Business Case Study: LoanTap - Logistic Regression

Problem statement

- Generate insights and recommendations that could help **LoanTap** to understand
  - Highly influencing factors in loan status
  - Extension of credit line based on the significant variables

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats as st
from scipy import interp
from itertools import cycle
import math
import re

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder, label_binar
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import (
    auc,
    average_precision_score,
    confusion_matrix,
    precision_recall_curve,
    roc_auc_score,
    roc_curve,
    classification_report
)

import warnings
warnings.simplefilter('ignore')
```

In [2]:
```
df = pd.read_csv('LoanTapData.csv')
```

Structure and Charactersistics of the dataset

In [3]:
```
df.shape
```

Out[3]:
```
(396030, 27)
```

Checking dtypes and missing values

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object 
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object 
 5   sub_grade             396030 non-null  object 
 6   emp_title             373103 non-null  object 
 7   emp_length            377729 non-null  object 
 8   home_ownership        396030 non-null  object 
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object 
 11  issue_d               396030 non-null  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 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 [5]:
```
{col:(df[col].isnull().sum()/df.shape[0]*100).round(2) for col in df.column
```

Out[5]:
```
{'emp_title': 5.79,
 'emp_length': 4.62,
 'title': 0.44,
 'revol_util': 0.07,
 'mort_acc': 9.54,
 'pub_rec_bankruptcies': 0.14}
```

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

Glimpse of data using head and tail

In [7]:
```
df.head()
```

Out[7]:

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

5 rows × 27 columns

In [8]:
```python
df.tail()
```

Out[8]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	ho
396025	10000.0	60 months	10.99	217.38	B	B4	licensed bankere	2 years	
396026	21000.0	36 months	12.29	700.42	C	C1	Agent	5 years	
396027	5000.0	36 months	9.99	161.32	B	B1	City Carrier	10+ years	
396028	21000.0	60 months	15.31	503.02	C	C2	Gracon Services, Inc	10+ years	
396029	2000.0	36 months	13.61	67.98	C	C2	Internal Revenue Service	10+ years	

5 rows × 27 columns

In [9]:
```python
df.select_dtypes(object).nunique()
```

Out[9]:
```
term                        2
grade                       7
sub_grade                  35
emp_title              173105
emp_length                 11
home_ownership              6
verification_status         3
issue_d                   115
loan_status                 2
purpose                    14
title                   48817
earliest_cr_line          684
initial_list_status         2
application_type            3
address                393700
dtype: int64
```

In [10]:
```python
# As we  have many levels for address and it's doesn'timpact much on loan_s
df.drop('address',axis=1,inplace=True)
```

- **Missing** values (missing percentage) for few columns
  - {'emp_title': 5.79, 'emp_length': 4.62, 'title': 0.44, 'revol_util': 0.07, 'mort_acc': 9.54, 'pub_rec_bankruptcies': 0.14}
- No duplicate records
- Removing 'address' as it is a kind of ID column

In [11]:
```python
df.describe(include='all').T
```

Out[11]:

	count	unique	top	freq	mean	std
loan_amnt	396030.0	NaN	NaN	NaN	14113.888089	8357.441341
term	396030	2	36 months	302005	NaN	NaN
int_rate	396030.0	NaN	NaN	NaN	13.6394	4.472157
installment	396030.0	NaN	NaN	NaN	431.849698	250.72779
grade	396030	7	B	116018	NaN	NaN
sub_grade	396030	35	B3	26655	NaN	NaN
emp_title	373103	173105	Teacher	4389	NaN	NaN
emp_length	377729	11	10+ years	126041	NaN	NaN
home_ownership	396030	6	MORTGAGE	198348	NaN	NaN
annual_inc	396030.0	NaN	NaN	NaN	74203.175798	61637.621158
verification_status	396030	3	Verified	139563	NaN	NaN
issue_d	396030	115	Oct-2014	14846	NaN	NaN
loan_status	396030	2	Fully Paid	318357	NaN	NaN
purpose	396030	14	debt_consolidation	234507	NaN	NaN
title	394275	48817	Debt consolidation	152472	NaN	NaN
dti	396030.0	NaN	NaN	NaN	17.379514	18.019092
earliest_cr_line	396030	684	Oct-2000	3017	NaN	NaN
open_acc	396030.0	NaN	NaN	NaN	11.311153	5.137649
pub_rec	396030.0	NaN	NaN	NaN	0.178191	0.530671
revol_bal	396030.0	NaN	NaN	NaN	15844.539853	20591.836109
revol_util	395754.0	NaN	NaN	NaN	53.791749	24.452193
total_acc	396030.0	NaN	NaN	NaN	25.414744	11.886991
initial_list_status	396030	2	f	238066	NaN	NaN
application_type	396030	3	INDIVIDUAL	395319	NaN	NaN
mort_acc	358235.0	NaN	NaN	NaN	1.813991	2.14793
pub_rec_bankruptcies	395495.0	NaN	NaN	NaN	0.121648	0.356174

- Transforming columns to reduce memory usage

In [12]:
```
df['emp_length'].unique()
```

Out[12]:
```
array(['10+ years', '4 years', '< 1 year', '6 years', '9 years',
       '2 years', '3 years', '8 years', '7 years', '5 years', '1 year',
       nan], dtype=object)
```

In [13]:
```
df['emp_length'] = df['emp_length'].apply(lambda x: int(re.findall(r'\d+',x
```

In [14]:
```
df['term'] = df['term'].apply(lambda x: int(re.findall(r'\d+',x)[0]) if not
```

Feature Engineering

In [15]:
```
# Checking if any column has majority of records for a level
```

```
near_const = [col for col in df.columns
                if any(
                [
                    cnt > 0.7 * df.shape[0]
                    for cnt in df[col].value_counts().tolist()
                ]
                    )
            ]
```

In [16]:
```
for col in near_const+['mort_acc']:
    display(df[col].value_counts())
```

```
36    302005
60     94025
Name: term, dtype: int64
Fully Paid     318357
Charged Off     77673
Name: loan_status, dtype: int64
0.0     338272
1.0      49739
2.0       5476
3.0       1521
4.0        527
5.0        237
6.0        122
7.0         56
8.0         34
9.0         12
10.0        11
11.0         8
13.0         4
12.0         4
19.0         2
40.0         1
17.0         1
86.0         1
24.0         1
15.0         1
Name: pub_rec, dtype: int64
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: application_type, dtype: int64
0.0    350380
1.0     42790
2.0      1847
3.0       351
4.0        82
5.0        32
6.0         7
7.0         4
8.0         2
Name: pub_rec_bankruptcies, dtype: int64
```

```
0.0     139777
1.0      60416
2.0      49948
3.0      38049
4.0      27887
5.0      18194
6.0      11069
7.0       6052
8.0       3121
9.0       1656
10.0       865
11.0       479
12.0       264
13.0       146
14.0       107
15.0        61
16.0        37
17.0        22
18.0        18
19.0        15
20.0        13
24.0        10
22.0         7
21.0         4
25.0         4
27.0         3
32.0         2
31.0         2
23.0         2
26.0         2
28.0         1
30.0         1
34.0         1
Name: mort_acc, dtype: int64
```

In [17]:
```
## combining rare levels
for col in ['pub_rec','mort_acc','pub_rec_bankruptcies']:
    # print(df[col].astype(np.float64).unique())
    df[col] = np.where(df[col].astype(np.float64)>0,1,0)
```

In [18]:
```
## As we have lesser number of levels for some columns and changing them to
for col in df.columns:
    if df[col].nunique() <= 14:
        df[col] =df[col].astype('category')
```

Categorical variables

In [19]:
```
df.describe(include='category').T
```

Out[19]:

	count	unique	top	freq
term	396030	2	36	302005
grade	396030	7	B	116018
emp_length	396030	11	10	126041
home_ownership	396030	6	MORTGAGE	198348
verification_status	396030	3	Verified	139563
loan_status	396030	2	Fully Paid	318357
purpose	396030	14	debt_consolidation	234507
pub_rec	396030	2	0	338272
initial_list_status	396030	2	f	238066
application_type	396030	3	INDIVIDUAL	395319
mort_acc	396030	2	1	218458
pub_rec_bankruptcies	396030	2	0	350915

Numeric variables

In [20]:
```
df.select_dtypes(np.number).describe().T
```

Out[20]:

	count	mean	std	min	25%	50%	75%	
loan_amnt	396030.0	14113.888089	8357.441341	500.00	8000.00	12000.00	20000.00	400
int_rate	396030.0	13.639400	4.472157	5.32	10.49	13.33	16.49	
installment	396030.0	431.849698	250.727790	16.08	250.33	375.43	567.30	15
annual_inc	396030.0	74203.175798	61637.621158	0.00	45000.00	64000.00	90000.00	87065
dti	396030.0	17.379514	18.019092	0.00	11.28	16.91	22.98	99
open_acc	396030.0	11.311153	5.137649	0.00	8.00	10.00	14.00	
revol_bal	396030.0	15844.539853	20591.836109	0.00	6025.00	11181.00	19620.00	17432
revol_util	395754.0	53.791749	24.452193	0.00	35.80	54.80	72.90	8
total_acc	396030.0	25.414744	11.886991	2.00	17.00	24.00	32.00	1

In [21]:
```
df.info()
```

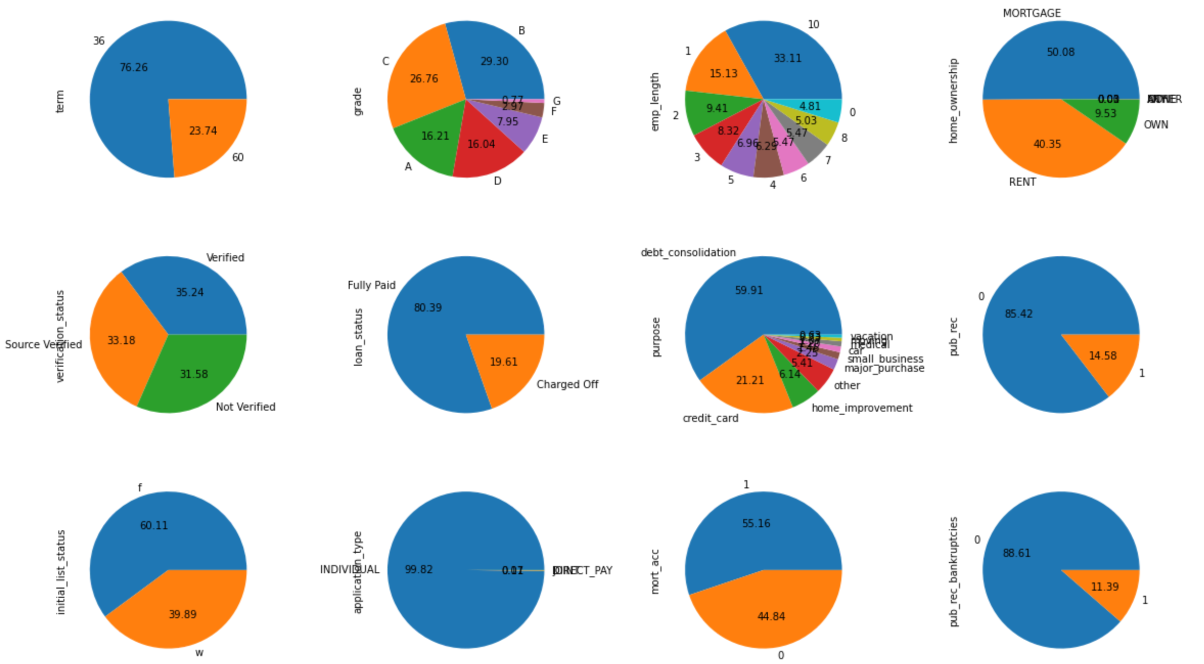
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 26 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  category
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  category
 5   sub_grade             396030 non-null  object
 6   emp_title             373103 non-null  object
 7   emp_length            396030 non-null  category
 8   home_ownership        396030 non-null  category
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  category
 11  issue_d               396030 non-null  object
 12  loan_status           396030 non-null  category
 13  purpose               396030 non-null  category
 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  category
 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  category
 23  application_type      396030 non-null  category
 24  mort_acc              396030 non-null  category
 25  pub_rec_bankruptcies  396030 non-null  category
dtypes: category(12), float64(9), object(5)
memory usage: 46.8+ MB
```

