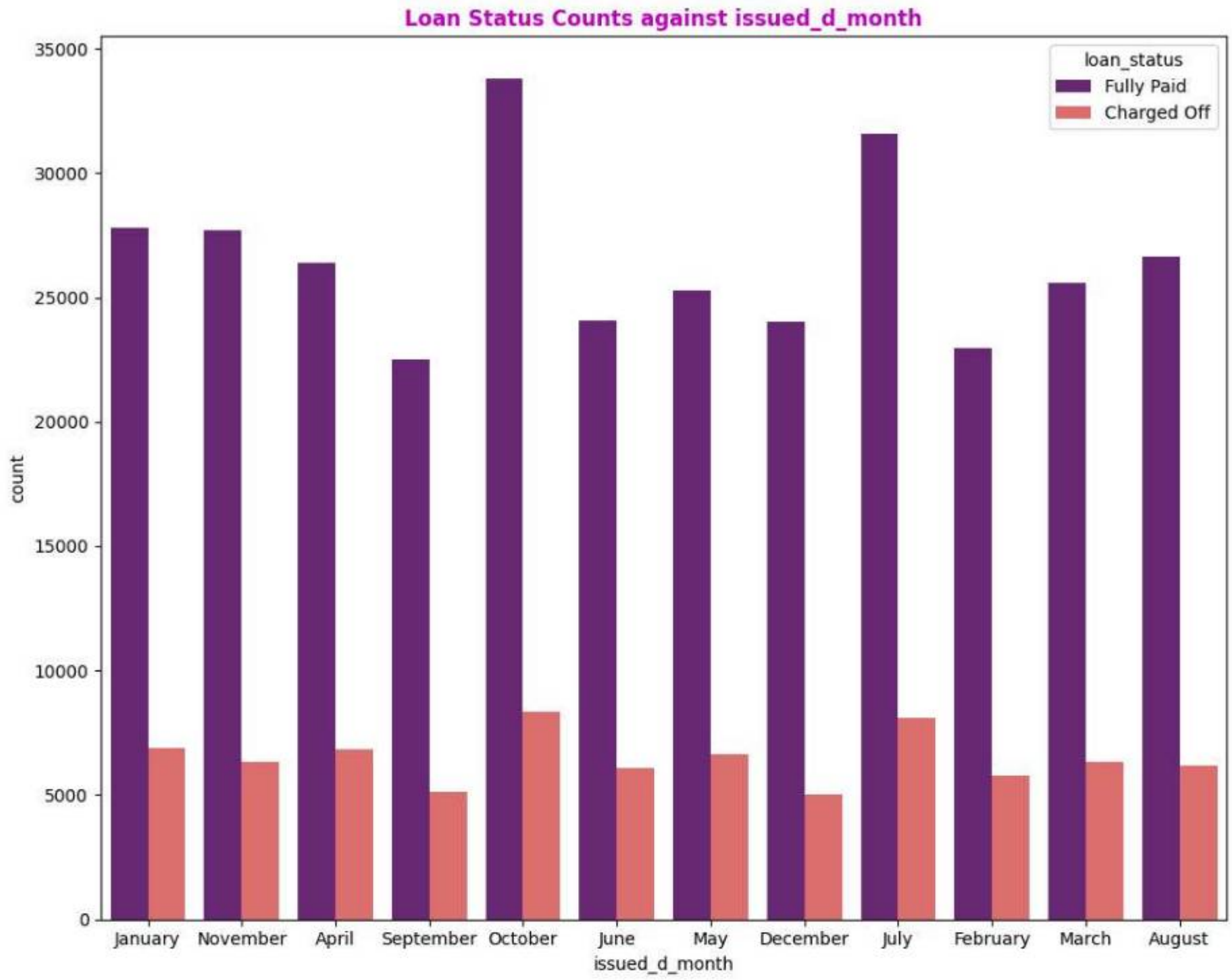
```
cat_col2 = ["term","grade","issued_d_month","pincode","verification_status","home_ownership"]
```

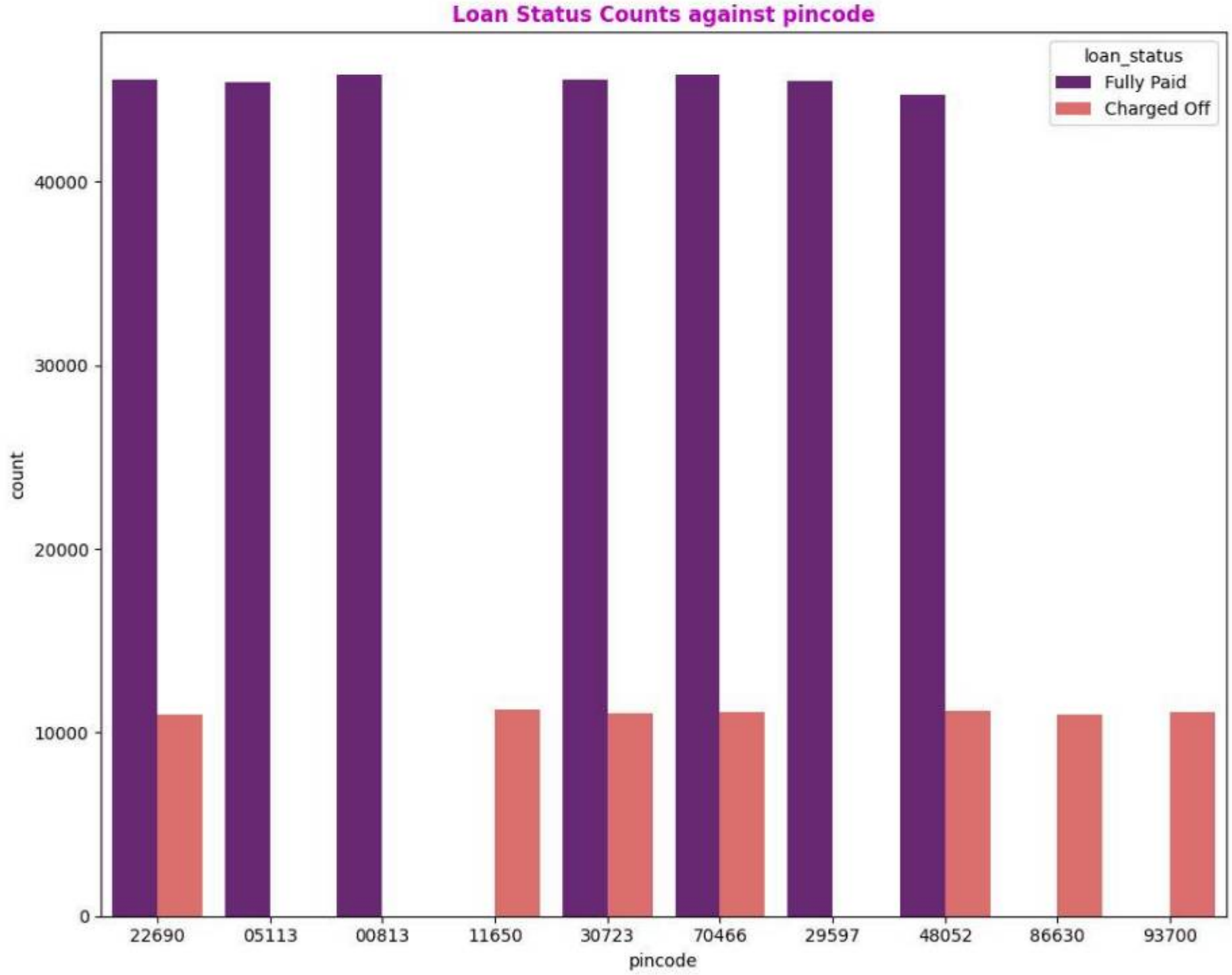
In [376…

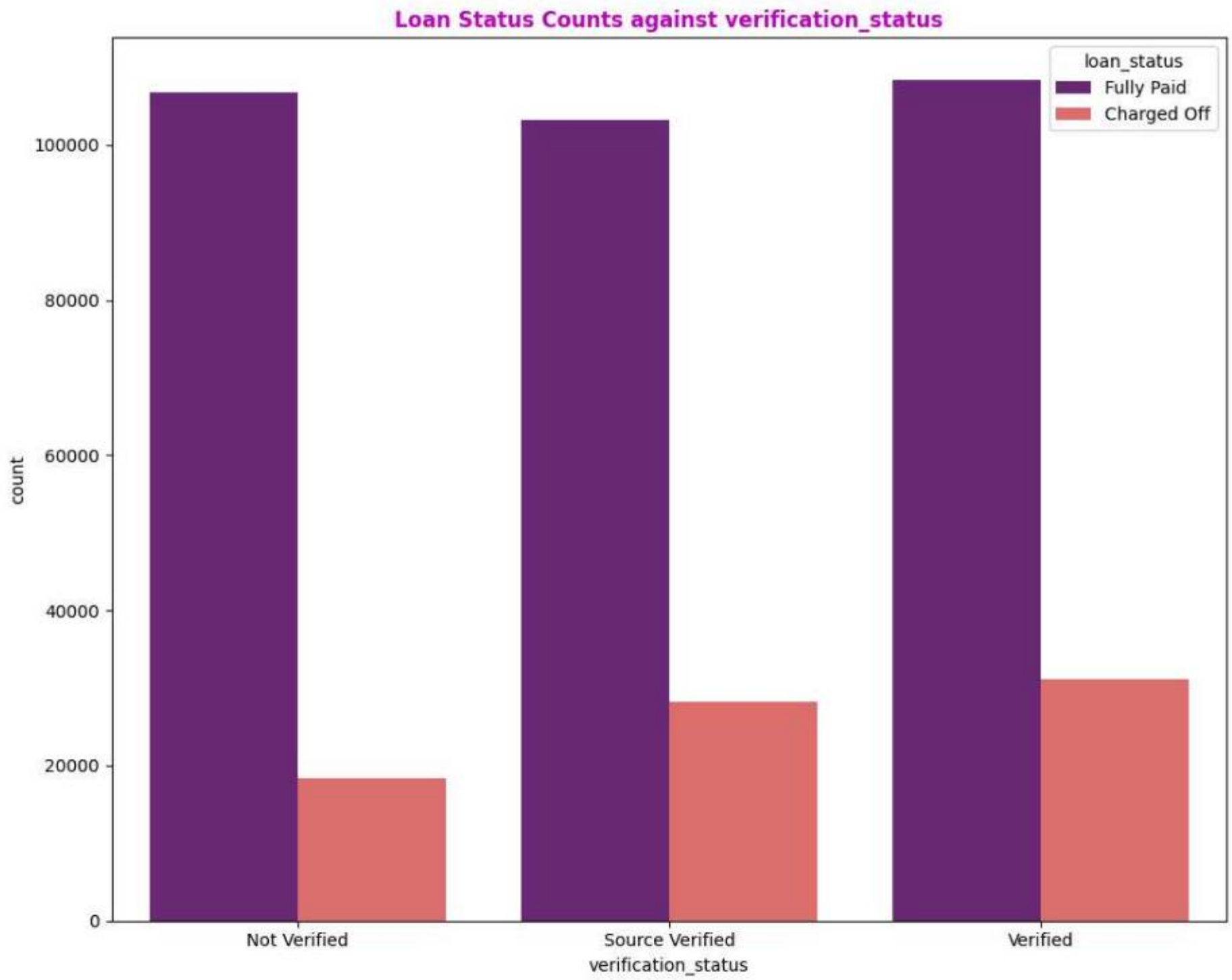
```
for  j in cat_col2:
  plt.figure(figsize=(10,8))
  sns.countplot(hue = data["loan_status"], x = data[j],palette = "magma")
  plt.title(f'Loan Status Counts against {j}',fontsize=12,fontweight='bold',color='m')
  plt.tight_layout()
  plt.show()
```

