BusinessCase : LoanTap - Logistic Regression

OverView of LoanTap:

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

Data Dictionary:

```
loan_amnt : The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
term : The number of payments on the loan. Values are in months and can be either 36 or 60.
int_rate : Interest Rate on the loan
installment : The monthly payment owed by the borrower if the loan originates.
grade : LoanTap assigned loan grade
sub_grade : LoanTap assigned loan subgrade
emp_title :The job title supplied by the Borrower when applying for the loan.*
emp_length : Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
home_ownership : The home ownership status provided by the borrower during registration or obtained from the credit report.
annual_inc : The self-reported annual income provided by the borrower during registration.
verification_status : Indicates if income was verified by LoanTap, not verified, or if the income source was verified
issue_d : The month which the loan was funded
loan_status : Current status of the loan - Target Variable
purpose : A category provided by the borrower for the loan request.
title : The loan title provided by the borrower
dti : A ratio calculated using the borrower’s total monthly debt payments on the total debt obligations, excluding mortgage and the requested LoanTap loan, divided by the borrower’s self-reported monthly income.
```

```
earliest_cr_line :The month the borrower's earliest reported credit
line was opened
open_acc : The number of open credit lines in the borrower's credit
file.
pub_rec : Number of derogatory public records
revol_bal : Total credit revolving balance
revol_util : Revolving line utilization rate, or the amount of credit
the borrower is using relative to all available revolving credit.
total_acc : The total number of credit lines currently in the
borrower's credit file
initial_list_status : The initial listing status of the loan. Possible
values are – W, F
application_type : Indicates whether the loan is an individual
application or a joint application with two co-borrowers
mort_acc : Number of mortgage accounts.
pub_rec_bankruptcies : Number of public record bankruptcies
Address: Address of the individual
```

Problem Statement:

Given a set of attributes for an Individual, determine if a credit line should be extended to them. If so, what should the repayment terms be in business recommendations?

```
#Data processing
import numpy as np
import pandas as pd

#Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns

#Stats & model building
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import (accuracy_score, confusion_matrix,
                             roc_curve, auc, ConfusionMatrixDisplay,
                             f1_score, recall_score,
                             precision_score, precision_recall_curve,
                             average_precision_score,
classification_report)
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

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

Loading LoanTap data

```
#Loading LoanTap data
data = pd.read_csv('logistic_regression.csv')

data.head()
```

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

                 emp_title emp_length home_ownership  annual_inc  ...
\
0                Marketing  10+ years           RENT    117000.0  ...

1          Credit analyst     4 years       MORTGAGE     65000.0  ...

2             Statistician   < 1 year           RENT     43057.0  ...

3          Client Advocate    6 years           RENT     54000.0  ...

4  Destiny Management Inc.    9 years       MORTGAGE     55000.0  ...


  open_acc pub_rec revol_bal revol_util total_acc  initial_list_status
\
0     16.0     0.0   36369.0       41.8      25.0                    w

1     17.0     0.0   20131.0       53.3      27.0                    f

2     13.0     0.0   11987.0       92.2      26.0                    f

3      6.0     0.0    5472.0       21.5      13.0                    f

4     13.0     0.0   24584.0       69.8      43.0                    f


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

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

```
4             679 Luna Roads\r\nGreggshire, VA 11650

[5 rows x 27 columns]
```

```
data.shape
```

```
(396030, 27)
```

Insights: There are 3.96L datapoints with 27 features in the LoanTap dataset among which 'loan_status' is the Target column.

```
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  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 394275 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
```

Checking for null values and filling with their respective appropriates

```
data.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1755
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

Insigths:

```
There are null values observed in the data 6 columns.

- Out of which 3 are categorical columns and 3 are numerical columns.
- Null values of Categorical columns 'emp_title','emp_length','title'
can be replaced by their respective modes.
- Null values of Nunerical columns
'revol_util','mort_acc','pub_rec_bankruptcies' can be replaced by
their respective means.
```

```
#Filling missing values with 'Unknown' for object dtype
fill_values = {'title': 'Unknown', 'emp_title': 'Unknown'}
data.fillna(value=fill_values, inplace=True)

#Mean aggregation of mort_acc by total_acc to fill missing values
```

```
avg_mort = data.groupby('total_acc')['mort_acc'].mean()

def fill_mort(total_acc, mort_acc):
  if np.isnan(mort_acc):
    return avg_mort[total_acc].round()
  else:
    return mort_acc

data['mort_acc'] = data.apply(lambda x:
fill_mort(x['total_acc'],x['mort_acc']), axis=1)

data.dropna(inplace=True)

data.isna().sum().sum()

0
```

Insights: Data is free of null values and read to analyze.

Exploratory Data Analysis

Splitting features based on Categorical and Numerical values

```
cols = data.columns
print(cols)

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

col_dtypes = data.dtypes
col_dtypes = col_dtypes.reset_index()
col_dtypes

                   index        0
0              loan_amnt  float64
1                   term   object
2               int_rate  float64
3            installment  float64
4                  grade   object
5              sub_grade   object
6              emp_title   object
7             emp_length   object
8         home_ownership   object
```

```
9              annual_inc  float64
10   verification_status   object
11               issue_d   object
12           loan_status   object
13               purpose   object
14                 title   object
15                   dti  float64
16      earliest_cr_line   object
17              open_acc  float64
18               pub_rec  float64
19             revol_bal  float64
20            revol_util  float64
21             total_acc  float64
22   initial_list_status   object
23      application_type   object
24              mort_acc  float64
25  pub_rec_bankruptcies  float64
26               address   object
```

```
cat_cols=list(col_dtypes[col_dtypes[0]=='object']['index'])
num_cols= list(col_dtypes[col_dtypes[0]=='float']['index'])

print(f"{len(cat_cols)} Categorical Columns:\n\n",cat_cols)
```

```
15 Categorical Columns:

 ['term', 'grade', 'sub_grade', 'emp_title', 'emp_length',
'home_ownership', 'verification_status', 'issue_d', 'loan_status',
'purpose', 'title', 'earliest_cr_line', 'initial_list_status',
'application_type', 'address']
```

```
print(f"{len(num_cols)} Numerical Columns:\n\n",num_cols)
```

```
12 Numerical Columns:

 ['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti',
'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
'mort_acc', 'pub_rec_bankruptcies']
```

Value counts of each feature

```
data[num_cols].describe().transpose()
```

```
                        count           mean            std      min
25%  \
loan_amnt            376929.0  14264.626826   8374.474814   500.00
8000.00
int_rate             376929.0     13.637716      4.475179     5.32
10.49
installment          376929.0    435.874478    251.195910    16.08
254.12
```

```
annual_inc             376929.0  75397.440575  62240.311742  4000.00
46000.00
dti                    376929.0     17.296733      8.111768     0.00
11.27
open_acc               376929.0     11.374460      5.144134     1.00
8.00
pub_rec                376929.0      0.172603      0.524977     0.00
0.00
revol_bal              376929.0  15979.636046  20683.822466     0.00
6106.00
revol_util             376929.0     53.961622     24.420020     0.00
36.10
total_acc              376929.0     25.482693     11.884231     2.00
17.00
mort_acc               376929.0      1.779234      2.060396     0.00
0.00
pub_rec_bankruptcies   376929.0      0.117200      0.350677     0.00
0.00
```

```
                           50%       75%         max
loan_amnt                12000.00  20000.00    40000.00
int_rate                    13.33     16.49       30.99
installment                379.63    572.98     1533.81
annual_inc               65000.00  90000.00  8706582.00
dti                         16.85     22.89      380.53
open_acc                    11.00     14.00       90.00
pub_rec                      0.00      0.00       86.00
revol_bal                11303.00  19785.00  1743266.00
revol_util                  55.00     73.00      892.30
total_acc                   24.00     32.00      151.00
mort_acc                     1.00      3.00       34.00
pub_rec_bankruptcies         0.00      0.00        8.00
```

