Business Case: LoanTap Logistic Regression

Context:

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

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?

Dataset: LoanTapData.csv

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 Concept Used:

Exploratory Data Analysis Feature Engineering Logistic Regression Precision Vs Recall Tradeoff What does ‘good’ look like?

Import the dataset and do usual exploratory data analysis steps like checking the structure & characteristics of the dataset Check how much target variable (Loan_Status) depends on different predictor variables (Use count plots, box plots, heat maps etc) Check correlation among independent variables and how they interact with each other Simple Feature Engineering steps: E.g.: 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

Missing values and Outlier Treatment Scaling - Using MinMaxScaler or StandardScaler Use Logistic Regression Model from Sklearn/Statsmodel library and explain the results Results Evaluation: Classification Report ROC AUC curve Precision recall curve 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. 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 Provide actionable Insights & Recommendations

Evaluation Criteria (100 points)

Define Problem Statement and perform Exploratory Data Analysis (10 points) Definition of problem (as per given problem statement with additional views) Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary. Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables) Bivariate Analysis (Relationships between important variable) Illustrate the insights based on EDA Comments on range of attributes, outliers of various attributes Comments on the distribution of the variables and relationship between them Comments for each univariate and bivariate plots Data Preprocessing (20 Points) Duplicate value check Missing value treatment Outlier treatment Feature engineering Data preparation for modeling Model building (10 Points) Build the Logistic Regression model and comment on the model statistics Display model coefficients with column names Results Evaluation (50 Points) ROC AUC Curve & comments (10 Points) Precision Recall Curve & comments (10 Points) Classification Report (Confusion Matrix etc) (10

Points) 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. (10 Points) 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. (10 Points) Actionable Insights & Recommendations (10 Points)

Questionnaire (Answers should present in the text editor along with insights):

What percentage of customers have fully paid their Loan Amount? Comment about the correlation between Loan Amount and Installment features. The majority of people have home ownership as _______. People with grades ‘A’ are more likely to fully pay their loan. (T/F) Name the top 2 afforded job titles. Thinking from a bank's perspective, which metric should our primary focus be on.. ROC AUC Precision Recall F1 Score How does the gap in precision and recall affect the bank? Which were the features that heavily affected the outcome? Will the results be affected by geographical location? (Yes/No)

Problem Statement

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

Personal Loan EMI Free Loan Personal Overdraft Advance Salary Loan But the main focus is to interpret the underwriting process behind the Personal Loan only 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?

Additional views We need to track the users previous credit line history and repayment status. Analysing the previous loans tenure and the total liability. As we are focusing more on salaried individual, we need to take salary of the person into consideration.

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.
- 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

Installing Dependencies

In [1]:
```
##!pip install imblearn
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import (
    accuracy_score, confusion_matrix, classification_report,
    roc_auc_score, roc_curve, auc,
    ConfusionMatrixDisplay, RocCurveDisplay
)
from statsmodels.stats.outliers_influence import variance_inflation_factor
from imblearn.over_sampling import SMOTE
```

Loading Dataset

In [3]:
```python
loantap = pd.read_csv('logistic_regression.csv')

loantap.head(5)
```

Out[3]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_lengt
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ yea
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 yea
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 ye
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 yea
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 yea

5 rows × 27 columns

In [4]:
```python
print(f"The dataset has {loantap.shape[0]} rows and {loantap.shape[1]} columns")
```

```
The dataset has 396030 rows and 27 columns
```

In [5]:
```python
loantap.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
```

In [6]:
```
display(loantap.dtypes)
```

```
loan_amnt               float64
term                     object
int_rate                float64
installment             float64
grade                    object
sub_grade                object
emp_title                object
emp_length               object
home_ownership           object
annual_inc              float64
verification_status      object
issue_d                  object
loan_status              object
purpose                  object
title                    object
dti                     float64
earliest_cr_line         object
open_acc                float64
pub_rec                 float64
revol_bal               float64
revol_util              float64
total_acc               float64
initial_list_status      object
application_type         object
mort_acc                float64
pub_rec_bankruptcies    float64
address                  object
dtype: object
```

In [7]:
```
loantap.duplicated().sum()
```

Out[7]:
```
0
```

Insights

- Dataset has no duplicate values

In [8]:
```
loantap.isna().sum()
```

Out[8]:

```
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
```

In [9]:

```
missing_vars = np.round(loantap.loc[:,loantap.isna().sum() > 0].isna().sum()/loa
display(missing_vars)
```

```
emp_title               5.79
emp_length              4.62
title                   0.44
revol_util              0.07
mort_acc                9.54
pub_rec_bankruptcies    0.14
dtype: float64
```

In [10]:

```
loantap.isnull().sum()
```

```
Out[10]: 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
```

Null Value Analysis:

1. **emp_title (22,927 nulls)**: A significant number of borrowers did not provide their employment title. This could be due to privacy concerns or the borrower being unemployed.

2. **emp_length (18,301 nulls)**: Many borrowers did not specify the length of their current employment. This can be crucial information, as employment length can be indicative of loan repayment capability.

3. **title (1,756 nulls)**: Some borrowers did not provide a title for their loan request. This might not be critical for loan approval but can offer insights into the purpose of the loan.

4. **revol_util (276 nulls)**: A small fraction of borrowers did not provide their revolving line utilization rate. This metric can be critical as it indicates the borrower's credit usage relative to their available credit.

5. **mort_acc (37,795 nulls)**: A large number of borrowers did not specify the number of mortgage accounts they have. This information can be essential, especially for home loans or large sum loans.

6. **pub_rec_bankruptcies (535 nulls)**: A few borrowers did not provide the number of public record bankruptcies. This is vital information as it directly relates to a borrower's creditworthiness.

Statistical Analysis

In [11]:

```
#loantap.describe()
display(loantap.describe().T)
```

	count	mean	std	min	25%	50%
loan_amnt	396030.0	14113.888089	8357.441341	500.00	8000.00	12000.00
int_rate	396030.0	13.639400	4.472157	5.32	10.49	13.33
installment	396030.0	431.849698	250.727790	16.08	250.33	375.43
annual_inc	396030.0	74203.175798	61637.621158	0.00	45000.00	64000.00
dti	396030.0	17.379514	18.019092	0.00	11.28	16.91
open_acc	396030.0	11.311153	5.137649	0.00	8.00	10.00
pub_rec	396030.0	0.178191	0.530671	0.00	0.00	0.00
revol_bal	396030.0	15844.539853	20591.836109	0.00	6025.00	11181.00
revol_util	395754.0	53.791749	24.452193	0.00	35.80	54.80
total_acc	396030.0	25.414744	11.886991	2.00	17.00	24.00
mort_acc	358235.0	1.813991	2.147930	0.00	0.00	1.00
pub_rec_bankruptcies	395495.0	0.121648	0.356174	0.00	0.00	0.00

- The wide ranges in many of the metrics (e.g., annual_inc, dti, revol_bal, revol_util) suggest the presence of outliers or possibly incorrect data entries.
- While averages provide an overall sense of the data distribution, the large differences between the mean and median in some metrics indicate skewed distributions.

Categorical Features:-

In [12]:

```
display(loantap.describe(include = 'object').T)
```

	count	unique	top	freq
term	396030	2	36 months	302005
grade	396030	7	B	116018
sub_grade	396030	35	B3	26655
emp_title	373103	173105	Teacher	4389
emp_length	377729	11	10+ years	126041
home_ownership	396030	6	MORTGAGE	198348
verification_status	396030	3	Verified	139563
issue_d	396030	115	Oct-2014	14846
loan_status	396030	2	Fully Paid	318357
purpose	396030	14	debt_consolidation	234507
title	394275	48817	Debt consolidation	152472
earliest_cr_line	396030	684	Oct-2000	3017
initial_list_status	396030	2	f	238066
application_type	396030	3	INDIVIDUAL	395319
address	396030	393700	USCGC Smith\r\nFPO AE 70466	8

In [13]:
```
for feat in (loantap.columns[loantap.nunique() < 15]):
    print(f'\nValue Counts {loantap[feat].value_counts(normalize=True) * 100}:-\
    print('-'*40)
```

```
Value Counts  36 months    76.258112
 60 months    23.741888
Name: term, dtype: float64:-

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

Value Counts B    29.295255
C    26.762366
A    16.207611
D    16.040199
E     7.950913
F     2.972502
G     0.771154
Name: grade, dtype: float64:-

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

Value Counts 10+ years    33.368103
2 years       9.484842
< 1 year      8.398879
3 years       8.382994
5 years       7.014288
1 year        6.852002
4 years       6.341054
6 years       5.517448
7 years       5.511623
8 years       5.074538
9 years       4.054229
Name: emp_length, dtype: float64:-

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

Value Counts MORTGAGE    50.084085
RENT        40.347953
OWN          9.531096
OTHER        0.028281
NONE         0.007828
ANY          0.000758
Name: home_ownership, dtype: float64:-

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

Value Counts Verified           35.240512
Source Verified    33.175517
Not Verified       31.583971
Name: verification_status, dtype: float64:-

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

Value Counts Fully Paid     80.387092
Charged Off    19.612908
Name: loan_status, dtype: float64:-

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

Value Counts debt_consolidation    59.214453
credit_card           20.962806
home_improvement       6.067722
other                  5.349342
major_purchase         2.219529
```

```
small_business         1.439537
car                    1.186021
medical                1.059516
moving                 0.720652
vacation               0.619145
house                  0.555766
wedding                0.457541
renewable_energy       0.083075
educational            0.064894
Name: purpose, dtype: float64:-

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

Value Counts f    60.113123
w    39.886877
Name: initial_list_status, dtype: float64:-

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

Value Counts INDIVIDUAL    99.820468
JOINT           0.107315
DIRECT_PAY      0.072217
Name: application_type, dtype: float64:-

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

Value Counts 0.0    88.592776
1.0    10.819353
2.0     0.467010
3.0     0.088750
4.0     0.020734
5.0     0.008091
6.0     0.001770
7.0     0.001011
8.0     0.000506
Name: pub_rec_bankruptcies, dtype: float64:-

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

In [125…

```
loantap['emp_title'].value_counts(normalize=True).rename_axis('unique_values').t
```

Out[125]:

unique_values	counts
0.000000	0.417055
1.000000	0.074480
0.224012	0.031307
0.243391	0.031261
0.500000	0.023202

Insights Summary:

- Most borrowers prefer a shorter loan term of 36 months.
- The loan grades are mostly concentrated around the B, C, and A categories.

- A significant proportion of borrowers have been employed for a long duration, which could be indicative of stability.

- Debt consolidation seems to be the primary reason for seeking loans, which suggests many borrowers are trying to streamline their finances.

- Most of the loans are individual applications, and a majority have managed to fully pay off their loans.

- There is significant difference found in the mean and median of the following attributes

    - loan_amnt
    - terms
    - installment
    - revol_bal etc.
- These attributes might contain outliers

Outliers Detection

In [14]:

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_palette(palette="Set2",n_colors=18)
sns.set_style("whitegrid", {'axes.facecolor': '0.97'})
```

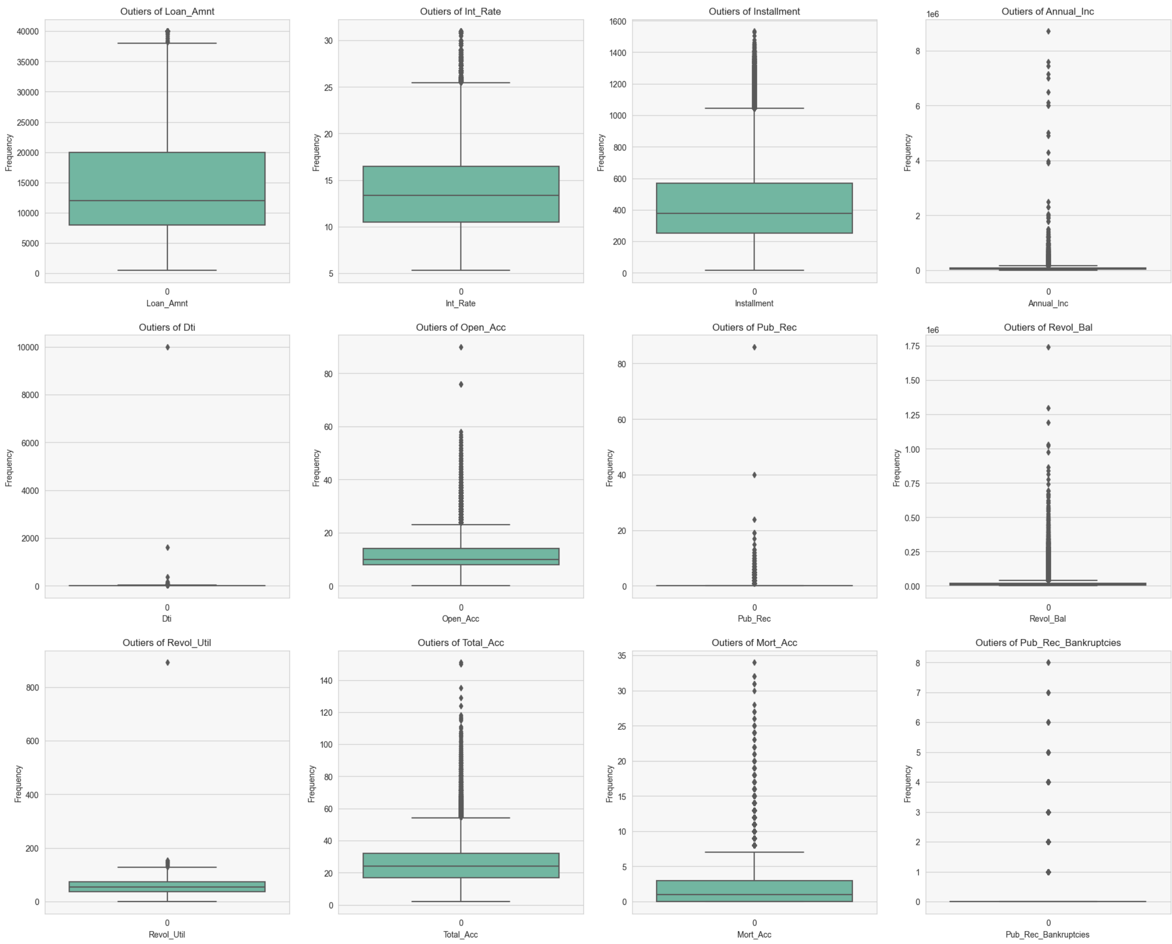
In [15]:

```
print('Outliers Detection:- ')
print(50*'-')
for i,var in enumerate(loantap.columns[loantap.dtypes != 'object']):
    q1 = np.quantile(loantap[var],0.25)
    q3 = np.quantile(loantap[var],0.75)

    iqr = q3 - q1

    upper_limit = q3 + 1.5 *iqr
    lower_limit = max(q1 - 1.5 *iqr,0)

    total_length = len(loantap)
    upper_outliers = len(loantap.loc[loantap[var] < lower_limit, var])
    lower_outliers = len(loantap.loc[loantap[var] > upper_limit, var])
    total_outliers = len(
        loantap.loc[(loantap[var] > upper_limit) | (loantap[var] < lower_limit),
    )

    upper_outliers_perc = round((upper_outliers/total_length) * 100,2)
    lower_outliers_perc = round((lower_outliers/total_length) * 100,2)

    total_outliers_perc = round((total_outliers/total_length) * 100,2)

    print(var.title(),":-")
    print(f'Percent of outliers in lower region: {upper_outliers_perc} %')
    print(f'Percent of outliers in upper region: {lower_outliers_perc} %')
    print(f'Total Outliers: {total_outliers_perc} %',end='\n\n')
```

```
Outliers Detection:-
--------------------------------------------------
Loan_Amnt :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.05 %
Total Outliers: 0.05 %

Int_Rate :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.95 %
Total Outliers: 0.95 %

Installment :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 2.84 %
Total Outliers: 2.84 %

Annual_Inc :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 4.22 %
Total Outliers: 4.22 %

Dti :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.07 %
Total Outliers: 0.07 %

Open_Acc :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 2.6 %
Total Outliers: 2.6 %

Pub_Rec :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 14.58 %
Total Outliers: 14.58 %

Revol_Bal :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 5.37 %
Total Outliers: 5.37 %

Revol_Util :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.0 %
Total Outliers: 0.0 %

Total_Acc :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 2.15 %
Total Outliers: 2.15 %

Mort_Acc :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.0 %
Total Outliers: 0.0 %

Pub_Rec_Bankruptcies :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.0 %
```

```
Total Outliers: 0.0 %
```

In [16]:
```python
plt.figure(figsize=(25,20))

for i,var in enumerate(loantap.columns[loantap.dtypes != 'object'],1):
    plt.subplot(3,4,i)

    sns.boxplot(loantap[var])

    plt.title(f'Outiers of {var.title()}')
    plt.xlabel(var.title())
    plt.ylabel('Frequency')

plt.show()
```



- All outliers are in the upper region.
- The feature Pub_Rec has the highest outlier percentage at 14.58%, followed by Revol_Bal at 5.37% and Annual_Inc at 4.22%.
- Features Revol_Util, Mort_Acc, and Pub_Rec_Bankruptcies have no outliers.