- Observations:
  - Number of rows: 396030, Number of columns: 26
  - 9 numeric variables, 17 categorical variables
  - Few columns with Missing values
  - No duplicates

Univariate Analysis

Pie charts for showing distribution of categorical variables

In [22]:
```
cat_cols_ = df.describe(include='category').columns.tolist()
# cat_cols_.remove()
cat_cols = np.array(cat_cols_).reshape(3,4)
fig, axs = plt.subplots(cat_cols.shape[0],cat_cols.shape[1], figsize=(20,12
for i in range(cat_cols.shape[0]):
    for j in range(cat_cols.shape[1]):
        if cat_cols[i][j] is not None:
            plt.figure()
            df[cat_cols[i][j]].value_counts().nlargest(10).plot(kind='pie',
```




```
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
```

Observations:

- Around 19% of the customers have charged off.
- Around 76% of the customers have term repayment as 36 months.
- Around 55% of the customers have mortgage accounts
- Around 11% of the customers have public record bankruptcies
- Around 14% of the customers have derogatory public records
- Around 31% of the customers didn't have their sources verified
- Around 59% of the customers purpose of the loan is debt consolidation and around 21% customers purpose of loan is credit card.

Statistical Summary and box plots for numeric variables

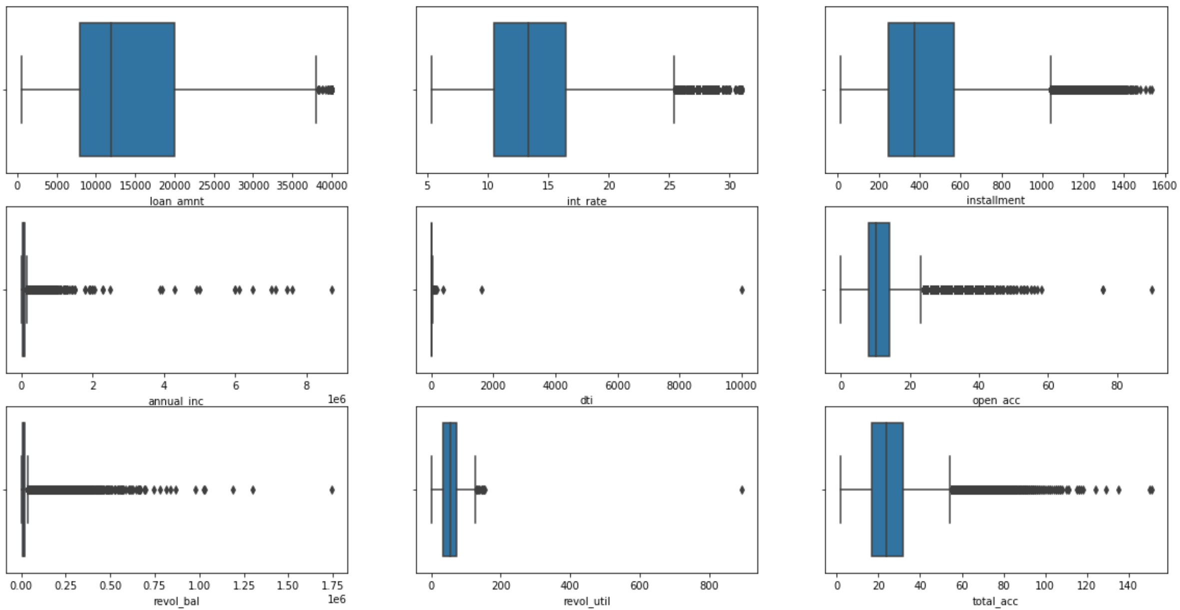
In [23]:
```
num_cols_ = df.select_dtypes(np.number).columns.tolist()
df[num_cols_] = df[num_cols_].astype(np.float64)
df[num_cols_].describe().T
```

Out[23]:

	count	mean	std	min	25%	50%	75%	
loan_amnt	396030.0	14113.888089	8357.441341	500.00	8000.00	12000.00	20000.00	400
int_rate	396030.0	13.639400	4.472157	5.32	10.49	13.33	16.49	
installment	396030.0	431.849698	250.727790	16.08	250.33	375.43	567.30	15
annual_inc	396030.0	74203.175798	61637.621158	0.00	45000.00	64000.00	90000.00	87065
dti	396030.0	17.379514	18.019092	0.00	11.28	16.91	22.98	99
open_acc	396030.0	11.311153	5.137649	0.00	8.00	10.00	14.00	
revol_bal	396030.0	15844.539853	20591.836109	0.00	6025.00	11181.00	19620.00	17432
revol_util	395754.0	53.791749	24.452193	0.00	35.80	54.80	72.90	8
total_acc	396030.0	25.414744	11.886991	2.00	17.00	24.00	32.00	1

Boxplots

In [24]:
```python
num_cols = np.array(num_cols_).reshape(3,3)
fig, axs = plt.subplots(num_cols.shape[0],num_cols.shape[1], figsize=(20,10
for i in range(num_cols.shape[0]):
    for j in range(num_cols.shape[1]):
        if num_cols[i][j] is not None:
            plt.figure()
            sns.boxplot(df[num_cols[i][j]],ax=axs[i][j])
            axs[i][j].set_xlabel(num_cols[i][j])
```




```
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
<Figure size 432x288 with 0 Axes>
```

Outliers

In [25]:
```python
def getOutliers(Series_):
    q1 = np.quantile(Series_,0.25)
```