Insights:

```
- Maximum amount applied for loan is 4000 and minimum is 500.
- Maximum number of pu_rec_bankruptcies are 8 and minimum is 0.
- Maximum total number of credit lines currently in the borrower's
credit file(total_acc) are 151
  and minimum are 2.
- Maximum installment rate is 30.99 and minimum is 5.32.
- Maximum number of pub_rec(Number of derogatory public records) are
86 and minimum are 0.
- Maximum no of mortgage accounts are 34 and minimum are 0.
```

```
for col in cat_cols:
    print(f'"{col}" has {data[col].nunique()} unique values',':\n',
data[col].unique(),'\n','-'*25)
```

```
"term" has 2 unique values :
 [' 36 months' ' 60 months']
 -------------------------
"grade" has 7 unique values :
 ['B' 'A' 'C' 'E' 'D' 'F' 'G']
 -------------------------
"sub_grade" has 35 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']
 -------------------------
"emp_title" has 172567 unique values :
 ['Marketing' 'Credit analyst ' 'Statistician' ...
 "Michael's Arts & Crafts" 'licensed bankere' 'Gracon Services, Inc']
 -------------------------
"emp_length" has 11 unique values :
 ['10+ years' '4 years' '< 1 year' '6 years' '9 years' '2 years' '3
years'
 '8 years' '7 years' '5 years' '1 year']
 -------------------------
"home_ownership" has 6 unique values :
 ['RENT' 'MORTGAGE' 'OWN' 'OTHER' 'ANY' 'NONE']
 -------------------------
"verification_status" has 3 unique values :
 ['Not Verified' 'Source Verified' 'Verified']
 -------------------------
"issue_d" has 112 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' 'Feb-2014' 'Dec-2011' 'Mar-2013' 'Jun-2016'
 'Mar-2014' 'Nov-2013' 'Dec-2014' 'Sep-2013' 'May-2016' 'Jul-2015'
 'Jul-2013' 'Aug-2013' 'Aug-2014' 'May-2008' 'Mar-2010' 'Dec-2013'
 'Mar-2012' 'Mar-2015' 'Sep-2011' 'Jul-2012' 'Dec-2012' 'Sep-2014'
 'Nov-2012' 'Apr-2016' 'Nov-2015' 'Jan-2011' 'May-2012' 'Jun-2012'
 'Aug-2012' 'May-2015' 'Oct-2016' 'Aug-2015' 'Jul-2016' 'Feb-2016'
 'May-2009' 'Aug-2016' 'Jan-2012' 'Jan-2013' 'Nov-2010' 'Jul-2011'
 'Mar-2011' 'Feb-2012' 'May-2011' 'Aug-2010' 'Jan-2016' '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' 'Nov-2008' 'Dec-2009' 'Oct-2010' 'Sep-2009' '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'
 'Sep-2008' 'Jun-2008' 'Jul-2008' 'Mar-2008' 'Oct-2007' 'Dec-2007'
 'Feb-2008' 'Jan-2008' 'Nov-2007' 'Aug-2007']
 -------------------------
"loan_status" has 2 unique values :
 ['Fully Paid' 'Charged Off']
```

```
 -------------------------
"purpose" has 14 unique values :
 ['vacation' 'debt_consolidation' 'credit_card' 'home_improvement'
 'small_business' 'major_purchase' 'other' 'medical' 'wedding' 'car'
 'moving' 'house' 'educational' 'renewable_energy']
 -------------------------
"title" has 46766 unique values :
 ['Vacation' 'Debt consolidation' 'Credit card refinancing' ...
 'Credit buster ' 'Loanforpayoff' 'Toxic Debt Payoff']
 -------------------------
"earliest_cr_line" has 665 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' 'May-1997' 'Jul-2006' 'Sep-2003' 'Aug-1992' 'Dec-1988'
 'Feb-2002' 'Jan-1992' 'Aug-2001' 'Dec-2010' 'Oct-1999' 'Sep-2004'
 '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' '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'
 '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' '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' 'Jul-1985' 'May-1978' 'Sep-2010' 'Oct-1996' 'Sep-2009'
 'Jun-1999' 'Jan-2000' 'Sep-1987' 'Aug-1998' 'Jan-1995' 'May-2000'
 'Jun-1981' 'Nov-1996' 'Feb-1998' '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-1988' 'Jul-1997'
 'Dec-2005' 'Mar-2003' 'Feb-1999' 'Nov-1990' 'Jun-2000' 'Dec-1996'
 'Apr-2010' 'Jan-2004' 'May-1999' 'Sep-1972' 'Jul-1981' 'Sep-1993'
 'Feb-2009' 'Nov-2002' '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' 'Aug-1994'
'Dec-2009' 'Sep-2000' 'Jan-1982' 'Jun-1998' 'Jan-1996' 'Nov-1987'
'May-2010' 'Jun-1987' '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-1992' 'Jul-1990' 'Jan-1988' '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-1987'
'Oct-1984' 'May-1988' 'Aug-1984' 'Nov-1988' 'May-1974' 'Nov-1982'
'Sep-1991' 'Feb-1984' 'Feb-1991' 'Jun-2009' 'Jan-1981' 'Jul-1989'
'Dec-1976' 'Dec-1994' 'Dec-1980' 'Sep-1984' 'Aug-1990' 'Jun-2007'
'Aug-1979' 'Sep-2008' 'Apr-1983' 'Mar-2006' 'Jul-1984' 'Jan-1985'
'Dec-1995' 'Apr-2008' 'Aug-1995' 'Mar-2008' 'Jan-1983' 'Jun-1979'
'Jul-1986' 'Nov-1977' 'Dec-1982' 'May-1979' 'May-1985' 'Feb-1983'
'Aug-1982' 'Oct-1980' 'Mar-1979' 'Jan-1978' 'Jun-2011' 'Mar-1984'
'May-1983' 'Jul-2008' 'Dec-1986' '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' 'Jun-1985'
'Aug-1981' 'Feb-2011' 'Nov-1974' 'Feb-1978' 'Sep-1983' 'Jul-2011'
'Nov-1979' 'Oct-1985' 'Aug-1983' 'Apr-1985' 'Jul-2009' 'Jul-1987'
'Nov-1983' 'Aug-1978' 'Aug-2010' 'Oct-1976' 'Aug-1986' 'Jan-1991'
'Dec-1991' 'May-2009' 'Aug-2011' 'Jan-1974' 'May-1981' 'Jun-1972'
'Jun-1978' 'Sep-1986' 'Jan-1987' 'Jan-1975' 'Jan-1980' 'Sep-1980'
'Jul-1974' 'Dec-1975' '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' 'Nov-1976' 'Nov-1970' 'Nov-2011' 'Nov-1973' 'Sep-1981'
'Jul-1980' 'Mar-2012' 'Mar-1977' 'Dec-1977' 'May-2012' 'Dec-1979'
'Oct-1983' 'Jan-2009' 'Jan-1970' 'Dec-2011' 'Mar-1976' 'Jan-1973'
'Oct-1973' 'Mar-1969' 'Oct-1977' 'Mar-1975' 'Jun-1969' 'Oct-1963'
'Nov-1960' 'Aug-1970' 'Feb-1979' 'Sep-1974' 'May-1966' 'Apr-1972'
'Apr-1973' 'May-1975' 'Sep-1966' 'May-1980' 'Feb-1982' 'Feb-1969'
'Feb-2012' 'Jan-1961' 'Aug-1973' 'Feb-1972' 'Apr-1975' 'Jul-1978'
'Mar-1980' 'Sep-1976' 'Apr-2011' 'Nov-2012' 'Jun-1984' 'Aug-1976'
'Apr-1981' 'Mar-2009' 'Jun-1977' 'Apr-2012' 'Apr-1971' 'Sep-1969'
'Feb-1977' 'Jun-2012' 'Apr-1976' 'Feb-1965' 'Jul-1977' 'Jun-1976'
'Mar-1973' 'Oct-1972' 'Aug-1977' 'Dec-1978' 'Jun-1975' 'Nov-1967'
'Nov-1971' 'Jun-1980' 'May-1964' 'Feb-1975' 'Feb-1971' 'Apr-1970'
'Apr-1969' 'Jun-1974' 'Oct-1974' 'May-1977' 'Dec-1981' 'Feb-1976'
'Mar-1970' 'Aug-1968' 'Jun-1963' 'Jun-2013' 'Mar-1972' 'Aug-2012'
'Feb-1968' 'Dec-1969' 'Jan-1977' 'Jul-1970' 'Feb-1973' 'Mar-1974'
'Feb-1974' 'Jul-1972' 'Jul-1973' 'Sep-1964' 'Jul-1965' 'Jul-2012'
'Jun-1973' 'Nov-1975' 'Jul-1963' 'Nov-1978' 'Jan-1964' 'Jun-1971'
'Mar-1971' 'Dec-1968' 'May-1958' 'Sep-1973' 'May-1971' 'Dec-1972'
'Aug-1965' 'Dec-1974' 'Jul-1976' 'Oct-2012' 'May-1973' 'Apr-1955'
'Sep-1978' 'Apr-1966' 'Jan-1968' 'Nov-1968' 'Mar-2013' 'Jan-2013'
```

```
 'Oct-1965' 'Jan-1966' 'Aug-1972' 'Jul-1969' 'May-1965' 'Oct-1969'
 'Aug-1974' 'May-1968' 'Aug-1969' 'May-2013' 'Jul-1967' 'Oct-1967'
 'May-1970' 'Aug-1975' 'Apr-1974' 'Jan-1963' 'Apr-1968' 'Jul-1971'
 'Dec-1973' 'Jan-1969' 'Nov-1972' 'Oct-1959' 'Apr-1967' 'Sep-1967'
 'Nov-1963' 'Feb-1970' 'Oct-1971' 'Jun-1960' 'Jan-1960' 'Sep-2013'
 'Sep-1971' 'May-1969' 'Dec-1966' 'Oct-1970' 'Nov-1969' 'Jan-1972'
 'Dec-1967' 'Sep-1968' 'Oct-1964' 'Aug-1966' 'Jan-1971' 'Jul-1966'
 'Apr-1964' 'Sep-1962' 'Jul-2013' 'Jun-1967' 'Apr-1965' 'Jun-1966'
 'Jan-1962' 'Oct-1968' 'Aug-1958' 'Dec-1959' 'Sep-1963' 'Dec-2012'
 'Dec-1963' 'Jan-1944' 'Jun-1965' 'May-1962' 'Jun-1970' 'Dec-1970'
 'Mar-1968' 'Jan-1967' 'Aug-2013' 'Jun-1968' 'Oct-1957' 'Dec-1958'
 'Mar-1967' 'Feb-1963' 'Feb-1967' 'Dec-1960' 'May-1955' 'Feb-1966'
 'Nov-1950' 'Mar-1964' 'Jan-1958' 'Nov-1966' 'Dec-1962' 'Sep-1961'
 'Jun-1957' 'Dec-1964' 'Nov-1953' 'Jan-1965' 'Mar-1966' 'Oct-1960'
 'Feb-1964' 'Jul-1959' 'Jul-1968' 'Mar-1963' 'Mar-1962' 'Nov-1965'
 'Jul-1960' 'May-1967' 'Oct-1962' 'Dec-1950' 'Jul-1958' 'Nov-1954'
 'Nov-1957' 'May-1963' 'Jul-1955' 'Oct-1950' 'Dec-1961' 'Oct-2013'
 'Jun-1964' 'Nov-1964' 'Aug-1964' 'Apr-1962' 'Jun-1962' 'Sep-1959'
 'Jul-1962' 'Jan-1957' 'Sep-1965' 'Jan-1956' 'Nov-1958' 'Jul-1951'
 'Jan-1959' 'Apr-1958' 'Mar-1960' 'Sep-1957' 'Sep-1960' 'May-1959'
 'Oct-1966' 'Jun-1959' 'Feb-1962' 'Sep-1956' 'Jul-1964' 'Aug-1960'
 'Feb-1961' 'Jan-1948' 'Aug-1963' 'Oct-1961' 'Aug-1962']
 -------------------------
"initial_list_status" has 2 unique values :
 ['w' 'f']
 -------------------------
"application_type" has 3 unique values :
 ['INDIVIDUAL' 'JOINT' 'DIRECT_PAY']
 -------------------------

"address" has 374809 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']
 -------------------------
```