Feature Extraction

In [17]:
```python
loantap['state'] = loantap['address'].str.split(',').str[1]
loantap['state'] = loantap['address'].str.split(',').str[1]
loantap['pincode'] = loantap['state'].str.split(' ').str[2].str.strip()
loantap['state'] = loantap['state'].str.split(' ').str[1].str.strip()
loantap['city'] = loantap['address'].str.split(',').str[0].str.split('\r\n').str
```

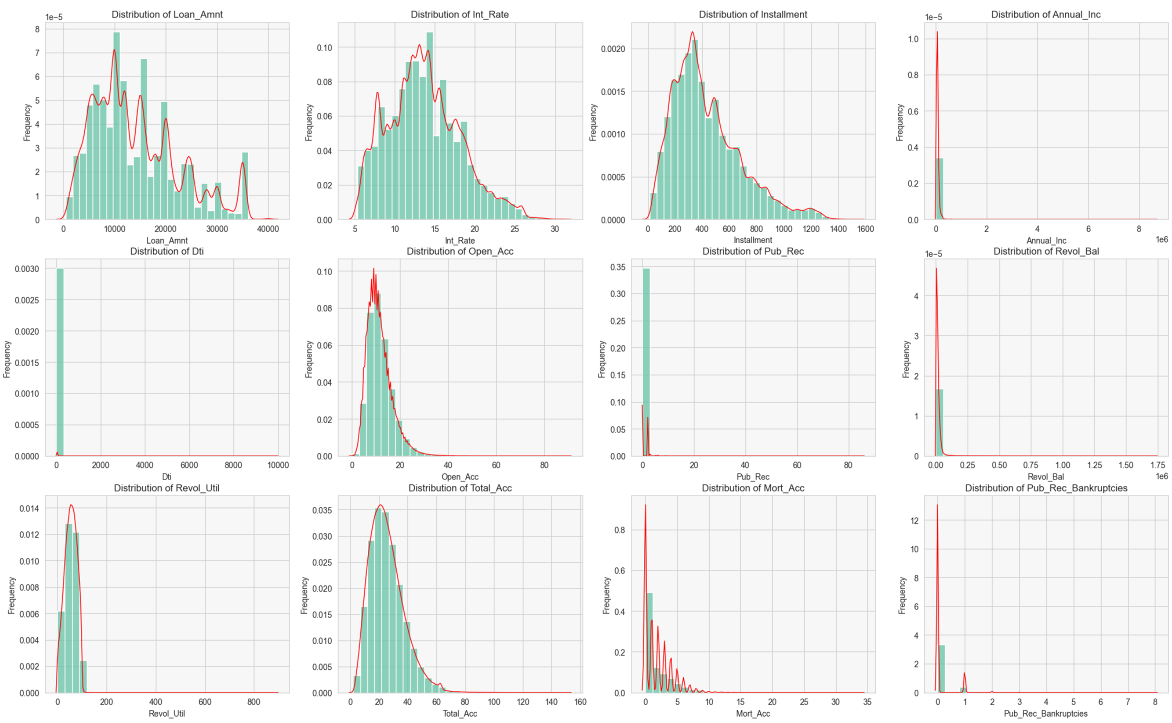
```
loantap.loc[loantap['city'].isna(),'city'] = loantap.loc[loantap['city'].isna(),
loantap.loc[loantap['state'].isna(),'state'] = loantap.loc[loantap['state'].isna
loantap.loc[loantap['pincode'].isna(),'pincode'] = loantap.loc[loantap['pincode'
```

In [18]:
```
target_var = 'loan_status'
continuous_vars = loantap.columns[loantap.dtypes != 'object'].to_list()
categorical_vars = loantap.columns[loantap.dtypes == 'object'].to_list()
categorical_vars.remove(target_var)
temp_vars = loantap.columns[(loantap.dtypes == 'object') & (loantap.nunique() <
```

Univariate Analysis

Continuous Variables:-

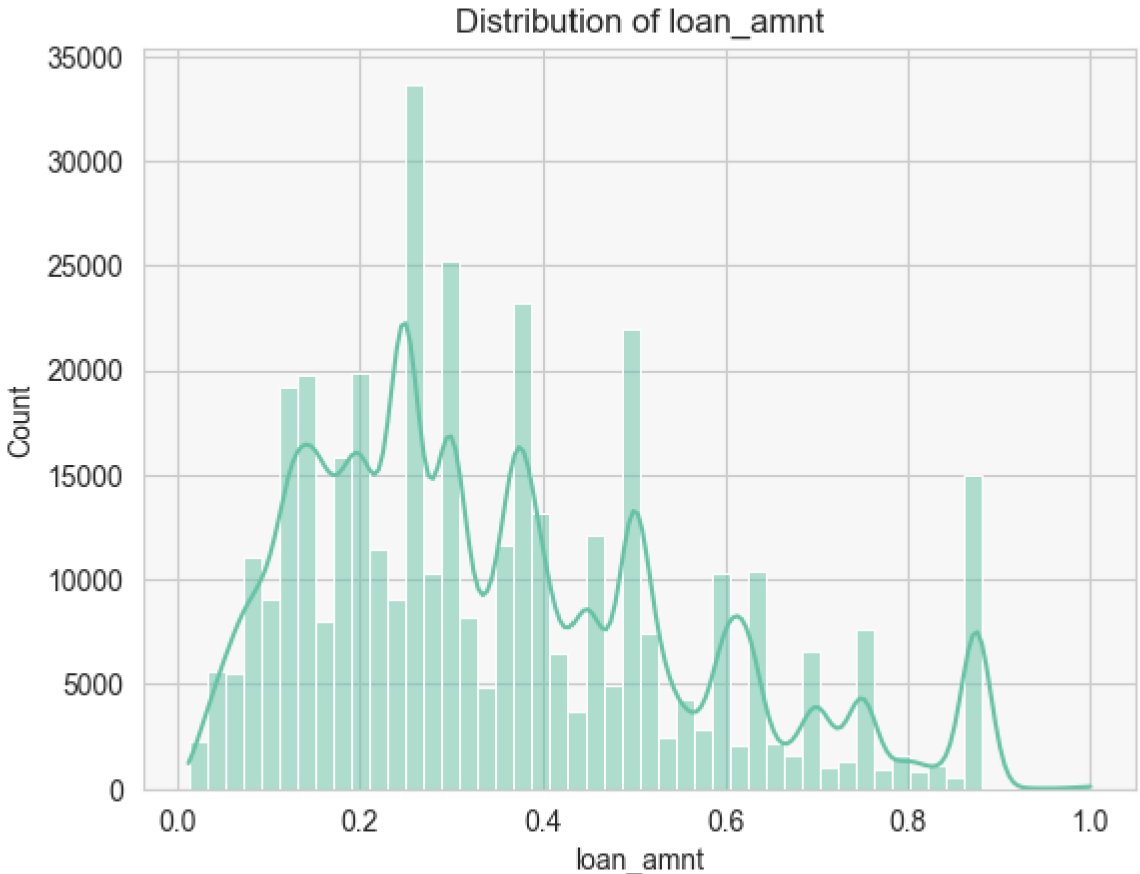
In [19]:
```
plt.figure(figsize=(25,15))

for i,var in enumerate(continuous_vars,1):
    plt.subplot(3,4,i)

    sns.histplot(loantap[var], bins=30,kde=False, label = var.title(), stat='den
    sns.kdeplot(loantap[var], color="r", lw=1, warn_singular=False)

    plt.title(f'Distribution of {var.title()}')
    plt.xlabel(var.title())
    plt.ylabel('Frequency')

plt.show()
```

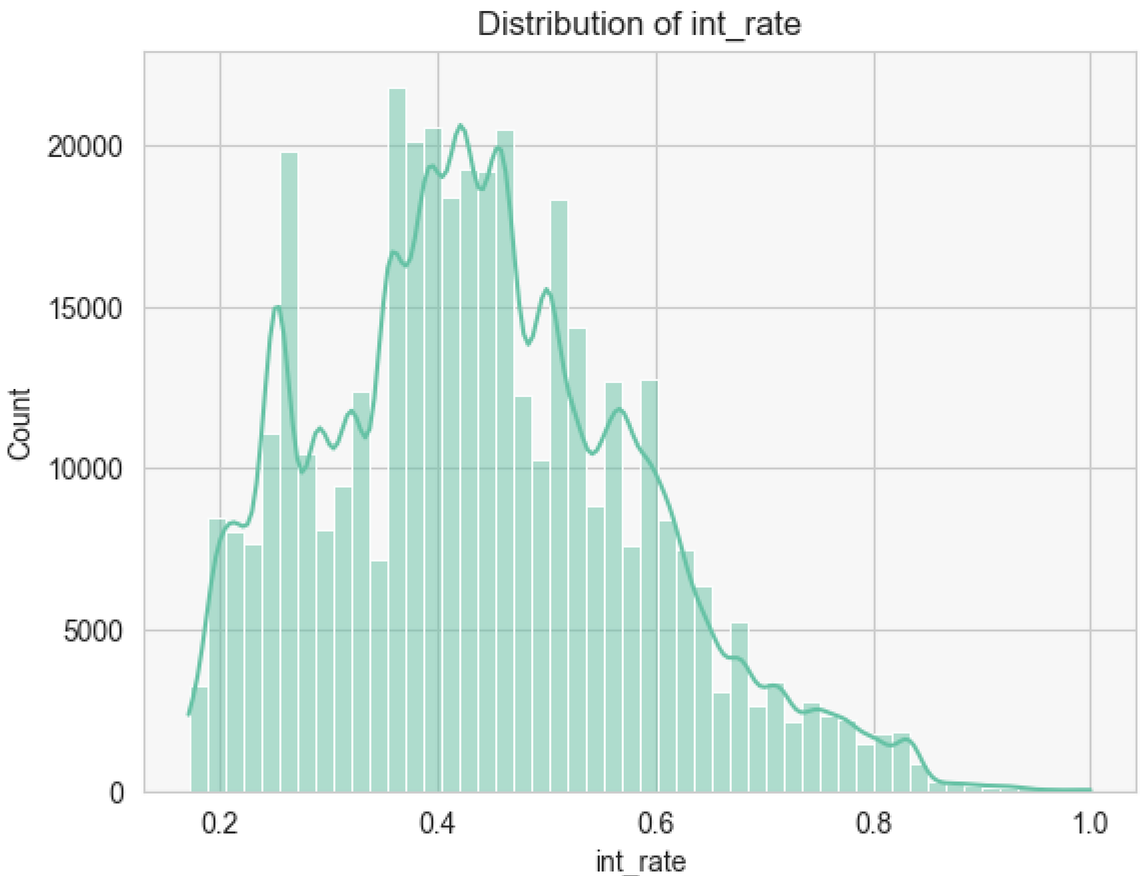


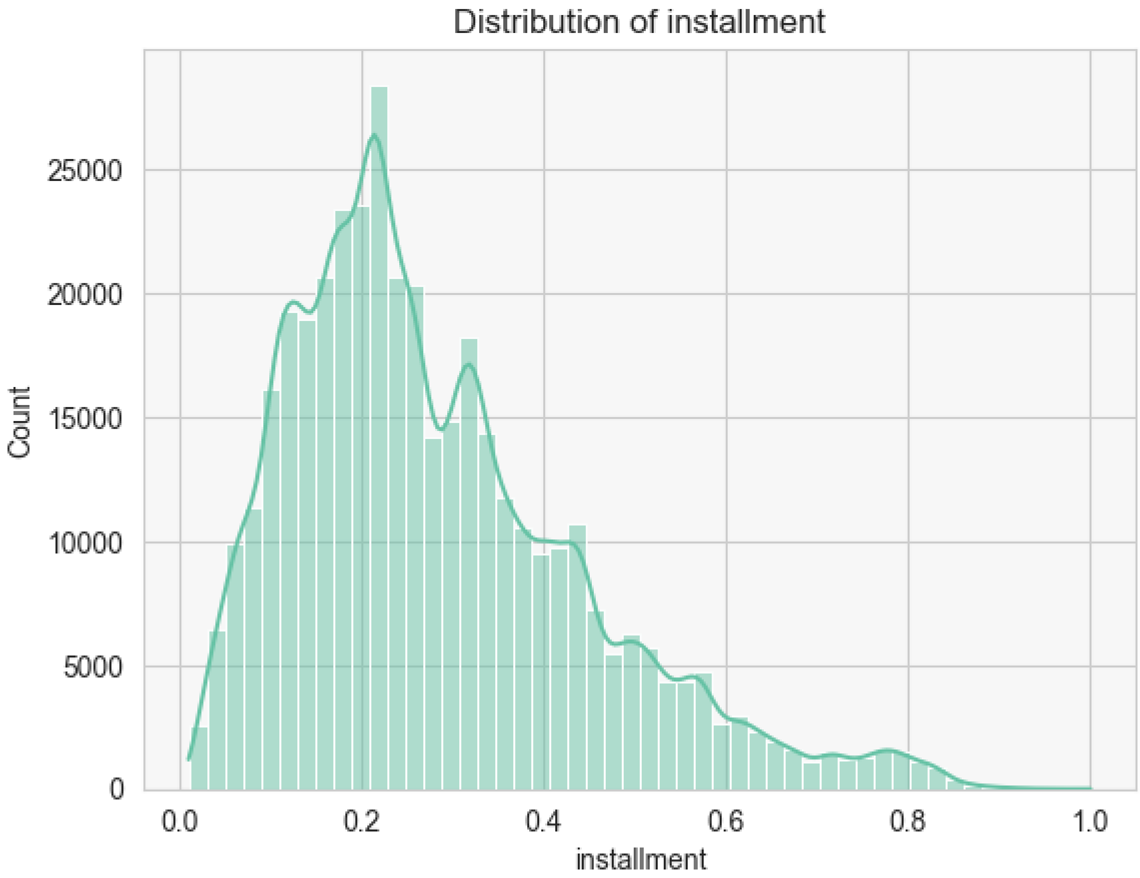
Visualization - Univariate Analysis

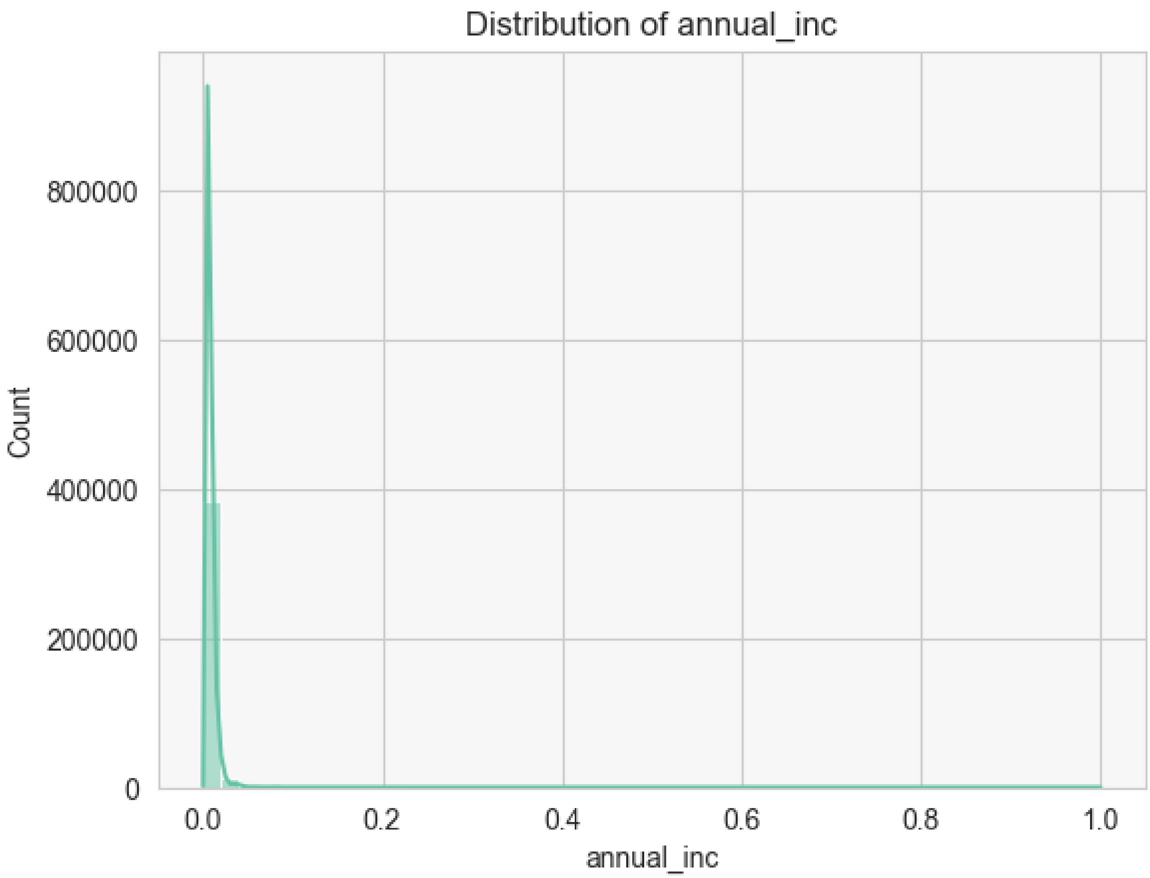
In [20]:
```
num_vars = loantap.select_dtypes('float64').columns.tolist()
```

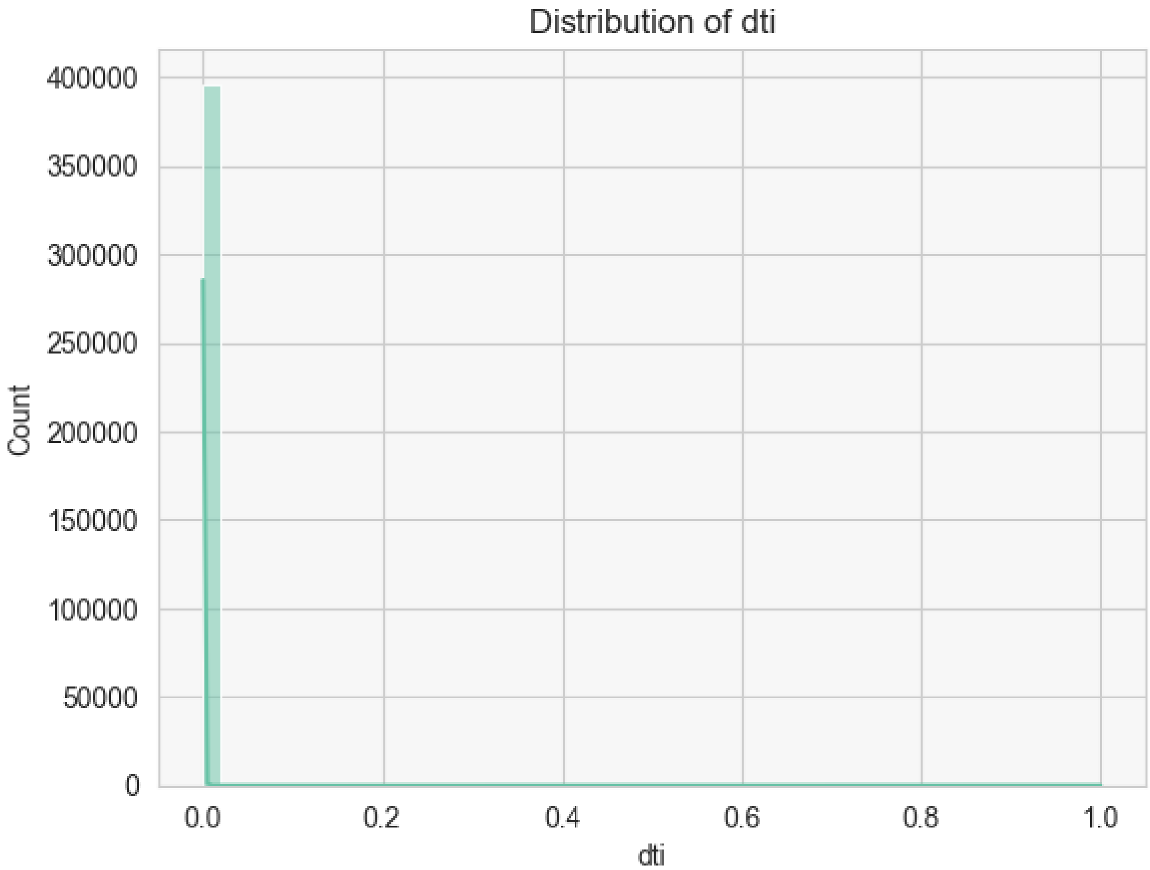
In [21]:
```
for i in num_vars:
#     plt.figure(figsize=(12,5))
    plt.title("Distribution of {}".format(i))
    sns.histplot(loantap[i]/loantap[i].max(), kde=True, bins=50)
    plt.show()
```