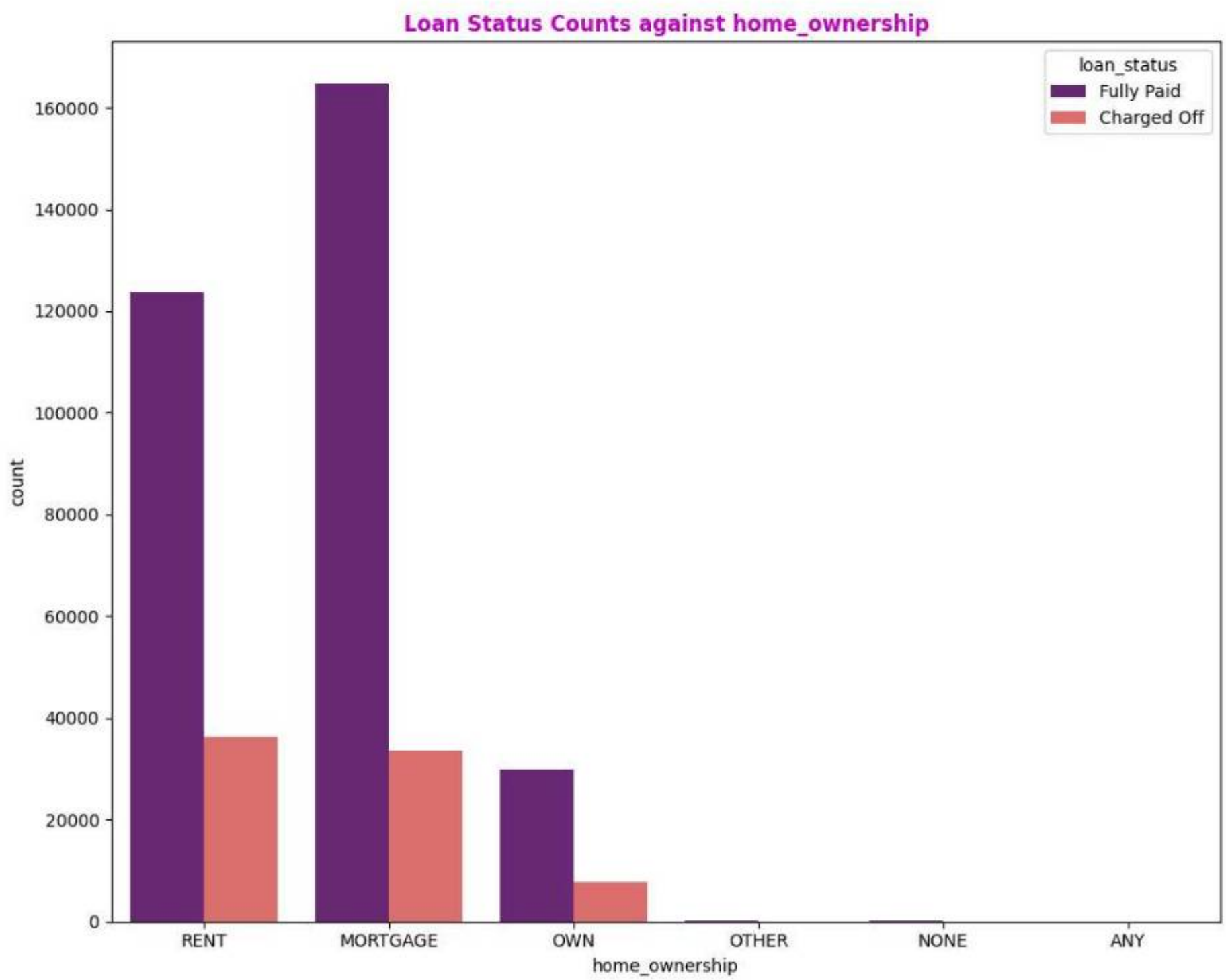

Loan Status Counts against term
loan_status
Fully Paid
Charged Off
250000
200000
150000
100000
50000
0
count
36 months
60 months
term


Loan Status Counts against grade
loan_status
Fully Paid
Charged Off
100000
80000
60000
40000
20000
0
count
B
A
C
E
D
F
G
grade


Loan Status Counts against issued_d_month
loan_status
Fully Paid
Charged Off
count
35000
30000
25000
20000
15000
10000
5000
0
January
November
April
September
October
June
May
December
July
February
March
August
issued_d_month

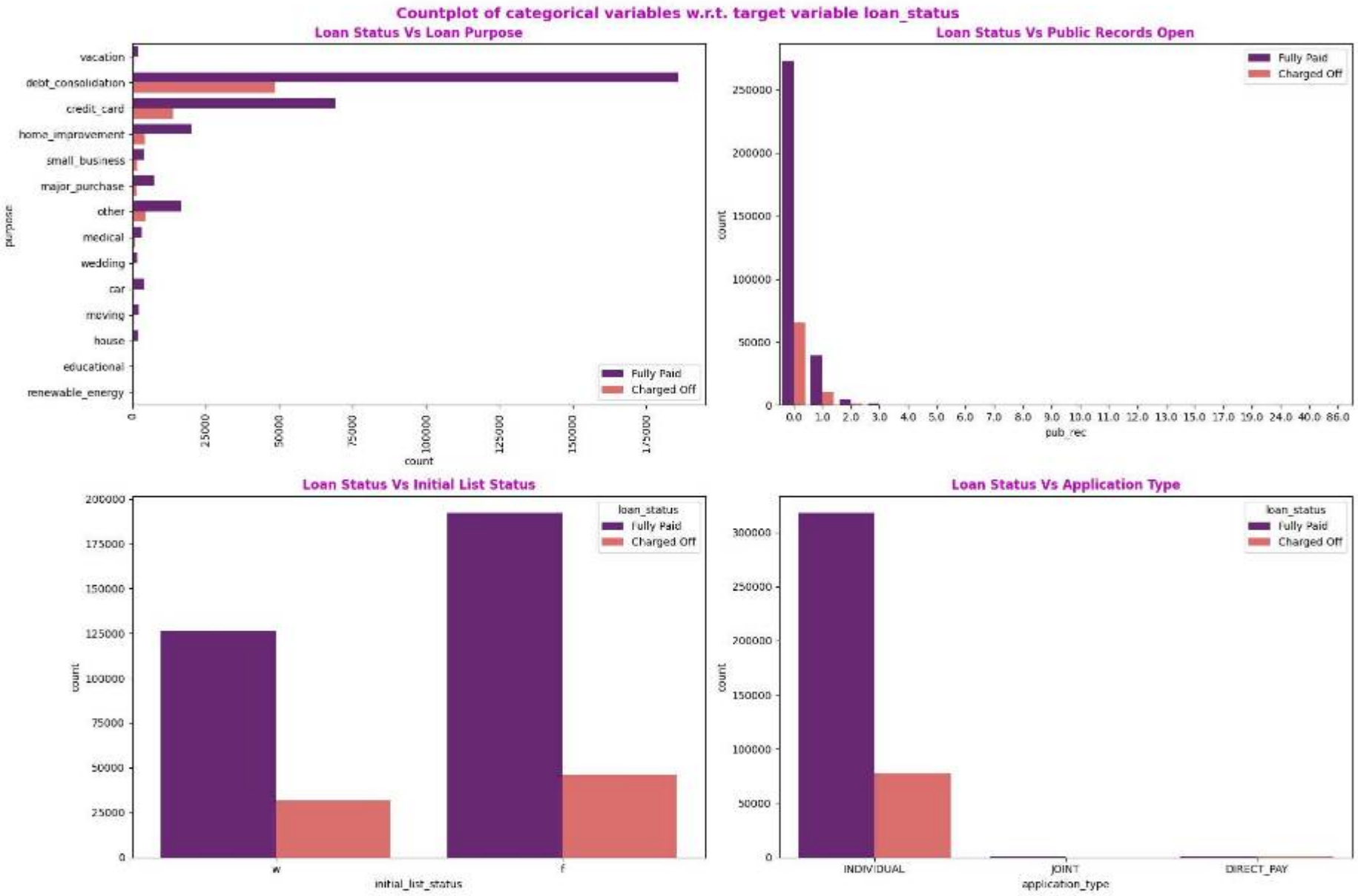

Loan Status Counts against pincode
loan_status
Fully Paid
Charged Off
count
40000
30000
20000
10000
0
22690
05113
00813
11650
30723
70466
29597
48052
86630
93700
pincode


Loan Status Counts against verification_status
loan_status
Fully Paid
Charged Off
count
100000
80000
60000
40000
20000
0
Not Verified
Source Verified
Verified
verification_status


Loan Status Counts against home_ownership
loan_status
Fully Paid
Charged Off
count
160000
140000
120000
100000
80000
60000
40000
20000
0
RENT
MORTGAGE
OWN
OTHER
NONE
ANY
home_ownership