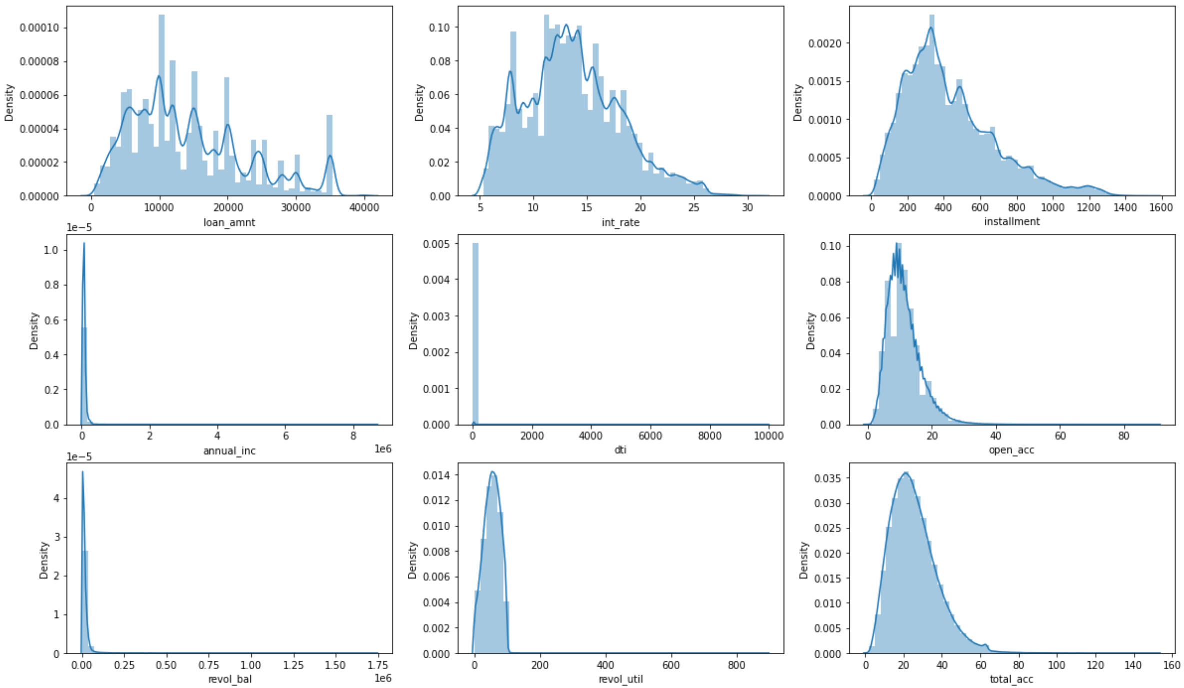
```
    q3 = np.quantile(Series_,0.75)
    IQR = (q3-q1)
    ub_ = q3+1.5*IQR
    lb_ = q1-1.5*IQR
    if lb_ < min(Series_): lb_ = min(Series_)
    #print(q1,q3,IQR,lb_,ub_)
    return {Series_.name:[lb_,ub_,100*Series_[((Series_ < lb_) | (Series_ >
for col in num_cols_:
    print(getOutliers(df[col]))
```

```
{'loan_amnt': [500.0, 38000.0, 0.04822866954523647]}
{'int_rate': [5.32, 25.489999999999995, 0.9537156276039694]}
{'installment': [16.08, 1042.7549999999999, 2.8406938868267555]}
{'annual_inc': [0.0, 157500.0, 4.216852258667273]}
{'dti': [0.0, 40.53, 0.06943918390020958]}
{'open_acc': [0.0, 23.0, 2.602580612579855]}
{'revol_bal': [0.0, 40012.5, 5.368027674671111]}
{'revol_util': [nan, nan, 0.0]}
{'total_acc': [2.0, 54.5, 2.146049541701386]}
```

- Outliers
    - 'revol_bal' and 'annual_inc' has more than 4% outliers,

Distplots

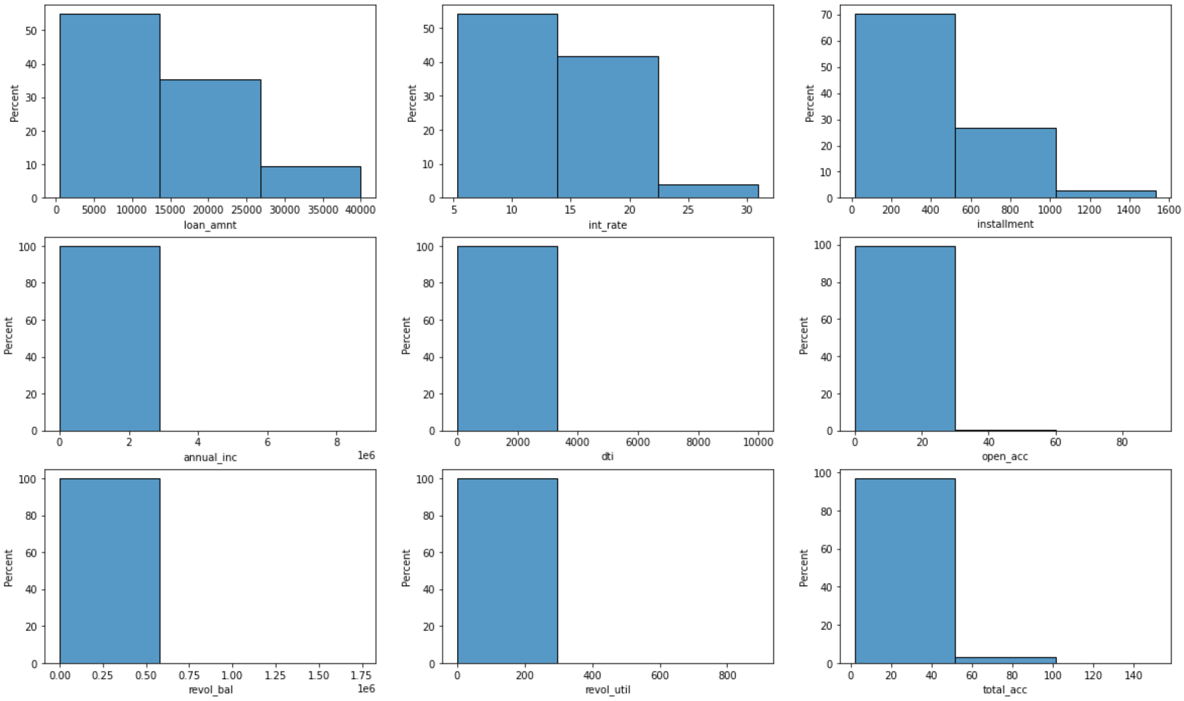
In [26]:
```
# num_cols = np.array(num_cols_).reshape(2,2)
fig, axs = plt.subplots(num_cols.shape[0],num_cols.shape[1], figsize=(20,12
for i in range(num_cols.shape[0]):
    for j in range(num_cols.shape[1]):
        if num_cols[i][j] is not None:
            sns.distplot(df[num_cols[i][j]],ax=axs[i][j])
            axs[i][j].set_xlabel(num_cols[i][j])
```



Histplots

In [27]:
```
# num_cols = np.array(num_cols_).reshape(2,2)
fig, axs = plt.subplots(num_cols.shape[0],num_cols.shape[1], figsize=(20,12
for i in range(num_cols.shape[0]):
    for j in range(num_cols.shape[1]):
```

```
        if num_cols[i][j] is not None:
            sns.histplot(df[num_cols[i][j]],bins=3,stat='percent',ax=axs[i]
            axs[i][j].set_xlabel(num_cols[i][j])
```



Observations

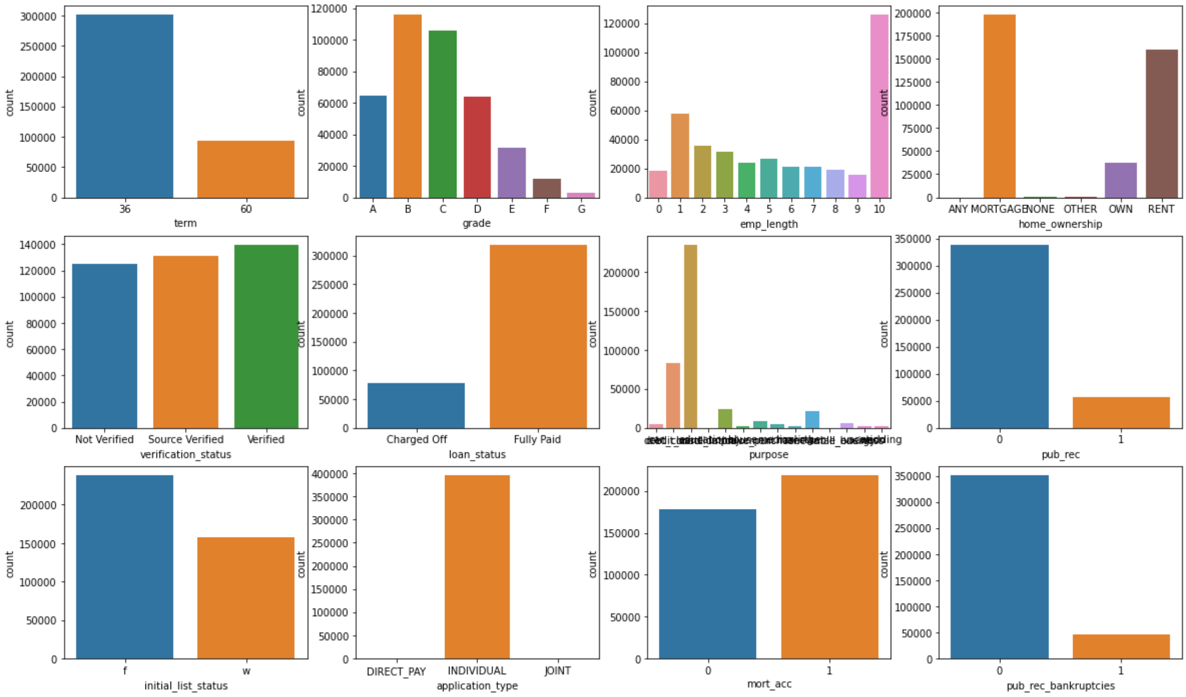
- More than 50% customers have less than 15000 loan amount
- More than 40% customers have interest rate more than 15%
- More than 40% customers have installments greater than 500

Barplots

In [28]:
```
# cat_cols = np.array(df.describe(include='category').columns.tolist()).res
fig, axs = plt.subplots(cat_cols.shape[0],cat_cols.shape[1], figsize=(20,12
for i in range(cat_cols.shape[0]):
    for j in range(cat_cols.shape[1]):
        if cat_cols[i][j] != 'NA':
            sns.countplot(x=cat_cols[i][j],data=df,ax=axs[i][j])
            display(df[cat_cols[i][j]].value_counts())
```

```
36    302005
60     94025
Name: term, dtype: int64
B    116018
C    105987
A     64187
D     63524
E     31488
F     11772
G      3054
Name: grade, dtype: int64
```

```
10    126041
1      57607
2      35827
3      31665
5      26495
4      23952
6      20841
7      20819
8      19168
0      18301
9      15314
Name: emp_length, dtype: int64
MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: home_ownership, dtype: int64
Verified           139563
Source Verified    131385
Not Verified       125082
Name: verification_status, dtype: int64
Fully Paid     318357
Charged Off     77673
Name: loan_status, dtype: int64
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
house                   2201
wedding                 1812
renewable_energy         329
educational              257
Name: purpose, dtype: int64
0    338272
1     57758
Name: pub_rec, dtype: int64
f    238066
w    157964
Name: initial_list_status, dtype: int64
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: application_type, dtype: int64
1    218458
0    177572
Name: mort_acc, dtype: int64
0    350915
1     45115
Name: pub_rec_bankruptcies, dtype: int64
```



Observations:

- Around 19%(>80k) of the customers have charged off.
- Around 24%(~100k) of the customers have term repayment as 60 months.
- Around 55%(>175k) of the customers have mortgage accounts
- Around 11%(~50k) of the customers have public record bankruptcies
- Around 14%(>50k) of the customers have derogatory public records
- Around 31%(>120k) of the customers didn't have their sources verified
- Around 59%(>230k) of the customers purpose of the loan is debt consolidation and around 21%(>90k) customers purpose of loan is credit card.