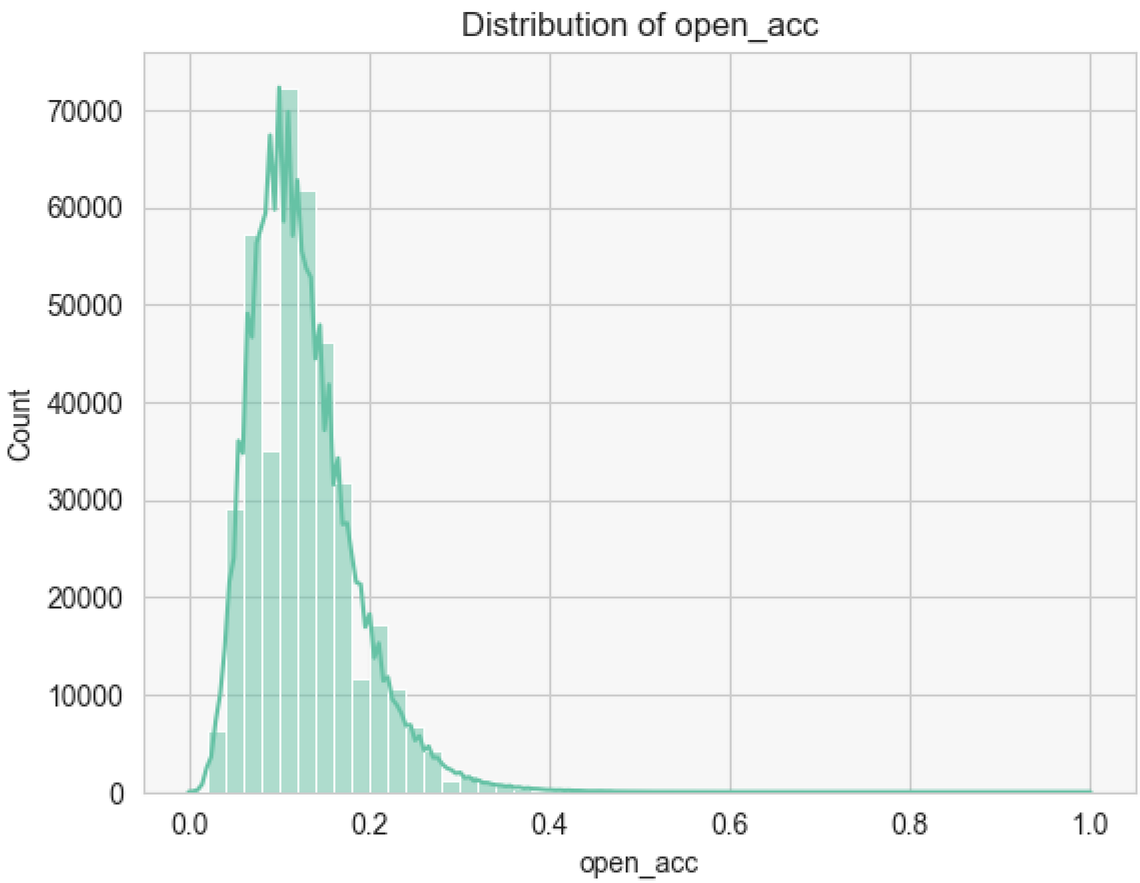



Distribution of installment
Count
installment
0
5000
10000
15000
20000
25000
0.0
0.2
0.4
0.6
0.8
1.0

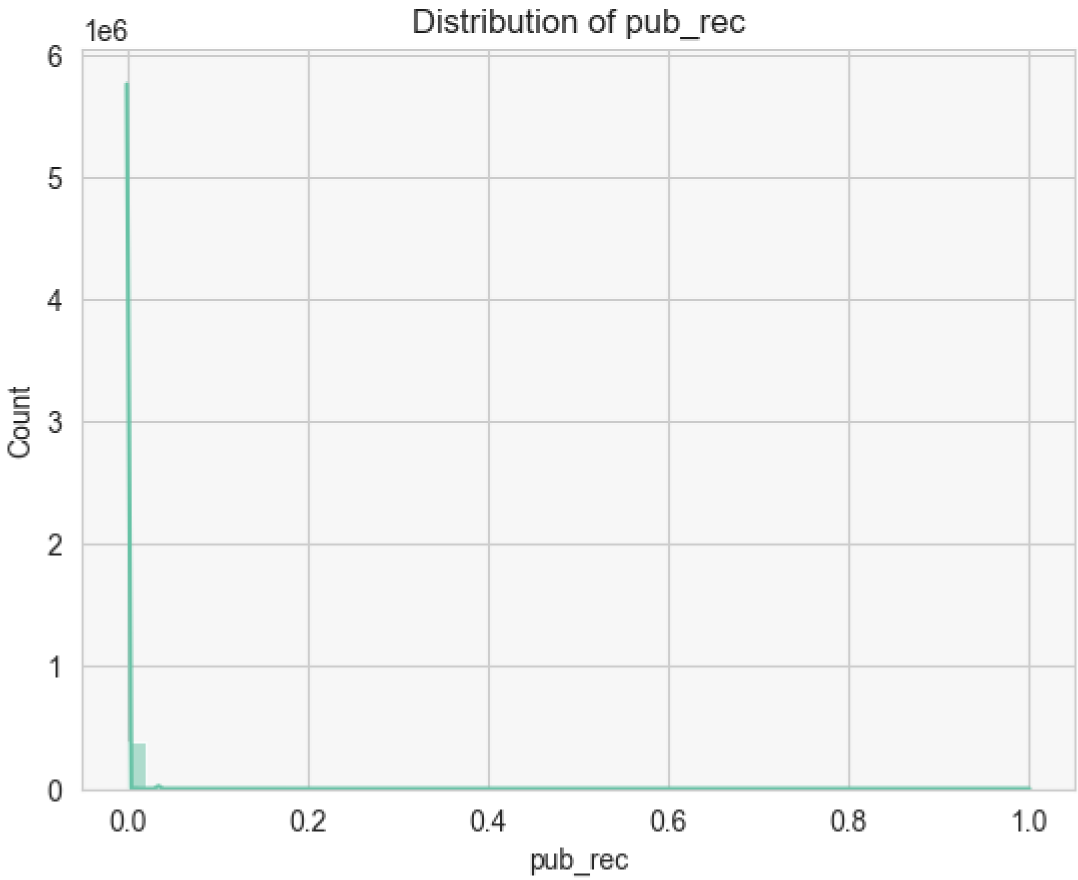

Distribution of annual_inc
Count
annual_inc
0
200000
400000
600000
800000
0.0
0.2
0.4
0.6
0.8
1.0

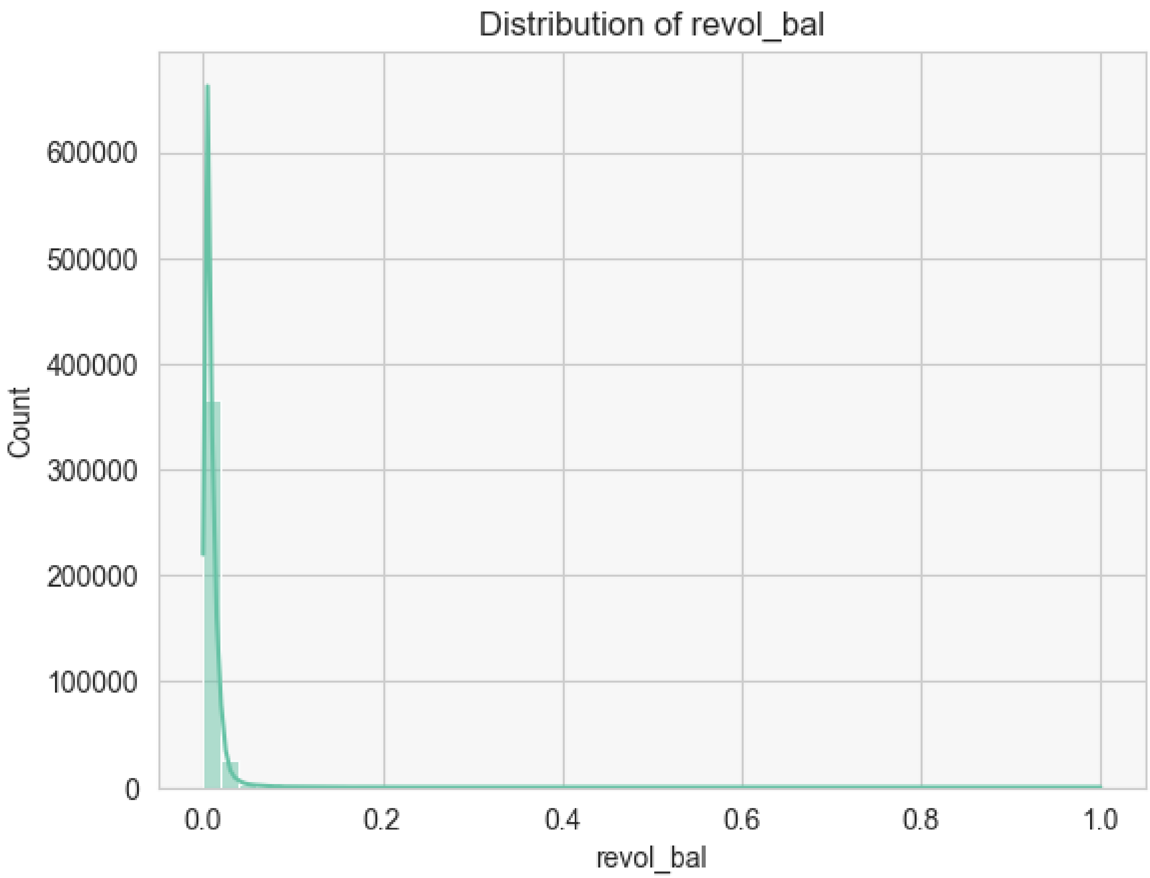


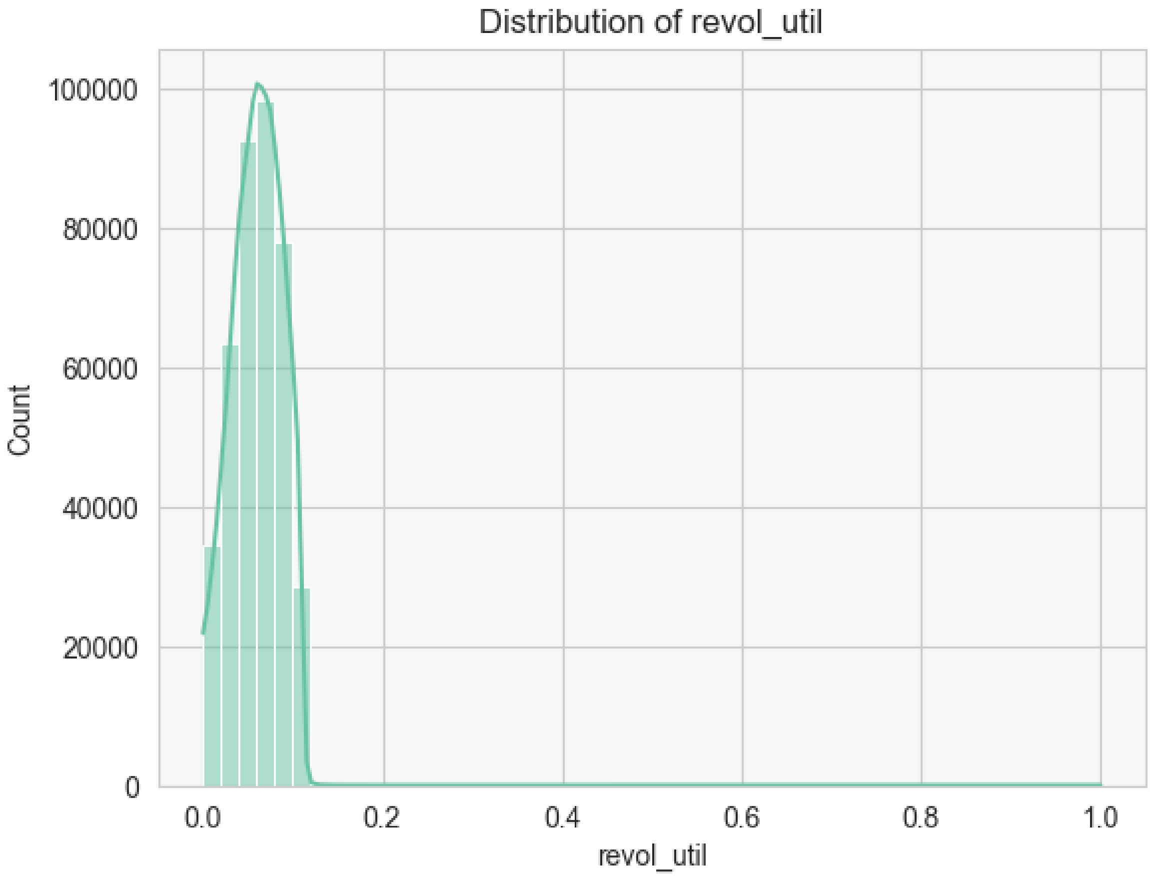
Distribution of dti
400000
350000
300000
250000
200000
150000
100000
50000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
dti

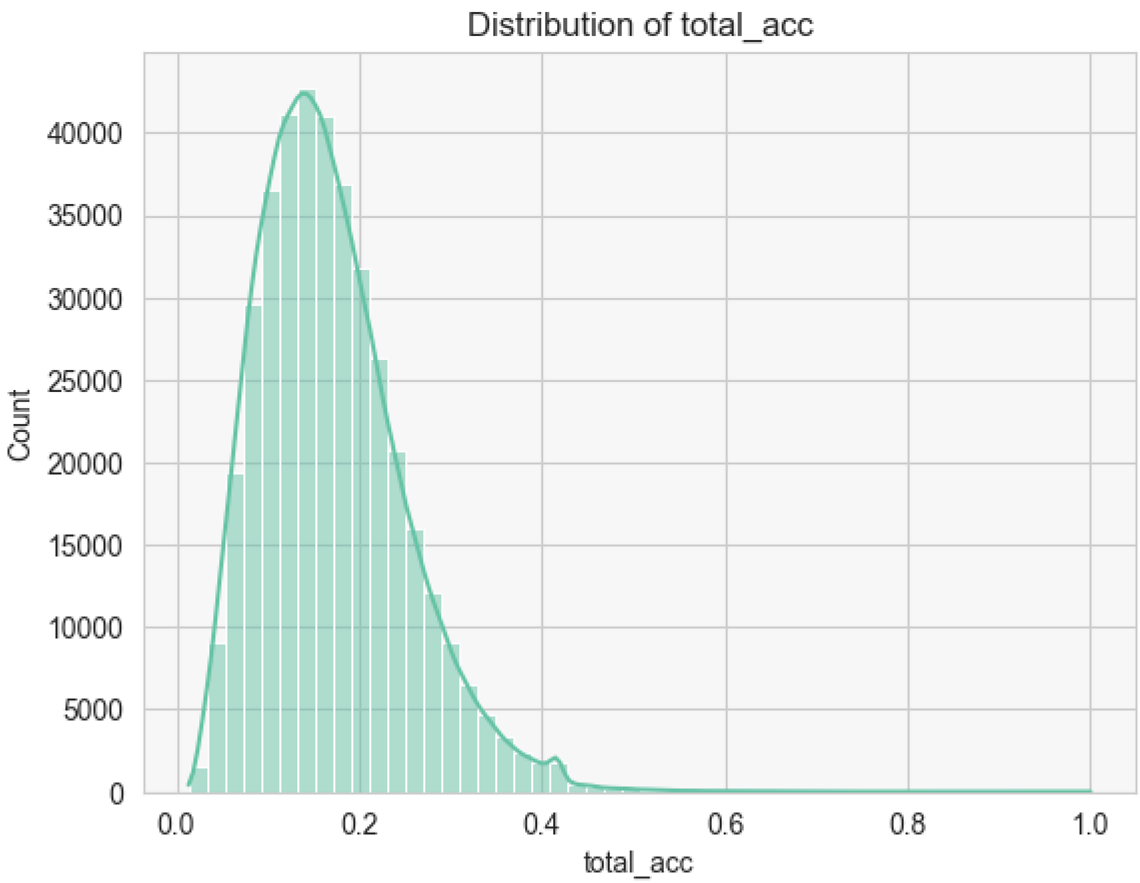


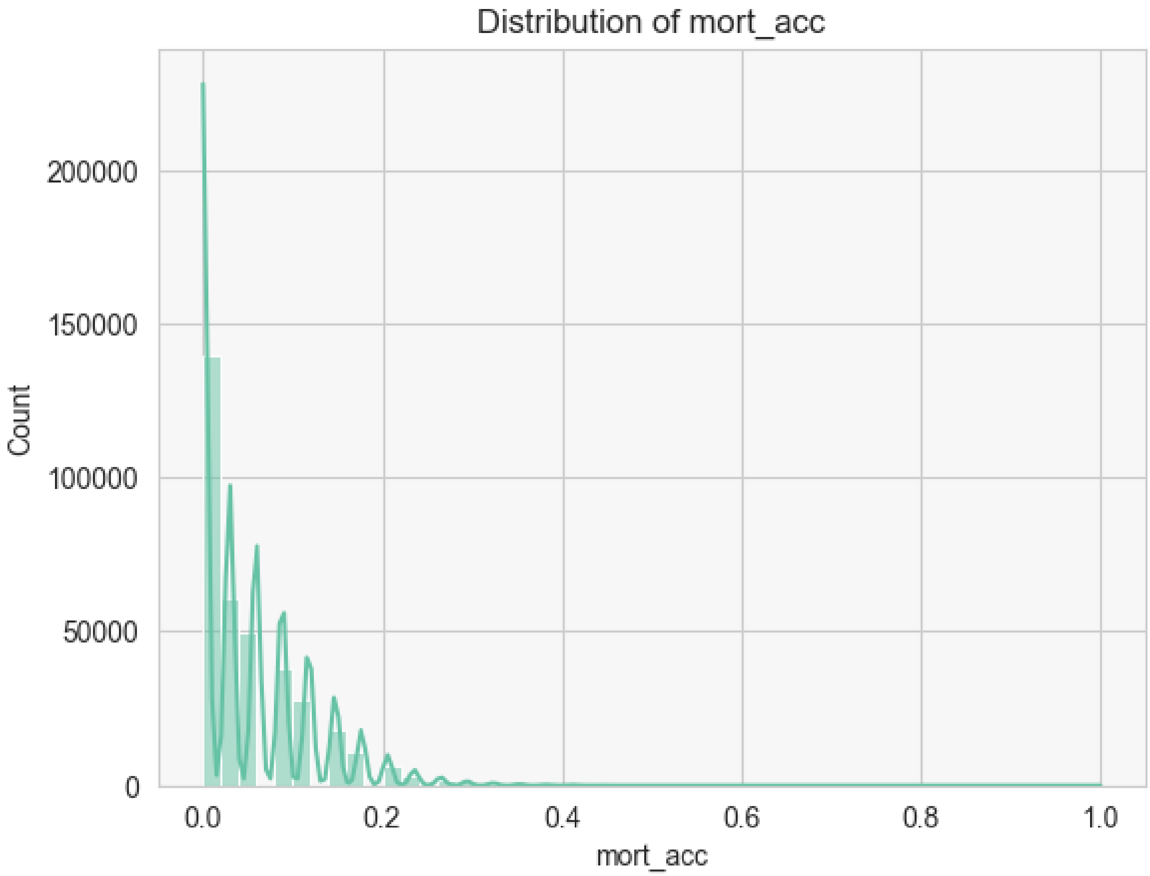
Distribution of open_acc
70000
60000
50000
40000
30000
20000
10000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
open_acc