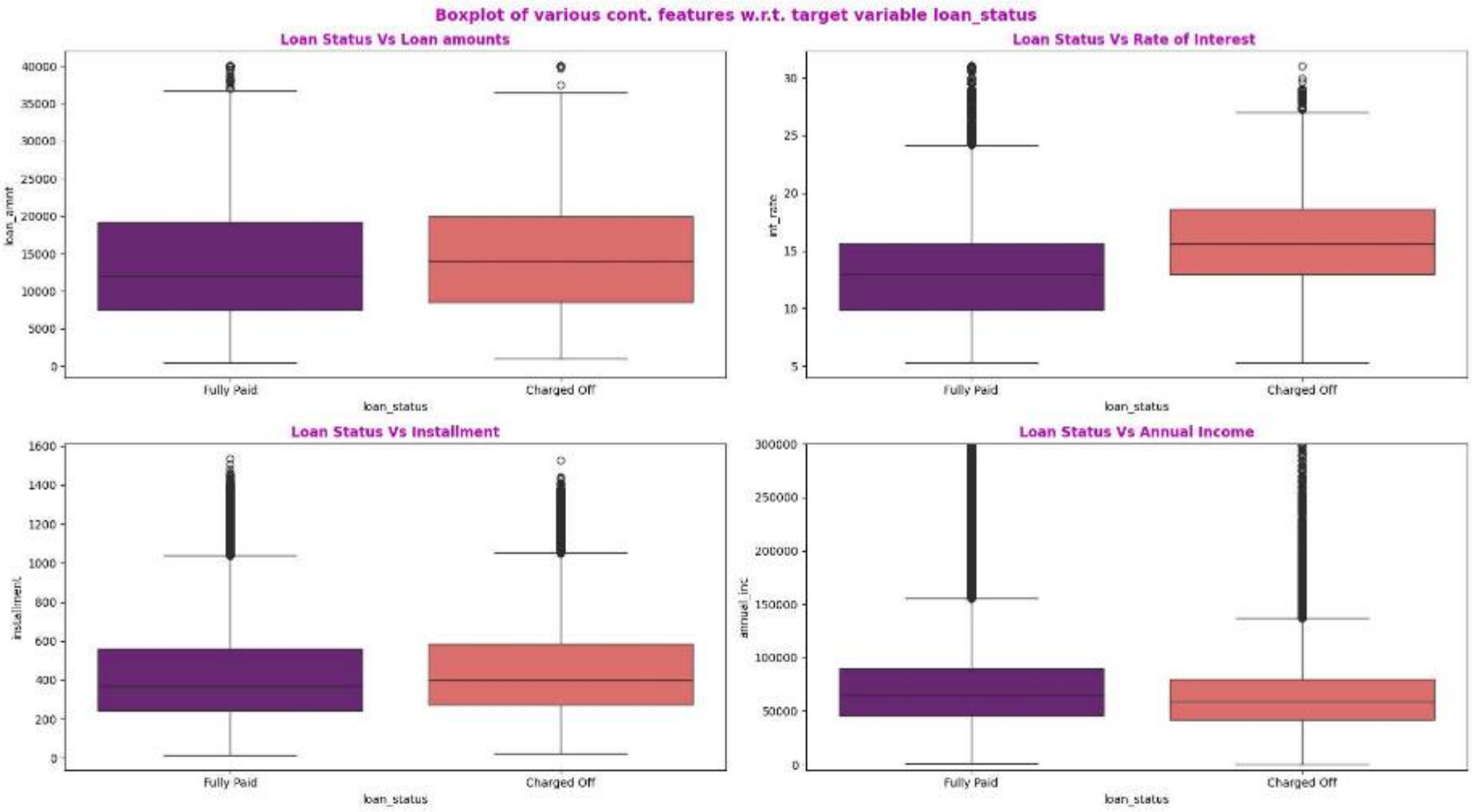
In [377…

```
#Countplot of categorical variables w.r.t. target variable loan_status
plt.figure(figsize=(18,12))
plt.suptitle('Countplot of categorical variables w.r.t. target variable loan_status',
             fontsize=14,fontweight='bold',color='m')
plt.subplot(221)
sns.countplot(data=data, y='purpose', hue='loan_status',palette="magma")
plt.xticks(rotation=90)
plt.title('Loan Status Vs Loan Purpose',fontsize=12,fontweight='bold',color='m')
plt.legend(loc=4)
plt.subplot(222)
sns.countplot(data=data, x='pub_rec',hue='loan_status',palette="magma")
plt.title('Loan Status Vs Public Records Open',fontsize=12,fontweight='bold',color='m')
plt.legend(loc=1)
plt.subplot(223)
sns.countplot(data=data, x='initial_list_status', hue='loan_status',palette="magma")
plt.title('Loan Status Vs Initial List Status',fontsize=12,fontweight='bold',color='m')
plt.subplot(224)
sns.countplot(data=data, x='application_type',hue='loan_status',palette="magma")
plt.title('Loan Status Vs Application Type',fontsize=12,fontweight='bold',color='m')
plt.tight_layout()
plt.show()
```



In [378…

```
#Boxplot of various cont. features w.r.t. target variable loan_status
plt.figure(figsize=(18,10))
plt.suptitle('Boxplot of various cont. features w.r.t. target variable loan_status',
             fontsize=14,fontweight='bold',color='m')
plt.subplot(221)
sns.boxplot(data=data, x='loan_status', y='loan_amnt',palette="magma")
plt.title('Loan Status Vs Loan amounts',fontsize=12,fontweight='bold',color='m')
plt.subplot(222)
sns.boxplot(data=data, x='loan_status', y='int_rate',palette="magma")
plt.title('Loan Status Vs Rate of Interest ',fontsize=12,fontweight='bold',color='m')
plt.subplot(223)
sns.boxplot(data=data, x='loan_status', y='installment',palette="magma")
plt.title('Loan Status Vs Installment',fontsize=12,fontweight='bold',color='m')
plt.subplot(224)
sns.boxplot(data=data, x='loan_status', y='annual_inc',palette="magma")
plt.ylim(bottom=-5000, top=300000)
plt.title('Loan Status Vs Annual Income',fontsize=12,fontweight='bold',color='m')
plt.tight_layout()
plt.show()
```

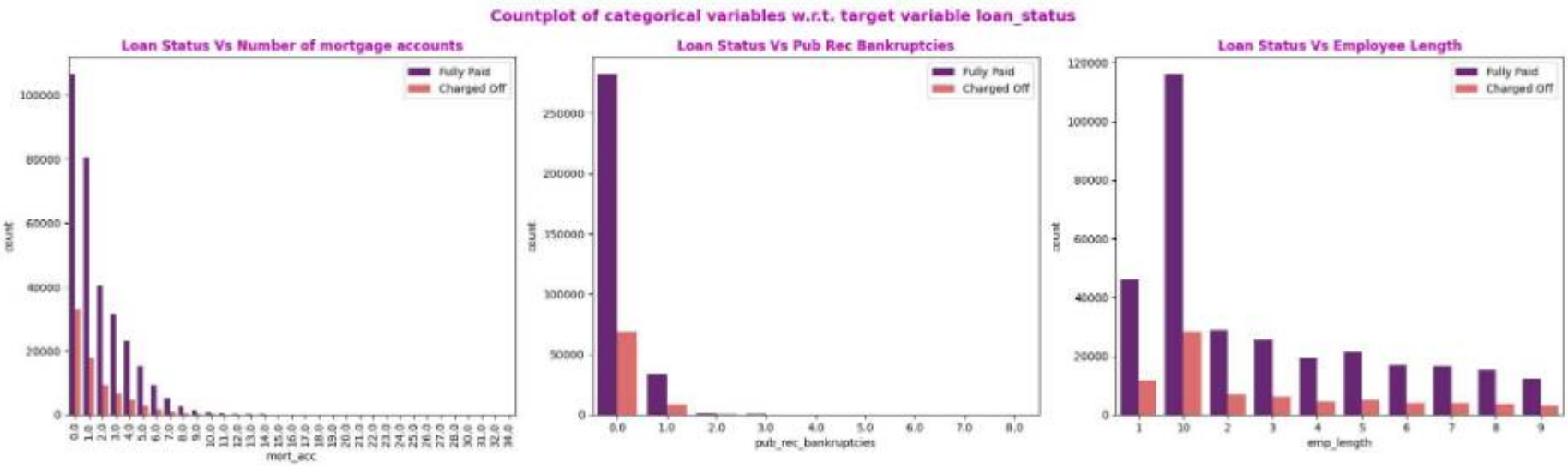


Insights :

- Customers with a Charged Off status tend to have a noticeably higher median interest rate and loan amount than those with a Fully Paid status.
- The median annual income and EMI are generally lower and higher, respectively, for Charged Off customers compared to Fully Paid customers.

In [379…
```
data['emp_length'] = data['emp_length'].str.extract('(\d+)')
```

In [380…
```
#Countplot for various categorical features w.r.t. target variable loan_status

plt.figure(figsize=(20,6))
plt.suptitle('Countplot of categorical variables w.r.t. target variable loan_status',
             fontsize=14,fontweight='bold',color='m')
plt.subplot(131)
sns.countplot(data=data, x='mort_acc',hue='loan_status',palette="magma")
plt.xticks(rotation=90)
plt.title('Loan Status Vs Number of mortgage accounts',fontsize=12,fontweight='bold',color='m')
plt.legend(loc=1)
plt.subplot(132)
sns.countplot(data=data, x='pub_rec_bankruptcies',hue='loan_status',palette="magma")
plt.title('Loan Status Vs Pub Rec Bankruptcies',fontsize=12,fontweight='bold',color='m')
plt.legend(loc=1)
plt.subplot(133)
order = sorted(data.emp_length.unique().tolist())
sns.countplot(data=data, x='emp_length',hue='loan_status',order=order,palette="magma")
plt.title('Loan Status Vs Employee Length',fontsize=12,fontweight='bold',color='m')
plt.legend(loc=1)
plt.tight_layout()
plt.show()
```



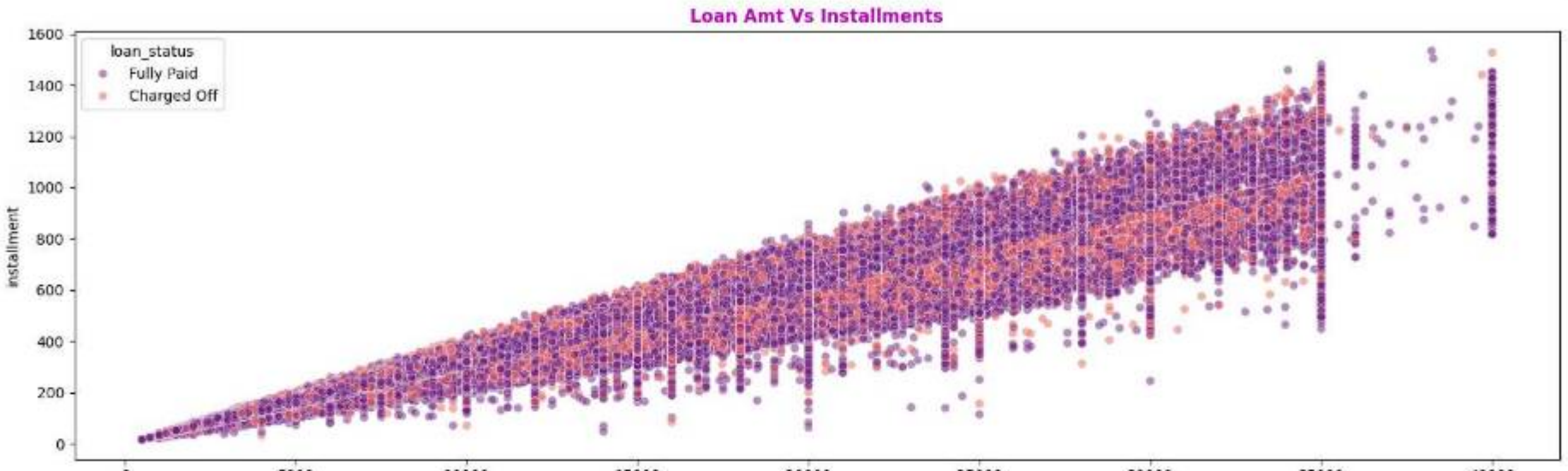
Q2. Comment about the correlation between Loan Amount and Installment features.

In [381…
```
data[['loan_amnt', 'installment']].corr()
```

Out[381…

	loan_amnt	installment
loan_amnt	1.000000	0.953929
installment	0.953929	1.000000

In [382…
```
plt.figure(figsize = (15,5))
sns.scatterplot(data = data, x = 'loan_amnt', y = 'installment', alpha = 0.5, hue = 'loan_status', palette = "magma")
plt.title('Loan Amt Vs Installments',fontsize=12,fontweight='bold',color='m')
plt.tight_layout()
plt.show()
```



Insights :

- Loan Terms: It's important to understand how loan amount and installment payments are related, as this helps lenders set suitable terms, like interest rates and repayment periods, based on a borrower's ability to repay.
- Multicollinearity Risk: High correlation between variables like loan amount and installment payments can cause multicollinearity, leading to unstable model estimates. Techniques like variable selection or regularization may be needed to address this issue.

Q3. The majority of people have which type of home ownership.