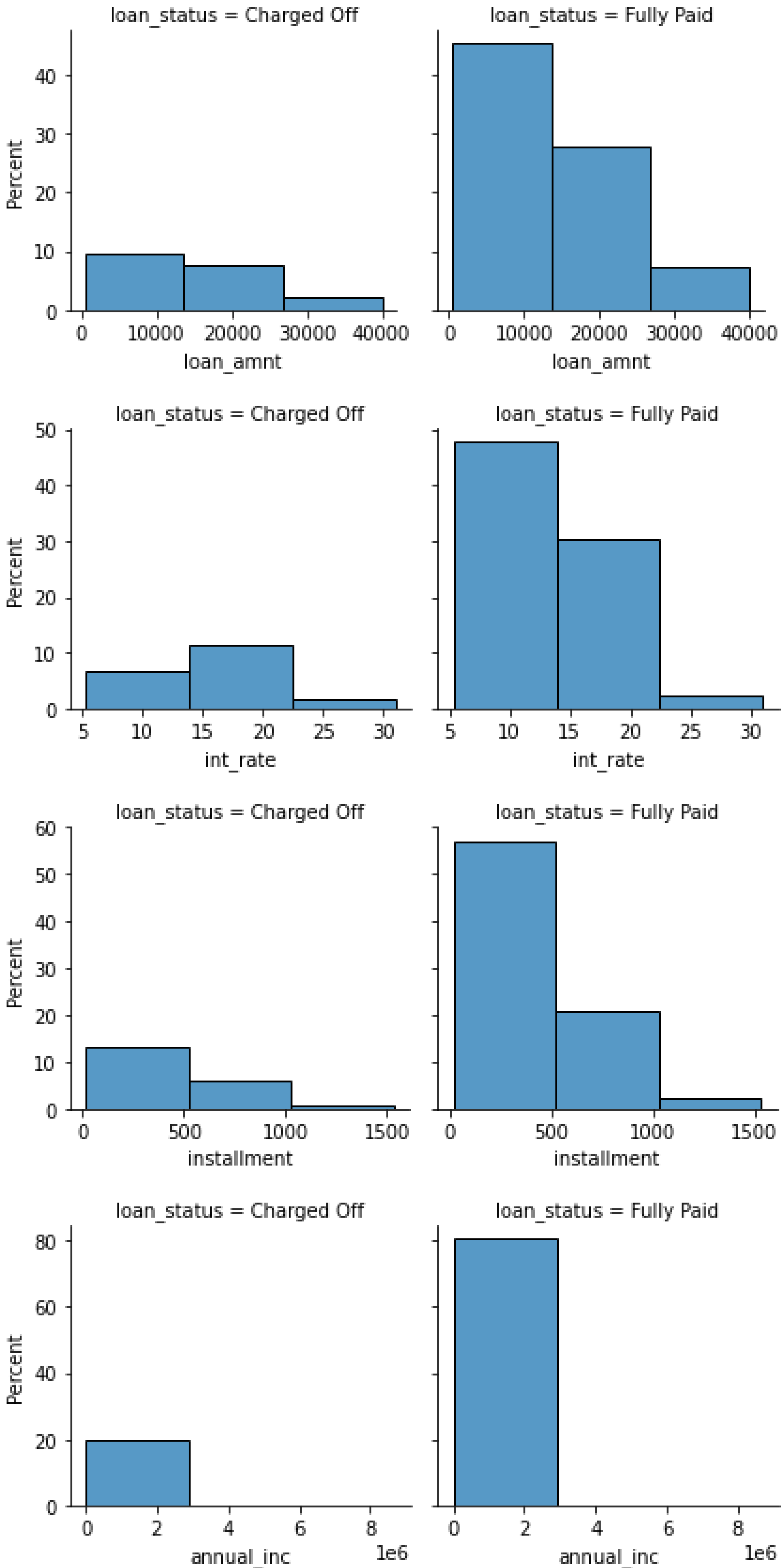
Bivariate Analysis

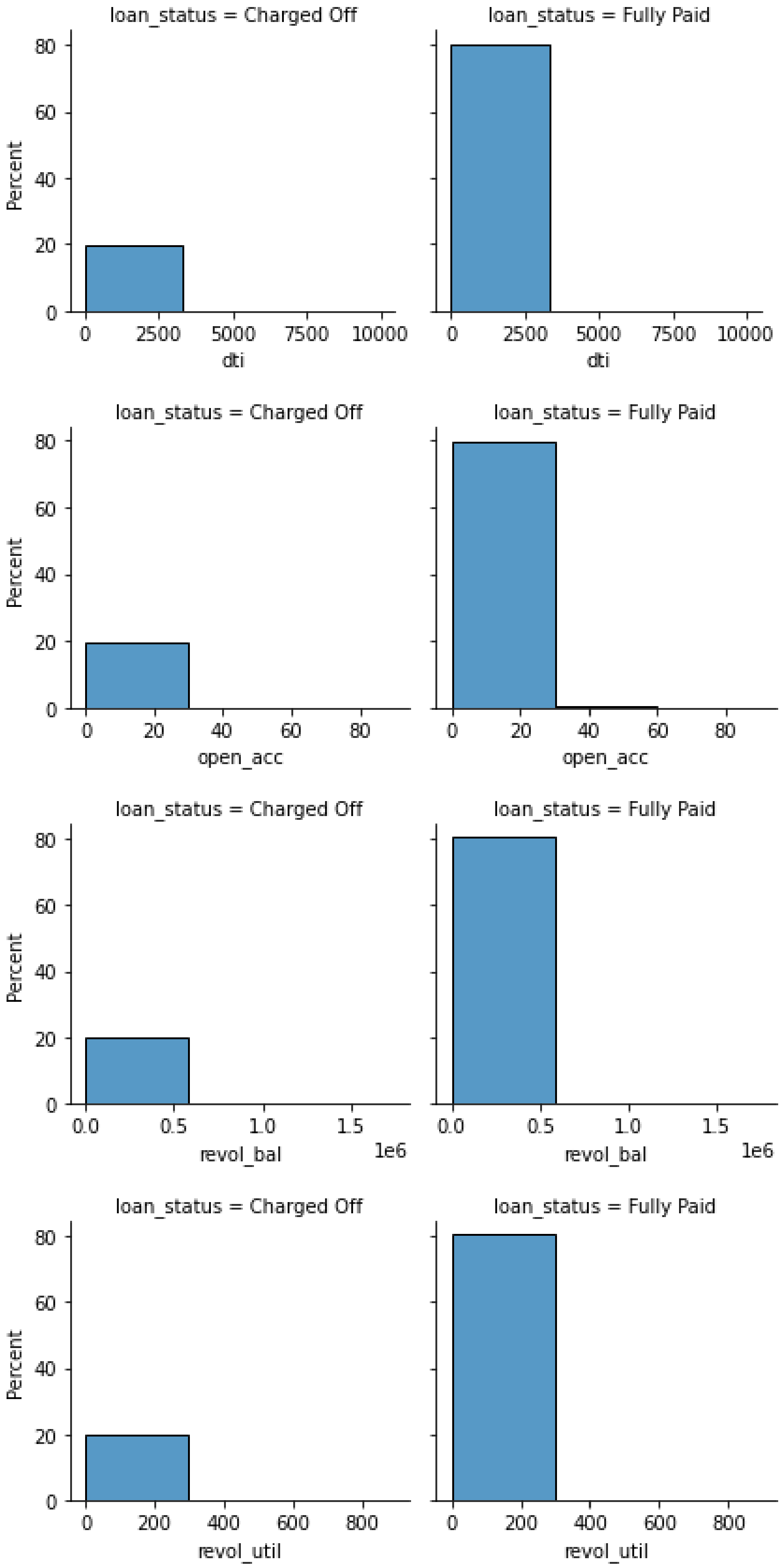
Effect of independant variables on **Loan Status**

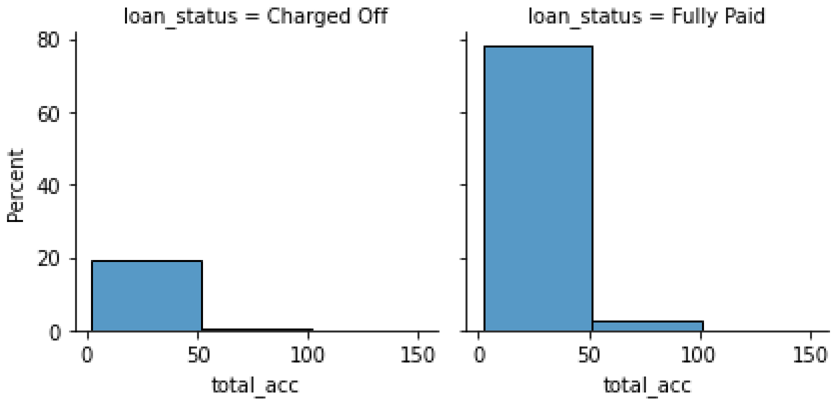
Distplots

In [29]:
```
target_ = 'loan_status'
```

In [30]:
```
for col in num_cols_:
    sns.displot(data=df, x=col, stat='percent', bins=3, col=target_,height=
```