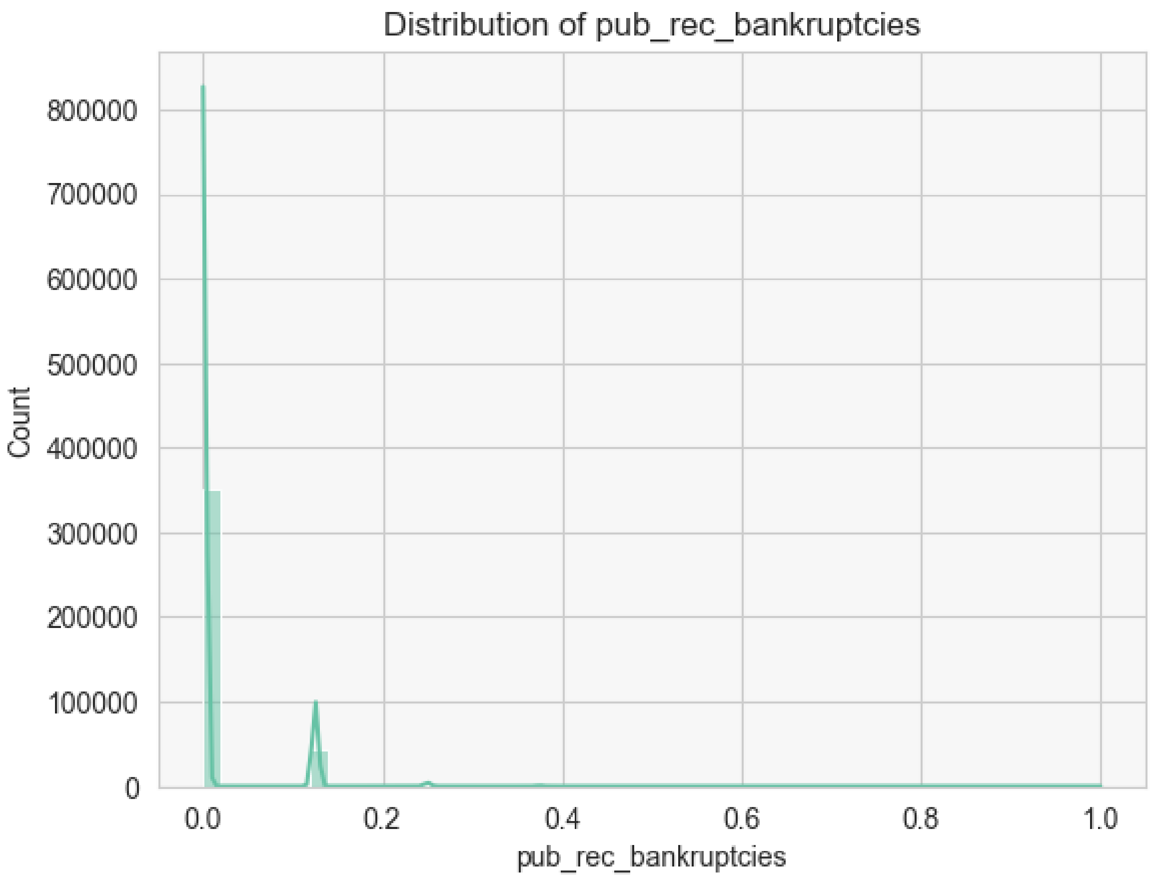

Distribution of pub_rec
1e6
6
5
4
3
2
1
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
pub_rec


Distribution of revol_bal
600000
500000
400000
300000
200000
100000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
revol_bal


Distribution of revol_util
Count
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0
revol_util


Distribution of total_acc
Count
40000
35000
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0
total_acc





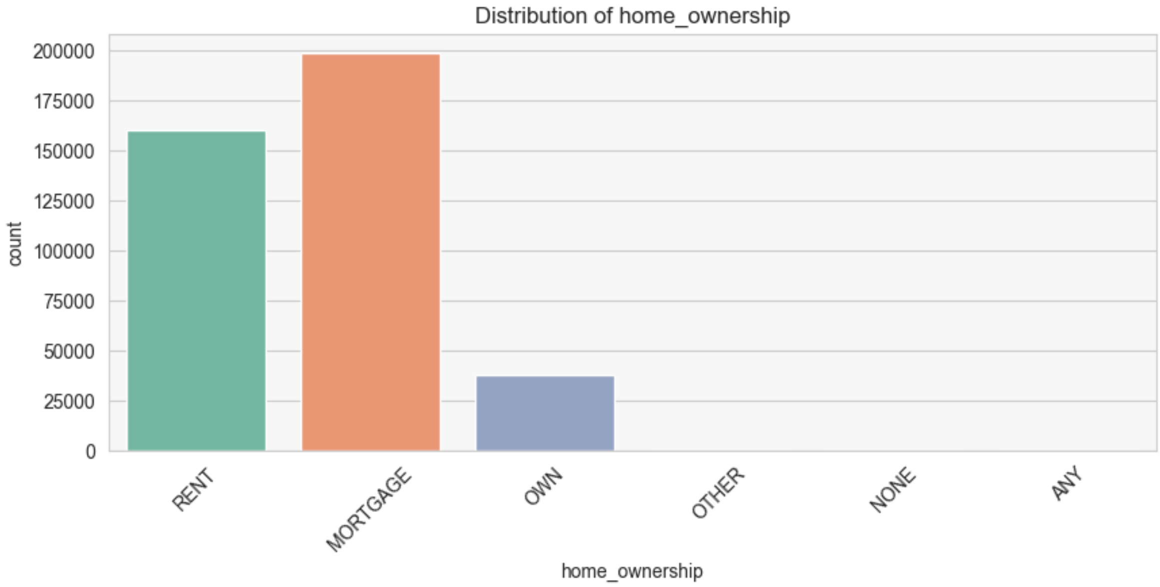
Insights

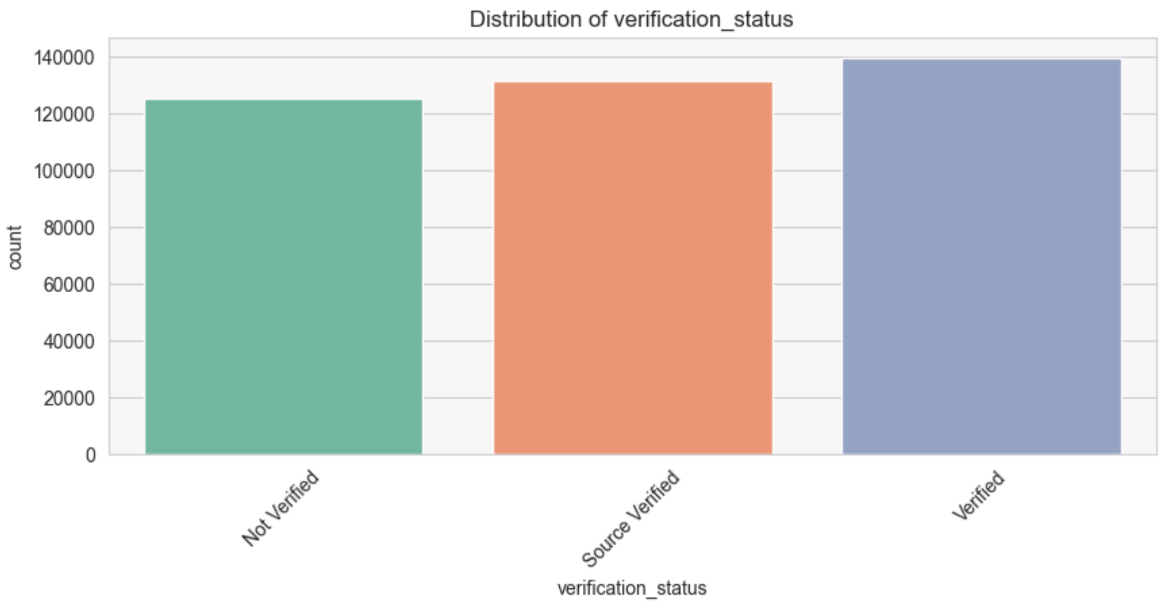
- Most of the distribution is highly skewed which tells us that they might contain outliers

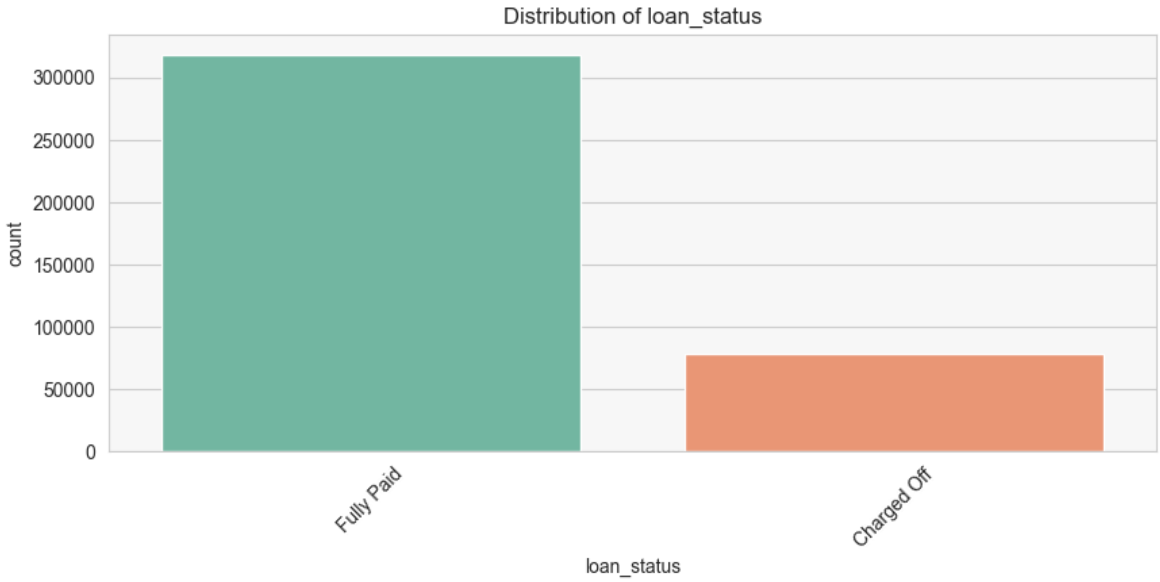
- Almost all the continuous features have outliers present in the dataset.
- **Log-normal distributions**: Int_Rate, Installment, Annual_Inc, Open_Acc, Revol_Bal, and Total_Acc.
- **Discrete distributions**: Open Acc, Pub_Rec, Mort_Acc, and Pub_Rec_Bankruptcies.

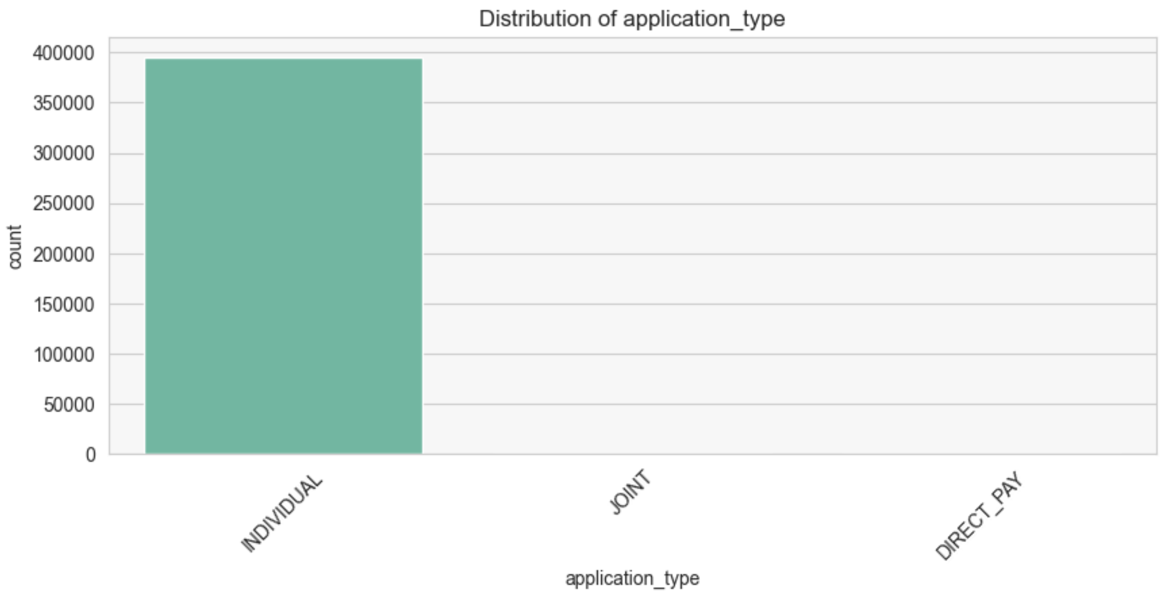
Please note, determining the exact nature of a distribution based on visual inspection is subjective, and statistical tests can be employed for a more precise classification.

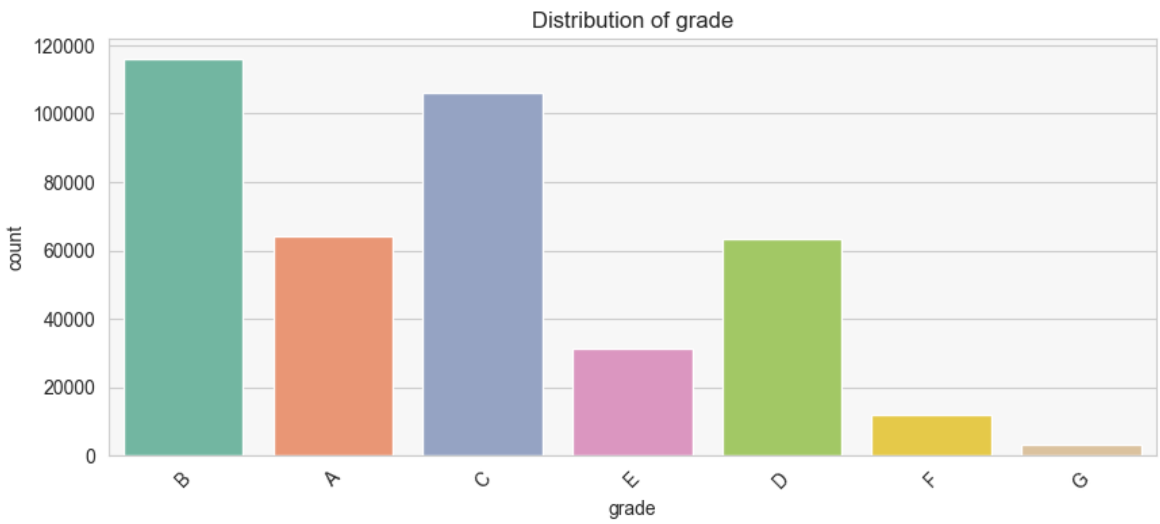
In [22]:
```
cat_vars = ['home_ownership', 'verification_status', 'loan_status', 'application
for i in cat_vars:
    plt.figure(figsize=(10, 4))
    plt.title(f'Distribution of {i}')
    sns.countplot(data=loantap, x=i)
    plt.xticks(rotation = 45)
    plt.show()
```

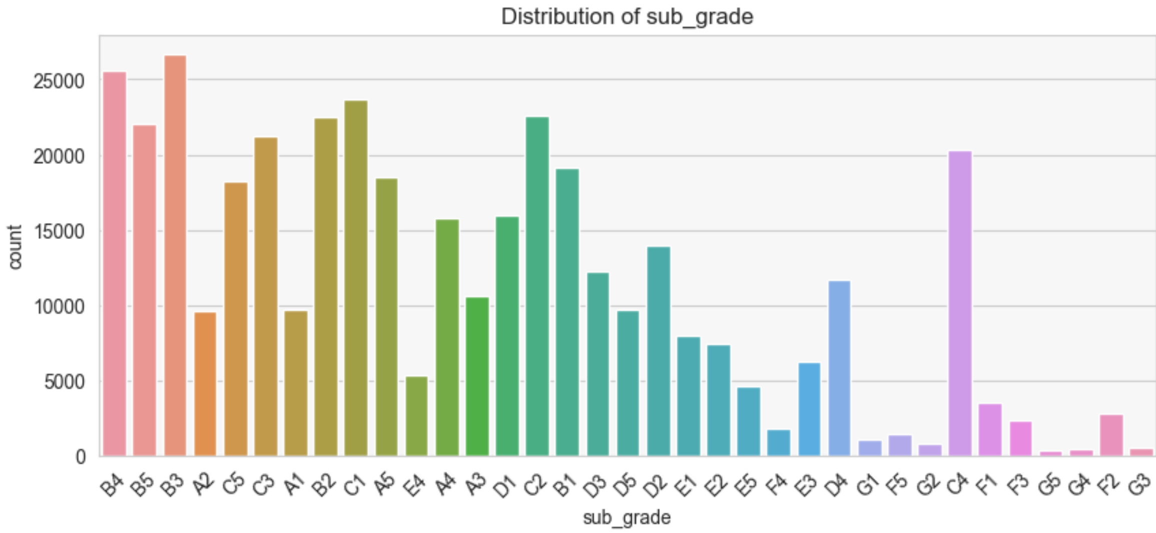


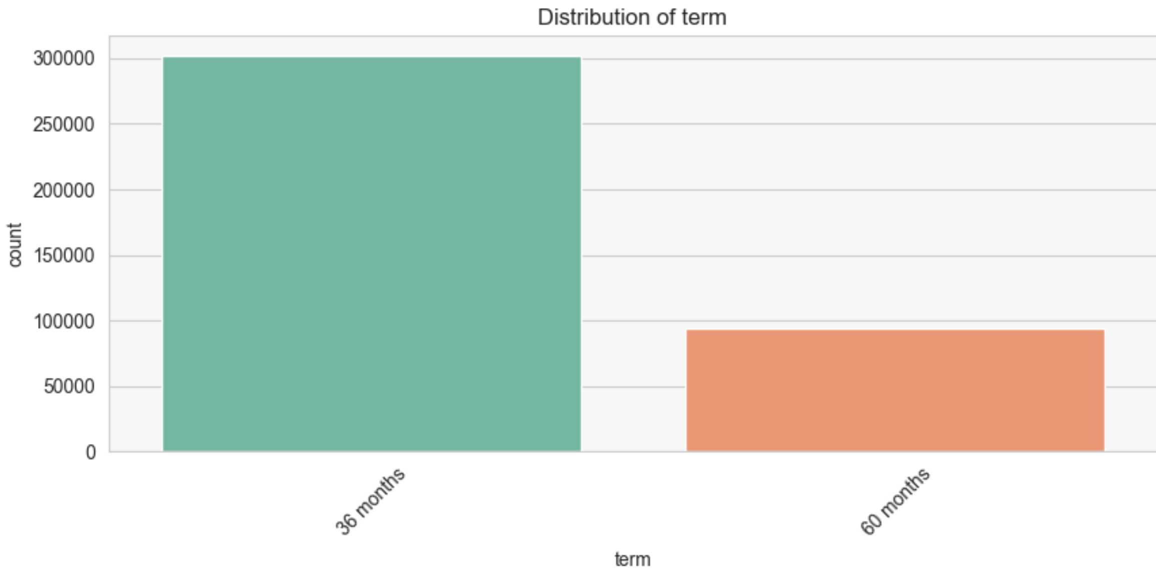



Distribution of loan_status
count
300000
250000
200000
150000
100000
50000
0
Fully Paid
Charged Off
loan_status