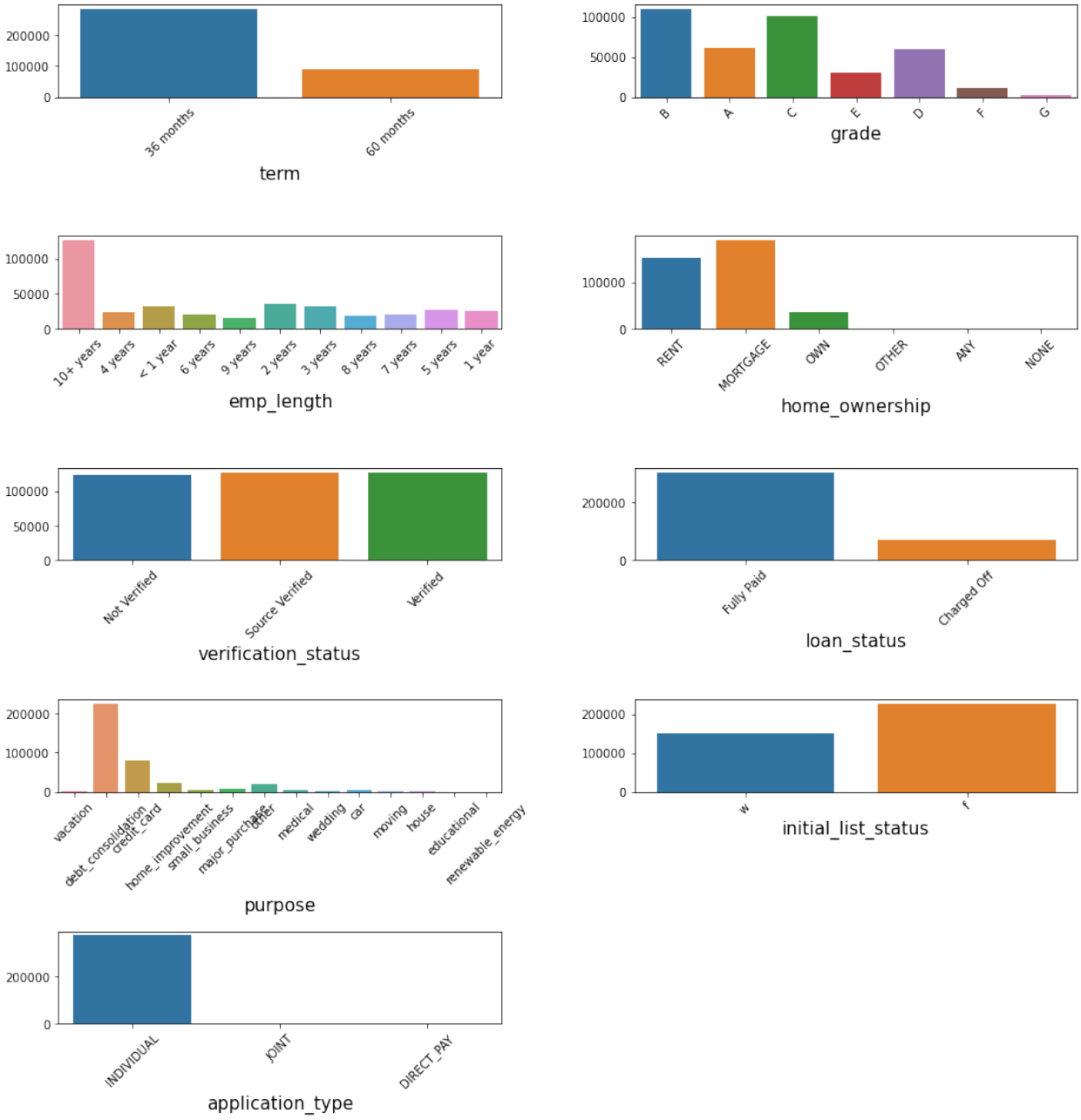
UNIVARIATE ANALYSIS

Selecting Categorical columns based on no of categories

```
# selected category cols
sel_cat_cols = ['term', 'grade', 'emp_length', 'home_ownership',
'verification_status', 'loan_status', 'purpose',
'initial_list_status', 'application_type']

#univariate analysis on categorical features
fig = plt.figure(figsize=(15,15))
```

```
for n,col in enumerate(sel_cat_cols):

    plt.subplot(int(len(sel_cat_cols)/2 +1), 2, n+1)
    plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1,
wspace=0.3, hspace=1.5)
    #plt.pie(list(data[col].value_counts().values),labels =
list(data[col].value_counts().index), startangle = 90)
    sns.countplot(x = data[col], dodge='False')
    #plt.title(col, fontsize = 15)
    plt.xticks(rotation = 45)
    plt.xlabel(col, color='black', fontsize='15')
    plt.ylabel('')
fig.suptitle("Univariate Analysis Categorical Features", fontsize= 20,
color = 'blue')
plt.show()
```

Univariate Analysis Categorical Features

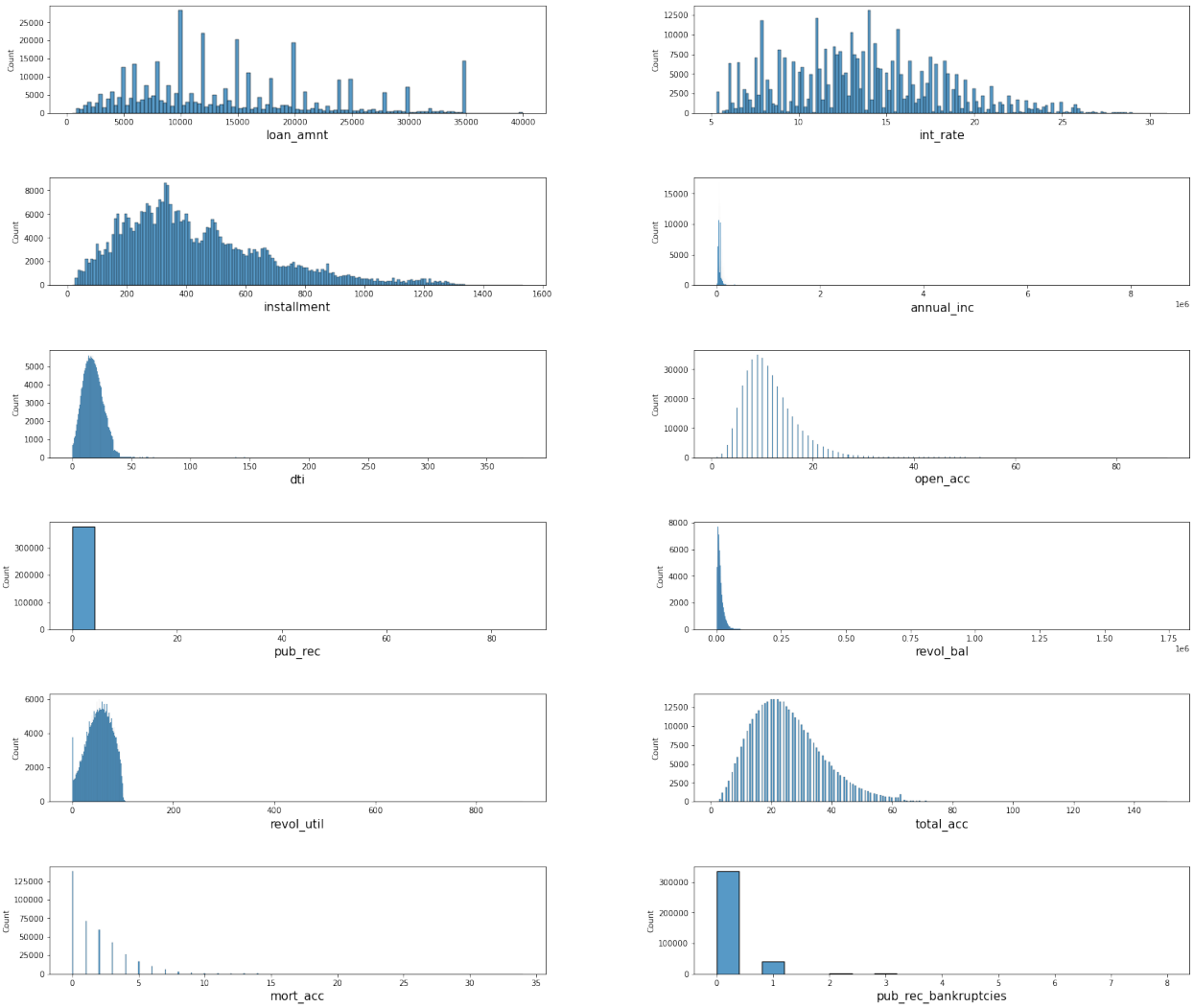


Insights:

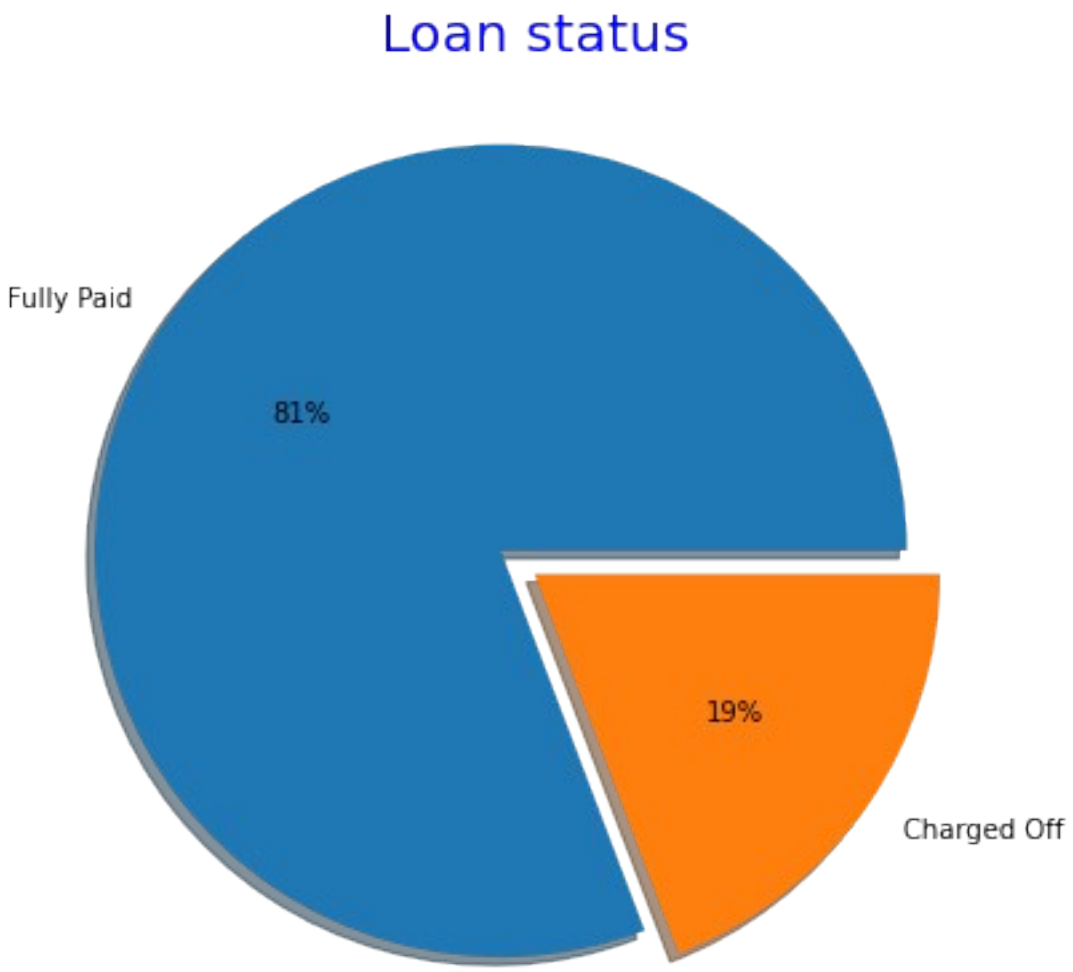
```
- Maximum no of customers applyed for loan took term as '30 months'
than '60 months'.
- Customers with grade B are more followed by 'C','A','D' and 'E'.
- More no of customers who applied for loan are with >10 years
emp_length(experience of employee).
- Maximum customers have their home on motgage, followed by rented
```

```
home and then own house.
- Maximum number of customers had their income verified by LoanTap.
- 80% of the customers have fully paid their loan and 20% didnt.
- Maximum of the customers took the loan for vacation purpose.
- The initial listing status of the loan is 'w' for maximum number of
customers.
- Maximum number of customers applied for loan individually
ie.application_type is 'INIDVIDUAL' .

#univariate analysis on numerical features
fig = plt.figure(figsize=(25,25))
for i,col in enumerate(num_cols):
    plt.subplot(int(len(num_cols)/2 +1), 2, i+1)
    plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1,
wspace=0.3, hspace=0.6)
    sns.histplot(x=data[col] )
    plt.xlabel(col,fontsize =15)
    #plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
fig.suptitle("Univariate Analysis of Numerical cols ", fontsize= 20,
color = 'blue')
plt.show()
```

Univariate Analysis of Numerical cols



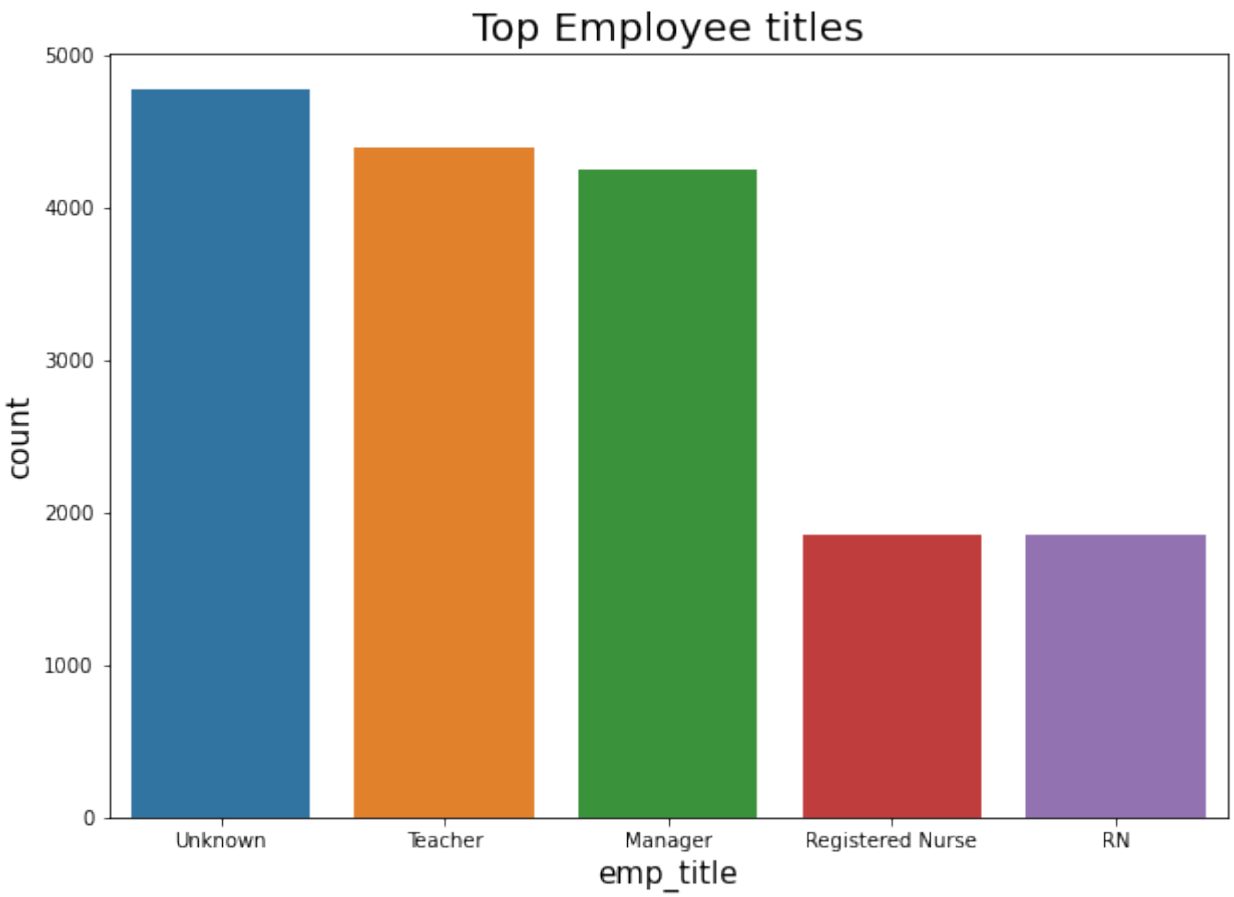

```
#pie chart for loan_status
fig = plt.figure(figsize=(10,7))
plt.pie(list(data['loan_status'].value_counts().values),labels =
list(data['loan_status'].value_counts().index),
        autopct='%.0f%%', explode = [0.1, 0], shadow = True)
plt.title('Loan status', fontsize='20', color='blue')
plt.show()
```



Insights:

```
81% of customers fully paid the loan. 19% chargerd off.
```

```
#top 5 employee titles of customers
fig = plt.figure(figsize=(10,7))
sns.barplot(x= data['emp_title'].value_counts()[:5].index, y =
data['emp_title'].value_counts()[:5].values)
plt.xlabel('emp_title', fontsize= 15, color = 'black')
plt.ylabel("count",fontsize= 15, color = 'black')
plt.title('Top Employee titles', fontsize='20', color='black')
plt.show()
```



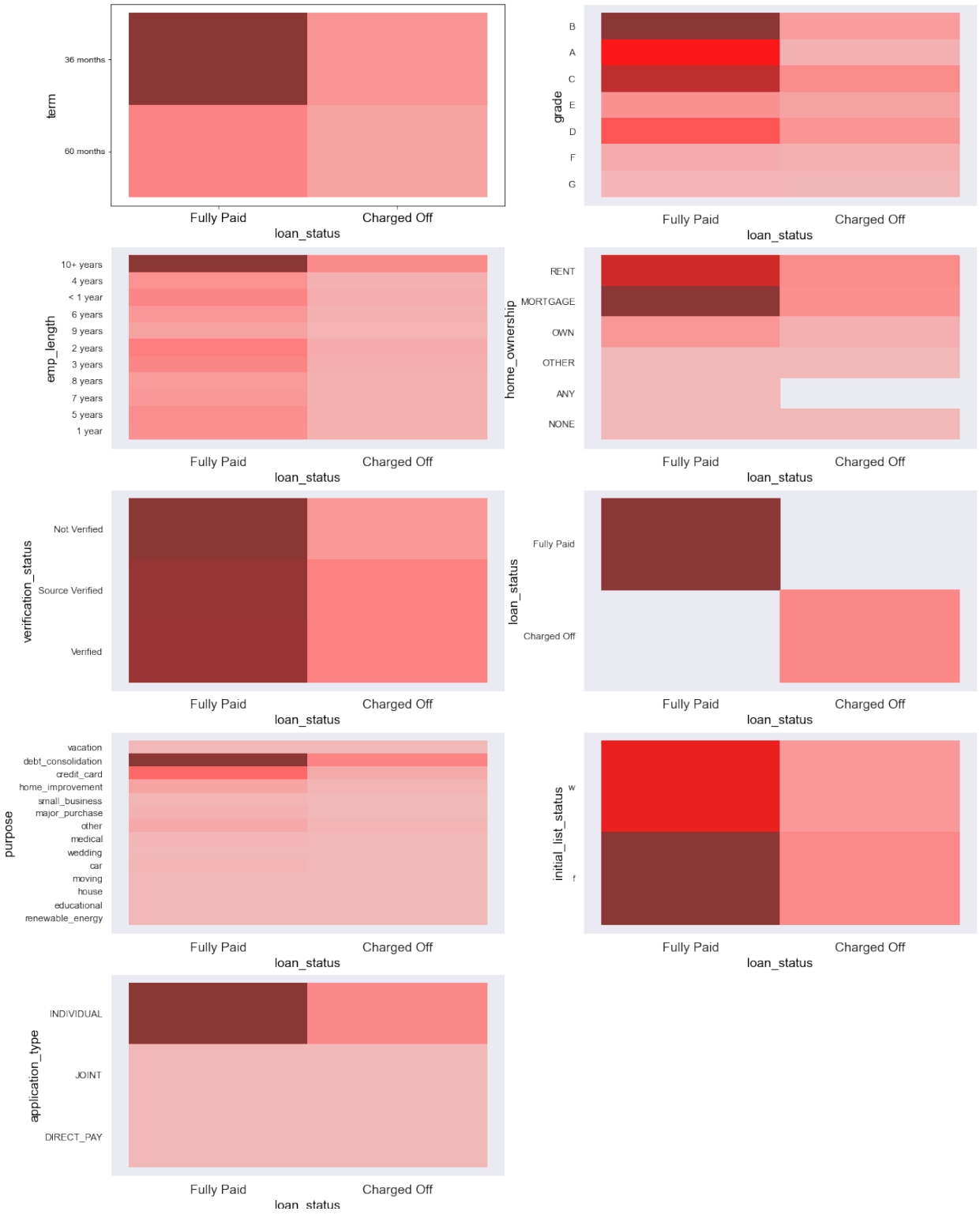
Insights:

```
Teacher and Manager are top 2 employee titles.
```