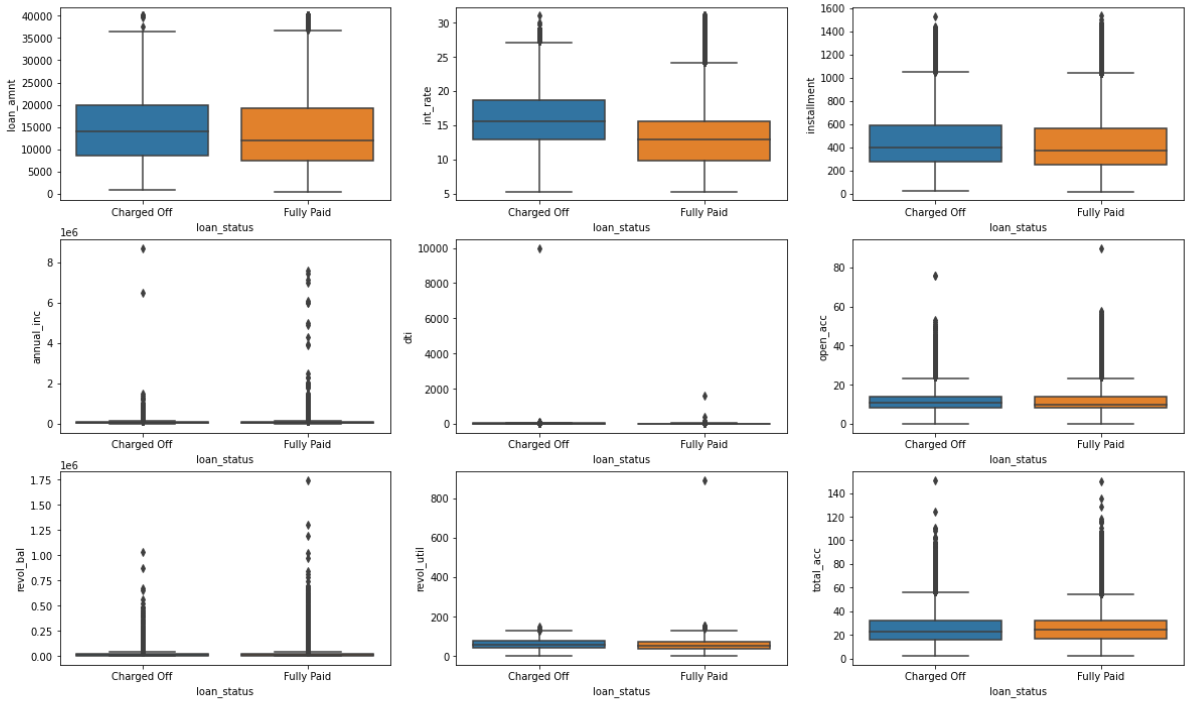

loan_status = Charged Off
loan_status = Fully Paid
Percent
loan_amnt
loan_status = Charged Off
loan_status = Fully Paid
Percent
int_rate
loan_status = Charged Off
loan_status = Fully Paid
Percent
installment
loan_status = Charged Off
loan_status = Fully Paid
Percent
annual_inc
1e6


loan_status = Charged Off
loan_status = Fully Paid
Percent
0
20
40
60
80
0
2500
5000
7500
10000
dti
0
20
40
60
80
open_acc
0.0
0.5
1.0
1.5
1e6
revol_bal
0
200
400
600
800
revol_util



Boxplots

In [31]:
```
fig, axs = plt.subplots(num_cols.shape[0],num_cols.shape[1], figsize=(20,12
for i in range(num_cols.shape[0]):
    for j in range(num_cols.shape[1]):
        if num_cols[i][j] is not None:
            sns.boxplot(y=num_cols[i][j], x=target_,data=df,ax=axs[i][j])
            axs[i][j].set_xlabel('loan_status')
```



In [32]:
```
for col in df.describe(include='category'):
    if col!= 'loan_status':
        print("========================================")
        print ("Loan Status vs {}".format(col))
        display(df[['loan_status',col]].value_counts())
```

```
=========================================
Loan Status vs term
loan_status  term
Fully Paid   36      254365
             60       63992
Charged Off  36       47640
             60       30033
dtype: int64
=========================================
Loan Status vs grade
```

```
loan_status  grade
Fully Paid   B        101431
             C         83538
             A         60151
             D         45186
Charged Off  C         22449
Fully Paid   E         19723
Charged Off  D         18338
             B         14587
             E         11765
Fully Paid   F          6735
Charged Off  F          5037
             A          4036
Fully Paid   G          1593
Charged Off  G          1461
dtype: int64
=========================================
Loan Status vs emp_length
loan_status  emp_length
Fully Paid   10            102826
             1              45890
             2              28903
             3              25483
Charged Off  10             23215
Fully Paid   5              21403
             4              19344
             6              16898
             7              16764
             8              15339
             0              13263
             9              12244
Charged Off  1              11717
             2               6924
             3               6182
             5               5092
             0               5038
             4               4608
             7               4055
             6               3943
             8               3829
             9               3070
dtype: int64
=========================================
Loan Status vs home_ownership
loan_status  home_ownership
Fully Paid   MORTGAGE          164716
             RENT              123578
Charged Off  RENT               36212
             MORTGAGE           33632
Fully Paid   OWN                29940
Charged Off  OWN                 7806
Fully Paid   OTHER                 96
             NONE                  24
Charged Off  OTHER                 16
             NONE                   7
Fully Paid   ANY                    3
dtype: int64
=========================================
Loan Status vs verification_status
```

```
loan_status  verification_status
Fully Paid   Verified              108411
             Not Verified          106775
             Source Verified       103171
Charged Off  Verified               31152
             Source Verified        28214
             Not Verified           18307
dtype: int64
=========================================
Loan Status vs purpose
loan_status  purpose
Fully Paid   debt_consolidation    185867
             credit_card            69145
Charged Off  debt_consolidation     48640
Fully Paid   home_improvement       19943
             other                  16690
Charged Off  credit_card            13874
Fully Paid   major_purchase          7342
Charged Off  other                   4495
             home_improvement        4087
Fully Paid   car                     4064
             small_business          4022
             medical                 3285
             moving                  2184
             vacation                1988
             house                   1767
Charged Off  small_business          1679
Fully Paid   wedding                 1593
Charged Off  major_purchase          1448
             medical                  911
             moving                   670
             car                      633
             vacation                 464
             house                    434
Fully Paid   renewable_energy         252
Charged Off  wedding                  219
Fully Paid   educational              215
Charged Off  renewable_energy          77
             educational               42
dtype: int64
=========================================
Loan Status vs pub_rec
loan_status  pub_rec
Fully Paid   0          272933
Charged Off  0           65339
Fully Paid   1           45424
Charged Off  1           12334
dtype: int64
=========================================
Loan Status vs initial_list_status
loan_status  initial_list_status
Fully Paid   f                      192105
             w                      126252
Charged Off  f                       45961
             w                       31712
dtype: int64
=========================================
Loan Status vs application_type
```

```
loan_status  application_type
Fully Paid   INDIVIDUAL          317802
Charged Off  INDIVIDUAL           77517
Fully Paid   JOINT                  371
             DIRECT_PAY             184
Charged Off  DIRECT_PAY             102
             JOINT                   54
dtype: int64
=========================================
Loan Status vs mort_acc
loan_status  mort_acc
Fully Paid   1           179492
             0           138865
Charged Off  1            38966
             0            38707
dtype: int64
=========================================
Loan Status vs pub_rec_bankruptcies
loan_status  pub_rec_bankruptcies
Fully Paid   0                       282507
Charged Off  0                        68408
Fully Paid   1                        35850
Charged Off  1                         9265
dtype: int64
```

In [33]:

```
df1 =df.copy()
df1['loan_amnt_binned'] = pd.qcut(df1['loan_amnt'], q=3)
df1['int_rate_binned'] = pd.qcut(df1['int_rate'], q=3)
```

In [34]:

```
df1.groupby(['loan_amnt_binned','int_rate_binned','term'])['loan_status'].a
```

Out[34]:
```
loan_amnt_binned    int_rate_binned  term
(499.999, 9600.0]   (5.319, 11.53]   36    Fully Paid     0.907505
                                           Charged Off    0.092495
                                     60    Fully Paid     0.856164
                                           Charged Off    0.143836
                    (11.53, 15.31]   36    Fully Paid     0.827383
                                           Charged Off    0.172617
                                     60    Fully Paid     0.753024
                                           Charged Off    0.246976
                    (15.31, 30.99]   36    Fully Paid     0.747463
                                           Charged Off    0.252537
                                     60    Fully Paid     0.689698
                                           Charged Off    0.310302
(9600.0, 16000.0]   (5.319, 11.53]   36    Fully Paid     0.910249
                                           Charged Off    0.089751
                                     60    Fully Paid     0.860030
                                           Charged Off    0.139970
                    (11.53, 15.31]   36    Fully Paid     0.825379
                                           Charged Off    0.174621
                                     60    Fully Paid     0.754782
                                           Charged Off    0.245218
                    (15.31, 30.99]   36    Fully Paid     0.729346
                                           Charged Off    0.270654
                                     60    Fully Paid     0.611837
                                           Charged Off    0.388163
(16000.0, 40000.0]  (5.319, 11.53]   36    Fully Paid     0.920335
                                           Charged Off    0.079665
                                     60    Fully Paid     0.854080
                                           Charged Off    0.145920
                    (11.53, 15.31]   36    Fully Paid     0.834041
                                           Charged Off    0.165959
                                     60    Fully Paid     0.763983
                                           Charged Off    0.236017
                    (15.31, 30.99]   36    Fully Paid     0.734062
                                           Charged Off    0.265938
                                     60    Fully Paid     0.622203
                                           Charged Off    0.377797
Name: loan_status, dtype: float64
```