Distribution of application_type
count
400000
350000
300000
250000
200000
150000
100000
50000
0
INDIVIDUAL
JOINT
DIRECT_PAY
application_type


Distribution of grade
count
120000
100000
80000
60000
40000
20000
0
B
A
C
E
D
F
G
grade





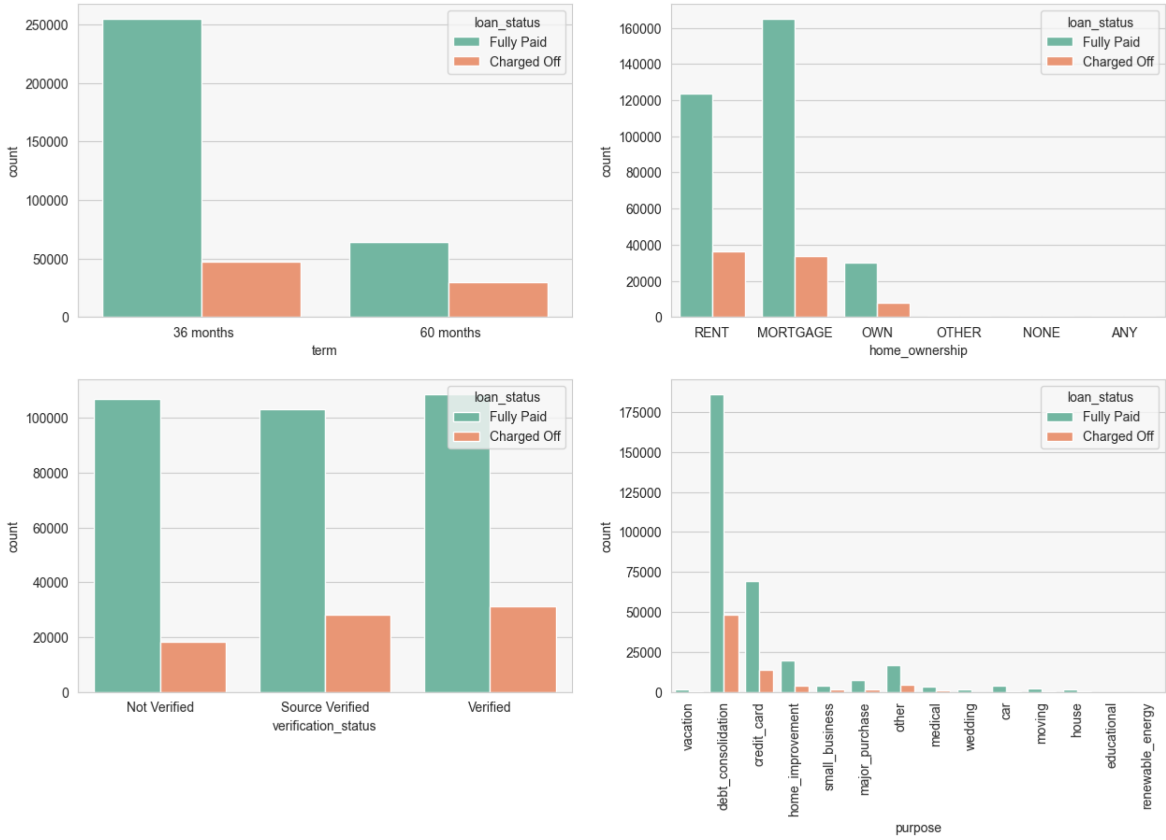
Insights

- All the application type is Individual
- Most of the loan tenure is disbursed for 36 months
- The grade of majority of people those who have took the loan is 'B' and have subgrade 'B3'.
- So from that we can infer that people with grade 'B' and subgrade 'B3' are more likely to fully pay the loan.

Visualization - Bivariate Analysis

In [23]:
```
plt.figure(figsize=(15,20))

plt.subplot(4,2,1)
sns.countplot(x='term',data=loantap,hue='loan_status')

plt.subplot(4,2,2)
sns.countplot(x='home_ownership',data=loantap,hue='loan_status')

plt.subplot(4,2,3)
sns.countplot(x='verification_status',data=loantap,hue='loan_status')
```

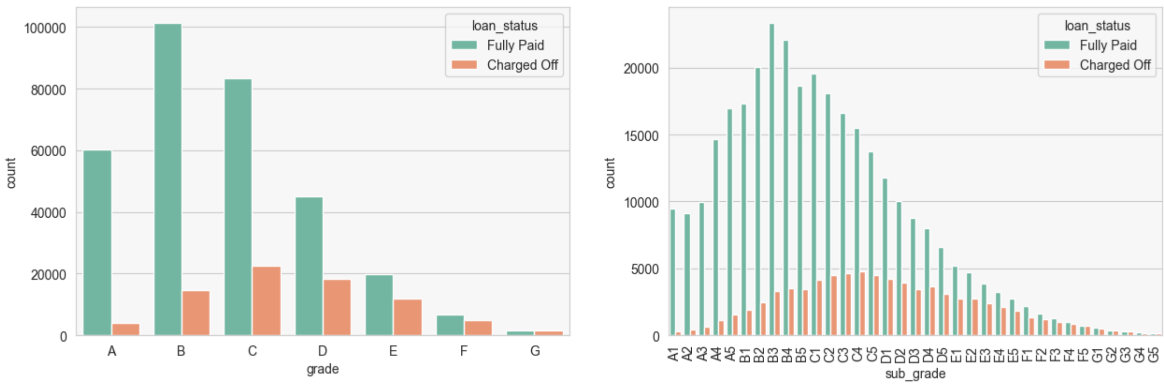
```
plt.subplot(4,2,4)
g=sns.countplot(x='purpose',data=loantap,hue='loan_status')
g.set_xticklabels(g.get_xticklabels(),rotation=90)

plt.show()
```



Insights

- Most of the people took loan for 36 months and full paid on time
- Most of people have home ownership as mortgage and rent
- Most of the people took loan for debt consolidations

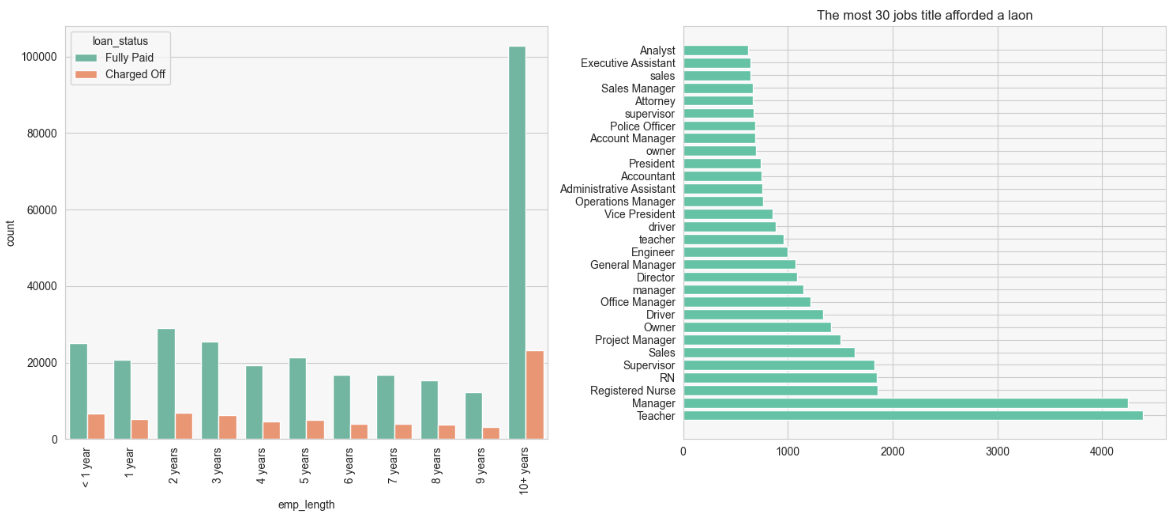
In [24]:
```
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
grade = sorted(loantap.grade.unique().tolist())
sns.countplot(x='grade', data=loantap, hue='loan_status', order=grade)

plt.subplot(2, 2, 2)
sub_grade = sorted(loantap.sub_grade.unique().tolist())
g = sns.countplot(x='sub_grade', data=loantap, hue='loan_status', order=sub_grad
g.set_xticklabels(g.get_xticklabels(), rotation=90)

plt.show()
```



Insights

- The grade of majority of people those who have fully paid the loan is 'B' and have subgrade 'B3'.
- So from that we can infer that people with grade 'B' and subgrade 'B3' are more likely to fully pay the loan.

In [25]:
```
plt.figure(figsize=(15,12))

plt.subplot(2,2,1)
order = ['< 1 year', '1 year', '2 years', '3 years', '4 years', '5 years',
          '6 years', '7 years', '8 years', '9 years', '10+ years',]
g=sns.countplot(x='emp_length',data=loantap,hue='loan_status',order=order)
g.set_xticklabels(g.get_xticklabels(),rotation=90)

plt.subplot(2,2,2)
plt.barh(loantap.emp_title.value_counts()[:30].index,loantap.emp_title.value_cou
plt.title("The most 30 jobs title afforded a laon")
plt.tight_layout()

plt.show()
```



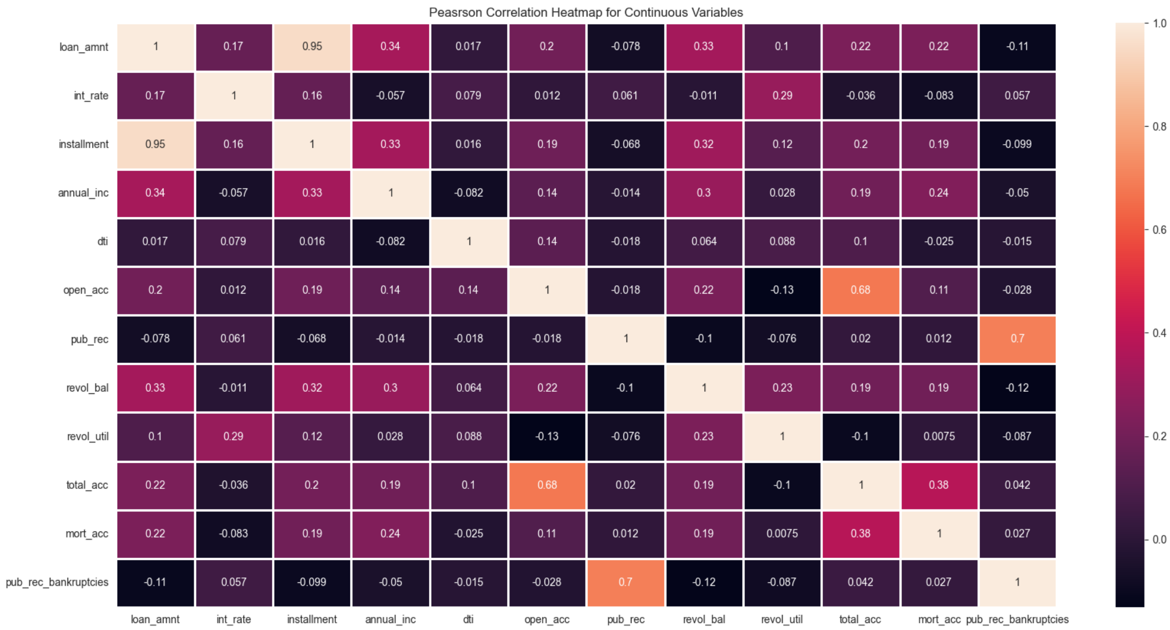
Insights

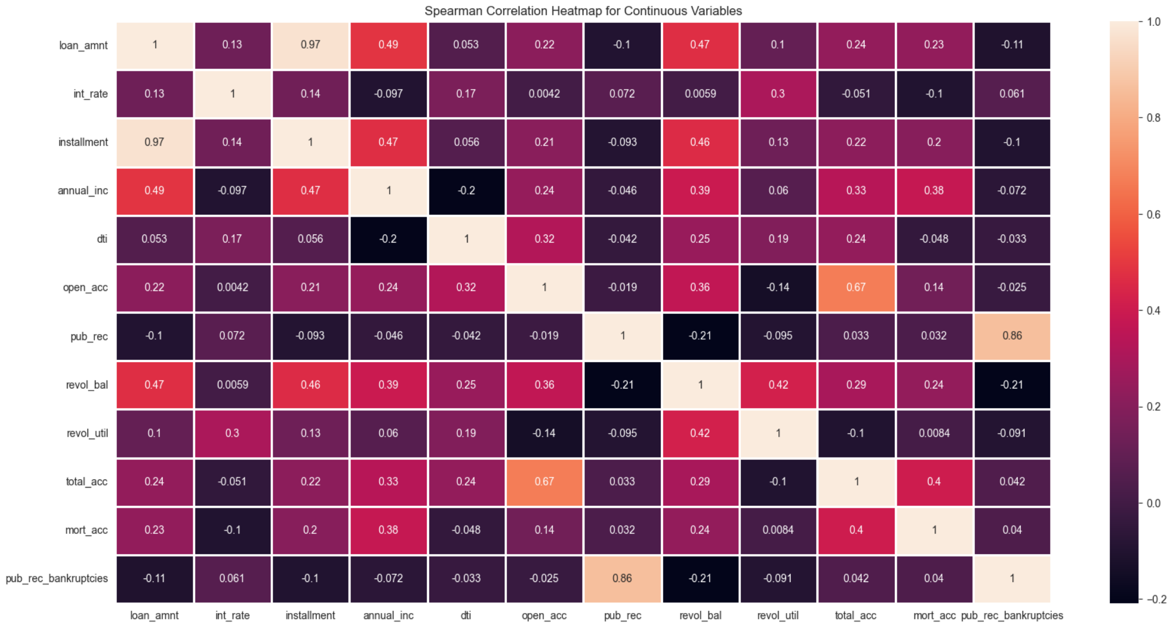
- Manager and Teacher are the most afforded loan on titles
- Person who employed for more than 10 years has successfully paid of the loan

Multivariate Analysis

In [26]:

```python
pearson_corr_data = loantap[continuous_vars].corr()
spearman_corr_data = loantap[continuous_vars].corr(method='spearman')

plt.figure(figsize=(20,10))
sns.heatmap(
    pearson_corr_data,
    annot=True,
    linewidth = 1
)
plt.title("Peasrson Correlation Heatmap for Continuous Variables")


plt.figure(figsize=(20,10))
sns.heatmap(
    spearman_corr_data,
    annot=True,
    linewidth = 1
)
plt.title("Spearman Correlation Heatmap for Continuous Variables")
plt.show()
```





Pearson Correlation:

1. **High Correlation:**
    - `loan_amnt` and `installment` : 0.95 - This suggests that the loan amount and installment have a very strong positive linear relationship. As the loan amount increases, the installment amount also tends to increase.
2. **Low Correlation:**
    - Variables like `int_rate` with most of the other variables have low correlations. This suggests interest rate doesn't have a strong linear relationship with many of the variables.

Spearman Correlation:

1. **High Correlation:**
    - `loan_amnt` and `installment` : 0.97 - This relationship remains consistent with the Pearson correlation, suggesting a strong monotonic relationship between loan amount and installment
2. **High Negative Correlation:**
    - `pub_rec` and `pub_rec_bankruptcies` : 0.86 - This suggests that there's a strong monotonic relationship between the number of derogatory public records and the number of public record bankruptcies.

- While Pearson correlation measures linear relationships, Spearman measures monotonic relationships. Some variables may have a stronger Spearman correlation than Pearson if their relationship is monotonic but not strictly linear.
- The difference in correlation values between Pearson and Spearman for certain pairs suggests that there might be non-linear monotonic relationships among some variables.