In [383…
```
round(data['home_ownership'].value_counts(normalize=True)*100,2)
```

Out[383…

home_ownership	proportion
MORTGAGE	50.08
RENT	40.35
OWN	9.53
OTHER	0.03
NONE	0.01
ANY	0.00

dtype: float64

Insights :

- Home Ownership Distribution: Mortgage holders make up about 50.08%, while renters account for 40.35%, highlighting a large portion of individuals who either own homes through mortgages or prefer renting over owning.

Q4. People with grades 'A' are more likely to fully pay their loan. (T/F)

In [384…
```
round(pd.crosstab(data['grade'],data['loan_status'], normalize = 'index')*100,2)
```

Out[384...

loan_status grade	Charged Off	Fully Paid
A	6.29	93.71
B	12.57	87.43
C	21.18	78.82
D	28.87	71.13
E	37.36	62.64
F	42.79	57.21
G	47.84	52.16

Loan Repayment Behavior: Borrowers with an 'A' grade have a very high repayment rate, with around 93.71% of their loans fully paid, indicating that those with the best credit ratings are more likely to meet their loan obligations.

So, Answer is True.

Q5. Name the top 2 afforded job titles.

In [385...
```
data["emp_title"].value_counts().head()
```

Out[385...

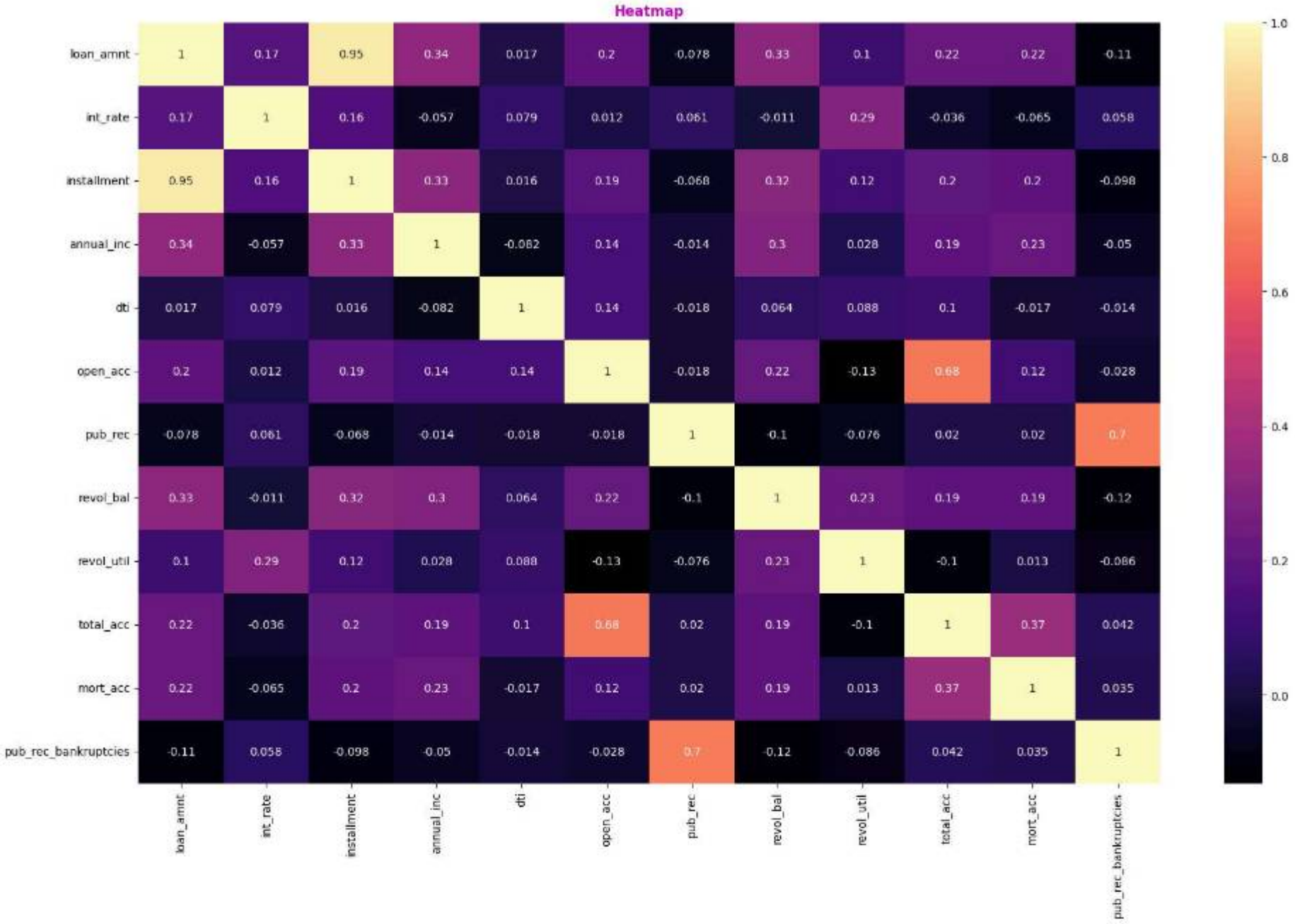
emp_title	count
Not Available	22927
Teacher	4389
Manager	4250
Registered Nurse	1856
RN	1846

dtype: int64

Top 2 most affoarded job titles are Teachers and Managers

HeatMap

In [386...
```
plt.figure(figsize=(20,12))
sns.heatmap(data.corr(numeric_only = True), annot=True, cmap='magma')
plt.title('Heatmap',fontsize=12,fontweight='bold',color='m')
plt.show()
```



Insights :

- Loan Amount and Installment: A strong correlation exists between loan amount and installment, suggesting that larger loans are associated with higher installment payments.
- Credit History: Total accounts (total_acc) and open accounts (open_acc) show a significant correlation, as does the relationship between public records of bankruptcies (pub_rec_bankruptcies) and public records (pub_rec).

Feature Engineering

In [387…
```
data['pub_rec'] = [1 if i > 1 else 0 for i in data['pub_rec']]
data['mort_acc'] = [1 if i > 1 else 0 for i in data['mort_acc']]
data['pub_rec_bankruptcies'] = [1 if i > 1 else 0 for i in data['pub_rec_bankruptcies']]
```

In [388…
```
data.head(3)
```

Out[388…

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	verification_status	i
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10	RENT	117000.0	Not Verified	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4	MORTGAGE	65000.0	Not Verified	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	1	RENT	43057.0	Source Verified	

Outlier Treatment

In [389…
```
for i in num_col2:
  Q1 = np.quantile(data[i],0.25)
  Q3 = np.quantile(data[i],0.75)
  IQR = Q3 - Q1
  upper_bound = Q3 + (1.5*IQR)
  lower_bound = Q1 - (1.5*IQR)
```

```
outliers = data[(data[i]< lower_bound)| (data[i]>upper_bound)]
print('Column Name :', i)
print(f'25th Percentile Value : {Q1} , 75th Percentile Value : {Q3} ')
print(f'IQR Value : {IQR}')
print(f'Lower Bound : {round(lower_bound,2)} , Upper Bound : {round(upper_bound,2)}')
print(f"Number of Outliers : {len(outliers)}")
print(f"Percentage Of Outliers : {(len(outliers)/len(data[i]))*100} \n")
print("-"*80)
```

```
Column Name : loan_amnt
25th Percentile Value : 8000.0 , 75th Percentile Value : 20000.0
IQR Value : 12000.0
Lower Bound : -10000.0 , Upper Bound : 38000.0
Number of Outliers : 191
Percentage Of Outliers : 0.04822866954523647

--------------------------------------------------------------------------------
Column Name : int_rate
25th Percentile Value : 10.49 , 75th Percentile Value : 16.49
IQR Value : 5.999999999999998
Lower Bound : 1.49 , Upper Bound : 25.49
Number of Outliers : 3777
Percentage Of Outliers : 0.9537156276039694

--------------------------------------------------------------------------------
Column Name : installment
25th Percentile Value : 250.33 , 75th Percentile Value : 567.3
IQR Value : 316.9699999999999
Lower Bound : -225.12 , Upper Bound : 1042.75
Number of Outliers : 11250
Percentage Of Outliers : 2.8406938868267555

--------------------------------------------------------------------------------
Column Name : annual_inc
25th Percentile Value : 45000.0 , 75th Percentile Value : 90000.0
IQR Value : 45000.0
Lower Bound : -22500.0 , Upper Bound : 157500.0
Number of Outliers : 16700
Percentage Of Outliers : 4.216852258667273

--------------------------------------------------------------------------------
Column Name : dti
25th Percentile Value : 11.28 , 75th Percentile Value : 22.98
IQR Value : 11.700000000000001
Lower Bound : -6.27 , Upper Bound : 40.53
Number of Outliers : 275
Percentage Of Outliers : 0.06943918390020958

--------------------------------------------------------------------------------
Column Name : open_acc
25th Percentile Value : 8.0 , 75th Percentile Value : 14.0
IQR Value : 6.0
Lower Bound : -1.0 , Upper Bound : 23.0
Number of Outliers : 10307
Percentage Of Outliers : 2.602580612579855

--------------------------------------------------------------------------------
Column Name : pub_rec
25th Percentile Value : 0.0 , 75th Percentile Value : 0.0
IQR Value : 0.0
Lower Bound : 0.0 , Upper Bound : 0.0
Number of Outliers : 8019
Percentage Of Outliers : 2.0248466025301113

--------------------------------------------------------------------------------
Column Name : revol_bal
25th Percentile Value : 6025.0 , 75th Percentile Value : 19620.0
IQR Value : 13595.0
Lower Bound : -14367.5 , Upper Bound : 40012.5
Number of Outliers : 21259
Percentage Of Outliers : 5.368027674671111

--------------------------------------------------------------------------------
Column Name : revol_util
25th Percentile Value : 35.9 , 75th Percentile Value : 72.9
IQR Value : 37.00000000000001
Lower Bound : -19.6 , Upper Bound : 128.4
Number of Outliers : 12
Percentage Of Outliers : 0.0030300734792818723

--------------------------------------------------------------------------------
Column Name : total_acc
25th Percentile Value : 17.0 , 75th Percentile Value : 32.0
IQR Value : 15.0
Lower Bound : -5.5 , Upper Bound : 54.5
Number of Outliers : 8499
Percentage Of Outliers : 2.1460495417013865

--------------------------------------------------------------------------------
Column Name : mort_acc
25th Percentile Value : 0.0 , 75th Percentile Value : 1.0
IQR Value : 1.0
Lower Bound : -1.5 , Upper Bound : 2.5
Number of Outliers : 0
Percentage Of Outliers : 0.0
```

```
--------------------------------------------------------------------------------
Column Name : pub_rec_bankruptcies
25th Percentile Value : 0.0 , 75th Percentile Value : 0.0
IQR Value : 0.0
Lower Bound : 0.0 , Upper Bound : 0.0
Number of Outliers : 2325
Percentage Of Outliers : 0.5870767366108629

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

Clipping the outlier

In [390…

```
# After clipping we can see the results
for i in num_col2:
  q1 =  data[i].quantile(0.25)
  q3 =  data[i].quantile(0.75)
  iqr = q3 - q1
  lower_bound = q1 - 1.5 * iqr
  upper_bound = q3 + 1.5 * iqr
  data[i] = np.where(data[i] < lower_bound,lower_bound,data[i])
  data[i] = np.where(data[i] > upper_bound,upper_bound,data[i])
  outliers =data[(data[i] < lower_bound) | ( data[i] > upper_bound)]
  print(f"No.of outliers in col {i}: ",len(outliers))
  print("-"*50)
```