BIVARIATE ANALYSIS

```
fig = plt.figure(figsize=(18,25))
for i,col in enumerate(sel_cat_cols):
    plt.subplot(int(len(sel_cat_cols)/2 +1), 2, i+1)
    sns.histplot(data= data, x= "loan_status", y= col, color = 'red')
    sns.set_theme(style='dark')
    plt.xlabel('loan_status', fontsize= 15, color = 'black')
    plt.ylabel(col,fontsize= 15, color = 'black')
    plt.xticks(fontsize = 15)

plt.suptitle("LoanStatus Vs Categorical Features",fontsize=20, color =
'red')
plt.show()
```

LoanStatus Vs Categorical Features



Insights:

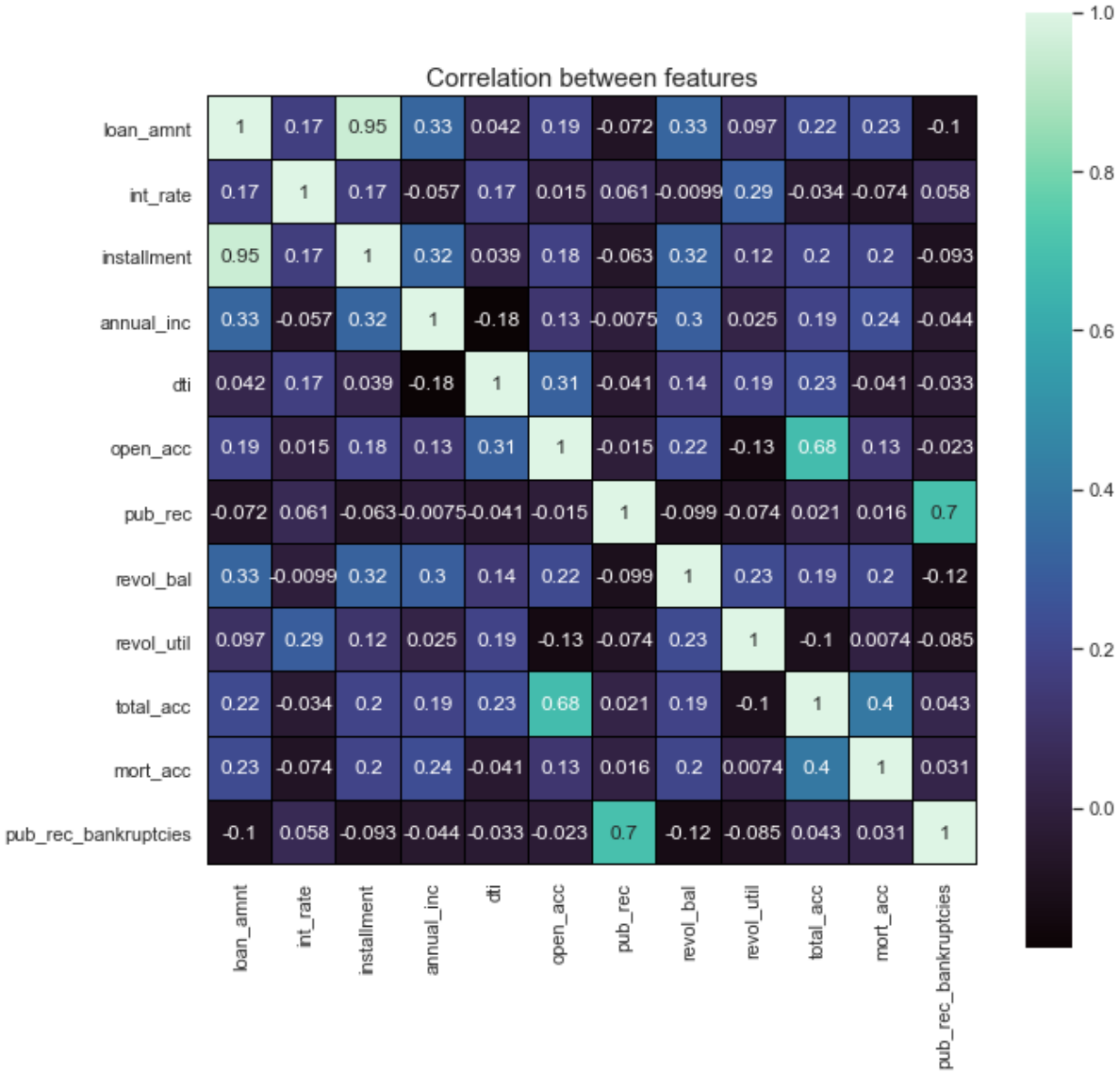
```
- Maximum number of Customers with 'application_type' INIDIVIDUAL had
paid the loan fully.
- Maximum number of customers with 'purpose' debt_consolidation had
paid the loan fully.
- Almost all the customers of all verification_types fully paid the
loan.
- Maximum number of customers with 'home_ownership' mortgage had paid
the loan fully.
- Maximum number of customers with 'emp_lenght' >10 years had paid the
loan fully.
- Maximum number of customers with 'term' 36 months had paid the loan
fully.
- Maximum number of customers with 'grade' B had paid the loan fully.
```

```
pd.crosstab(data['loan_status'],data['grade'],
normalize='columns',margins= True,
            margins_name = 'Fraction').round(2)
```

grade	A	B	C	D	E	F	G	Fraction
loan_status								
Charged Off	0.06	0.12	0.21	0.28	0.37	0.43	0.48	0.19
Fully Paid	0.94	0.88	0.79	0.72	0.63	0.57	0.52	0.81

CORRELATION

```
#Heat map for correlation
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(), annot = True, cmap = 'mako', linewidths =
0.1, square= True, linecolor = 'Black')
plt.yticks(rotation=0)
plt.title("Correlation between features", fontsize= 15)
plt.show()
```



Insights:

```
The features loan_amnt and installment are more positively correlated
with 95% .
pub_rec and pub_rec_bankruptcies are correlated only by 70%.
total_acc and open_acc are 68% correlated.

#correlation using spearmsmans corr coeff
for col in num_cols:
    val = np.corrcoef(data[col].rank(), data['loan_status'].rank())
[0,1]
    if val> 0 :
        print('There is +ve relation between loan_status and', col,"-
corrcoef: ",np.round(val,2))
```

```
    if val == 0:
        print('There is no relation between loan_status and ', col,"-
corrcoef: ",np.round(val,2))
    if val< 0:
        print('There is -ve relation between loan_status and ', col,"-
corrcoef: ",np.round(val,2))
```