Data Preprocessing

Feature Engineering

In [27]:

```
from statsmodels.graphics.gofplots import qqplot
```

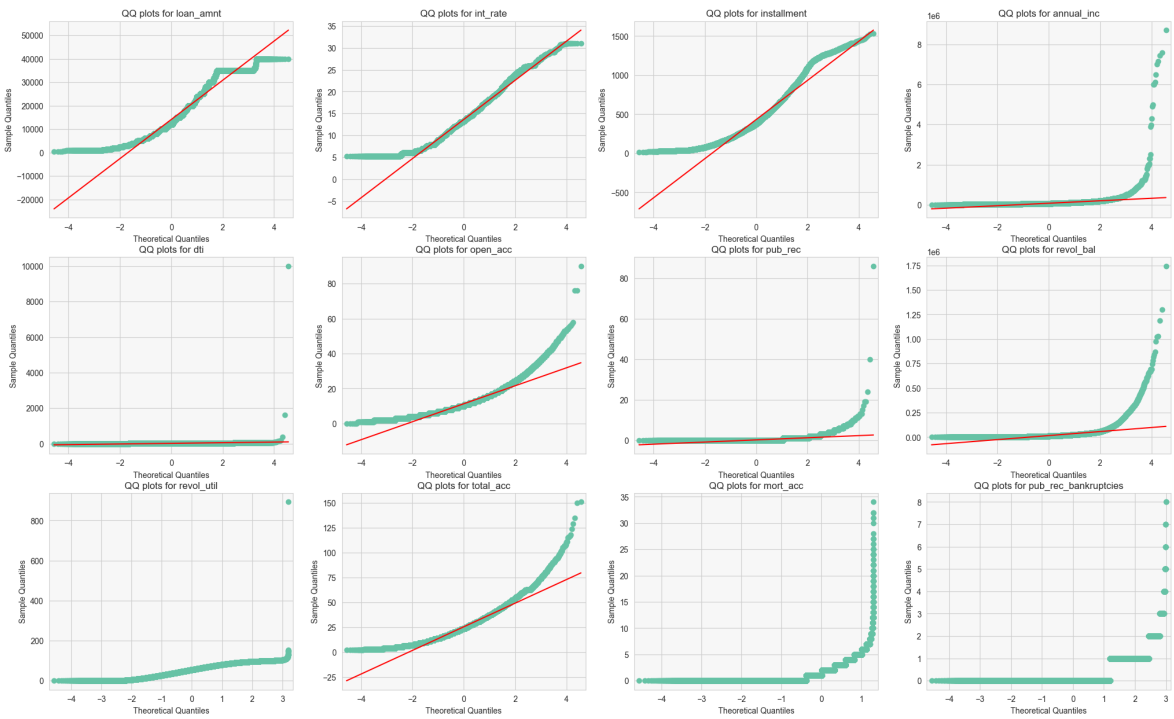
In [28]:

```
log_normal_vars = [
      'loan_amnt',
      'int_rate',
      'installment',
      'annual_inc',
      'total_acc',
      'open_acc'
]
```

Before Log Transformation:-

In [29]:

```
plt.figure(figsize = (25, 15))

for i,var in enumerate(continuous_vars,1):

    plt.subplot(3, 4, i)
    plt.title(f'QQ plots for {var}')
    qqplot(loantap[var], line='s', ax=plt.gca())

plt.show()
```

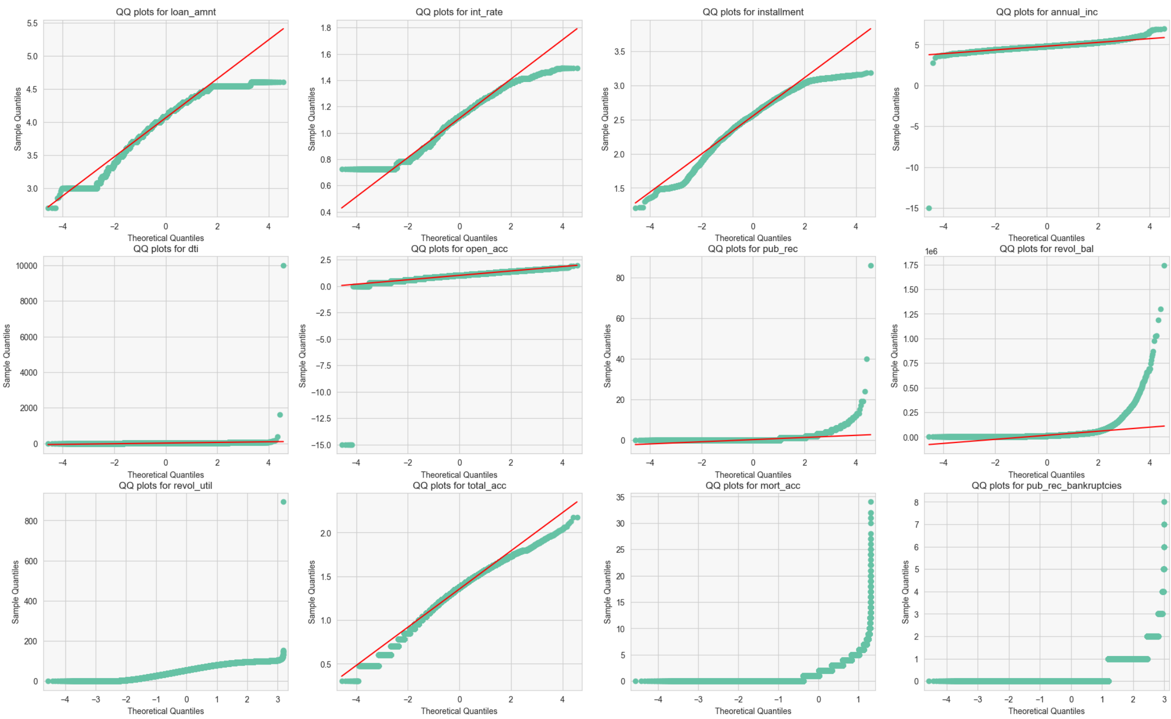


After Log Transformation:-

In [30]:

```
for var in log_normal_vars:
    loantap[var] = np.log10(loantap[var]+1e-15)
```

In [31]:
```
plt.figure(figsize = (25, 15))

for i,var in enumerate(continuous_vars,1):

    plt.subplot(3, 4, i)
    plt.title(f'QQ plots for {var}')
    qqplot(loantap[var], line='s', ax=plt.gca())

plt.show()
```



- `int_rate` and `annual income` are near to normal

Statistically Insignificant Features: ['revol_bal', 'pub_rec_bankruptcies', 'initial_list_status', 'state']

Missing Values Imputation

In [32]:
```
missing_vars
```

Out[32]:
```
emp_title               5.79
emp_length              4.62
title                   0.44
revol_util              0.07
mort_acc                9.54
pub_rec_bankruptcies    0.14
dtype: float64
```

emp_title Impute:-

In [33]:
```
mode = lambda x: x.mode().iloc[0] if not x.mode().empty else 'Unknown'

emp_title_impute_loantap = loantap.groupby(['state', 'home_ownership', 'grade'])
```

```
def emp_title_imputer(x):
    x['emp_title'] = emp_title_impute_loantap.loc[(x['state'],x['home_ownership'
    return x['emp_title']

loantap.loc[loantap['emp_title'].isna(),'emp_title'] = loantap.loc[loantap['emp_
```

emp_length Impute:-

In [34]:
```
mode = lambda x: x.mode().iloc[0] if not x.mode().empty else 'Unknown'

emp_length_impute_loantap = loantap.groupby(['state', 'home_ownership', 'grade']

def emp_length_imputer(x):
    x['emp_length'] = emp_length_impute_loantap.loc[(x['state'],x['home_ownershi
    return x['emp_length']

loantap.loc[loantap['emp_length'].isna(),'emp_length'] = loantap.loc[loantap['em
```

title Impute:-

In [35]:
```
mode = lambda x: x.mode().iloc[0] if not x.mode().empty else 'Unknown'

title_impute_loantap = loantap.groupby(['purpose'])['emp_length'].apply(mode)

def title_imputer(x):
    x['title'] = title_impute_loantap.loc[x['purpose']]
    return x['title']

loantap.loc[loantap['title'].isna(),'title'] = loantap.loc[loantap['title'].isna
```

revol_util Impute:-

In [36]:
```
revol_util_impute_loantap = loantap.groupby(['state','home_ownership','grade','s

def revol_util_imputer(x):
    x['revol_util'] = revol_util_impute_loantap.loc[(x['state'],x['home_ownershi
    return x['revol_util']

loantap.loc[loantap['revol_util'].isna(),'revol_util'] = loantap.loc[loantap['re
```

In [37]:
```
revol_util_impute_loantap = loantap.groupby(['state'])['revol_util'].median()

def revol_util_imputer(x):
    x['revol_util'] = revol_util_impute_loantap.loc[(x['state'])]
    return x['revol_util']

loantap.loc[loantap['revol_util'].isna(),'revol_util'] = loantap.loc[loantap['re
```

mort_acc Impute:-

In [38]:
```
bin_edges = [0, 5000, 15000, 25000, 50000, 75000, 100000, 200000] + list(range(3
loantap['annual_inc_bins'] = pd.cut(loantap['annual_inc'], bins=bin_edges, inclu
```

In [39]:
```
mort_acc_impute_loantap = loantap.groupby(['state','home_ownership','grade','ann

def mort_acc_imputer(x):
```

```
    x['mort_acc'] = mort_acc_impute_loantap.loc[(x['state'],x['home_ownership'],
    return x['mort_acc']

loantap.loc[loantap['mort_acc'].isna(),'mort_acc'] = loantap.loc[loantap['mort_a
```

In [40]:
```
mort_acc_impute_loantap = loantap.groupby(['state'])['mort_acc'].median()

def mort_acc_imputer(x):
    x['mort_acc'] = mort_acc_impute_loantap.loc[(x['state'])]
    return x['mort_acc']

loantap.loc[loantap['mort_acc'].isna(),'mort_acc'] = loantap.loc[loantap['mort_a
```

pub_rec_bankruptcies Impute:-

In [41]:
```
pub_rec_bankruptcies_impute_loantap = loantap.groupby(['state','home_ownership',

def pub_rec_bankruptcies_imputer(x):
    x['pub_rec_bankruptcies'] = pub_rec_bankruptcies_impute_loantap.loc[(x['stat
    return x['pub_rec_bankruptcies']

loantap.loc[loantap['pub_rec_bankruptcies'].isna(),'pub_rec_bankruptcies'] = loa
```

In [42]:
```
pub_rec_bankruptcies_impute_loantap = loantap.groupby(['state'])['pub_rec_bankru

def pub_rec_bankruptcies_imputer(x):
    x['pub_rec_bankruptcies'] = pub_rec_bankruptcies_impute_loantap.loc[(x['stat
    return x['pub_rec_bankruptcies']

loantap.loc[loantap['pub_rec_bankruptcies'].isna(),'pub_rec_bankruptcies'] = loa
```

Outliers Treatment

In [43]:
```
print('Outliers Detection:- ')
print(50*'-')
for i,var in enumerate(loantap.select_dtypes(include=['number']).columns.tolist(
    q1 = np.quantile(loantap[var],0.25)
    q3 = np.quantile(loantap[var],0.75)

    iqr = q3 - q1

    upper_limit = q3 + 1.5 *iqr
    lower_limit = max(q1 - 1.5 *iqr,0)

    total_length = len(loantap)
    upper_outliers = len(loantap.loc[loantap[var] < lower_limit, var])
    lower_outliers = len(loantap.loc[loantap[var] > upper_limit, var])
    total_outliers = len(
        loantap.loc[(loantap[var] > upper_limit) | (loantap[var] < lower_limit),
    )

    upper_outliers_perc = round((upper_outliers/total_length) * 100,2)
    lower_outliers_perc = round((lower_outliers/total_length) * 100,2)

    total_outliers_perc = round((total_outliers/total_length) * 100,2)
```

```
    print(var.title(),":-")
    print(f'Percent of outliers in lower region: {upper_outliers_perc} %')
    print(f'Percent of outliers in upper region: {lower_outliers_perc} %')
    print(f'Total Outliers: {total_outliers_perc} %',end='\n\n')
```

```
Outliers Detection:-
--------------------------------------------------
Loan_Amnt :-
Percent of outliers in lower region: 1.98 %
Percent of outliers in upper region: 0.0 %
Total Outliers: 1.98 %

Int_Rate :-
Percent of outliers in lower region: 0.62 %
Percent of outliers in upper region: 0.0 %
Total Outliers: 0.62 %

Installment :-
Percent of outliers in lower region: 2.09 %
Percent of outliers in upper region: 0.0 %
Total Outliers: 2.09 %

Annual_Inc :-
Percent of outliers in lower region: 0.55 %
Percent of outliers in upper region: 0.82 %
Total Outliers: 1.38 %

Dti :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.07 %
Total Outliers: 0.07 %

Open_Acc :-
Percent of outliers in lower region: 1.6 %
Percent of outliers in upper region: 0.31 %
Total Outliers: 1.91 %

Pub_Rec :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 14.58 %
Total Outliers: 14.58 %

Revol_Bal :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 5.37 %
Total Outliers: 5.37 %

Revol_Util :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 0.0 %
Total Outliers: 0.0 %

Total_Acc :-
Percent of outliers in lower region: 1.65 %
Percent of outliers in upper region: 0.06 %
Total Outliers: 1.71 %

Mort_Acc :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 1.73 %
Total Outliers: 1.73 %

Pub_Rec_Bankruptcies :-
Percent of outliers in lower region: 0.0 %
Percent of outliers in upper region: 11.39 %
```

```
Total Outliers: 11.39 %
```

In [44]:
```python
def drop_outliers(loantap):
    numeric_cols = loantap.select_dtypes(include=['number']).columns.tolist()

    for col in numeric_cols:
        Q1 = loantap[col].quantile(0.25)
        Q3 = loantap[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        loantap_out = loantap.loc[(loantap[col] >= lower_bound) & (loantap[col]

    return loantap_out

loantap = drop_outliers(loantap).reset_index()
```

In [45]:
```python
loantap.shape
```

Out[45]:
```
(350915, 32)
```

In [46]:
```python
loantap.isnull().sum()
```

Out[46]:
```
index                   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
state                   0
pincode                 0
city                    0
annual_inc_bins         1
dtype: int64
```

Converting term values to numerical val

One hot encoding

In [47]:
```
term_values={' 36 months': 36, ' 60 months':60}
loantap['term'] = loantap.term.map(term_values)

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

# Initial List Status
loantap['initial_list_status'].unique()
np.array(['w', 'f'], dtype=object)
list_status = {'w': 0, 'f': 1}
loantap['initial_list_status'] = loantap.initial_list_status.map(list_status)

# Let's fetch ZIP from address and then drop the remaining details -
loantap['zip_code'] = loantap.address.apply(lambda x: x[-5:])
loantap['zip_code'].value_counts(normalize=True)*100
```

Out[47]:
```
70466    14.397219
22690    14.266703
30723    14.255874
48052    14.145591
00813    11.584287
29597    11.506205
05113    11.467734
11650     2.816921
93700     2.788425
86630     2.771041
Name: zip_code, dtype: float64
```

In [48]:
```
loantap.isnull().sum()
```

Out[48]:
```
index                   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
state                   0
pincode                 0
city                    0
annual_inc_bins         1
zip_code                0
dtype: int64
```

In [49]:
```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.preprocessing import OneHotEncoder
```

In [50]:
```
"""mapping = {
    'A1': 35, 'A2': 34, 'A3': 33, 'A4': 32, 'A5': 31,
    'B1': 30, 'B2': 29, 'B3': 28, 'B4': 27, 'B5': 26,
    'C1': 25, 'C2': 24, 'C3': 23, 'C4': 22, 'C5': 21,
    'D1': 20, 'D2': 19, 'D3': 18, 'D4': 17, 'D5': 16,
    'E1': 15, 'E2': 14, 'E3': 13, 'E4': 12, 'E5': 11,
    'F1': 10, 'F2': 9,  'F3': 8,  'F4': 7,  'F5': 6,
    'G1': 5,  'G2': 4,  'G3': 3,  'G4': 2,  'G5': 1
}
loantap['sub_grade'] = loantap['sub_grade'].map(mapping)
"""
```

Out[50]:
```
"mapping = {\n    'A1': 35, 'A2': 34, 'A3': 33, 'A4': 32, 'A5': 31,\n    'B1':
30, 'B2': 29, 'B3': 28, 'B4': 27, 'B5': 26,\n    'C1': 25, 'C2': 24, 'C3': 23,
'C4': 22, 'C5': 21,\n    'D1': 20, 'D2': 19, 'D3': 18, 'D4': 17, 'D5': 16,\n
'E1': 15, 'E2': 14, 'E3': 13, 'E4': 12, 'E5': 11,\n    'F1': 10, 'F2': 9,  'F
3': 8,  'F4': 7,  'F5': 6,\n    'G1': 5,  'G2': 4,  'G3': 3,  'G4': 2,  'G5': 1
\n}\nloantap['sub_grade'] = loantap['sub_grade'].map(mapping)\n"
```

In [51]:
```
## employee_length Encoding
ordered_categories = ['Unknown', '< 1 year', '1 year', '2 years', '3 years', '4
```

```
encoder = OrdinalEncoder(categories=[ordered_categories])
loantap['emp_length'] = encoder.fit_transform(loantap[['emp_length']])
```

In [52]:
```
## emp_title Encoding
mean_encode = loantap.groupby('emp_title')['loan_status'].mean()
loantap['emp_title'] = loantap['emp_title'].map(mean_encode)
```

In [53]:
```
## Dropping redundant Features
drop_vars = ['annual_inc_bins','title','address','city','state','pincode']
loantap.drop(columns = drop_vars,inplace = True)
```

issue_d Encoding:-

In [54]:
```
loantap['issue_d'] = pd.to_datetime(loantap['issue_d'], format='%b-%Y')
min_date = loantap['issue_d'].min()

loantap['issue_d'] = ((loantap['issue_d'] - min_date) / np.timedelta64(1, 'M')).
```

In [55]:
```
#loantap.isnull().sum()
```

loan_status Encoding:-

In [56]:
```
#loantap['loan_status'] = loantap['loan_status'].map({'Fully Paid': 0, 'Charged
```

In [57]:
```
#loantap.isnull().sum()
```

earliest_cr_line Encding:-

In [58]:
```
loantap['earliest_cr_line'] = pd.to_datetime(loantap['earliest_cr_line'], format
min_date = loantap['earliest_cr_line'].min()

loantap['earliest_cr_line'] = ((loantap['earliest_cr_line'] - min_date) / np.tim
```

initial_list_status Encoding:-

In [59]:
```
#loantap['initial_list_status'] = loantap['initial_list_status'].map({'w': 0, 'f
```

In [ ]:
```

```

In [61]:
```
#drop_vars = ['city','state']
#loantap.drop(columns = drop_vars,inplace = True)
```

In [62]:
```
dummies=['purpose','zip_code','term', 'grade','sub_grade', 'verification_status'
data=pd.get_dummies(loantap,columns=dummies,drop_first=True)
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
```

In [63]:
```
data.isnull().sum()
```

```
Out[63]: index                          0
         loan_amnt                      0
         int_rate                       0
         installment                    0
         emp_title                      0
         emp_length                     0
         annual_inc                     0
         issue_d                        0
         loan_status                    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
         mort_acc                       0
         pub_rec_bankruptcies           0
         purpose_credit_card            0
         purpose_debt_consolidation     0
         purpose_educational            0
         purpose_home_improvement       0
         purpose_house                  0
         purpose_major_purchase         0
         purpose_medical                0
         purpose_moving                 0
         purpose_other                  0
         purpose_renewable_energy       0
         purpose_small_business         0
         purpose_vacation               0
         purpose_wedding                0
         zip_code_05113                 0
         zip_code_11650                 0
         zip_code_22690                 0
         zip_code_29597                 0
         zip_code_30723                 0
         zip_code_48052                 0
         zip_code_70466                 0
         zip_code_86630                 0
         zip_code_93700                 0
         term_60                        0
         grade_B                        0
         grade_C                        0
         grade_D                        0
         grade_E                        0
         grade_F                        0
         grade_G                        0
         sub_grade_A2                   0
         sub_grade_A3                   0
         sub_grade_A4                   0
         sub_grade_A5                   0
         sub_grade_B1                   0
         sub_grade_B2                   0
         sub_grade_B3                   0
         sub_grade_B4                   0
         sub_grade_B5                   0
         sub_grade_C1                   0
         sub_grade_C2                   0
         sub_grade_C3                   0
```

```
sub_grade_C4                           0
sub_grade_C5                           0
sub_grade_D1                           0
sub_grade_D2                           0
sub_grade_D3                           0
sub_grade_D4                           0
sub_grade_D5                           0
sub_grade_E1                           0
sub_grade_E2                           0
sub_grade_E3                           0
sub_grade_E4                           0
sub_grade_E5                           0
sub_grade_F1                           0
sub_grade_F2                           0
sub_grade_F3                           0
sub_grade_F4                           0
sub_grade_F5                           0
sub_grade_G1                           0
sub_grade_G2                           0
sub_grade_G3                           0
sub_grade_G4                           0
sub_grade_G5                           0
verification_status_Source Verified    0
verification_status_Verified           0
application_type_INDIVIDUAL            0
application_type_JOINT                 0
home_ownership_MORTGAGE                0
home_ownership_NONE                    0
home_ownership_OTHER                   0
home_ownership_OWN                     0
home_ownership_RENT                    0
dtype: int64
```