```
No.of outliers in col loan_amnt:  0
--------------------------------------------------
No.of outliers in col int_rate:  0
--------------------------------------------------
No.of outliers in col installment:  0
--------------------------------------------------
No.of outliers in col annual_inc:  0
--------------------------------------------------
No.of outliers in col dti:  0
--------------------------------------------------
No.of outliers in col open_acc:  0
--------------------------------------------------
No.of outliers in col pub_rec:  0
--------------------------------------------------
No.of outliers in col revol_bal:  0
--------------------------------------------------
No.of outliers in col revol_util:  0
--------------------------------------------------
No.of outliers in col total_acc:  0
--------------------------------------------------
No.of outliers in col mort_acc:  0
--------------------------------------------------
No.of outliers in col pub_rec_bankruptcies:  0
--------------------------------------------------
```

Encoding

In [391…

```
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})
```

In [392…

```
data[["loan_status","initial_list_status"]].head()
```

Out[392…

	loan_status	initial_list_status
0	1	0
1	1	1
2	1	1
3	1	1
4	0	1

Dropping Unwanted Columns

In [393…

```
data2 = data.drop(columns=['emp_title','title','sub_grade','initial_list_status',
                           'state','pub_rec','pub_rec_bankruptcies','address','issued_d_month','earliest_cr_line','issue_d'],axis
```

In [394…

```
data2.sample(3)
```

Out[394…

	loan_amnt	term	int_rate	installment	grade	emp_length	home_ownership	annual_inc	verification_status	loan_status	
385466	19200.0	60 months	14.16	448.35	C	8	RENT	80000.0	Source Verified	1	debt_
368745	18000.0	60 months	24.89	527.17	F	1	RENT	60000.0	Verified	0	debt_
222324	20000.0	60 months	13.99	465.27	C	10	RENT	55320.0	Not Verified	1	ma

In [395…
```
data2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 19 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   emp_length           396030 non-null  object 
 6   home_ownership       396030 non-null  object 
 7   annual_inc           396030 non-null  float64
 8   verification_status  396030 non-null  object 
 9   loan_status          396030 non-null  int64  
 10  purpose              396030 non-null  object 
 11  dti                  396030 non-null  float64
 12  open_acc             396030 non-null  float64
 13  revol_bal            396030 non-null  float64
 14  revol_util           396030 non-null  float64
 15  total_acc            396030 non-null  float64
 16  application_type     396030 non-null  object 
 17  mort_acc             396030 non-null  float64
 18  pincode              396030 non-null  object 
dtypes: float64(10), int64(1), object(8)
memory usage: 57.4+ MB
```

In [396…
```
data2.shape
```

Out[396… (396030, 19)

In [397…
```
data2["term"] = data2["term"].str.split().str[0].astype(int)
```

In [398…
```
data2["term"]
```

Out[398…

	term
0	36
1	36
2	36
3	36
4	60
...	...
396025	60
396026	36
396027	36
396028	60
396029	36

396030 rows × 1 columns

dtype: int64

ONE HOT ENCODING

In [399…
```
dummies=['pincode', 'grade','purpose','home_ownership','verification_status','application_type']
data2_ohe = pd.get_dummies(data2, columns=dummies, drop_first=True)*1
```

In [400…
```
data2_ohe.shape
```

Out[400… (396030, 50)

In [401…
```
data2_ohe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 50 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   loan_amnt                             396030 non-null  float64
 1   term                                  396030 non-null  int64  
 2   int_rate                              396030 non-null  float64
 3   installment                           396030 non-null  float64
 4   emp_length                            396030 non-null  object 
 5   annual_inc                            396030 non-null  float64
 6   loan_status                           396030 non-null  int64  
 7   dti                                   396030 non-null  float64
 8   open_acc                              396030 non-null  float64
 9   revol_bal                             396030 non-null  float64
 10  revol_util                            396030 non-null  float64
 11  total_acc                             396030 non-null  float64
 12  mort_acc                              396030 non-null  float64
 13  pincode_05113                         396030 non-null  int64  
 14  pincode_11650                         396030 non-null  int64  
 15  pincode_22690                         396030 non-null  int64  
 16  pincode_29597                         396030 non-null  int64  
 17  pincode_30723                         396030 non-null  int64  
 18  pincode_48052                         396030 non-null  int64  
 19  pincode_70466                         396030 non-null  int64  
 20  pincode_86630                         396030 non-null  int64  
 21  pincode_93700                         396030 non-null  int64  
 22  grade_B                               396030 non-null  int64  
 23  grade_C                               396030 non-null  int64  
 24  grade_D                               396030 non-null  int64  
 25  grade_E                               396030 non-null  int64  
 26  grade_F                               396030 non-null  int64  
 27  grade_G                               396030 non-null  int64  
 28  purpose_credit_card                   396030 non-null  int64  
 29  purpose_debt_consolidation            396030 non-null  int64  
 30  purpose_educational                   396030 non-null  int64  
 31  purpose_home_improvement              396030 non-null  int64  
 32  purpose_house                         396030 non-null  int64  
 33  purpose_major_purchase                396030 non-null  int64  
 34  purpose_medical                       396030 non-null  int64  
 35  purpose_moving                        396030 non-null  int64  
 36  purpose_other                         396030 non-null  int64  
 37  purpose_renewable_energy              396030 non-null  int64  
 38  purpose_small_business                396030 non-null  int64  
 39  purpose_vacation                      396030 non-null  int64  
 40  purpose_wedding                       396030 non-null  int64  
 41  home_ownership_MORTGAGE               396030 non-null  int64  
 42  home_ownership_NONE                   396030 non-null  int64  
 43  home_ownership_OTHER                  396030 non-null  int64  
 44  home_ownership_OWN                    396030 non-null  int64  
 45  home_ownership_RENT                   396030 non-null  int64  
 46  verification_status_Source Verified   396030 non-null  int64  
 47  verification_status_Verified          396030 non-null  int64  
 48  application_type_INDIVIDUAL           396030 non-null  int64  
 49  application_type_JOINT                396030 non-null  int64  
dtypes: float64(10), int64(39), object(1)
memory usage: 151.1+ MB
```

In [402…
```
data2_ohe.sample(3)
```

Out[402…

	loan_amnt	term	int_rate	installment	emp_length	annual_inc	loan_status	dti	open_acc	revol_bal	revol_util	total_acc	mo
17403	5000.0	36	11.22	164.22	10	26000.0	1	21.61	10.0	5415.0	30.4	22.0	
393642	15000.0	36	11.99	498.15	10	99000.0	1	11.99	8.0	7704.0	50.4	13.0	
248764	15000.0	60	18.99	389.03	3	101000.0	1	26.51	17.0	32471.0	78.6	29.0	

Model Creation

In [403…
```
X = data2_ohe.drop(['loan_status'], axis=1)
y = data2_ohe['loan_status']
```

In [404…
```
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)
```

```
(316824, 49)
(79206, 49)
(316824,)
(79206,)
```

Scaling The data

In [405…
```
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)
```

In [406…
```
X_train.head()
```

Out[406…

	loan_amnt	term	int_rate	installment	emp_length	annual_inc	dti	open_acc	revol_bal	revol_util	total_acc	mort_acc	pincod
0	0.480000	1.0	0.264254	0.315738	1.000000	0.251115	0.649889	0.347826	0.000200	0.000779	0.476190	0.0	
1	0.338000	0.0	0.556767	0.438990	0.333333	0.187380	0.055761	0.260870	0.026142	0.123053	0.114286	0.0	
2	0.920000	1.0	0.621715	0.847425	1.000000	0.506055	0.848261	0.521739	0.505817	0.447819	0.647619	1.0	
3	0.530667	0.0	0.337134	0.645453	1.000000	0.410453	0.525537	0.782609	0.317826	0.384735	0.552381	0.0	
4	0.920000	1.0	0.607338	0.842068	1.000000	1.000000	0.125339	0.217391	0.364186	0.491433	0.114286	0.0	

In [407…
```
#Fit the Model on training data
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
```

Out[407…

▾ LogisticRegression

LogisticRegression()

In [408…
```
y_train_pred = lr_model.predict(X_train)
y_test_pred = lr_model.predict(X_test)
```

In [409…
```
lr_model.score(X_train,y_train)
```

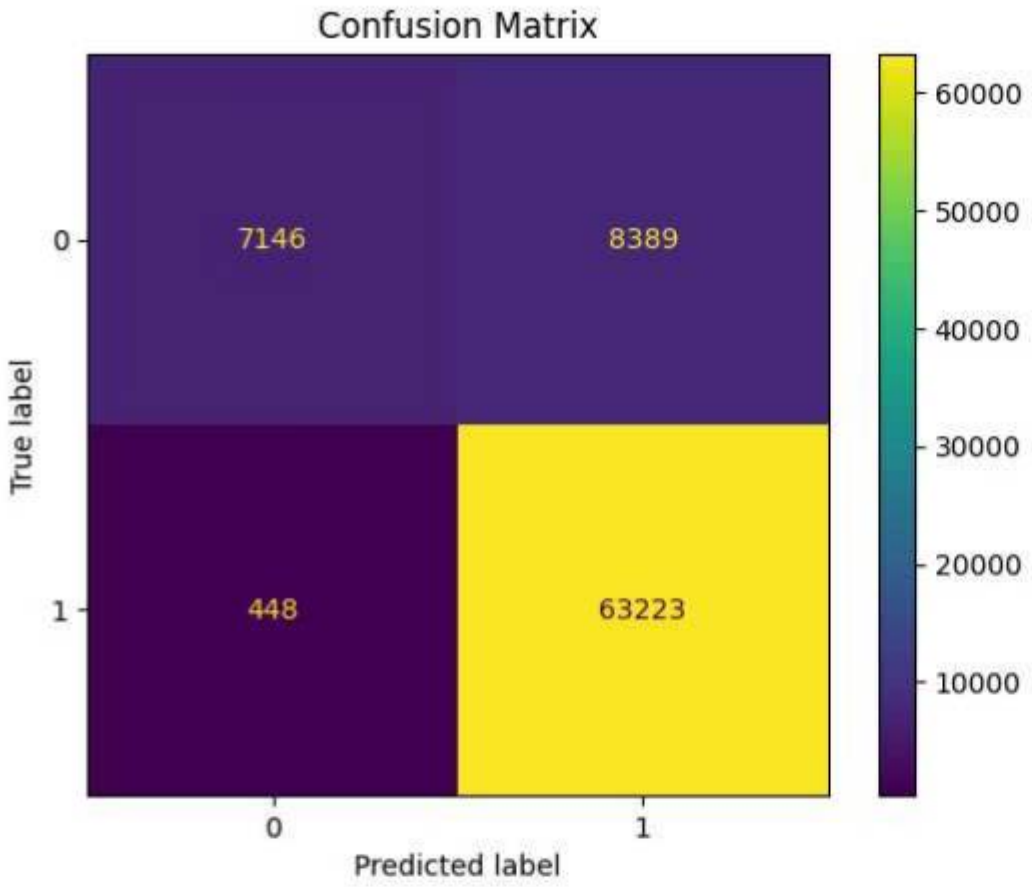
Out[409… 0.8893675984142615

In [410…
```
lr_model.score(X_test,y_test)
```

Out[410… 0.8884301694316087

In [411…
```
print('Train Accuracy :', round(lr_model.score(X_train, y_train),2))
print('Train F1 Score:',round(f1_score(y_train,y_train_pred),2))
print('Train Recall Score:',round(recall_score(y_train,y_train_pred),2))
print('Train Precision Score:',round(precision_score(y_train,y_train_pred),2))

print('\nTest Accuracy :',round(lr_model.score(X_test,y_test),2))
print('Test F1 Score:',round(f1_score(y_test,y_test_pred),2))
print('Test Recall Score:',round(recall_score(y_test,y_test_pred),2))
print('Test Precision Score:',round(precision_score(y_test,y_test_pred),2))