In [35]:
```python
df1.groupby(['mort_acc','pub_rec','pub_rec_bankruptcies'])['loan_status'].a
```

Out[35]:
```
mort_acc  pub_rec  pub_rec_bankruptcies
0         0        0                     Fully Paid     0.786819
                                         Charged Off    0.213181
          1        0                     Fully Paid     0.711773
                                         Charged Off    0.288227
                   1                     Fully Paid     0.756915
                                         Charged Off    0.243085
1         0        0                     Fully Paid     0.824167
                                         Charged Off    0.175833
          1        0                     Fully Paid     0.788755
                                         Charged Off    0.211245
                   1                     Fully Paid     0.814396
                                         Charged Off    0.185604
Name: loan_status, dtype: float64
```

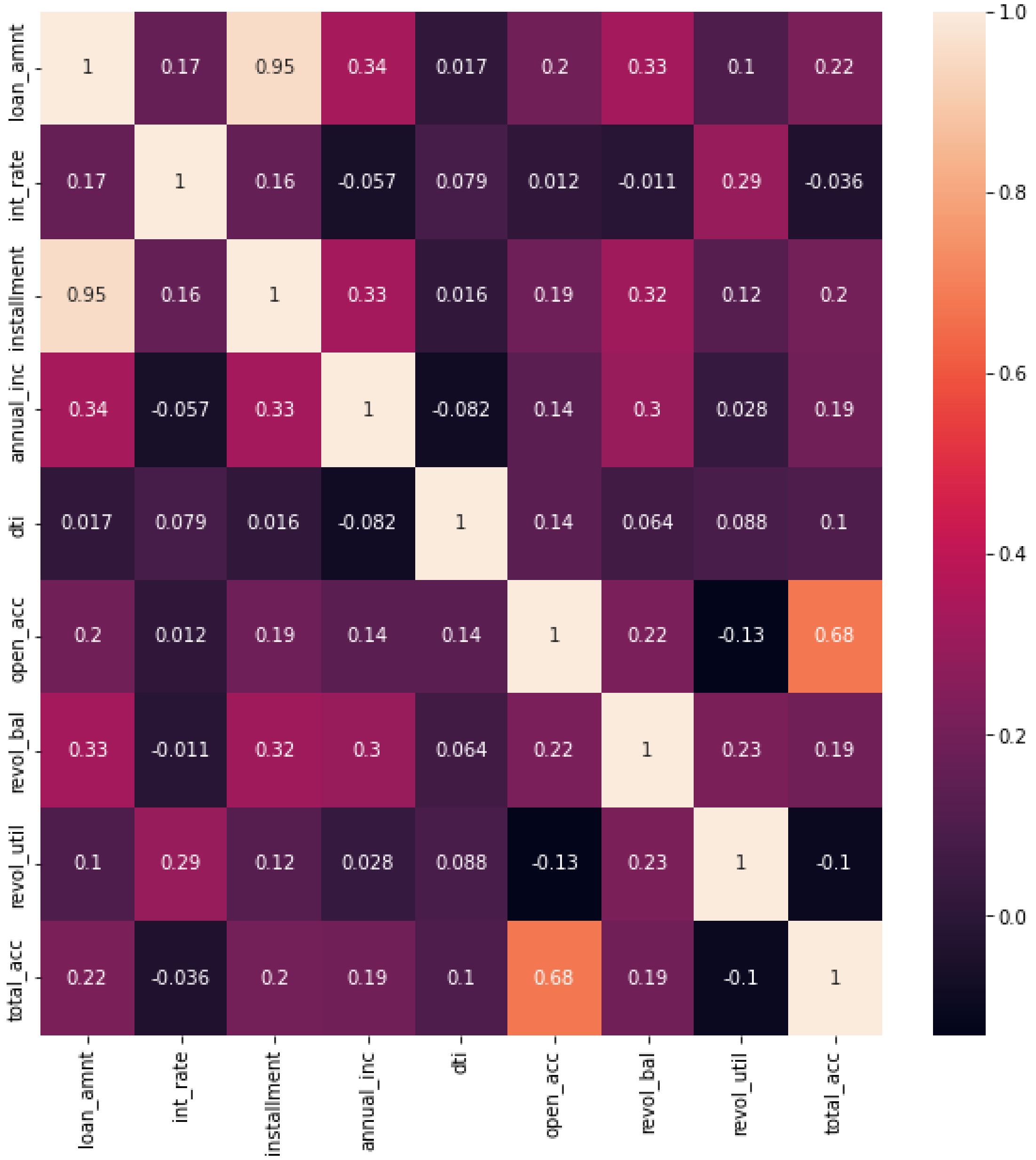
In [36]:
```python
df1['open_acc_binned'] = pd.qcut(df1['open_acc'], q=3)
df1['dti_binned'] = pd.qcut(df1['dti'], q=5)
df1.groupby(['mort_acc','open_acc_binned','dti_binned'])['loan_status'].app
```

Out[36]:
```
mort_acc  open_acc_binned  dti_binned
0         (-0.001, 9.0]    (-0.001, 10.01]  Fully Paid     0.844464
                                            Charged Off    0.155536
                           (10.01, 14.7]    Fully Paid     0.813989
                                            Charged Off    0.186011
                           (14.7, 19.16]    Fully Paid     0.791791
                                            Charged Off    0.208209
                           (19.16, 24.48]   Fully Paid     0.764754
                                            Charged Off    0.235246
                           (24.48, 9999.0]  Fully Paid     0.693031
                                            Charged Off    0.306969
          (9.0, 13.0]      (-0.001, 10.01]  Fully Paid     0.839115
                                            Charged Off    0.160885
                           (10.01, 14.7]    Fully Paid     0.817829
                                            Charged Off    0.182171
                           (14.7, 19.16]    Fully Paid     0.797963
                                            Charged Off    0.202037
                           (19.16, 24.48]   Fully Paid     0.770010
                                            Charged Off    0.229990
                           (24.48, 9999.0]  Fully Paid     0.681481
                                            Charged Off    0.318519
          (13.0, 90.0]     (-0.001, 10.01]  Fully Paid     0.835336
                                            Charged Off    0.164664
                           (10.01, 14.7]    Fully Paid     0.817880
                                            Charged Off    0.182120
                           (14.7, 19.16]    Fully Paid     0.799638
                                            Charged Off    0.200362
                           (19.16, 24.48]   Fully Paid     0.769966
                                            Charged Off    0.230034
                           (24.48, 9999.0]  Fully Paid     0.677249
                                            Charged Off    0.322751
1         (-0.001, 9.0]    (-0.001, 10.01]  Fully Paid     0.881673
                                            Charged Off    0.118327
                           (10.01, 14.7]    Fully Paid     0.854149
                                            Charged Off    0.145851
                           (14.7, 19.16]    Fully Paid     0.831122
                                            Charged Off    0.168878
                           (19.16, 24.48]   Fully Paid     0.796901
                                            Charged Off    0.203099
                           (24.48, 9999.0]  Fully Paid     0.748611
                                            Charged Off    0.251389
          (9.0, 13.0]      (-0.001, 10.01]  Fully Paid     0.880014
                                            Charged Off    0.119986
                           (10.01, 14.7]    Fully Paid     0.859303
                                            Charged Off    0.140697
                           (14.7, 19.16]    Fully Paid     0.822126
                                            Charged Off    0.177874
                           (19.16, 24.48]   Fully Paid     0.796613
                                            Charged Off    0.203387
                           (24.48, 9999.0]  Fully Paid     0.740274
                                            Charged Off    0.259726
          (13.0, 90.0]     (-0.001, 10.01]  Fully Paid     0.882748
                                            Charged Off    0.117252
                           (10.01, 14.7]    Fully Paid     0.867085
                                            Charged Off    0.132915
                           (14.7, 19.16]    Fully Paid     0.833287
                                            Charged Off    0.166713
                           (19.16, 24.48]   Fully Paid     0.803799
                                            Charged Off    0.196201
                           (24.48, 9999.0]  Fully Paid     0.742471
                                            Charged Off    0.257529
Name: loan_status, dtype: float64
```

- Observations
  - When the loan_amount, int_rate and term is more there are slightly more chances for charged off
  - When the customers didn't have any mortgage accounts and have derogatory public records and public record bankruptcies, more chances for charged off
  - When the customers didn't have any mortgage accounts and have no of open credit lines and higher dti, more chances for charged off

In [37]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df1[num_cols_+[target_]].corr(),annot=True)
```

Out[37]: <AxesSubplot:>



- loan amount and installment have higher correlation
- total_acc and open_acc have little high correlation

In [38]:
```
# plt.figure(figsize=(10,10))
# sns.pairplot(df)
#### As chances of getting inferences from parplot is low and as it is taki
```

Missing Value treatment

In [39]:
```
missing_cols = [col for col in df.columns if df[col].isnull().sum()>0]
missing_cols
```

Out[39]:
```
['emp_title', 'title', 'revol_util']
```

In [40]:
```
df[missing_cols].head()
```

Out[40]:

	emp_title	title	revol_util
0	Marketing	Vacation	41.8
1	Credit analyst	Debt consolidation	53.3
2	Statistician	Credit card refinancing	92.2
3	Client Advocate	Credit card refinancing	21.5
4	Destiny Management Inc.	Credit Card Refinance	69.8

In [41]:
```
# We will impute categoricalcols with unknown and numeric column with mean
df[missing_cols[:-1]]=df[missing_cols[:-1]].fillna('Unknown')

df['revol_util'] = df['revol_util'].fillna(df['revol_util'].mean())
```

Outlier treatment

In [42]:
```
for col in num_cols_:
    print(getOutliers(df[col]))
```

```
{'loan_amnt': [500.0, 38000.0, 0.04822866954523647]}
{'int_rate': [5.32, 25.489999999999995, 0.9537156276039694]}
{'installment': [16.08, 1042.7549999999999, 2.8406938868267555]}
{'annual_inc': [0.0, 157500.0, 4.216852258667273]}
{'dti': [0.0, 40.53, 0.06943918390020958]}
{'open_acc': [0.0, 23.0, 2.602580612579855]}
{'revol_bal': [0.0, 40012.5, 5.368027674671111]}
{'revol_util': [0.0, 128.40000000000003, 0.0030300734792818728]}
{'total_acc': [2.0, 54.5, 2.146049541701386]}
```

Handling outliers using winsorization

In [43]:
```
df_ = df.copy()
for col in num_cols_:
    out_limit= 0.055 if col in ['annual_inc','revol_bal'] else 0.03
    df_[col] = st.mstats.winsorize(df_[col],limits=[0.02,out_limit],inplace
```

Model building

In [44]:
```
le_dct={}
for col in df_.columns:
    try:
        df_[col] = pd.to_numeric(df_[col])
    except:
        le_dct[col] = LabelEncoder()
        df_[col] = le_dct[col].fit_transform(df_[col].astype(str))
```

In [45]:
```
df_.head()
```

Out[45]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_own
0	10000.0	36	11.44	329.48	1	8	80956	10	
1	8000.0	36	11.99	265.68	1	9	33317	4	
2	15600.0	36	10.49	506.97	1	7	127182	1	
3	7200.0	36	6.49	220.65	0	1	27760	6	
4	24375.0	60	17.27	609.33	2	14	38300	9	

5 rows × 26 columns

In [46]:
```
x = df_.copy()
y= x.pop(target_)
```

Train test split

In [47]:
```
 X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3, s
```

Standardization

In [48]:
```
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

In [49]:
```
X_train=pd.DataFrame(X_train,columns=x.columns)
y_train.reset_index(drop=True,inplace=True)
X_test=pd.DataFrame(X_test,columns=x.columns)
y_test.reset_index(drop=True,inplace=True)
```

Modelling

Linear Regression

In [50]:
```
log_reg = LogisticRegression(random_state=123)
log_reg = log_reg.fit(X_train,y_train)
```

In [51]:
```
display('Logistic Regression Model Coefficients')
display(pd.Series(log_reg.coef_.ravel(),index=x.columns))
```

```
'Logistic Regression Model Coefficients'
```

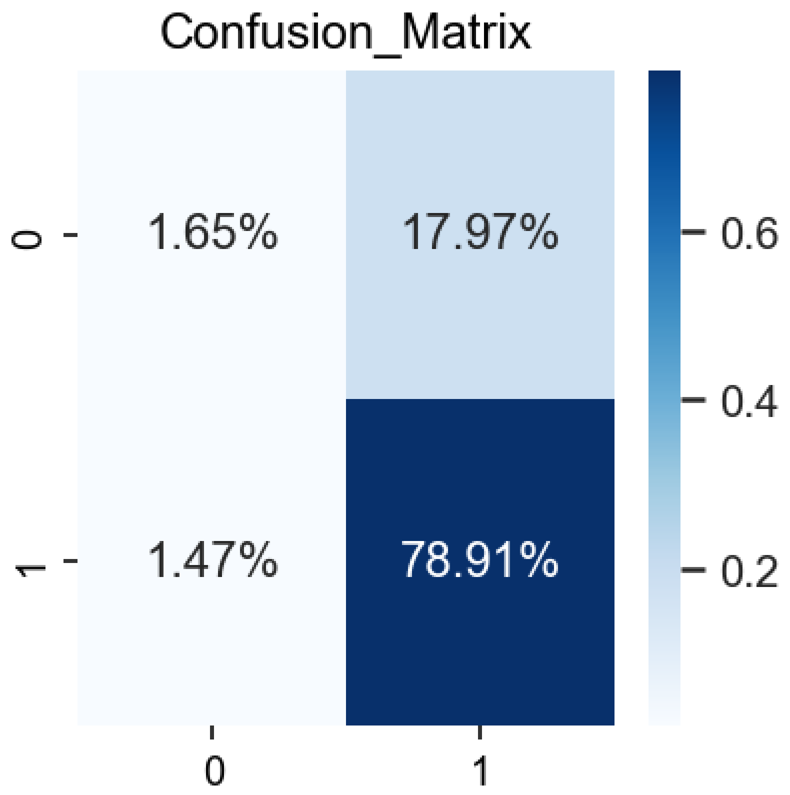
```
loan_amnt              -0.171235
term                   -0.162242
int_rate                0.126025
installment             0.038890
grade                   0.014925
sub_grade              -0.595487
emp_title              -0.111340
emp_length              0.048500
home_ownership         -0.124565
annual_inc              0.194326
verification_status    -0.025314
issue_d                 0.000082
purpose                -0.037640
title                   0.020696
dti                    -0.193199
earliest_cr_line        0.004703
open_acc               -0.148433
pub_rec                -0.085627
revol_bal               0.097170
revol_util             -0.121086
total_acc               0.110467
initial_list_status    -0.006754
application_type        0.016804
mort_acc                0.001775
pub_rec_bankruptcies    0.064063
dtype: float64
```

In [52]:
```
y_pred = log_reg.predict(X_test)
y_score = log_reg.predict_proba(X_test)
```

Confusion-matrix

In [53]:
```
cf_matrix = confusion_matrix(y_test, y_pred,normalize='all')
f, ax = plt.subplots(figsize=(3, 3), dpi=144)
sns.set(font_scale=1)
sns.heatmap(cf_matrix, annot=True, fmt=".2%", cmap="Blues", ax=ax)
plt.title("Confusion_Matrix")
```

Out[53]:
```
Text(0.5, 1.0, 'Confusion_Matrix')
```



Classification Report

In [54]:
```
print({0:'Charged Off',1:'Fully Paid'})
print(classification_report(y_test, y_pred))
```

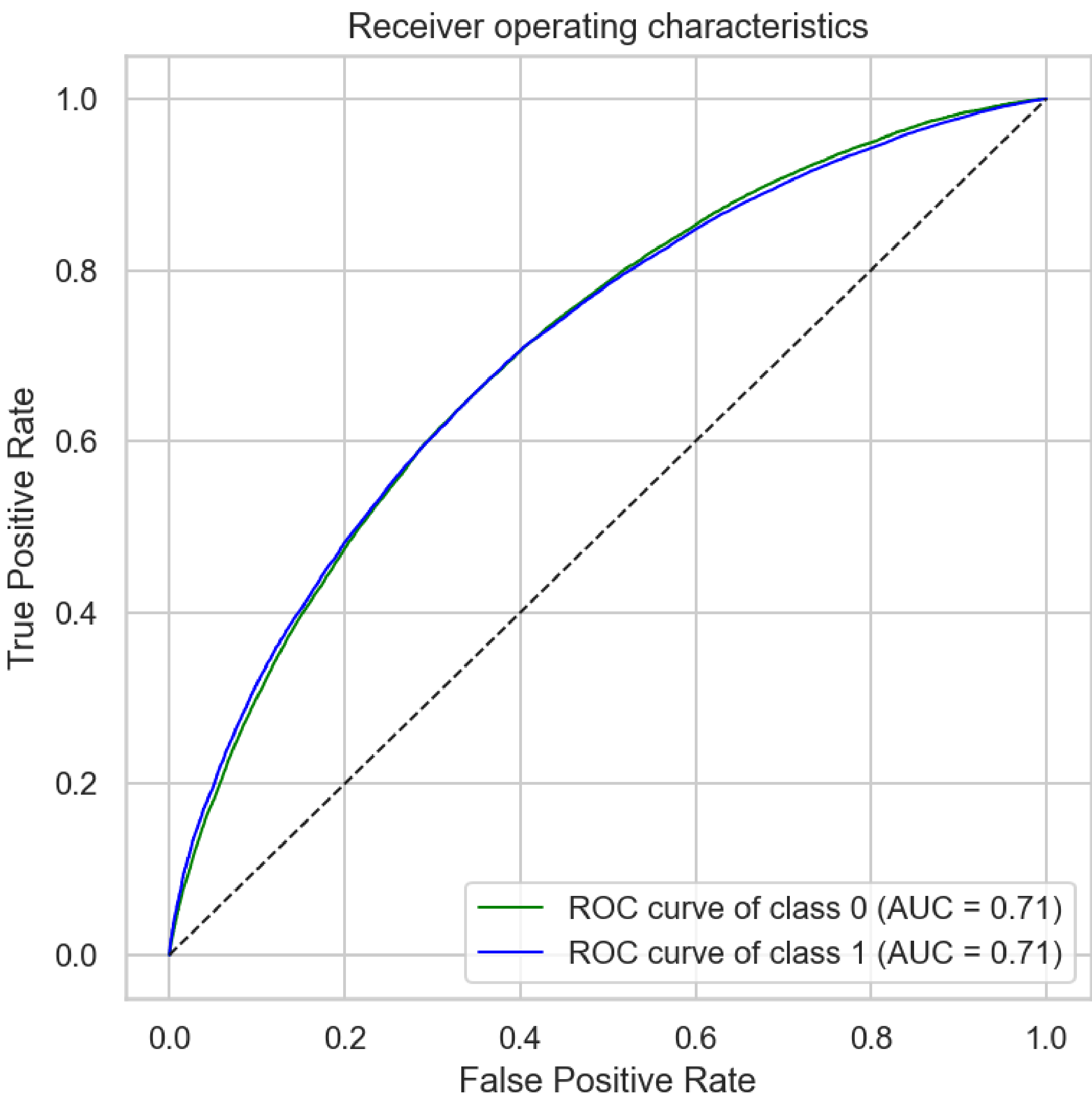
```
{0: 'Charged Off', 1: 'Fully Paid'}
              precision    recall  f1-score   support

           0       0.53      0.08      0.14     23302
           1       0.81      0.98      0.89     95507

    accuracy                           0.81    118809
   macro avg       0.67      0.53      0.52    118809
weighted avg       0.76      0.81      0.74    118809
```

ROC-AUC Curve

In [55]:
```
fpr = dict()
tpr = dict()
roc_auc = dict()
n_classes=2
y_score = log_reg.predict_proba(X_test)
y_testG = label_binarize(y_test, classes=range(n_classes + 1))
y_testG = y_testG[:, :n_classes]
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_testG[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

plt.figure(figsize=(6, 6), dpi=144)
sns.set(font_scale=1)
sns.set_style("whitegrid")

lw = 1
colors = cycle(["green", "blue"])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i],tpr[i],color=color,lw=lw,label="ROC curve of class {0}
```

```
plt.plot([0, 1], [0, 1], "k--", lw=lw)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
# plt.rcParams['font.size'] = 10
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristics")
plt.legend(loc="lower right")
```

Out[55]: <matplotlib.legend.Legend at 0x7f7a5b962f40>



Precision-Recall Curve

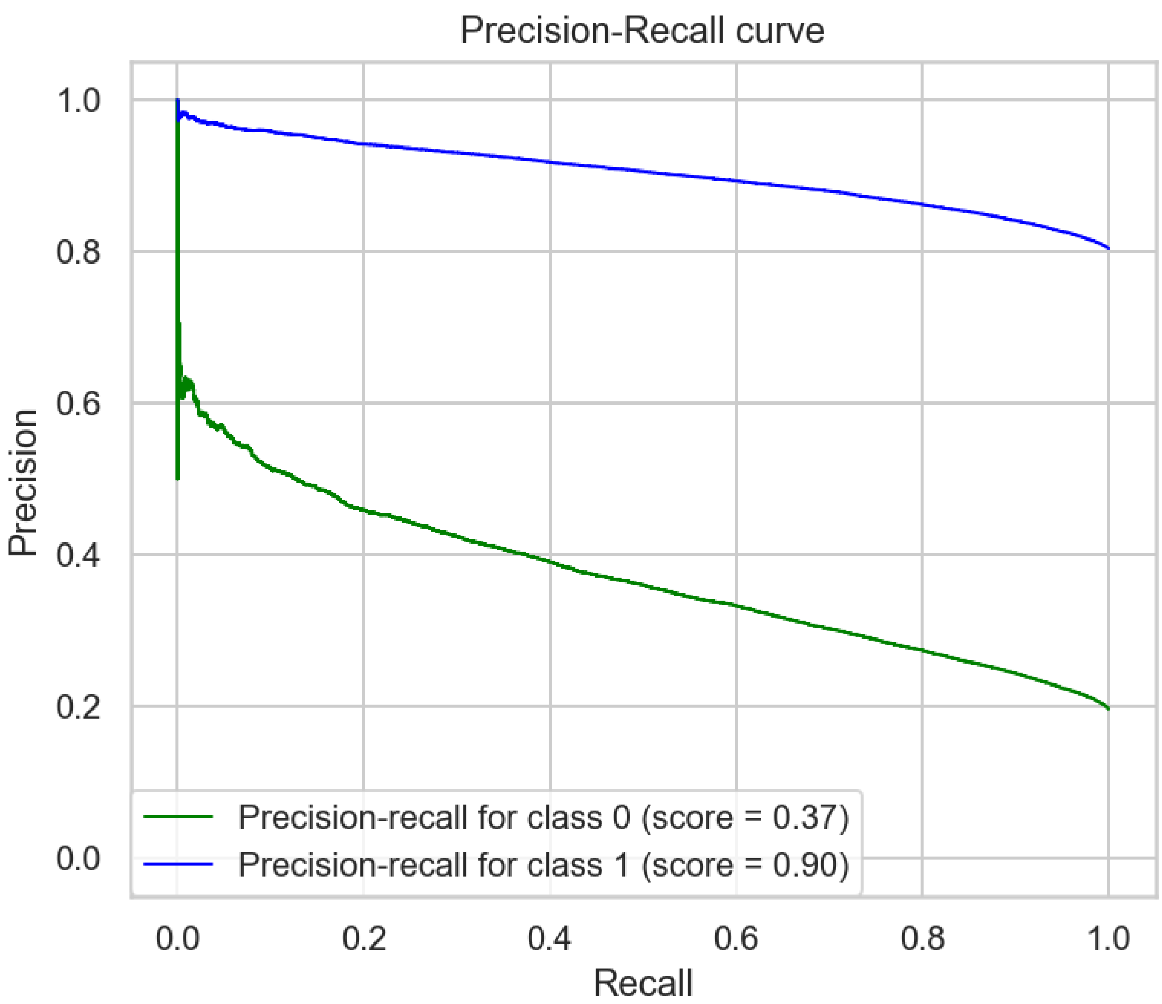
In [56]:

```
precision = dict()
recall = dict()
average_precision = dict()
for i in range(n_classes):
    precision[i], recall[i], _ = precision_recall_curve(
        y_testG[:, i], y_score[:, i]
    )
    average_precision[i] = average_precision_score(y_testG[:, i], y_score[:

colors = cycle(["green", "blue"])

plt.figure(figsize=(6, 6), dpi=144)
```

```
sns.set(font_scale=1)
sns.set_style("whitegrid")

lines = []
labels = []

for i, color in zip(range(n_classes), colors):
    (l,) = plt.plot(recall[i], precision[i], color=color, lw=1)
    lines.append(l)
    labels.append(
        "Precision-recall for class {0} (score = {1:0.2f})"
        "".format(i, average_precision[i])
    )

fig = plt.gcf()
fig.subplots_adjust(bottom=0.25)
plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])

plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("Precision-Recall curve")
plt.legend(lines, labels, loc=(0, 0))
```

Out[56]:

```
<matplotlib.legend.Legend at 0x7f7a5b8f79a0>
```



Insights and Recommendations

- Insights

    - More than 50% customers have less than 15000 loan amount
    - More than 40% customers have interest rate more than 15%
    - More than 40% customers have installments greater than 500
    - Around 19%(>80k) of the customers have charged off.
    - Around 24%(~100k) of the customers have term repayment as 60 months.
    - Around 55%(>175k) of the customers have mortgage accounts
    - Around 11%(~50k) of the customers have public record bankruptcies
    - Around 14%(>50k) of the customers have derogatory public records
    - Around 31%(>120k) of the customers didn't have their sources verified
    - Around 59%(>230k) of the customers purpose of the loan is debt consolidation and around 21%(>90k) customers purpose of loan is credit card.
    - When the loan_amount, int_rate and term is more there are slightly more chances for charged off
    - When the customers didn't have any mortgage accounts and have derogatory public records and public record bankruptcies, more chances for charged off
    - When the customers didn't have any mortgage accounts and have no of open credit lines and higher dti, more chances for charged off

    - When the loan_amount, int_rate and term is more there are slightly more chances for charged off
    - When the customers didn't have any mortgage accounts and have derogatory public records and public record bankruptcies, more chances for charged off
    - When the customers didn't have any mortgage accounts and have no of open credit lines and higher dti, more chances for charged off

- Model Performance
    - Accuracy of the model 0.81
    - Precision for 'Fully Paid': 0.81, 'Charged Off': 0.53
    - 1.47% False positives
    - Recall for 'Fully Paid': 0.98, 'Charged Off': 0.08
    - 17.97% False negatives
    - F1 score for 'Fully Paid': 0.89, 'Charged Off': 0.14
    - Observations
        - False negatives are more when compared to false positives, which means predicting charged off instead of fully paid is more.
        - Both false positives and false negatives should be minimal for a better model.
        - One of the ways we can achieve it is, by tuning the model with weighted metric using precision and recall
- Questionnaire
    - What percentage of customers have fully paid their Loan Amount?
        - 80.38%
    - Comment about the correlation between Loan Amount and Installment features.
        - Highly correlated (with a correlation of 0.95)

- The majority of people have home ownership as ___.
  - **MORTGAGE**
- People with grades ‘A’ are more likely to fully pay their loan. (T/F)
  - False
- Name the top 2 afforded job titles.
  - Correctional Sgt. and Interim Director of Case Management with an annual avergae income of 8706582 and 7600000
- Thinking from a bank's perspective, which metric should our primary focus be on..
  - Precision (to reduce False Positives)
- How does the gap in precision and recall affect the bank?
  - With False positives bank will affected and with false negatives customers won't get aloan, they will be affected.
- Which were the features that heavily affected the outcome?
  - 'sub_grade', 'annual_inc', 'dti', 'loan_amnt', and 'term'.
- Will the results be affected by geographical location? (Yes/No)
  - Can't say

- Recommendations
  - As there are very few data points on the extremes of annual_inc, dti and int_rate, we may need more data to improve the model performance.
  - More data on public record bankruptcies and derogatory public records can increase the model performance.
  - Increasing the tenure(credit line) by decreasing the EMI, to make it more affordable to the customer which can also profit the bank.