In [64]:

```
loantap.dtypes
```

Out[64]:
```
index                       int64
loan_amnt                 float64
term                        int64
int_rate                  float64
installment               float64
grade                      object
sub_grade                  object
emp_title                 float64
emp_length                float64
home_ownership             object
annual_inc                float64
verification_status        object
issue_d                     int32
loan_status                 int64
purpose                    object
dti                       float64
earliest_cr_line            int32
open_acc                  float64
pub_rec                   float64
revol_bal                 float64
revol_util                float64
total_acc                 float64
initial_list_status         int64
application_type           object
mort_acc                  float64
pub_rec_bankruptcies      float64
zip_code                   object
dtype: object
```

In [65]:
```
loantap.head()
```

Out[65]:

	index	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_le
0	0	4.000000	36	1.058426	2.517829	B	B4	0.262500	
1	1	3.903090	36	1.078819	2.424359	B	B5	0.333333	
2	2	4.193125	36	1.020775	2.704982	B	B3	0.111111	
3	3	3.857332	36	0.812245	2.343704	A	A2	0.000000	
4	4	4.386945	60	1.237292	2.784853	C	C5	1.000000	

Data processing for modelling

In [66]:
```
from sklearn.model_selection import train_test_split

X=data.drop('loan_status',axis=1)
y=data['loan_status']
#X_train, X_test, y_train, y_test =train_test_split(X,y,test_size=0.30,stratify=
X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random_
X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0.

print(X_train.shape)
print(X_test.shape)
```

```
(210549, 90)
(70183, 90)
```

In [67]:
```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.fit_transform(X_val)
X_test = scaler.transform(X_test)
```

Model Building

In [68]:
```
logreg=LogisticRegression(max_iter=1000)
logreg.fit(X_train,y_train)
```

Out[68]:
```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

In [69]:
```
# X.columns.shape
# # logreg.coef_[0]
pd.Series((zip(X.columns, logreg.coef_[0])))
```

```
Out[69]: 0                                (index, -0.06317249283981667)
         1                             (loan_amnt, 0.4330611233279029)
         2                              (int_rate, -1.396861909971435)
         3                           (installment, 0.2675330199824831)
         4                              (emp_title, 8.718454250252659)
         5                          (emp_length, -0.04227140548187026)
         6                            (annual_inc, -5.373710851840544)
         7                               (issue_d, 0.5901415098517242)
         8                                    (dti, 1.214729103974016)
         9                     (earliest_cr_line, -0.5863150421810531)
         10                             (open_acc, 5.4037876052335285)
         11                              (pub_rec, 1.5106656714932152)
         12                           (revol_bal, -0.6229032043336746)
         13                           (revol_util, 0.7533092057156349)
         14                            (total_acc, 0.2455019681375547)
         15                (initial_list_status, 0.06879559201401896)
         16                           (mort_acc, -1.2046569771380182)
         17                                (pub_rec_bankruptcies, 0.0)
         18                 (purpose_credit_card, 0.08634889563612501)
         19          (purpose_debt_consolidation, 0.12855700899819558)
         20                   (purpose_educational, 0.5006275216797242)
         21             (purpose_home_improvement, 0.1432855031537233)
         22                      (purpose_house, -0.17778605742131715)
         23               (purpose_major_purchase, 0.1429599505156378)
         24                      (purpose_medical, 0.15808316542971979)
         25                        (purpose_moving, 0.2670226852238125)
         26                        (purpose_other, 0.04967129110948002)
         27             (purpose_renewable_energy, 0.2932261837033369)
         28               (purpose_small_business, 0.32888480165388345)
         29                      (purpose_vacation, 0.1914973602798932)
         30                    (purpose_wedding, -0.17298180398941962)
         31                      (zip_code_05113, -2.8546572677553237)
         32                       (zip_code_11650, 11.734371396536341)
         33                         (zip_code_22690, 4.523265932450068)
         34                      (zip_code_29597, -2.8499741734720314)
         35                         (zip_code_30723, 4.509013377714138)
         36                         (zip_code_48052, 4.513104885082565)
         37                         (zip_code_70466, 4.535967194266227)
         38                       (zip_code_86630, 11.682719902360052)
         39                       (zip_code_93700, 11.734016861071009)
         40                             (term_60, 0.35956262279542184)
         41                               (grade_B, 1.2773863064386064)
         42                               (grade_C, 1.8953693107921514)
         43                                 (grade_D, 2.282869865279921)
         44                               (grade_E, 2.5966978166243226)
         45                                 (grade_F, 2.819105132285982)
         46                               (grade_G, 2.9463244607094925)
         47                          (sub_grade_A2, 0.2786319502695874)
         48                          (sub_grade_A3, 0.5179324776622494)
         49                          (sub_grade_A4, 0.6505298953791313)
         50                          (sub_grade_A5, 0.9780262854728808)
         51                        (sub_grade_B1, -0.0486387422235155)
         52                        (sub_grade_B2, 0.10553575560822792)
         53                          (sub_grade_B3, 0.2625608231631062)
         54                          (sub_grade_B4, 0.3707666637007489)
         55                          (sub_grade_B5, 0.5871618061900938)
         56                        (sub_grade_C1, 0.17094299162283827)
         57                        (sub_grade_C2, 0.20583434001086984)
         58                        (sub_grade_C3, 0.43672586727829793)
         59                        (sub_grade_C4, 0.48378345346284346)
```

```
60                      (sub_grade_C5, 0.5980826584173361)
61                     (sub_grade_D1, 0.31681166438134856)
62                     (sub_grade_D2, 0.35638662546966915)
63                      (sub_grade_D3, 0.544133310416285)
64                      (sub_grade_D4, 0.502870116711796 3)
65                      (sub_grade_D5, 0.562668127675565 3)
66                      (sub_grade_E1, 0.4855679369442056)
67                      (sub_grade_E2, 0.5056890233880389)
68                      (sub_grade_E3, 0.5756750484593444)
69                      (sub_grade_E4, 0.4469015005762894)
70                      (sub_grade_E5, 0.5828643072564298)
71                       (sub_grade_F1, 0.397290907005897)
72                     (sub_grade_F2, 0.32411452742650776)
73                      (sub_grade_F3, 0.6591442372380116)
74                      (sub_grade_F4, 0.6298869063376559)
75                      (sub_grade_F5, 0.8086685542779255)
76                     (sub_grade_G1, 0.23549867840201003)
77                      (sub_grade_G2, 0.6935957297025002)
78                      (sub_grade_G3, 0.6070991289365149)
79                      (sub_grade_G4, 0.8615136943451633)
80                      (sub_grade_G5, 0.5486172293231454)
81    (verification_status_Source Verified, 0.205937...
82    (verification_status_Verified, 0.2425169587491...
83    (application_type_INDIVIDUAL, 0.03552195794088...
84        (application_type_JOINT, -1.1213224376372675)
85       (home_ownership_MORTGAGE, -0.2326276189191058)
86          (home_ownership_NONE, -0.29535445168784086)
87           (home_ownership_OTHER, 0.6721754213878898)
88           (home_ownership_OWN, -0.11985480241915604)
89           (home_ownership_RENT, 0.04570728068510016)
dtype: object
```

In [70]:
```python
y_pred = logreg.predict(X_test)
print('Accuracy of Logistic Regression Classifier on test set: {:.3f}'.format(lo
```

```
Accuracy of Logistic Regression Classifier on test set: 0.799
```

In [71]:
```python
def accuracy(y_true, y_pred):
    return np.sum(y_true==y_pred)/y_true.shape[0]
```

In [72]:
```python
accuracy(y_test,y_pred)
```

Out[72]:
```
0.7990254050126099
```

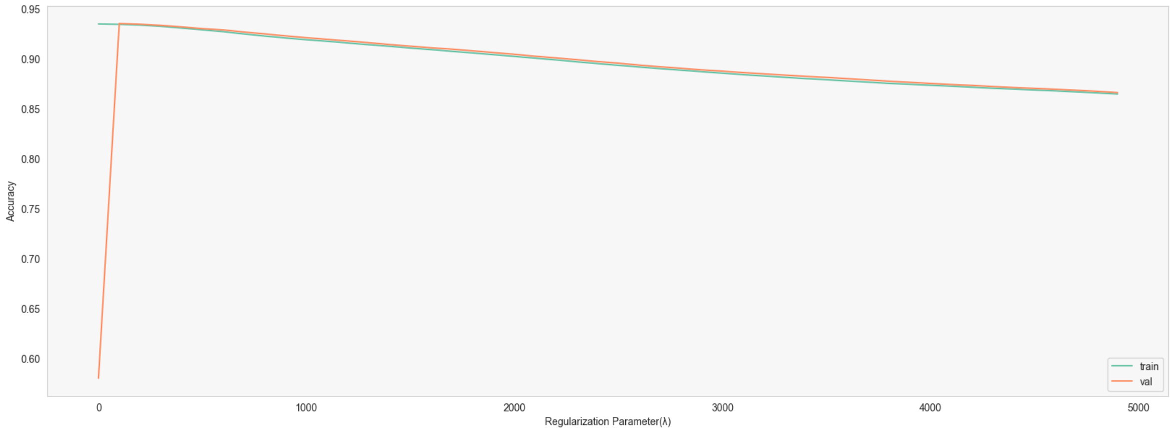
Cross Validation

In [73]:
```python
#X_tr_cv, X_test, y_tr_cv, y_test = train_test_split(X, y, test_size=0.2, random
#X_train, X_val, y_train, y_val = train_test_split(X_tr_cv, y_tr_cv, test_size=0
```

In [ ]:
```python
train_scores = []
val_scores = []

for la in np.arange(0.01, 5000.0, 100): # range of values of Lambda
    model =  LogisticRegression(C=1/la).fit(X_train, y_train)
    train_score = accuracy(y_train, model.predict(X_train))
    val_score = accuracy(y_val, model.predict(X_val))
```

```
    train_scores.append(train_score)
    val_scores.append(val_score)
```

In [75]:
```
plt.figure(figsize=(20,7))
plt.plot(list(np.arange(0.01, 5000.0, 100)), train_scores, label="train")
plt.plot(list(np.arange(0.01, 5000.0, 100)), val_scores, label="val")
plt.legend(loc='lower right')

plt.xlabel("Regularization Parameter(λ)")
plt.ylabel("Accuracy")
plt.grid()
plt.show()
```



In [76]:
```
idx = np.argmax(train_scores)
la = np.arange(0.01, 5000.0, 100)[idx]

print(f'Best Regularization Parameter {la}')
```

```
Best Regularization Parameter 0.01
```

In [ ]:
```
model = LogisticRegression(C= 1/la)
model.fit(X_train,y_train)

y_pred = model.predict(X_test)
```

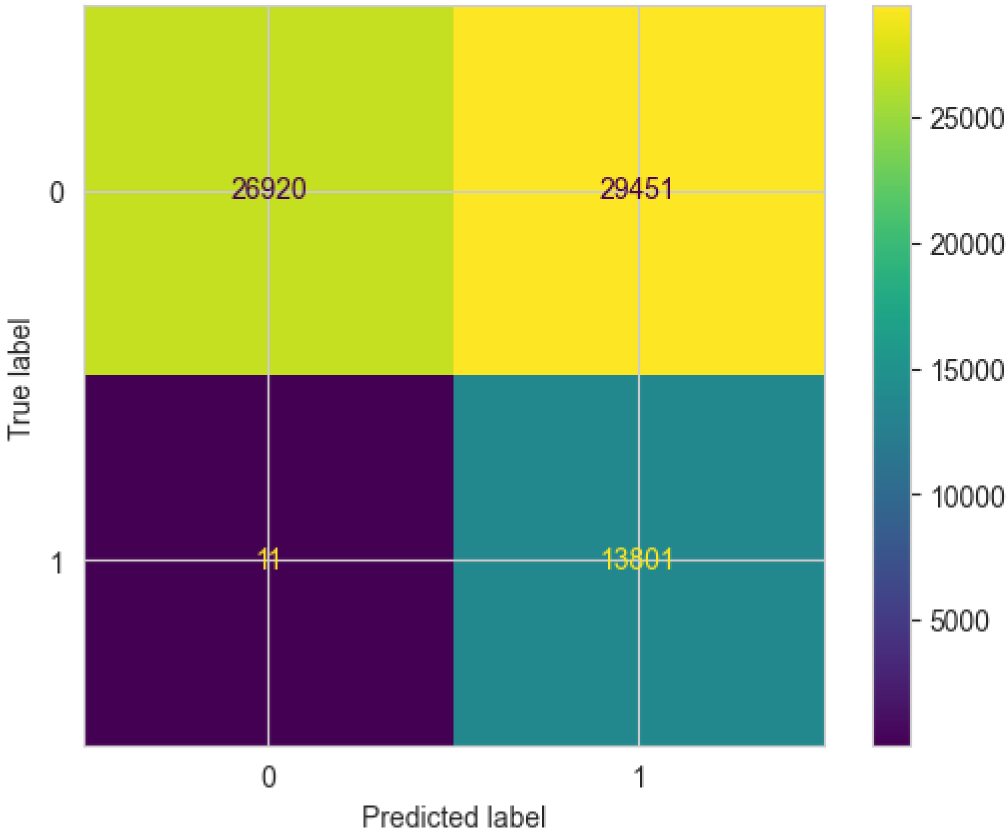
In [ ]:

Confusion Matrix

In [78]:
```
confusion_matrix=confusion_matrix(y_test,y_pred)
print(confusion_matrix)
ConfusionMatrixDisplay(confusion_matrix=confusion_matrix, display_labels=logreg.
```

```
[[26920 29451]
 [   11 13801]]
```

Out[78]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x159dabfe4d0
>
```



Insights

- There is insignificant value for false negative but significantly large false positive. Which will hamper our prediction due to type-1 errors.

In [79]:
```
def metrics(y_test, y_pred):
    conf_matrix = confusion_matrix(y_test, y_pred)
    fig, ax = plt.subplots(figsize=(10,5))
    ConfusionMatrixDisplay(conf_matrix).plot(ax = ax)
    print(classification_report(y_test,y_pred))
```

Assumption of Log. Reg. (Multicollinearity Check)

In [81]:
```
#X.head()
```

In [82]:
```
def calc_vif(X):
    # Calculating the VIF
    vif=pd.DataFrame()
    vif['Feature']=X.columns
    vif['VIF']=[variance_inflation_factor(X.values,i) for i in range(X.shape[1])]
    vif['VIF']=round(vif['VIF'],2)
    vif=vif.sort_values(by='VIF',ascending=False)
    return vif

calc_vif(X)[:5]
```

```
C:\Users\deepa\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmod
els\regression\linear_model.py:1783: RuntimeWarning: invalid value encountered in
scalar divide
  return 1 - self.ssr/self.uncentered_tss
C:\Users\deepa\AppData\Local\Programs\Python\Python310\lib\site-packages\statsmod
els\stats\outliers_influence.py:198: RuntimeWarning: divide by zero encountered i
n scalar divide
  vif = 1. / (1. - r_squared_i)
```

Out[82]:

	Feature	VIF
45	grade_F	inf
56	sub_grade_C1	inf
63	sub_grade_D3	inf
62	sub_grade_D2	inf
61	sub_grade_D1	inf

As the VIF score is infinity use Statistical tests to determine the inrelevant/insignificant features to be dropped

In [85]:

```
target_var = 'loan_status'
continuous_vars = loantap.columns[loantap.dtypes != 'object'].to_list()
categorical_vars = loantap.columns[loantap.dtypes == 'object'].to_list()
#categorical_vars.remove(target_var)
temp_vars = loantap.columns[(loantap.dtypes == 'object') & (loantap.nunique() <
```

In [92]:

```
loantap.head()
```

Out[92]:

	index	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_le
0	0	4.000000	36	1.058426	2.517829	B	B4	0.262500	
1	1	3.903090	36	1.078819	2.424359	B	B5	0.333333	
2	2	4.193125	36	1.020775	2.704982	B	B3	0.111111	
3	3	3.857332	36	0.812245	2.343704	A	A2	0.000000	
4	4	4.386945	60	1.237292	2.784853	C	C5	1.000000	

In [93]:

```
from scipy.stats import levene
def levene_test(against):
    print(f'\nLevene Test {against} and loan_status:-')

    df2 = loantap.copy()
    df2.dropna(inplace=True)

    test_stat, p_value = levene(
        df2.loc[loantap['loan_status'] == 0,against].sample(34000,random_state=4
        df2.loc[loantap['loan_status'] == 1,against].sample(8400,random_state=42
    )
    print(f'\tp_value: {p_value:.4f}')
    if p_value < 0.05:
        print('\tThe samples do not have Homogenous Variance')
```

```
    else:
        print('\tThe samples have Homogenous Variance ')

    return p_value > 0.05
```

In [94]:
```
from scipy.stats import ttest_ind, mannwhitneyu
def ttest(against):
    # Verify Assumptions
    homogenity_of_var = levene_test(against)

    df2 = loantap.copy()
    df2.dropna(inplace=True)

    group1 = df2.loc[loantap[target_var] == 0, against].sample(34000,random_stat
    group2 = df2.loc[loantap[target_var] == 1, against].sample(8400,random_state

    if homogenity_of_var:

        print(f'\nT Test Two Independent Samples {against} and loan_status:- ')

        t_stat,p_value = ttest_ind(group1,group2)
        print(f'\tp_value: {p_value:.4f}')

        if p_value < 0.05:
            print(f'\tReject Null:- \n\tThe means of {against} is not equal for
        else:
            print(f'\tFailed to Reject Null:- \n\tThe means of {against} is  equ

        if p_value > 0.05: return True

    else:

        print('\nMann Whitney - U Test {against} and loan_status:-')

        t_stat,p_value = mannwhitneyu(group1,group2)
        print(f'\tp_value: {p_value:.4f}')

        if p_value < 0.05:
            print(f'\tReject Null:- \n\tThe means of {against} is not equal for
        else:
            print(f'\tFailed to Reject Null:- \n\tThe means of {against} is  equ

        if p_value > 0.05: return True
```