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

```
Train Accuracy : 0.89
Train F1 Score: 0.94
Train Recall Score: 0.99
Train Precision Score: 0.88

Test Accuracy : 0.89
Test F1 Score: 0.93
Test Recall Score: 0.99
Test Precision Score: 0.88
```



In [412…
```
print(classification_report(y_test,y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.46      0.62     15535
           1       0.88      0.99      0.93     63671

    accuracy                           0.89     79206
   macro avg       0.91      0.73      0.78     79206
weighted avg       0.89      0.89      0.87     79206
```

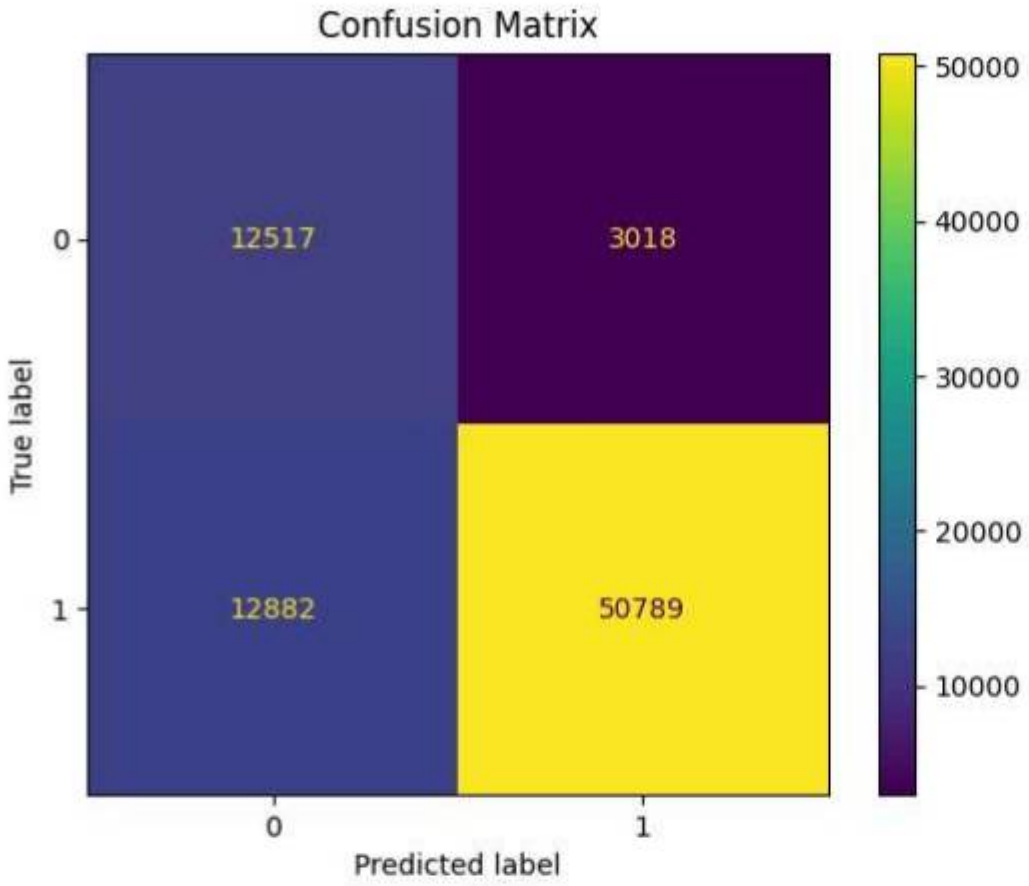
Oversampling to balance the target variable

In [413…
```
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: 254686
Before OverSampling, count of label 0: 62138
After OverSampling, count of label 1: 254686
After OverSampling, count of label 0: 254686
```

In [414…
```
model = LogisticRegression()
model.fit(X_train_res, y_train_res)
train_preds = model.predict(X_train)
test_preds = model.predict(X_test)

#Model Evaluation
print('Train Accuracy :', round(model.score(X_train, y_train),2))
print('Train F1 Score:',round(f1_score(y_train,y_train_pred),2))
print('Train Recall Score:',round(recall_score(y_train,y_train_pred),2))
print('Train Precision Score:',round(precision_score(y_train,y_train_pred),2))

print('\nTest Accuracy :',round(model.score(X_test,y_test),2))
print('Test F1 Score:',round(f1_score(y_test,y_test_pred),2))
print('Test Recall Score:',round(recall_score(y_test,y_test_pred),2))
print('Test Precision Score:',round(precision_score(y_test,y_test_pred),2))

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

```
Train Accuracy : 0.8
Train F1 Score: 0.94
Train Recall Score: 0.99
Train Precision Score: 0.88

Test Accuracy : 0.8
Test F1 Score: 0.93
Test Recall Score: 0.99
Test Precision Score: 0.88
```



In [415…
```
y_pred = test_preds
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.49      0.81      0.61     15535
           1       0.94      0.80      0.86     63671

    accuracy                           0.80     79206
   macro avg       0.72      0.80      0.74     79206
weighted avg       0.86      0.80      0.82     79206
```

Insights :

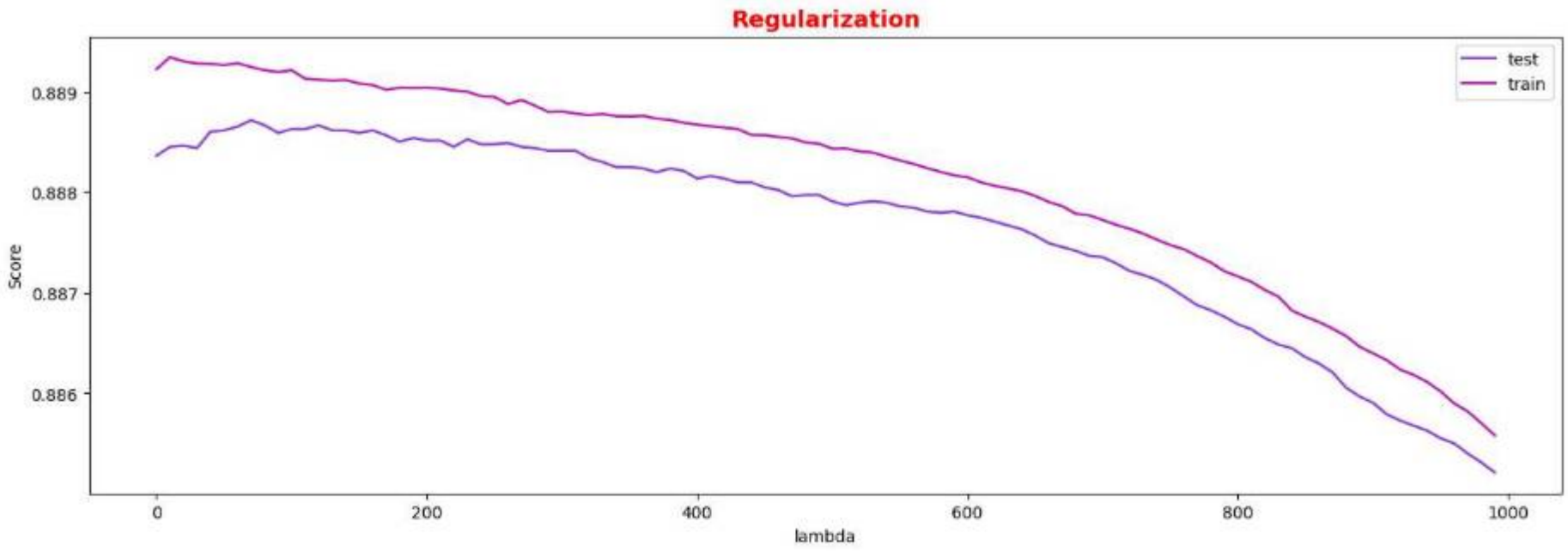
- The model achieves a high recall score, correctly identifying 80% of actual defaulters.
- However, the precision for predicting defaulters is low, as only 47% of predicted defaulters are true defaulters.
- The strong recall combined with low precision results in many false positives, meaning non-defaulters are incorrectly flagged as defaulters.
- As a result, some deserving customers may be unjustly denied loans due to the model's inability to accurately identify defaulters.
- Despite an 80% overall accuracy, the low precision negatively impacts the F1 score, bringing it down to 60%, highlighting the trade-off between precision and recall in the model's performance.

Regularizaton Model

In [416…
```
#Try with different regularization factor lamda and choose the best to build the model

lmd = np.arange(0.01, 1000, 10)

train_scores = []
test_scores = []

for i in lmd:
    model = LogisticRegression(C = 1/i)
    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)
```

In [417…
```
#Ploting the train and test scores with respect lambda values
rand = np.arange(0.01, 1000, 10)
plt.figure(figsize=(16,5))
```

```
sns.lineplot(x=rand,y=test_scores,color='blueviolet',label='test')
sns.lineplot(x=rand,y=train_scores,color='m',label='train')
plt.title('Regularization',fontsize=14,fontweight='bold',color='r')
plt.xlabel("lambda")
plt.ylabel("Score")
plt.show()
```



In [418…
```
print(np.argmax(test_scores))
print(test_scores[np.argmax(test_scores)])
```

```
7
0.8887205514733733
```

In [419…
```
#Calculate the best lambda value based on the index of best test score

best_lamb = 0.01 + (10*7)
best_lamb
```

Out[419…
```
70.01
```

In [420…
```
#Fit the model using best lambda

reg_model = LogisticRegression(C=1/best_lamb)
reg_model.fit(X_train, y_train)
```

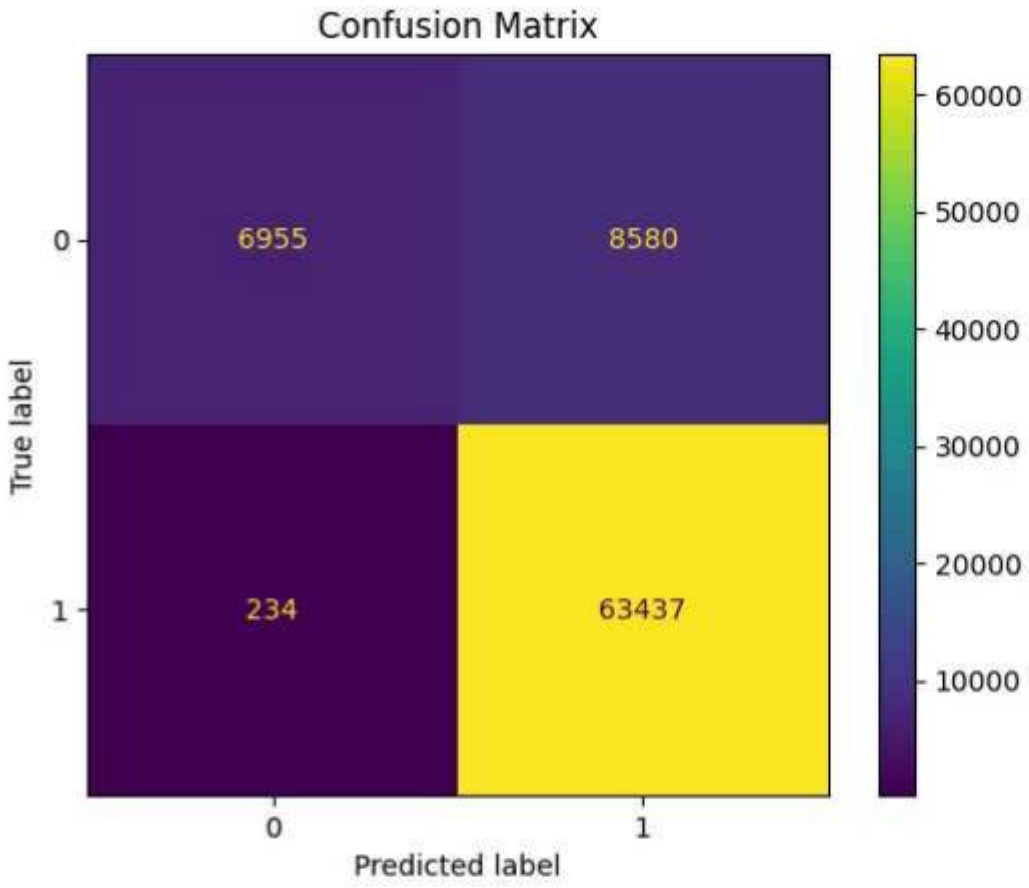
Out[420…
```
LogisticRegression
LogisticRegression(C=0.0142836737608913)
```

In [421…
```
#Predict the y_values and y_probability values

y_reg_pred = reg_model.predict(X_test)
y_reg_pred_proba = reg_model.predict_proba(X_test)
```

In [422…
```
#Print model score

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: 88.52 %
```

In [423…
```
# Confusion Matrix
cm = confusion_matrix(y_test, y_reg_pred)
disp = ConfusionMatrixDisplay(cm)
disp.plot()
plt.title('Confusion Matrix')
plt.show()
```



In [424…
```
print(classification_report(y_test, y_reg_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.45      0.61     15535
           1       0.88      1.00      0.94     63671

    accuracy                           0.89     79206
   macro avg       0.92      0.72      0.77     79206
weighted avg       0.90      0.89      0.87     79206
```

K-fold - Cross Validation

In [425…
```
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 : 0.889
```

In [426…
```
cm = confusion_matrix(y_test, y_reg_pred)
cm_df = pd.DataFrame(cm, index=['Charged Off','Fully paid'], columns=['Charged Off','Fully paid'])
cm_df
```

Out[426…

	Charged Off	Fully paid
Charged Off	6955	8580
Fully paid	234	63437

- TN = 5223 (True Negative: Correctly predicted Charged Off)
- TP = 50450 (True Positive: Correctly predicted Fully Paid)
- FP = 6455 (False Positive: Predicted Fully Paid but actually Charged Off)
- FN = 151 (False Negative: Predicted Charged Off but actually Fully Paid)

In [426…

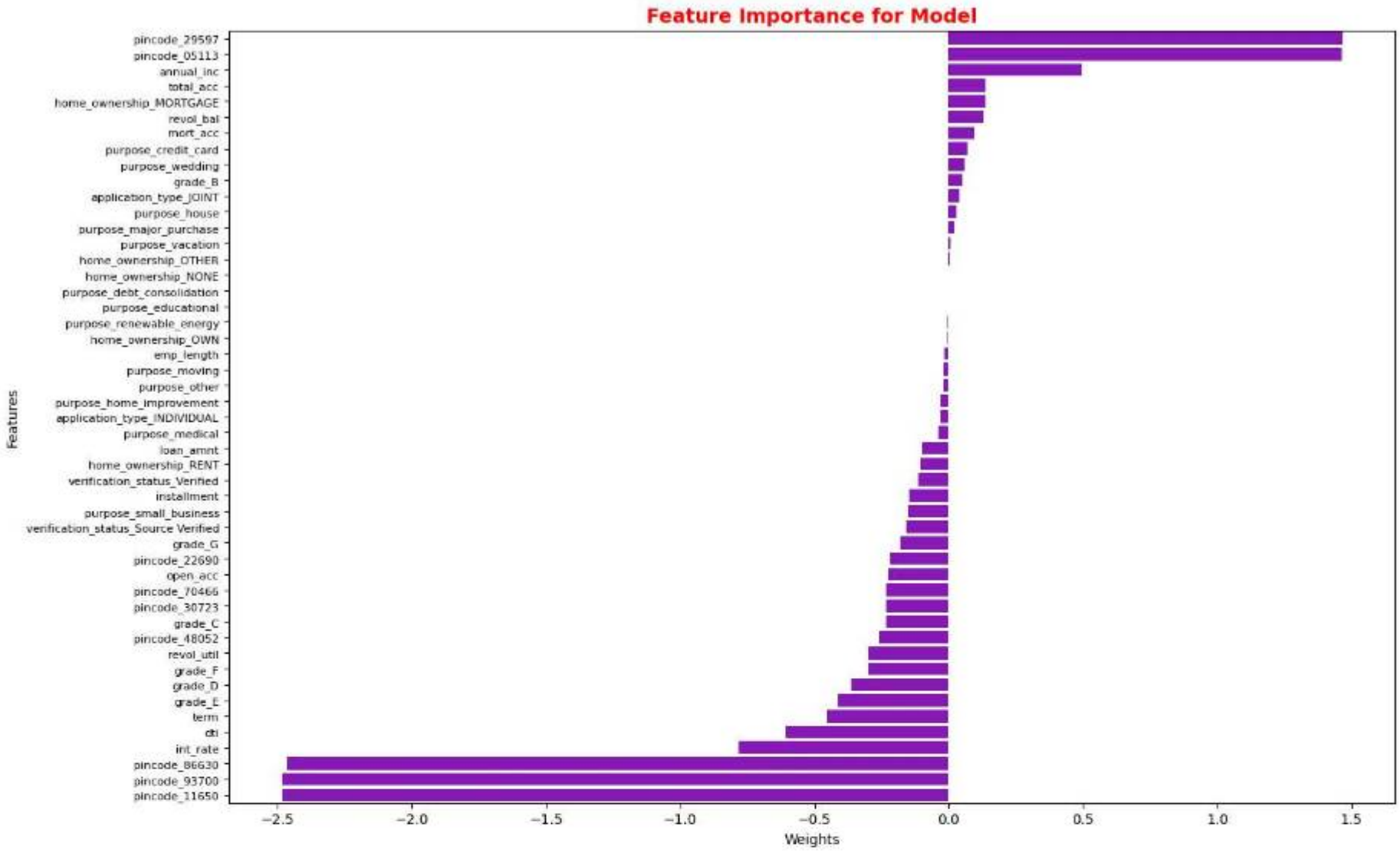
In [427…
```
#Collect the model coefficients and print those in dataframe format
coeff_df = pd.DataFrame()
coeff_df['Features'] = X_train_res.columns
coeff_df['Weights'] = model.coef_[0]
coeff_df['ABS_Weights'] = abs(coeff_df['Weights'])
coeff_df = coeff_df.sort_values(['ABS_Weights'], ascending=False)
coeff_df
```

Out[427...

	Features	Weights	ABS_Weights
13	pincode_11650	-2.483419	2.483419
20	pincode_93700	-2.482956	2.482956
19	pincode_86630	-2.464663	2.464663
15	pincode_29597	1.465068	1.465068
12	pincode_05113	1.461707	1.461707
2	int_rate	-0.783209	0.783209
6	dti	-0.608594	0.608594
5	annual_inc	0.493648	0.493648
1	term	-0.455549	0.455549
24	grade_E	-0.411222	0.411222
23	grade_D	-0.363713	0.363713
25	grade_F	-0.300042	0.300042
9	revol_util	-0.299706	0.299706
17	pincode_48052	-0.259785	0.259785
22	grade_C	-0.234915	0.234915
16	pincode_30723	-0.234624	0.234624
18	pincode_70466	-0.233330	0.233330
7	open_acc	-0.224575	0.224575
14	pincode_22690	-0.218044	0.218044
26	grade_G	-0.181749	0.181749
45	verification_status_Source Verified	-0.158471	0.158471
37	purpose_small_business	-0.150111	0.150111
3	installment	-0.148850	0.148850
10	total_acc	0.137117	0.137117
40	home_ownership_MORTGAGE	0.135933	0.135933
8	revol_bal	0.129773	0.129773
46	verification_status_Verified	-0.113046	0.113046
44	home_ownership_RENT	-0.107661	0.107661
0	loan_amnt	-0.098242	0.098242
11	mort_acc	0.096346	0.096346
27	purpose_credit_card	0.069839	0.069839
39	purpose_wedding	0.058896	0.058896
21	grade_B	0.049842	0.049842
48	application_type_JOINT	0.040907	0.040907
33	purpose_medical	-0.037336	0.037336
47	application_type_INDIVIDUAL	-0.030365	0.030365
30	purpose_home_improvement	-0.030140	0.030140
31	purpose_house	0.027106	0.027106
35	purpose_other	-0.021514	0.021514
34	purpose_moving	-0.020829	0.020829
32	purpose_major_purchase	0.019422	0.019422
4	emp_length	-0.017619	0.017619
43	home_ownership_OWN	-0.006945	0.006945
36	purpose_renewable_energy	-0.005839	0.005839
38	purpose_vacation	0.004816	0.004816
29	purpose_educational	-0.002889	0.002889
28	purpose_debt_consolidation	-0.002264	0.002264
42	home_ownership_OTHER	0.000998	0.000998

	Features	Weights	ABS_Weights
41	home_ownership_NONE	0.000067	0.000067

In [428…
```
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'],color='darkviolet')
plt.title("Feature Importance for Model",fontsize=14,fontweight='bold',color='r')
plt.xlabel("Weights")
plt.yticks(fontsize=8)
plt.ylabel("Features")
plt.show()
```



In [429…
```
#Logistic Regression model intercept
model.intercept_
```

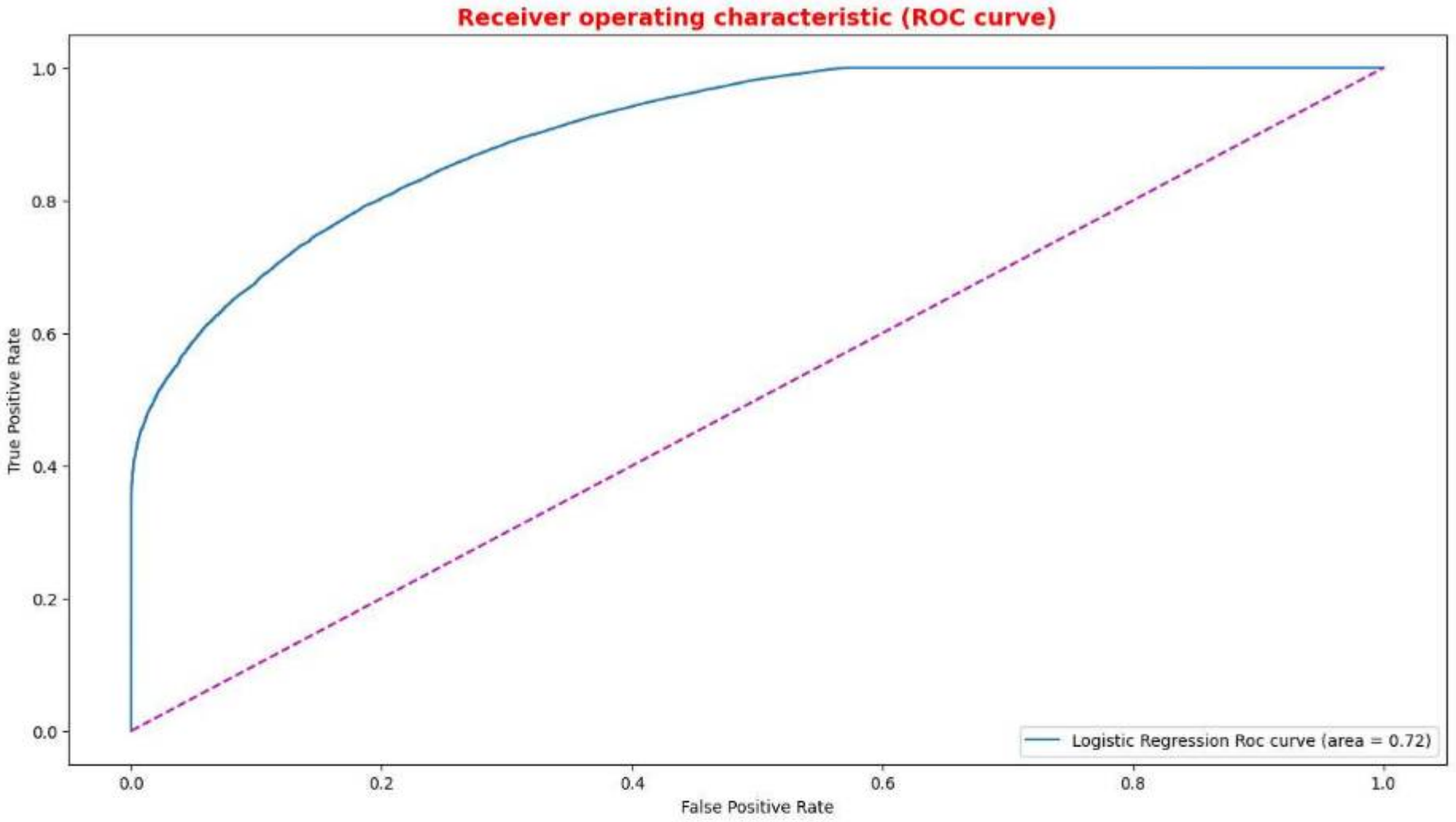
Out[429…
```
array([2.68665385])
```

Insights :

- The model places significant importance on features like zip code, annual income, and grade, indicating that certain zip codes play a strong role in predicting defaulters.
- Features such as debt-to-income ratio (dti), number of open accounts (open_acc), and loan amount (loan_amnt) also have high positive coefficients, while some zip codes with negative coefficients are linked to a lower likelihood of default.

ROC AUC Curve

In [430…
```
# area under ROC curve
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)

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],'m--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic (ROC curve)',fontsize=14,fontweight='bold',color='r')
plt.legend(loc="lower right")
plt.show()
```



In [431…
```
logit_roc_auc
```

Out[431…
```
0.7220118011201542
```

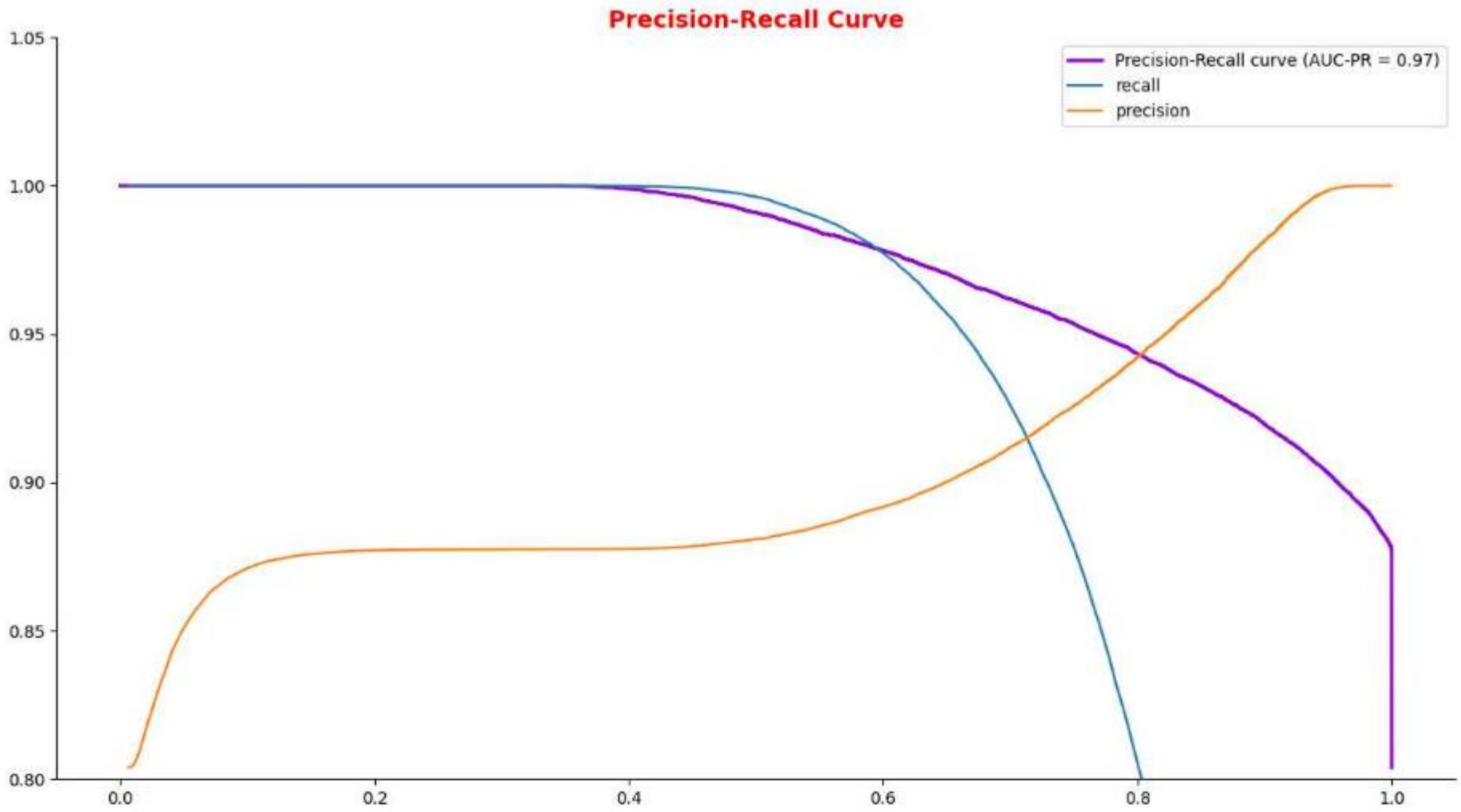
In [432…
```
roc_auc = auc(fpr, tpr)
roc_auc
```

Out[432…
```
0.9040218086420752
```

Insights :

- The model's ROC curve area is 72%, indicating it can distinguish between classes correctly 72% of the time.
- To improve predictions, it's important to increase the True Positive Rate (TPR) while minimizing the False Positive Rate (FPR).
- The trade-off between TPR and FPR suggests that while the model identifies more Fully Paid customers, it also risks misclassifying Charged Off customers as Fully Paid, potentially leading to Non-Performing Assets (NPAs).
- To enhance the model, reducing the FPR while keeping a high TPR is essential, which will improve the AUC and overall performance.

In [433…
```
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, color='darkviolet', lw=2, label=f'Precision-Recall curve (AUC-PR = {average_precision:.2f})')
plt.plot(thresholds, recall[0:thresholds.shape[0]], label='recall')
plt.plot(thresholds, precision[0:thresholds.shape[0]], label='precision')
plt.ylim([0.8, 1.05])
plt.title('Precision-Recall Curve',fontsize=14,fontweight='bold',color='r')
plt.legend(loc='upper right')
sns.despine()
plt.show()
```




```
In [434…  auc(recall, precision).round(2)

Out[434…  0.97
```

Insights :

- The model's precision-recall curve has an AUC of 0.97, demonstrating excellent performance in distinguishing between positive and negative classes.
- Precision-recall curves are particularly useful in imbalanced datasets, focusing on the accurate identification of the relevant class, in this case, Fully Paid customers.
- Precision and recall calculations exclude true negatives, allowing for a more focused evaluation on correctly predicting Fully Paid customers.
- The high AUC value (97%) highlights the model's effectiveness in differentiating between classes, and the key area for improvement is increasing precision by reducing false positives.

Q6: Thinking from a bank's perspective, which metric should our primary focus be on..

- ROC AUC
- Precision
- Recall
- F1 Score

Answer:

From a bank's standpoint, reducing risks and boosting profitability are top priorities. The ROC AUC (Receiver Operating Characteristic Area Under Curve) is an essential metric as it reflects both the True Positive Rate (TPR) and False Positive Rate (FPR).

Optimizing ROC AUC helps the bank balance identifying creditworthy customers while minimizing defaulter risk, enhancing the credit model's performance.

Q7. How does the gap in precision and recall affect the bank?

Answer

Evaluating false positives and false negatives through recall and precision is key to understanding model errors, as a low recall poses a risk for the bank.

A larger gap between precision and recall increases incorrect predictions; good precision reduces NPAs, while good recall ensures valuable customers are not missed.

Q8. Which were the features that heavily affected the outcome?

Answer

Pincode, Annual Income, and Grade are the most significant features, while Loan term, total credit balance (revol_bal), debt-to-income ratio (dti), and interest rate (int_rate) also have high weights in the model.

Q9. Will the results be affected by geographical location? (Yes/No)

Answer

Yes, pincode (Address) is a key feature, indicating that geographical location influences our results.

Recommendation

- To optimize loan approval and risk management, focus on maximizing the F1 score and the area under the Precision-Recall Curve, effectively balancing precision and recall to identify defaulters while minimizing false positives. This will improve the accuracy of the model and enhance risk management.
- Implement dynamic risk thresholds based on loan purpose, sub-grade analysis, and geographical data, as regional economic differences may impact the results. Additionally, increase verification rigor for applicants with lower grades, higher loan amounts, and high debt-to-income ratios (dti). Loans with lower grades should be scrutinized more carefully, and interest rates can be adjusted to compensate for higher risks.
- Consider using more advanced classifiers like Random Forests or XGBoost and fine-tuning hyperparameters to capture complex relationships within the data. Employ stratified k-fold cross-validation to ensure reliable estimates of model performance and account for minority class distribution.
- Targeted strategies can be applied for high-risk zip codes, such as additional verification steps or higher interest rates. Additionally, evaluate small business loans with extra financial health checks and collateral requirements to further mitigate default risks.

In [434…