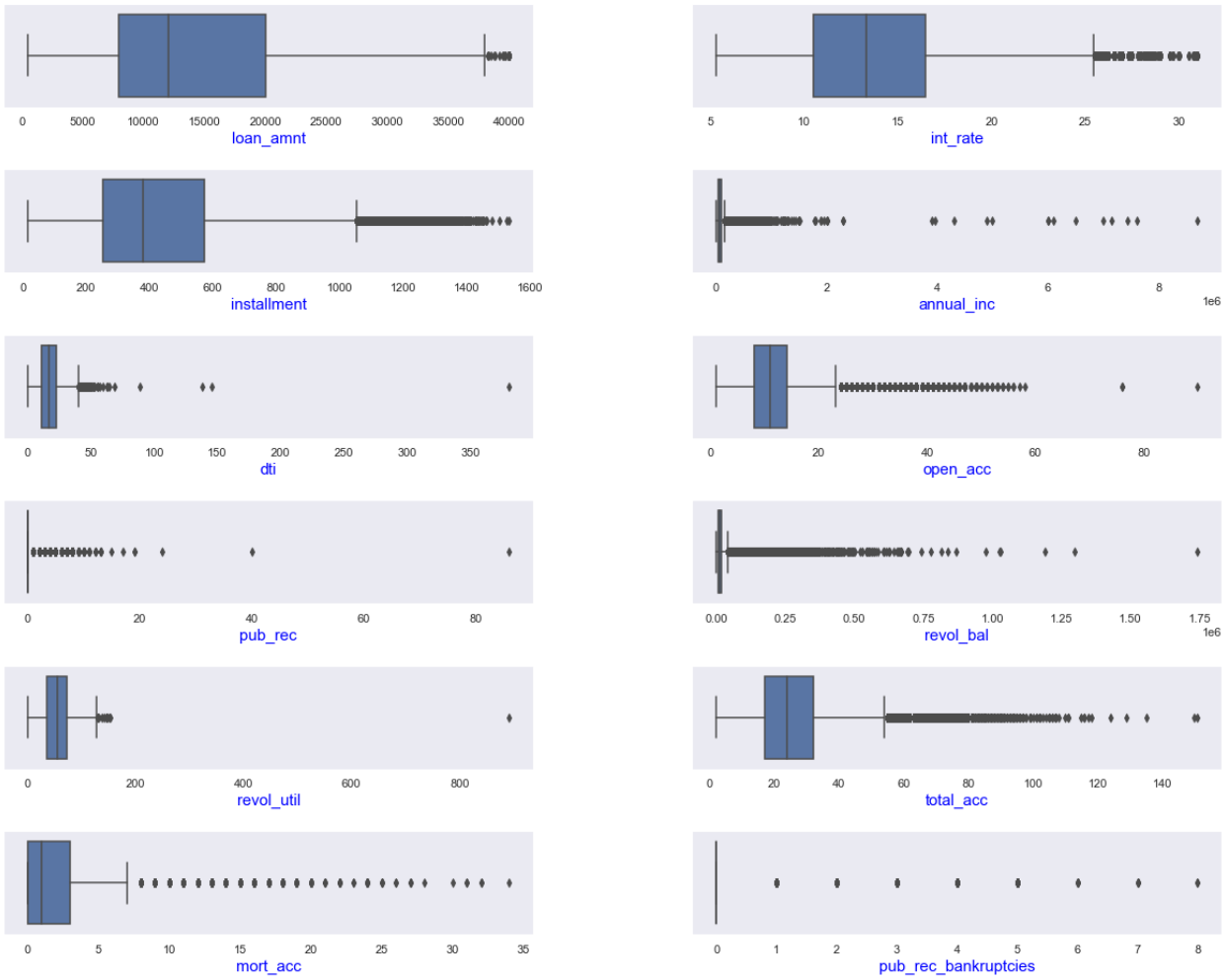
```
There is -ve relation between loan_status and  loan_amnt - corrcoef:
-0.07
There is -ve relation between loan_status and  int_rate - corrcoef:  -
0.25
There is -ve relation between loan_status and  installment - corrcoef:
-0.05
There is +ve relation between loan_status and annual_inc - corrcoef:
0.08
There is -ve relation between loan_status and  dti - corrcoef:  -0.13
There is -ve relation between loan_status and  open_acc - corrcoef:  -
0.03
There is -ve relation between loan_status and  pub_rec - corrcoef:  -
0.02
There is -ve relation between loan_status and  revol_bal - corrcoef:
-0.0
There is -ve relation between loan_status and  revol_util - corrcoef:
-0.08
There is +ve relation between loan_status and total_acc - corrcoef:
0.02
There is +ve relation between loan_status and mort_acc - corrcoef:
0.08
There is -ve relation between loan_status and  pub_rec_bankruptcies -
corrcoef:  -0.01
```

OUTLIER DETECTION AND TREATMENT

```
data.duplicated().sum()
```

```
0
```

```
#Outlier detection on numerical features
fig = plt.figure(figsize=(20,18))
for i,col in enumerate(num_cols):
    plt.subplot(int(len(num_cols)/2 +1), 2, i+1)
    plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1,
wspace=0.3, hspace=0.6)
    sns.boxplot(x=data[col] )
    plt.xlabel(col,fontsize =15, color = 'blue')
    #plt.ylabel("count of bikes", fontsize = 15, color = 'blue')
fig.suptitle("Outliers in Numerical cols ", fontsize= 20, color =
'blue')
plt.show()
```



Insights:

```
    From the above boxplot we can say that there are outliers in all
the numerical features.

# IQR based outlier detection
def iqr_outliers(col):
  q1 = np.percentile(col, 25)
  q3 = np.percentile(col, 75)
  iqr = q3 - q1
  lower_bound = q1 - 1.5 * iqr
  upper_bound = q3 + 1.5 * iqr
  outliers = [i for i, value in enumerate(col) if value < lower_bound
or value > upper_bound]
  return outliers
```

```
outliers= {}
print("Outliers in the data:\n")
for col in num_cols:
    outliers[col] = iqr_outliers(data[col])
    print(f'There are
{round((len(outliers[col])/data.shape[0])*100,2)}% in "{col}"')
```

```
Outliers in the data:

There are 0.05% in "loan_amnt"
There are 0.95% in "int_rate"
There are 2.77% in "installment"
There are 4.42% in "annual_inc"
There are 0.06% in "dti"
There are 2.65% in "open_acc"
There are 14.13% in "pub_rec"
There are 5.35% in "revol_bal"
There are 0.0% in "revol_util"
There are 2.16% in "total_acc"
There are 1.72% in "mort_acc"
There are 10.98% in "pub_rec_bankruptcies"
```

```
#outlier treatment: minimizing outliers by setting upperand lower
limits as mean+/-std
for col in num_cols:
    mean=data[col].mean()
    std=data[col].std()

    upper_limit=mean+3*std
    lower_limit=mean-3*std

    data=data[(data[col]<upper_limit) & (data[col]>lower_limit)]

data.shape
```

```
(344054, 27)
```

Insights:

```
From the above method 5% of data has been discarded considering them
as potential outliers.
```

Feature Engineering & Data Preprocessing

ENCODING

For encoding a feature let us first check the no of unique values in each category. This will help us choosing the type of encoding to be applied.

There are total 24 features in our data.

```
- 'loan_status' is target feature.
- Categorical columns:
['term', 'grade', 'sub_grade', 'emp_length', 'home_ownership',
'verification_status',
 'issue_d', 'loan_status', 'purpose', 'title', 'earliest_cr_line',
'initial_list_status', 'application_type',
 'address']
- Numerical columns:
['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti',
'open_acc', 'pub_rec', 'revol_bal',
 'revol_util', 'total_acc', 'mort_acc', 'pub_rec_bankruptcies']
```

Out of these following are the required fields based on the feature description:
term,int_rate,installment,gradeemp_title,home_ownership,annual_inc,verification_status,loan_status,purpose,dti,open_acc,pub_rec,revol_bal,revol_util,total_acc,initial_list_status,application_type,mort_acc ,pub_rec_bankruptcies

```
Below are the features that can be discarded:

    sub_grade, emp_length, issue_d, title,earliest_cr_line,Address
```

Let us extract zipcode from address begore dropping address column.

```
data['address'].sample(5)
```

```
304095      22998 Megan Shores\r\nChristopherfurt, DE 48052
251443    450 Christopher Prairie\r\nGarrisonshire, VT 2...
206656    48801 Thomas Vista Apt. 791\r\nCruzfurt, ID 00813
199790    66853 Cox Street Apt. 658\r\nWest Jamesborough...
177749    93272 Christina Mount Apt. 930\r\nMichaelburgh...
Name: address, dtype: object
```

```
# Deriving zip code and state from address
data['zip_code'] = data['address'].apply(lambda x: pd.Series(x[-5:]))

#dropping unwanted columns
data.drop(columns=['sub_grade','emp_title','emp_length', 'issue_d',
'title','earliest_cr_line','address'], axis=1,
          inplace=True)

#Since there are only 10 zipcodes, we can change the datatype of
zipcodes to categorical
data['zip_code'] = data['zip_code'].astype('object')

#updated cat_cols and num_cols
cat_cols = list(data.columns[(data.dtypes=='object') & (data.columns!
='loan_status')])
cat_cols
```

```
['term',
 'grade',
 'home_ownership',
 'verification_status',
 'purpose',
 'initial_list_status',
 'application_type',
 'zip_code']

for col in cat_cols:
    print(col,":",data[col].nunique(), data[col].unique())

term : 2 [' 36 months' ' 60 months']
grade : 7 ['B' 'A' 'C' 'E' 'D' 'F' 'G']
home_ownership : 6 ['RENT' 'MORTGAGE' 'OWN' 'OTHER' 'ANY' 'NONE']
verification_status : 3 ['Not Verified' 'Source Verified' 'Verified']
purpose : 14 ['vacation' 'debt_consolidation' 'credit_card'
'home_improvement'
 'small_business' 'major_purchase' 'other' 'medical' 'wedding' 'car'
 'moving' 'house' 'educational' 'renewable_energy']
initial_list_status : 2 ['w' 'f']
application_type : 3 ['INDIVIDUAL' 'JOINT' 'DIRECT_PAY']
zip_code : 10 ['22690' '05113' '00813' '11650' '30723' '70466' '29597'
'48052' '86630'
 '93700']
```

Creation of Flags

If value greater than 1.0 then 1 else 0. This can be done on:

1. pub_rec
2. mort_acc
3. pub_rec_bankruptcies

```
def pub_rec(number):
    if number == 0.0:
        return 0
    else:
        return 1

def mort_acc(number):
    if number == 0.0:
        return 0
    elif number >= 1.0:
        return 1
    else:
        return number


def pub_rec_bankruptcies(number):
```

```
    if number == 0.0:
        return 0
    elif number >= 1.0:
        return 1
    else:
        return number

data['pub_rec']=data.pub_rec.apply(pub_rec)
data['mort_acc']=data.mort_acc.apply(mort_acc)
data['pub_rec_bankruptcies']=data.pub_rec_bankruptcies.apply(pub_rec_b
ankruptcies)

#categorical columns encoding

# Converting term values to numerical val
term_values={' 36 months': 36, ' 60 months':60}
data['term'] = data.term.map(term_values)

# Mapping the target variable
data['loan_status']=data.loan_status.map({'Fully Paid':0, 'Charged
Off':1})

# Initial List Status
list_status = {'w': 0, 'f': 1}
data['initial_list_status'] =
data.initial_list_status.map(list_status)

data.select_dtypes('object').columns
```

```
Index(['grade', 'home_ownership', 'verification_status', 'purpose',
       'application_type', 'zip_code'],
      dtype='object')
```

```
data.select_dtypes('object').columns
```

```
Index(['grade', 'home_ownership', 'verification_status', 'purpose',
       'application_type', 'zip_code'],
      dtype='object')
```

```
#implementing one-hot encoding for categorical columns
cat_cols_en = data.select_dtypes('object').columns

encoder = OneHotEncoder(sparse=False)
encoded_data = encoder.fit_transform(data[cat_cols_en])
encoded_df = pd.DataFrame(encoded_data,
columns=encoder.get_feature_names_out(cat_cols_en))
data_en = pd.concat([data,encoded_df], axis=1)
data_en.drop(columns=cat_cols_en, inplace=True)
data_en.head()
```

```
   loan_amnt  term  int_rate  installment  annual_inc  loan_status
dti  \
0    10000.0  36.0     11.44       329.48    117000.0          0.0
26.24
1     8000.0  36.0     11.99       265.68     65000.0          0.0
22.05
2    15600.0  36.0     10.49       506.97     43057.0          0.0
12.79
3     7200.0  36.0      6.49       220.65     54000.0          0.0
2.60
4    24375.0  60.0     17.27       609.33     55000.0          1.0
33.95

   open_acc  pub_rec  revol_bal  ...  zip_code_00813
zip_code_05113  \
0      16.0      0.0    36369.0  ...             0.0             0.0

1      17.0      0.0    20131.0  ...             0.0             1.0

2      13.0      0.0    11987.0  ...             0.0             1.0

3       6.0      0.0     5472.0  ...             1.0             0.0

4      13.0      0.0    24584.0  ...             0.0             0.0


   zip_code_11650  zip_code_22690  zip_code_29597  zip_code_30723  \
0             0.0             1.0             0.0             0.0
1             0.0             0.0             0.0             0.0
2             0.0             0.0             0.0             0.0
3             0.0             0.0             0.0             0.0
4             1.0             0.0             0.0             0.0

   zip_code_48052  zip_code_70466  zip_code_86630  zip_code_93700
0             0.0             0.0             0.0             0.0
1             0.0             0.0             0.0             0.0
2             0.0             0.0             0.0             0.0
3             0.0             0.0             0.0             0.0
4             0.0             0.0             0.0             0.0

[5 rows x 58 columns]

data_en.columns

Index(['loan_amnt', 'term', 'int_rate', 'installment', 'annual_inc',
       'loan_status', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
'revol_util',
       'total_acc', 'initial_list_status', 'mort_acc',
'pub_rec_bankruptcies',
       'grade_A', 'grade_B', 'grade_C', 'grade_D', 'grade_E',
'grade_F',
```

```
       'grade_G', 'home_ownership_ANY', 'home_ownership_MORTGAGE',
       'home_ownership_NONE', 'home_ownership_OTHER',
'home_ownership_OWN',
       'home_ownership_RENT', 'verification_status_Not Verified',
       'verification_status_Source Verified',
'verification_status_Verified',
       'purpose_car', 'purpose_credit_card',
'purpose_debt_consolidation',
       'purpose_educational', 'purpose_home_improvement',
'purpose_house',
       'purpose_major_purchase', 'purpose_medical', 'purpose_moving',
       'purpose_other', 'purpose_renewable_energy',
'purpose_small_business',
       'purpose_vacation', 'purpose_wedding',
'application_type_DIRECT_PAY',
       'application_type_INDIVIDUAL', 'application_type_JOINT',
       'zip_code_00813', 'zip_code_05113', 'zip_code_11650',
'zip_code_22690',
       'zip_code_29597', 'zip_code_30723', 'zip_code_48052',
'zip_code_70466',
       'zip_code_86630', 'zip_code_93700'],
      dtype='object')
```

```
data_en.dropna(inplace=True)
```

```
data_en.isna().sum().sum()
```

```
0
```

```
X = data_en.drop(columns=['loan_status'])
X.reset_index(inplace=True, drop=True)
y = data_en['loan_status']
y.reset_index(drop=True, inplace=True)
```

```
y= data_en['loan_status']
data_en.drop(columns='loan_status', axis=1, inplace = True)
X = data_en
```

Logistic Regression Modelling

```
#Splitting data to test and train data

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=1)
```

```
X_train.shape
```

```
(239144, 57)
```

```
#Scaling data using minmax scaler
```

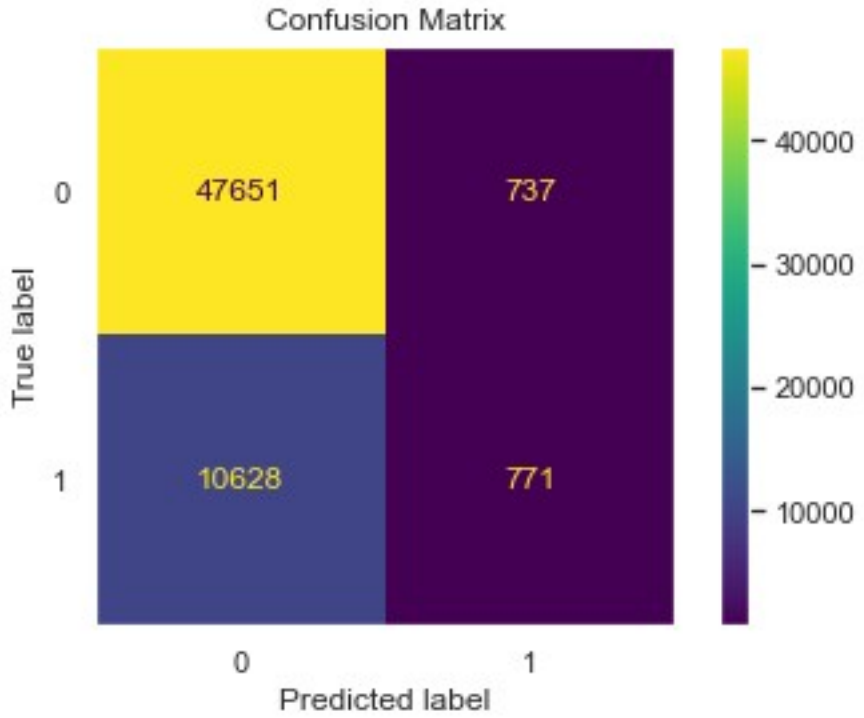
```
scaler = MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train),
columns=X_train.columns)
X_test = pd.DataFrame(scaler.transform(X_test),
columns=X_test.columns)
```

SMOTE, or Synthetic Minority Oversampling Technique, is a technique used in machine learning to address the issue of class imbalance in datasets. Class imbalance occurs when one class (the minority class) has significantly fewer instances compared to other classes (the majority class). SMOTE addresses class imbalance by oversampling the minority class. However, unlike simple oversampling (which simply duplicates existing minority class examples), SMOTE generates synthetic data points for the minority class.

```
# Oversampling to balance the target variable

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: 46054
Before OverSampling, count of label 0: 193090
After OverSampling, count of label 1: 193090
After OverSampling, count of label 0: 193090

model = LogisticRegression()
model.fit(X_train, y_train)
train_preds = model.predict(X_train)
test_preds = model.predict(X_test)

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

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

# Confusion Matrix
```

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

```
Train Accuracy : 0.81
Train F1 Score: 0.12
Train Recall Score: 0.07
Train Precision Score: 0.53

Test Accuracy : 0.81
Test F1 Score: 0.12
Test Recall Score: 0.07
Test Precision Score: 0.51
```




```
print(classification_report(y_test, test_preds))
```

```
              precision    recall  f1-score   support

         0.0       0.82      0.98      0.89     48388
         1.0       0.51      0.07      0.12     11399

    accuracy                           0.81     59787
   macro avg       0.66      0.53      0.51     59787
weighted avg       0.76      0.81      0.75     59787
```

```
print("Model Coefficents:\n",'-'*15)
for i,col in enumerate(data_en.columns):
    print(col,":",round(model.coef_[0][i],3))
```

```
Model Coefficents:
 ---------------
loan_amnt : -0.006
term : 0.582
int_rate : 2.125
installment : 0.466
annual_inc : -1.14
dti : 1.036
open_acc : 0.643
pub_rec : 0.276
revol_bal : -0.499
revol_util : 0.45
total_acc : -0.553
initial_list_status : -0.034
mort_acc : -0.231
pub_rec_bankruptcies : -0.253
grade_A : -0.159
grade_B : -0.147
grade_C : -0.168
grade_D : -0.147
grade_E : -0.145
grade_F : -0.139
grade_G : -0.217
home_ownership_ANY : -0.039
home_ownership_MORTGAGE : -0.276
home_ownership_NONE : 0.035
home_ownership_OTHER : -0.32
home_ownership_OWN : -0.267
home_ownership_RENT : -0.255
verification_status_Not Verified : -0.358
verification_status_Source Verified : -0.391
verification_status_Verified : -0.373
purpose_car : -0.039
purpose_credit_card : -0.047
purpose_debt_consolidation : -0.058
purpose_educational : -0.081
purpose_home_improvement : -0.081
purpose_house : -0.008
purpose_major_purchase : -0.063
purpose_medical : -0.016
purpose_moving : -0.178
purpose_other : -0.103
purpose_renewable_energy : -0.105
purpose_small_business : -0.103
purpose_vacation : -0.079
purpose_wedding : -0.163
```

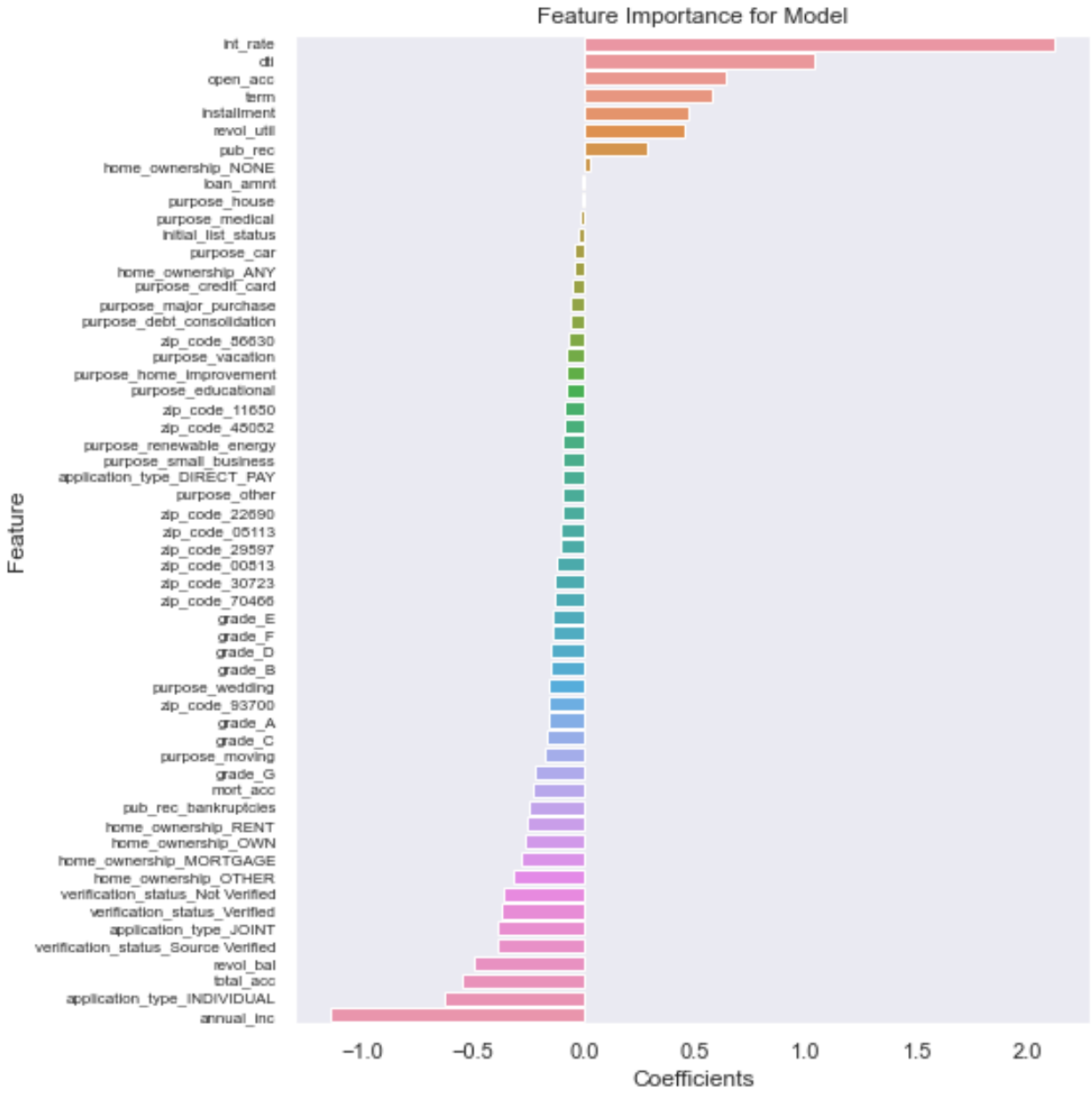
```
application_type_DIRECT_PAY : -0.102
application_type_INDIVIDUAL : -0.628
application_type_JOINT : -0.392
zip_code_00813 : -0.124
zip_code_05113 : -0.111
zip_code_11650 : -0.086
zip_code_22690 : -0.103
zip_code_29597 : -0.11
zip_code_30723 : -0.132
zip_code_48052 : -0.092
zip_code_70466 : -0.133
zip_code_86630 : -0.073
zip_code_93700 : -0.158
```

```
print("model intercept: ", model.intercept_)
```

```
model intercept:  [-1.21717848]
```

Feature Importance

```
feature_imp = pd.DataFrame({'Columns':X_train.columns,
'Coefficients':model.coef_[0]}).round(2).sort_values('Coefficients',
ascending=False)

plt.figure(figsize=(8,8))
sns.barplot(y = feature_imp['Columns'],
           x =  feature_imp['Coefficients'])
plt.title("Feature Importance for Model")
plt.yticks(fontsize=8)
plt.ylabel("Feature")
plt.tight_layout()
plt.show()
```



ROC Curve & AUC

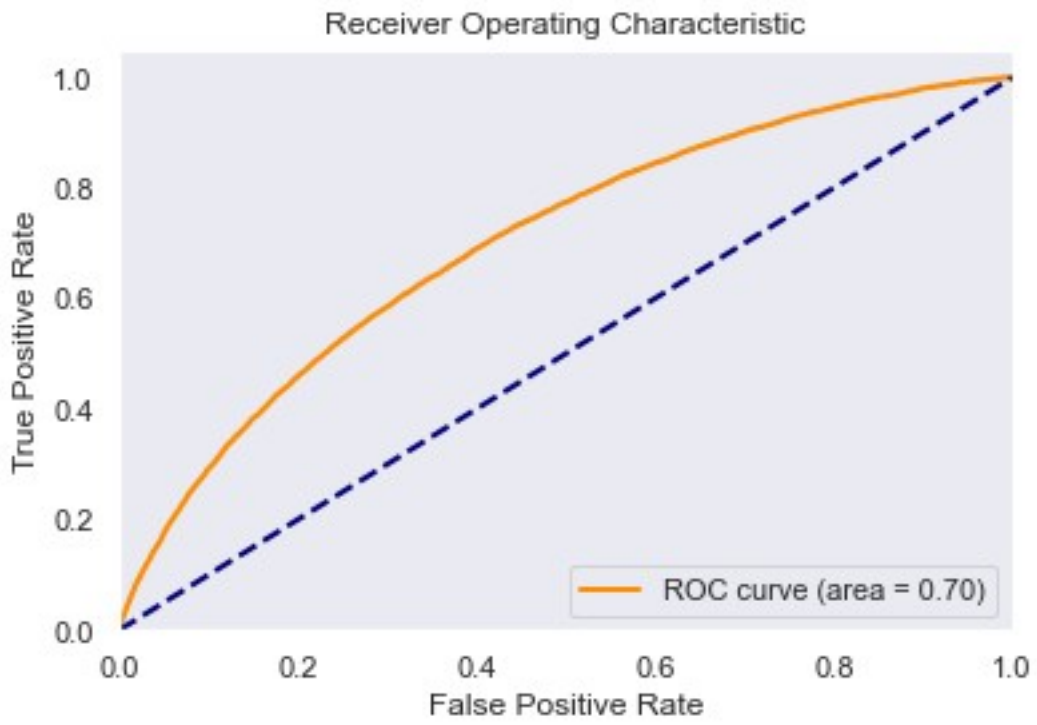
ROC AUC stands for Receiver Operating Characteristic Area Under the Curve.

ROC AUC score is a single number that summarizes the classifier's performance across all possible classification thresholds.

ROC AUC score shows how well the classifier distinguishes positive and negative classes. It can take values from 0 to 1.

A higher ROC AUC indicates better performance. A perfect model would have an AUC of 1, while a random model would have an AUC of 0.5.

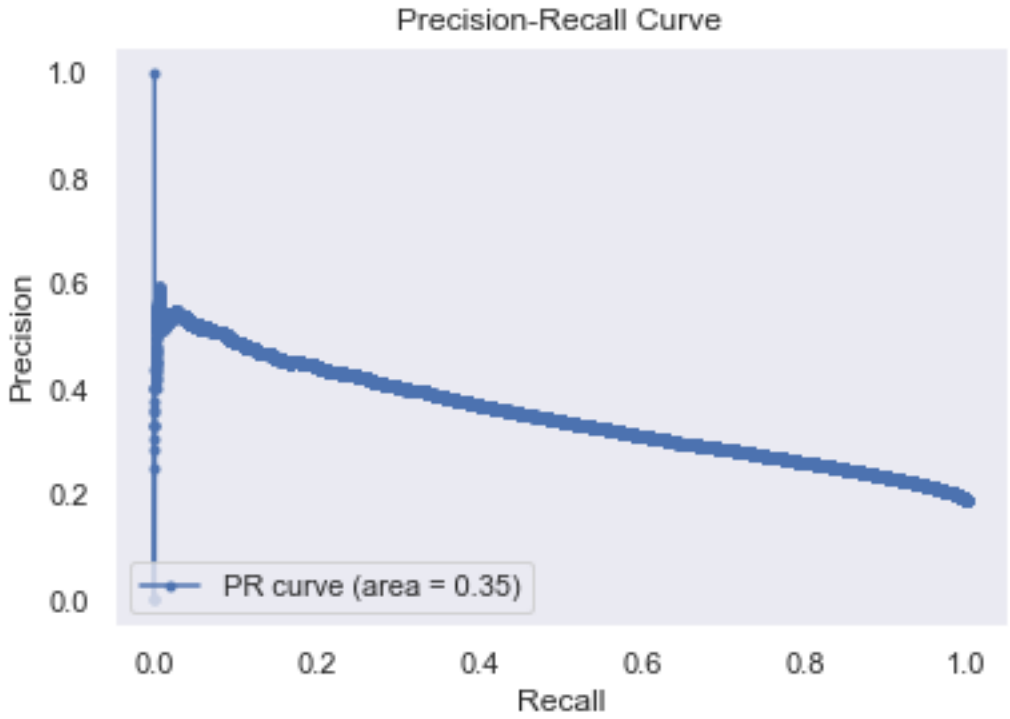
```
# Predict probabilities for the test set
probs = model.predict_proba(X_test)[:,1]

# Compute the false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, probs)

# Compute the area under the ROC curve
roc_auc = auc(fpr, tpr)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area =
%0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



Insights:

```
Model has area under ROC 70% which indicates it as a decent model,
however the score can be improved by
hyperparameter tunning and training a better model.
```

```
# Compute the false precision and recall at all thresholds
precision, recall, thresholds = precision_recall_curve(y_test, probs)

# Area under Precision Recall Curve
auprc = average_precision_score(y_test, probs)

# Plot the precision-recall curve
plt.plot(recall, precision, marker='.', label='PR curve (area =
%0.2f)' % auprc)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc="lower left")
plt.show()
```



Insights:

```
Precision-recall area is 0.35 which means its not a good value and
there is a lot of scope for improvement in the model.
```

Tradeoff Questions:

How can we make sure that our model can detect real defaulters and there are less false positives? This is important as we can lose out on an opportunity to finance more individuals and earn interest on it ?

An oversampling technique SMOTE is used to reduce imbalance and false positives in the data. For better performance of the model we can try various hyperparameter tunning.

Since NPA (non-performing asset) is a real problem in this industry, it's important we play safe and shouldn't disburse loans to anyone ?

Yes. LoanTap should not disburse loans to everyone.

There are many non-verified customers in the data. Every customer applying for loan should be verified either by company internally or by a third-party to identify the correct persons.

From data provided, 20% of people default on their loan, which inturn become NPAs for the company.