In [95]:
```
stat_insig_var = []

for var in continuous_vars:
    if ttest(var):
        stat_insig_var.append(var)
    print('-'*90)

print(f'\nStatistically Insignificant Features: {stat_insig_var}')
```

```
Levene Test index and loan_status:-
        p_value: 0.4491
        The samples have Homogenous Variance

T Test Two Independent Samples index and loan_status:-
        p_value: 0.9644
        Failed to Reject Null:-
        The means of index is  equal for Fully Paid and Charged Off customers
-----------------------------------------------------------------------------------------

Levene Test loan_amnt and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of loan_amnt is not equal for Fully Paid and Charged Off custom
ers
-----------------------------------------------------------------------------------------

Levene Test term and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of term is not equal for Fully Paid and Charged Off customers
-----------------------------------------------------------------------------------------

Levene Test int_rate and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of int_rate is not equal for Fully Paid and Charged Off custome
rs
-----------------------------------------------------------------------------------------

Levene Test installment and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of installment is not equal for Fully Paid and Charged Off cust
omers
-----------------------------------------------------------------------------------------

Levene Test emp_title and loan_status:-
        p_value: 0.0000
```

```
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of emp_title is not equal for Fully Paid and Charged Off custom
ers
--------------------------------------------------------------------------------
---------

Levene Test emp_length and loan_status:-
        p_value: 0.3713
        The samples have Homogenous Variance

T Test Two Independent Samples emp_length and loan_status:-
        p_value: 0.5606
        Failed to Reject Null:-
        The means of emp_length is  equal for Fully Paid and Charged Off customer
s
--------------------------------------------------------------------------------
---------

Levene Test annual_inc and loan_status:-
        p_value: 0.0113
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of annual_inc is not equal for Fully Paid and Charged Off custo
mers
--------------------------------------------------------------------------------
---------

Levene Test issue_d and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of issue_d is not equal for Fully Paid and Charged Off customer
s
--------------------------------------------------------------------------------
---------

Levene Test loan_status and loan_status:-
```

```
C:\Users\deepa\AppData\Local\Programs\Python\Python310\lib\site-packages\scipy\st
ats\_morestats.py:3189: RuntimeWarning: invalid value encountered in scalar divid
e
  W = numer / denom
```

```
        p_value: nan
        The samples have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of loan_status is not equal for Fully Paid and Charged Off cust
omers
---------------------------------------------------------------------------------
---------

Levene Test dti and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of dti is not equal for Fully Paid and Charged Off customers
---------------------------------------------------------------------------------
---------

Levene Test earliest_cr_line and loan_status:-
        p_value: 0.5509
        The samples have Homogenous Variance

T Test Two Independent Samples earliest_cr_line and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of earliest_cr_line is not equal for Fully Paid and Charged Off
customers
---------------------------------------------------------------------------------
---------

Levene Test open_acc and loan_status:-
        p_value: 0.1058
        The samples have Homogenous Variance

T Test Two Independent Samples open_acc and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of open_acc is not equal for Fully Paid and Charged Off custome
rs
---------------------------------------------------------------------------------
---------

Levene Test pub_rec and loan_status:-
        p_value: 0.0056
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0002
        Reject Null:-
        The means of pub_rec is not equal for Fully Paid and Charged Off customer
s
---------------------------------------------------------------------------------
---------

Levene Test revol_bal and loan_status:-
        p_value: 0.0935
```

```
        The samples have Homogenous Variance

T Test Two Independent Samples revol_bal and loan_status:-
        p_value: 0.2248
        Failed to Reject Null:-
        The means of revol_bal is  equal for Fully Paid and Charged Off customers
---------------------------------------------------------------------------------
---------

Levene Test revol_util and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of revol_util is not equal for Fully Paid and Charged Off custo
mers
---------------------------------------------------------------------------------
---------

Levene Test total_acc and loan_status:-
        p_value: 0.5780
        The samples have Homogenous Variance

T Test Two Independent Samples total_acc and loan_status:-
        p_value: 0.0001
        Reject Null:-
        The means of total_acc is not equal for Fully Paid and Charged Off custom
ers
---------------------------------------------------------------------------------
---------

Levene Test initial_list_status and loan_status:-
        p_value: 0.1783
        The samples have Homogenous Variance

T Test Two Independent Samples initial_list_status and loan_status:-
        p_value: 0.1783
        Failed to Reject Null:-
        The means of initial_list_status is  equal for Fully Paid and Charged Off
customers
---------------------------------------------------------------------------------
---------

Levene Test mort_acc and loan_status:-
        p_value: 0.0000
        The samples do not have Homogenous Variance

Mann Whitney - U Test {against} and loan_status:-
        p_value: 0.0000
        Reject Null:-
        The means of mort_acc is not equal for Fully Paid and Charged Off custome
rs
---------------------------------------------------------------------------------
---------

Levene Test pub_rec_bankruptcies and loan_status:-
        p_value: nan
        The samples have Homogenous Variance
```

```
Mann Whitney - U Test {against} and loan_status:-
        p_value: 1.0000
        Failed to Reject Null:-
        The means of pub_rec_bankruptcies is  equal for Fully Paid and Charged Of
f customers
--------------------------------------------------------------------------------
---------

Statistically Insignificant Features: ['index', 'emp_length', 'revol_bal', 'initi
al_list_status', 'pub_rec_bankruptcies']
```

In [96]:

```
from scipy.stats import chi2_contingency
def chi_squared_test(against):

    df2 = loantap.copy()
    df2.dropna(inplace=True)

    contingency_table = pd.crosstab(df2[target_var], df2[against])

    # Cell Expectancy: 80% of the cells should have an expected frequency of 5 o
    chi2, p, _, expected = chi2_contingency(contingency_table)

    n_cells = expected.size
    n_large_expect = (expected >= 5).sum()

    if n_large_expect / n_cells < 0.8:
        print(f"Assumption check for {against}: Cell expectancy failed!")
        return

    print(f'\nChi-squared Test between {against} and {target_var}:-')
    print(f'\tp_value: {p:.4f}')

    if p < 0.05:
        print(f'\tReject Null:- \n\t{against} and {target_var} are dependent')
    else:
        print(f'\tFailed to Reject Null:- \n\t{against} and {target_var} are ind

    return p > 0.05
```

In [97]:

```
for var in categorical_vars:
    if chi_squared_test(var):
        stat_insig_var.append(var)
    print('-'*90)

print(f'\nStatistically Insignificant Features: {stat_insig_var}')
```

```
Chi-squared Test between grade and loan_status:-
        p_value: 0.0000
        Reject Null:-
        grade and loan_status are dependent
---------------------------------------------------------------------------------
---------

Chi-squared Test between sub_grade and loan_status:-
        p_value: 0.0000
        Reject Null:-
        sub_grade and loan_status are dependent
---------------------------------------------------------------------------------
---------

Chi-squared Test between home_ownership and loan_status:-
        p_value: 0.0000
        Reject Null:-
        home_ownership and loan_status are dependent
---------------------------------------------------------------------------------
---------

Chi-squared Test between verification_status and loan_status:-
        p_value: 0.0000
        Reject Null:-
        verification_status and loan_status are dependent
---------------------------------------------------------------------------------
---------

Chi-squared Test between purpose and loan_status:-
        p_value: 0.0000
        Reject Null:-
        purpose and loan_status are dependent
---------------------------------------------------------------------------------
---------

Chi-squared Test between application_type and loan_status:-
        p_value: 0.0000
        Reject Null:-
        application_type and loan_status are dependent
---------------------------------------------------------------------------------
---------

Chi-squared Test between zip_code and loan_status:-
        p_value: 0.0000
        Reject Null:-
        zip_code and loan_status are dependent
---------------------------------------------------------------------------------
---------

Statistically Insignificant Features: ['index', 'emp_length', 'revol_bal', 'initi
al_list_status', 'pub_rec_bankruptcies']
```

Perfomance after dropping Insignificant Features:-

In [ ]:

```
X.drop(columns=stat_insig_var,inplace=True)
#X_train.drop(columns=stat_insig_var,inplace=True)
#X_val.drop(columns=stat_insig_var,inplace=True)
#X_test.drop(columns=stat_insig_var,inplace=True)
```

In [ ]:

```

```

Validation using KFold

In [100…

```
from sklearn.model_selection import train_test_split, KFold, cross_val_score
X=scaler.fit_transform(X)

kfold=KFold(n_splits=5)
accuracy=np.mean(cross_val_score(logreg,X,y,cv=kfold,scoring='accuracy',n_jobs=-
print("Cross Validation accuracy : {:.3f}".format(accuracy))
```

```
Cross Validation accuracy : 0.935
```

Insights

- Cross Validation accuracy and testing accuracy is almost same which infers model is performing the decent job.

SMOTE - Oversampling

In [106…

```
from imblearn.over_sampling import SMOTE
```

In [107…

```
# Initialize SMOTE and fit on data
smote = SMOTE(random_state=42,sampling_strategy=0.60)
X_resampled, y_resampled = smote.fit_resample(X_train, y_train)

print("Before SMOTE, counts of label '1':", sum(y_train==1))
print("Before SMOTE, counts of label '0':", sum(y_train==0))
print("After SMOTE, counts of label '1':", sum(y_resampled==1))
print("After SMOTE, counts of label '0':", sum(y_resampled==0))
```

```
Before SMOTE, counts of label '1': 40955
Before SMOTE, counts of label '0': 169594
After SMOTE, counts of label '1': 101756
After SMOTE, counts of label '0': 169594
```

Training Data:-

In [ ]:

```
model = LogisticRegression(C = 1/la)
model.fit(X_resampled,y_resampled)

y_pred = model.predict(X_train)
#metrics(y_train,y_pred)
```

In [111…

```
#lr1 = LogisticRegression(max_iter=1000)
#lr1.fit(X_train_res, y_train_res)
#predictions = lr1.predict(X_test)

# Classification Report
print(classification_report(y_train, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.95      0.95    169594
           1       0.79      0.83      0.81     40955

    accuracy                           0.92    210549
   macro avg       0.87      0.89      0.88    210549
weighted avg       0.92      0.92      0.92    210549
```

Unseen Data:-

In [ ]:

In [113…
```
model = LogisticRegression(C = 1/la)
model.fit(X_resampled,y_resampled)

y_pred = model.predict(X_test)
# Classification Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     56371
           1       0.78      0.83      0.81     13812

    accuracy                           0.92     70183
   macro avg       0.87      0.89      0.88     70183
weighted avg       0.92      0.92      0.92     70183
```

```
C:\Users\deepa\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Classification Report

In [114…
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95     56371
           1       0.78      0.83      0.81     13812

    accuracy                           0.92     70183
   macro avg       0.87      0.89      0.88     70183
weighted avg       0.92      0.92      0.92     70183
```

In [120…
```
# X.columns.shape
# # logreg.coef_[0]
pd.Series((zip(loantap.columns, logreg.coef_[0])))
```

Out[120]:
```
0                       (index, -0.06317249283981667)
1                     (loan_amnt, 0.4330611233279029)
2                          (term, -1.396861909971435)
3                      (int_rate, 0.2675330199824831)
4                   (installment, 8.718454250252659)
5                       (grade, -0.04227140548187026)
6                     (sub_grade, -5.373710851840544)
7                     (emp_title, 0.5901415098517242)
8                    (emp_length, 1.214729103974016)
9              (home_ownership, -0.5863150421810531)
10                   (annual_inc, 5.4037876052335285)
11       (verification_status, 1.5106656714932152)
12                     (issue_d, -0.6229032043336746)
13                 (loan_status, 0.7533092057156349)
14                      (purpose, 0.2455019681375547)
15                         (dti, 0.06879559201401896)
16          (earliest_cr_line, -1.2046569771380182)
17                                   (open_acc, 0.0)
18                     (pub_rec, 0.08634889563612501)
19                   (revol_bal, 0.12855700899819558)
20                   (revol_util, 0.5006275216797242)
21                    (total_acc, 0.1432855031537233)
22     (initial_list_status, -0.17778605742131715)
23            (application_type, 0.1429599505156378)
24                    (mort_acc, 0.15808316542971979)
25      (pub_rec_bankruptcies, 0.2670226852238125)
26                    (zip_code, 0.04967129110948002)
dtype: object
```

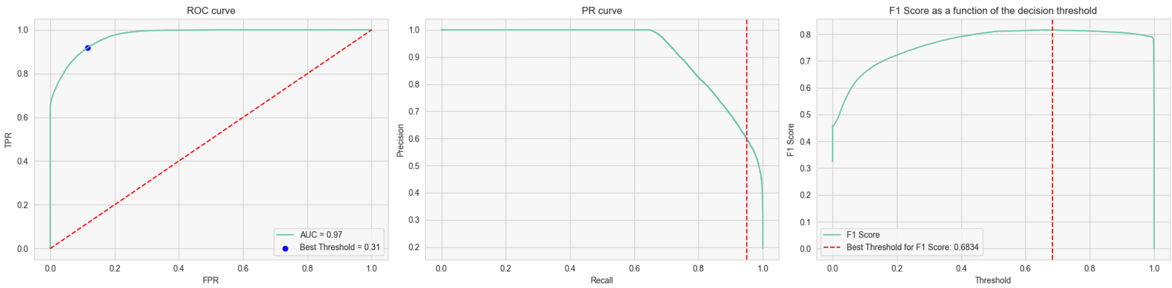
AUC/ROC

In [115…
```
from sklearn.metrics import roc_curve, precision_recall_curve, auc
```

In [116…
```
probability = model.predict_proba(X_val)
probabilities = probability[:, 1]

fpr, tpr, thr_roc = roc_curve(y_val, probabilities)
roc_auc = auc(fpr, tpr)

youdens_index = tpr - fpr
best_idx_roc = np.argmax(youdens_index)
best_threshold_roc = thr_roc[best_idx_roc]
roc_auc = auc(fpr, tpr)
print("Area Under Curve - ROC",roc_auc)

precision, recall, thr_pr = precision_recall_curve(y_val, probabilities)
pr_auc = auc(recall, precision)
print("Area Under Curve - PR",pr_auc)


desired_recall = 0.95
idx_pr = np.where(recall >= desired_recall)[0][-1]
best_threshold_pr = thr_pr[idx_pr]

f1_scores = (2*precision*recall)/(precision + recall + 1e-10)
idx_f1 = np.argmax(f1_scores)
best_threshold_f1 = thr_pr[idx_f1]
```

```
plt.figure(figsize=(20, 5))

plt.subplot(1, 3, 1)
plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
plt.plot(fpr, fpr, '--', color='red')
plt.scatter(fpr[best_idx_roc], tpr[best_idx_roc], color='blue', label=f'Best Thr
plt.title('ROC curve')
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.legend()

# Precision-Recall curve
plt.subplot(1, 3, 2)
plt.plot(recall, precision)
plt.axvline(desired_recall, color='red', linestyle='--')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('PR curve')

# F1 Score curve
plt.subplot(1, 3, 3)
plt.plot(thr_pr, f1_scores[:-1], label='F1 Score')
plt.axvline(x=best_threshold_f1, color='red', linestyle='--', label=f'Best Thres
plt.xlabel('Threshold')
plt.ylabel('F1 Score')
plt.title('F1 Score as a function of the decision threshold')
plt.legend()

plt.tight_layout()
plt.show()

print(f"Best threshold from ROC curve (Youden's index): {best_threshold_roc:.4f}
print(f"Best threshold for a recall of at least {desired_recall}: {best_threshol
print(f"Best threshold for maximum F1 Score: {best_threshold_f1:.4f}")

y_pred = np.where(probabilities >= best_threshold_roc, 1, 0)
```

```
Area Under Curve - ROC 0.9740087411700558
Area Under Curve - PR 0.9229203550105318
```




```
Best threshold from ROC curve (Youden's index): 0.3098
Best threshold for a recall of at least 0.95: 0.2284
Best threshold for maximum F1 Score: 0.6834
```

1. **ROC Model Performance**:
   - **Overall Accuracy**: High (92%), indicating effective predictions across both loan statuses.
   - **Precision for Non-Defaulters (Class 0)**: Very high (96%), showing the model's strength in correctly identifying non-defaulters.

- **Recall for Defaulters (Class 1)**: High (83%), meaning the model effectively identifies most actual defaulters.
- **F1-Score**: Balanced for both classes (0.95 for non-defaulters and 0.81 for defaulters), suggesting a good trade-off between precision and recall.

Tradeoff Questions

Trade-off Between Models: The ROC model offers a more balanced approach between precision and recall, suitable for scenarios where both loan approval opportunities and risk mitigation are equally prioritized. The Precision-Recall model, favoring recall, is better for situations where identifying defaulters is more critical, even at the expense of higher false positives.

1. 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.

   - `Answer` - Since data is imbalances by making the data balance we can try to avoid false positives. For evaluation metrics, we should be focusing on the macro average f1-score because we don't want to make false positive prediction and at the same we want to detect the defualers.
2. 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

   - `Answer` - Below are the most important features and their importance while making the prediction. So these variables can help the managers to identify which are customers who are more likely to pay the loan amount fully,

   Important Fetaures and their weightage coefficients are as below:

- `(term, -1.396861909971435)`
- `(installment, 8.718454250252659)`
- `(sub_grade, -5.373710851840544)`
- `(emp_title, 0.5901415098517242)`
- `(emp_length, 1.214729103974016)`
- `(home_ownership, -0.5863150421810531)`
- `(annual_inc, 5.4037876052335285)`
- `(verification_status, 1.5106656714932152)`
- `(issue_d, -0.6229032043336746)`
- `(loan_status, 0.7533092057156349)`
- `(earliest_cr_line, -1.2046569771380182)`
- `(revol_util, 0.5006275216797242)`

In [ ]:

```
```

Insights

1. **Loan Term Preference and Risk**: A majority of borrowers prefer a shorter loan term of 36 months. Shorter terms are generally associated with lower risk and might indicate a borrower's confidence in repayment ability.
2. **Loan Grade Concentration**: The concentration of loans in the B, C, and A categories suggests these grades are most common and potentially the most reliable segments.
3. **Employment Stability and Loan Purpose**: The long employment duration of many borrowers indicates stability, which is a positive sign for lenders. The primary purpose of loans being debt consolidation reflects a trend towards financial management and restructuring by borrowers.
4. **High Outlier Percentages**: The significant outlier percentages in features like Pub_Rec, Revol_Bal, and Annual_Inc highlight the need for careful outlier handling and data validation processes.
5. **Correlation Insights**: The high correlation between loan amount and installment (0.95 Pearson, 0.97 Spearman) suggests that these two variables are strongly linked in their behavior.
6. **Geographical Variations in Default Rates**: The variance in default rates between states like Minnesota (lowest) and Wyoming (highest) can indicate regional economic factors affecting loan repayment.
7. **Impact of Loan Terms and Grades on Risk**: Longer-term loans (60 months) and higher grades (towards G) are riskier, which may influence lending strategies and interest rates.
8. **Verification Status**: The fact that verification doesn't necessarily guarantee a loan being fully paid implies that other factors play a significant role in loan performance.

Recommendations

1. **Risk Management Strategies**: Given the association of higher interest rates, higher loan amounts, and higher DTI ratios with increased risk of default, lenders should consider adjusting their risk assessment models to factor in these elements more significantly.
2. **Tailored Loan Products**: Develop tailored loan products for the most common borrower segments (B, C, and A grade borrowers) to enhance product-market fit.
3. **Enhanced Data Validation**: Implement robust data validation and outlier management strategies to ensure data accuracy, especially for variables with high outlier percentages.

4. **Regional Risk Assessment**: Incorporate geographical data into risk assessments to account for regional variations in default rates.

5. **Loan Term Structuring**: Offer more flexible terms for higher-grade loans and consider stricter terms for longer-term and lower-grade loans to mitigate risk.

6. **Diversification of Loan Purposes**: While debt consolidation is predominant, diversifying the portfolio with other loan purposes could spread risk.

7. **Further Statistical Analysis**: Conduct further statistical tests to validate the significance of observed correlations and insights, ensuring that lending strategies are data-driven.

8. **Monitoring and Adjustment of Models**: Continuously monitor and adjust credit scoring models in response to changes in borrower behavior and economic conditions.

In [ ]: