loantap-logistic-regression

February 26, 2025

1 Business Case: LoanTap Logistic Regression

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?

[111]:
```
# Importing the necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

[112]:
```
# converting data into dataframe

loantap  = pd.read_csv("logistic_regression.csv")
loantap.head()
```

[112]:
```
   loan_amnt        term  int_rate  installment grade sub_grade  \
0    10000.0   36 months     11.44       329.48     B        B4
1     8000.0   36 months     11.99       265.68     B        B5
```

```
2   15600.0   36 months     10.49       506.97     B        B3
3    7200.0   36 months      6.49       220.65     A        A2
4   24375.0   60 months     17.27       609.33     C        C5

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

  verification_status   issue_d  loan_status            purpose  \
0        Not Verified  Jan-2015   Fully Paid           vacation
1        Not Verified  Jan-2015   Fully Paid  debt_consolidation
2     Source Verified  Jan-2015   Fully Paid        credit_card
3        Not Verified  Nov-2014   Fully Paid        credit_card
4            Verified  Apr-2013  Charged Off        credit_card

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

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

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

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

[113]:
```python
# making an copy of the dataset

df = loantap.copy()
```

1.0.1 Identification of variables

[114]:
```python
df.shape
```

[114]: (396030, 27)

[115]:
```python
# data info

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object 
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object 
 5   sub_grade             396030 non-null  object 
 6   emp_title             373103 non-null  object 
 7   emp_length            377729 non-null  object 
 8   home_ownership        396030 non-null  object 
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object 
 11  issue_d               396030 non-null  object 
 12  loan_status           396030 non-null  object 
 13  purpose               396030 non-null  object 
 14  title                 394274 non-null  object 
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object 
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object 
 23  application_type      396030 non-null  object 
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object 
dtypes: float64(12), object(15)
```

```
memory usage: 81.6+ MB
```

[116]:
```python
# Checking of null values

df.isna().sum()
```

[116]:
```
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                    1756
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
```

[117]:
```python
# Percentage of null values in each columns

df.isna().sum()/len(df)*100
```

[117]:
```
loan_amnt               0.000000
term                    0.000000
int_rate                0.000000
installment             0.000000
grade                   0.000000
sub_grade               0.000000
emp_title               5.789208
emp_length              4.621115
```

```
home_ownership          0.000000
annual_inc              0.000000
verification_status     0.000000
issue_d                 0.000000
loan_status             0.000000
purpose                 0.000000
title                   0.443401
dti                     0.000000
earliest_cr_line        0.000000
open_acc                0.000000
pub_rec                 0.000000
revol_bal               0.000000
revol_util              0.069692
total_acc               0.000000
initial_list_status     0.000000
application_type        0.000000
mort_acc                9.543469
pub_rec_bankruptcies    0.135091
address                 0.000000
dtype: float64
```

1.0.2 Analysing the basic metrics

[118]:
```python
df.describe(include='all').transpose()
```

[118]:
```
                        count  unique                 top    freq  \
loan_amnt            396030.0     NaN                 NaN     NaN
term                   396030       2           36 months  302005
int_rate             396030.0     NaN                 NaN     NaN
installment          396030.0     NaN                 NaN     NaN
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
annual_inc           396030.0     NaN                 NaN     NaN
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                  394274   48816  Debt consolidation  152472
dti                  396030.0     NaN                 NaN     NaN
earliest_cr_line       396030     684            Oct-2000    3017
open_acc             396030.0     NaN                 NaN     NaN
pub_rec              396030.0     NaN                 NaN     NaN
revol_bal            396030.0     NaN                 NaN     NaN
revol_util           395754.0     NaN                 NaN     NaN
```

```
total_acc             396030.0     NaN                           NaN     NaN
initial_list_status     396030       2                             f  238066
application_type        396030       3                    INDIVIDUAL  395319
mort_acc              358235.0     NaN                           NaN     NaN
pub_rec_bankruptcies  395495.0     NaN                           NaN     NaN
address                 396030  393700  USCGC Smith\r\nFPO AE 70466       8

                              mean           std    min      25%      50%  \
loan_amnt             14113.888089   8357.441341  500.0   8000.0  12000.0
term                           NaN           NaN    NaN      NaN      NaN
int_rate                   13.6394      4.472157   5.32    10.49    13.33
installment             431.849698     250.72779  16.08   250.33   375.43
grade                          NaN           NaN    NaN      NaN      NaN
sub_grade                      NaN           NaN    NaN      NaN      NaN
emp_title                      NaN           NaN    NaN      NaN      NaN
emp_length                     NaN           NaN    NaN      NaN      NaN
home_ownership                 NaN           NaN    NaN      NaN      NaN
annual_inc            74203.175798  61637.621158    0.0  45000.0  64000.0
verification_status            NaN           NaN    NaN      NaN      NaN
issue_d                        NaN           NaN    NaN      NaN      NaN
loan_status                    NaN           NaN    NaN      NaN      NaN
purpose                        NaN           NaN    NaN      NaN      NaN
title                          NaN           NaN    NaN      NaN      NaN
dti                      17.379514     18.019092    0.0    11.28    16.91
earliest_cr_line               NaN           NaN    NaN      NaN      NaN
open_acc                 11.311153      5.137649    0.0      8.0     10.0
pub_rec                   0.178191      0.530671    0.0      0.0      0.0
revol_bal             15844.539853  20591.836109    0.0   6025.0  11181.0
revol_util               53.791749     24.452193    0.0     35.8     54.8
total_acc                25.414744     11.886991    2.0     17.0     24.0
initial_list_status            NaN           NaN    NaN      NaN      NaN
application_type               NaN           NaN    NaN      NaN      NaN
mort_acc                  1.813991       2.14793    0.0      0.0      1.0
pub_rec_bankruptcies      0.121648      0.356174    0.0      0.0      0.0
address                        NaN           NaN    NaN      NaN      NaN

                          75%        max
loan_amnt             20000.0    40000.0
term                      NaN        NaN
int_rate                16.49      30.99
installment             567.3    1533.81
grade                     NaN        NaN
sub_grade                 NaN        NaN
emp_title                 NaN        NaN
emp_length                NaN        NaN
home_ownership            NaN        NaN
annual_inc            90000.0  8706582.0
```

```
verification_status        NaN        NaN
issue_d                    NaN        NaN
loan_status                NaN        NaN
purpose                    NaN        NaN
title                      NaN        NaN
dti                      22.98     9999.0
earliest_cr_line           NaN        NaN
open_acc                  14.0       90.0
pub_rec                    0.0       86.0
revol_bal              19620.0  1743266.0
revol_util                72.9      892.3
total_acc                 32.0      151.0
initial_list_status        NaN        NaN
application_type           NaN        NaN
mort_acc                   3.0       34.0
pub_rec_bankruptcies       0.0        8.0
address                    NaN        NaN
```

Insights **Outliers:** The significant differences between mean & median in key attributes like loan amount and revolving balance indicate potential outliers.

Loan Duration Preference: A preference for 36-month loan terms among borrowers suggests a balance between manageable installments.

Home Ownership Trends: The prevalence of applicants with mortgaged homes suggests financial stability or a need for substantial, property-secured loans.

Successful Loan Repayment: Most loans being fully paid off reflects positively on borrowers' financial commitment, indicating effective lending criteria.

Debt Consolidation Dominance: The primary use of loans for debt consolidation highlights a common strategy to manage or reduce high-interest debt.

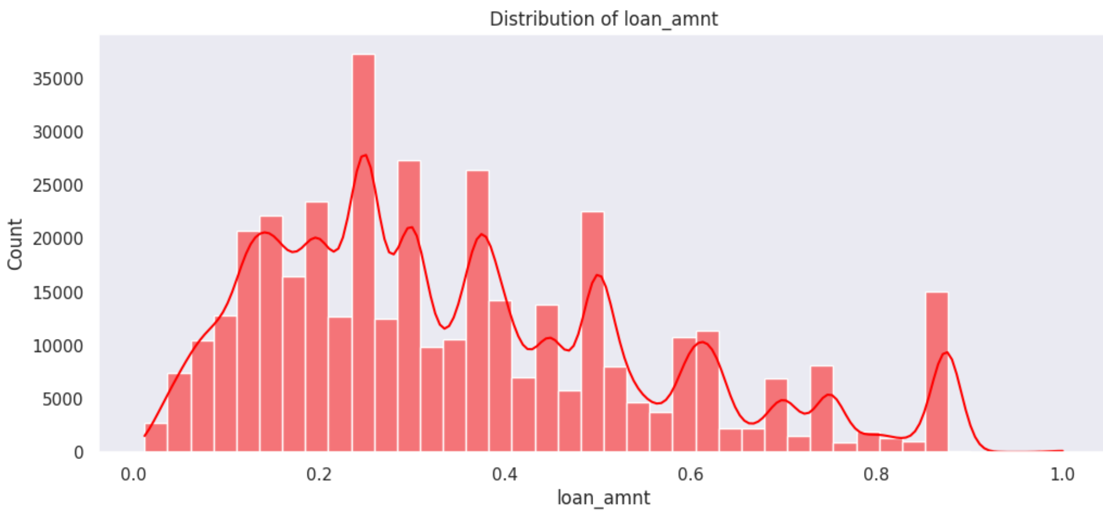
Individual Borrowers: The predominance of individual applicants suggests that personal loans are a major market segment.

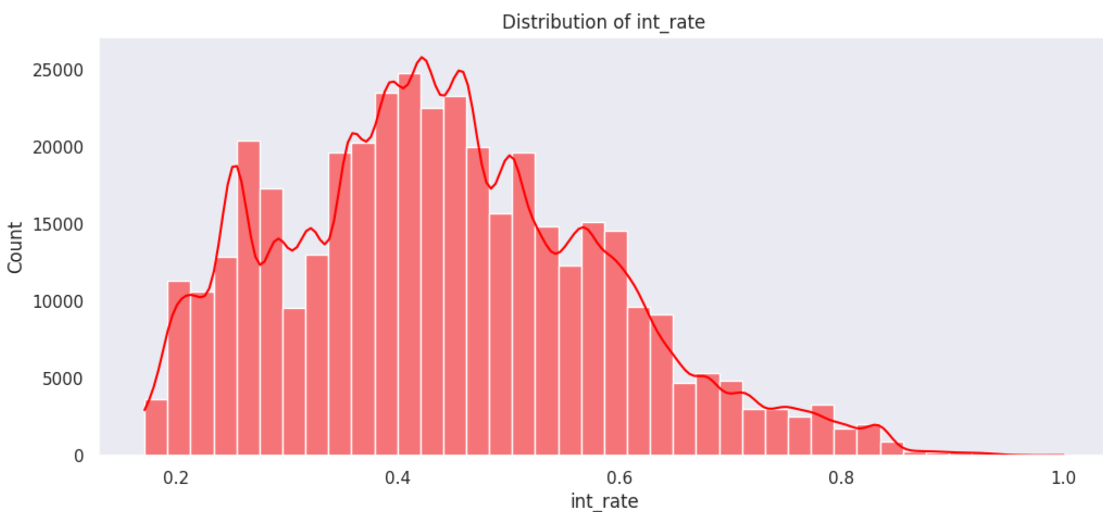
[119]:
```
n_columns = loantap.select_dtypes('float64').columns.tolist()\

n_columns
```

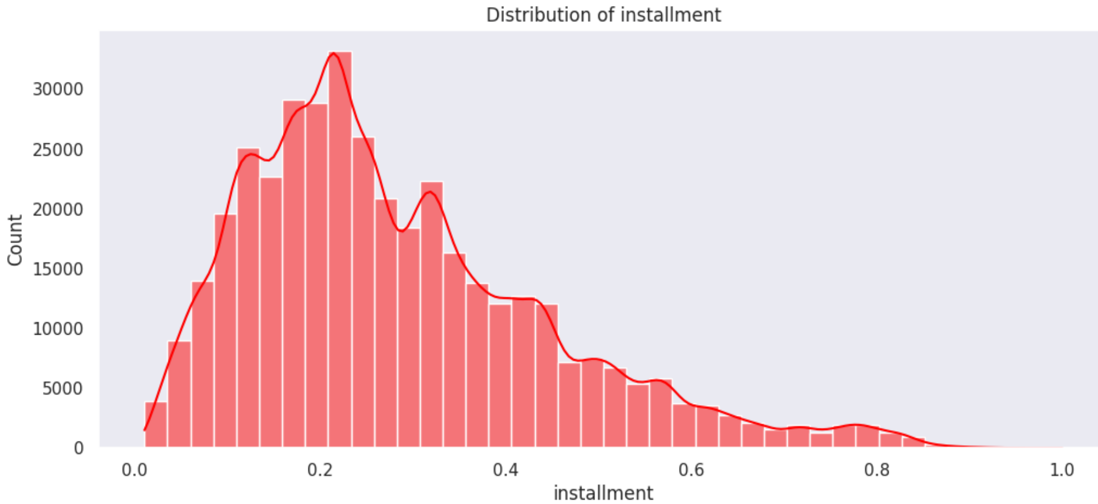
[119]:
```
['loan_amnt',
 'int_rate',
 'installment',
 'annual_inc',
 'dti',
 'open_acc',
 'pub_rec',
 'revol_bal',
 'revol_util',
```

```
'total_acc',
'mort_acc',
'pub_rec_bankruptcies']
```

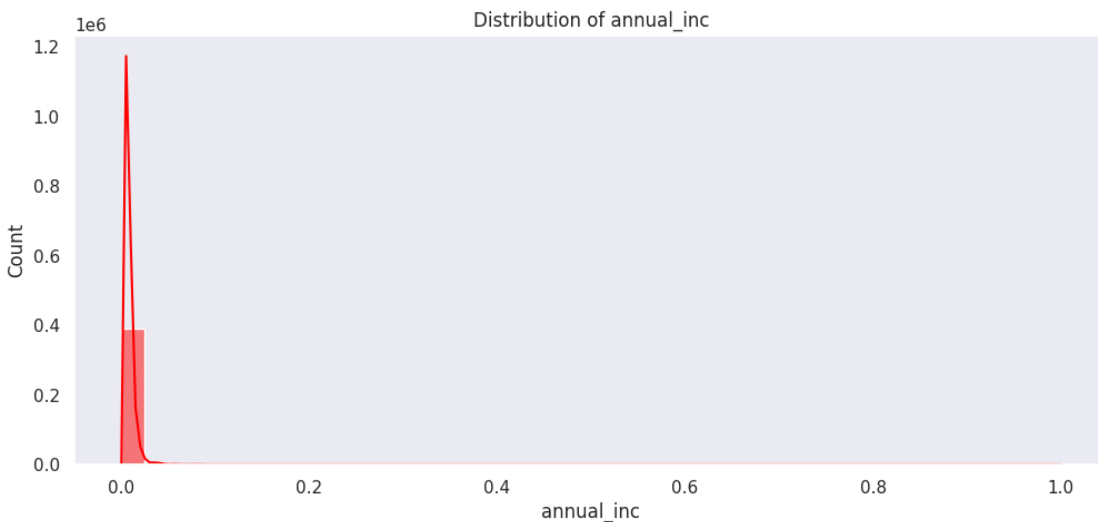
[120]:
```
for i in n_columns:
    plt.figure(figsize=(12,5))
    plt.title("Distribution of {}".format(i))
    sns.histplot(df[i]/df[i].max(), kde=True,color="red", bins=40)
    plt.show()
```



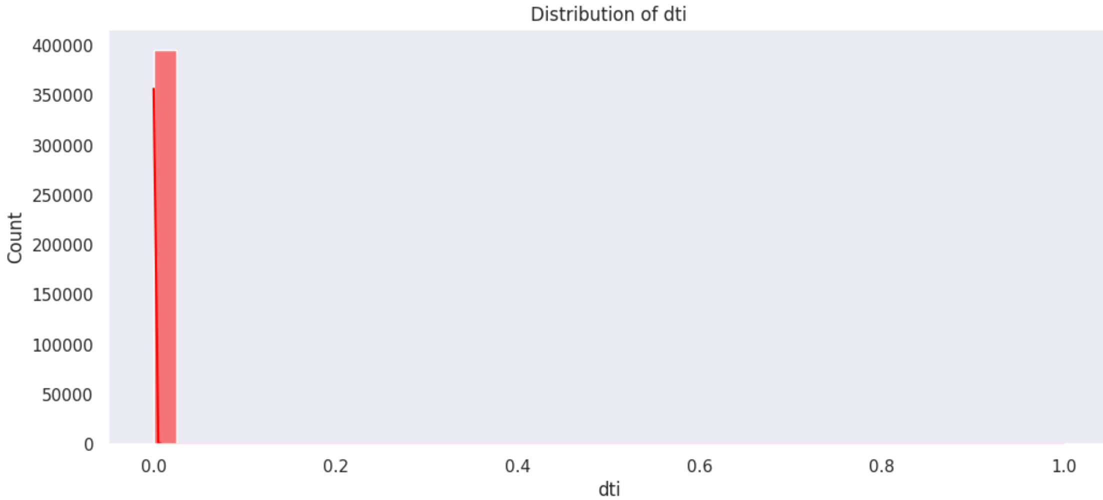




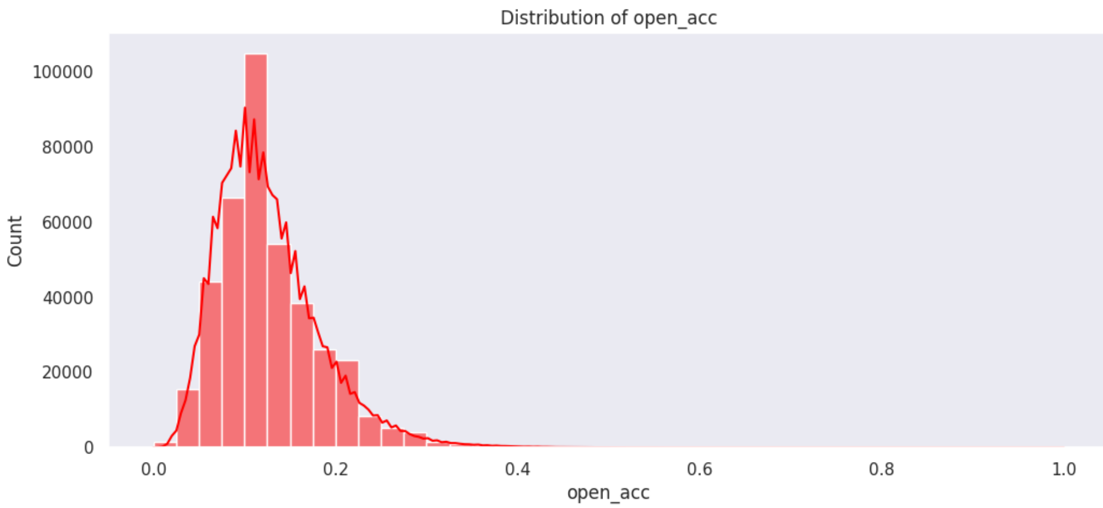
Distribution of installment
Count
installment
30000
25000
20000
15000
10000
5000
0
0.0
0.2
0.4
0.6
0.8
1.0



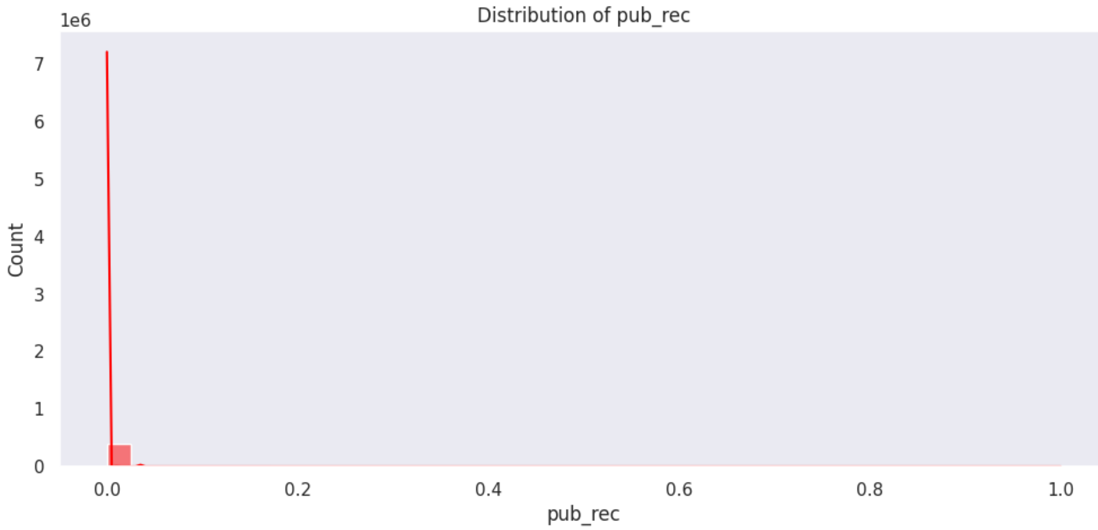
Distribution of annual_inc
1e6
Count
annual_inc
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

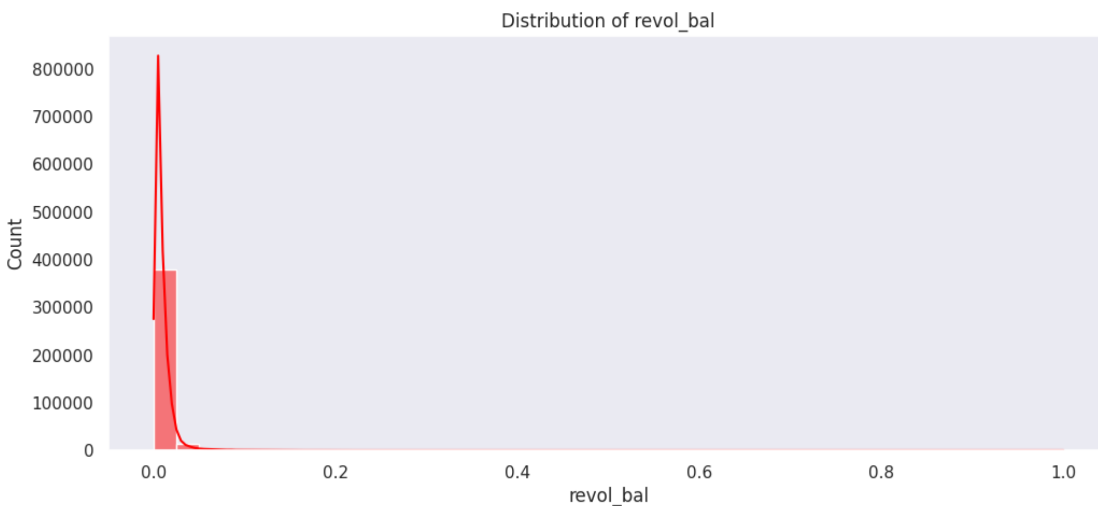


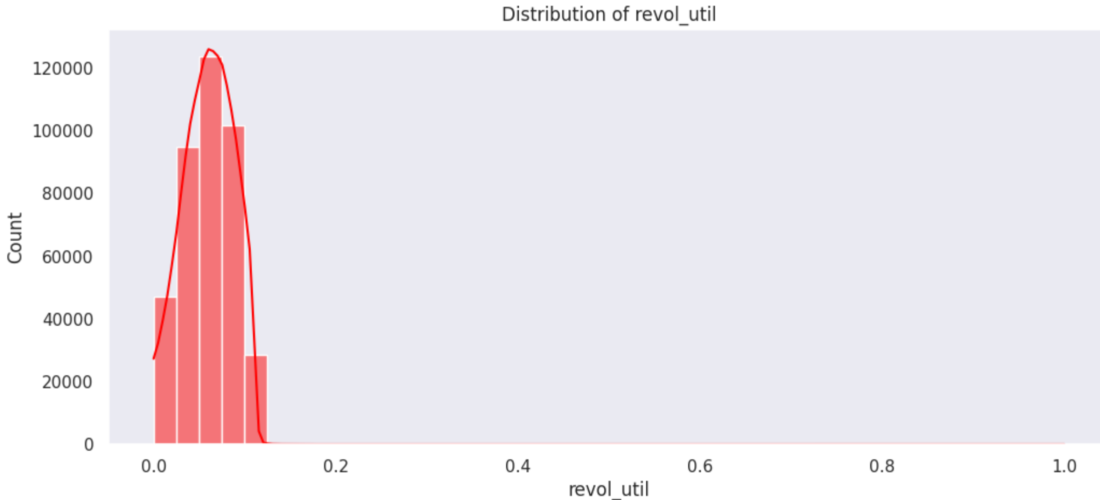
Distribution of dti
400000
350000
300000
250000
200000
150000
100000
50000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
dti



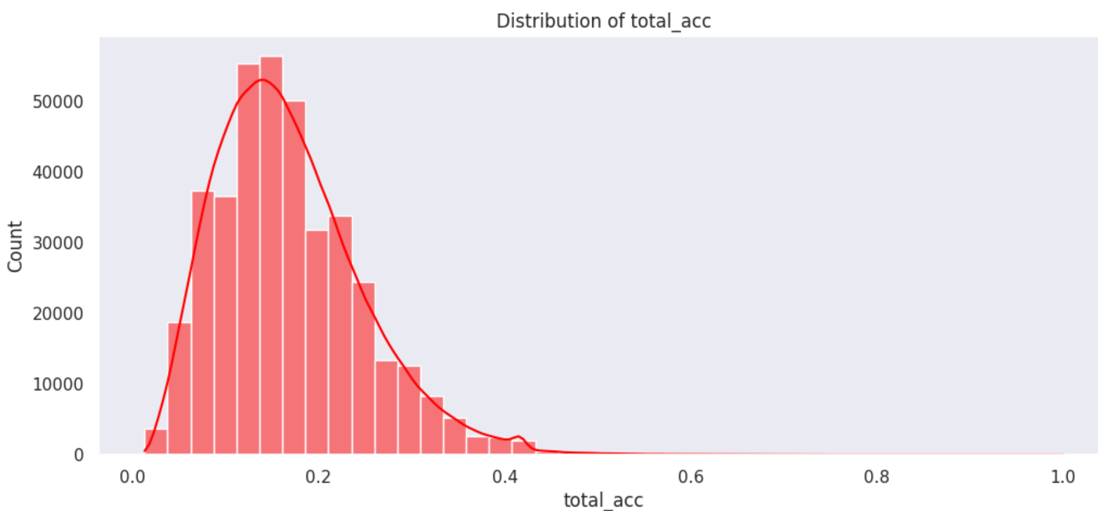
Distribution of open_acc
100000
80000
60000
40000
20000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
open_acc


Distribution of pub_rec
1e6
7
6
5
4
3
2
1
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
pub_rec

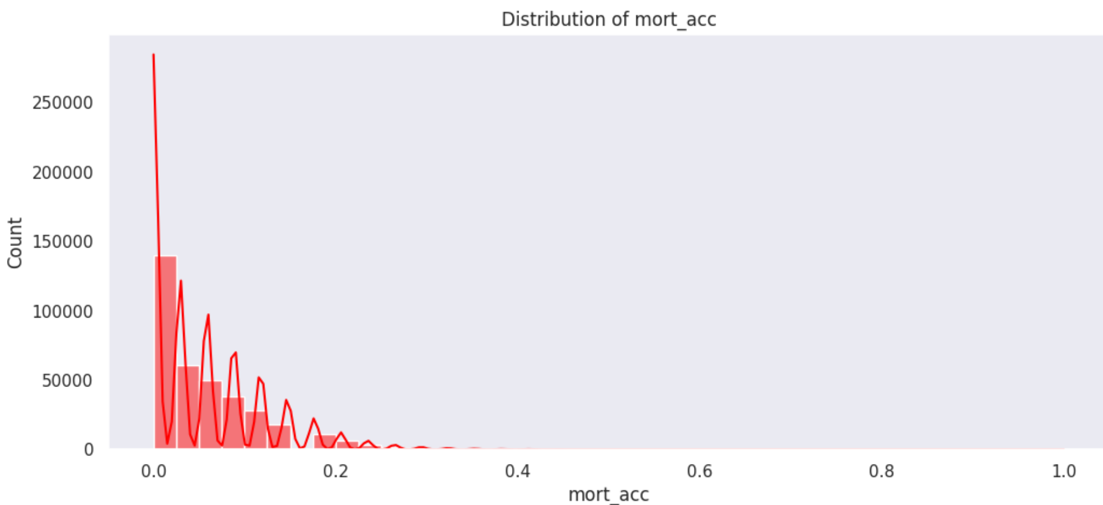

Distribution of revol_bal
800000
700000
600000
500000
400000
300000
200000
100000
0
Count
0.0
0.2
0.4
0.6
0.8
1.0
revol_bal

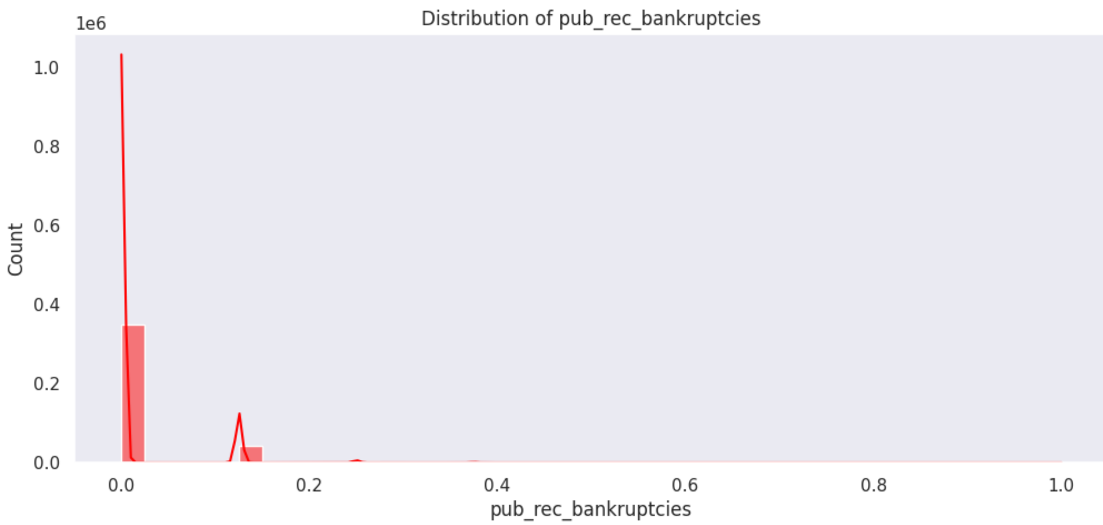


Distribution of revol_util
Count
revol_util
120000
100000
80000
60000
40000
20000
0
0.0
0.2
0.4
0.6
0.8
1.0



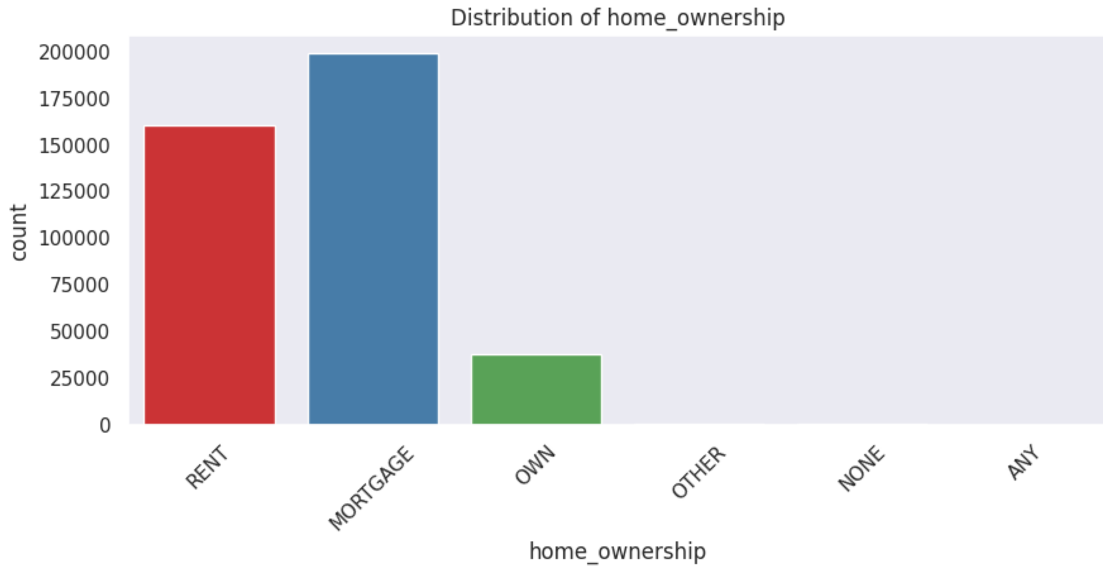
Distribution of total_acc
Count
total_acc
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0

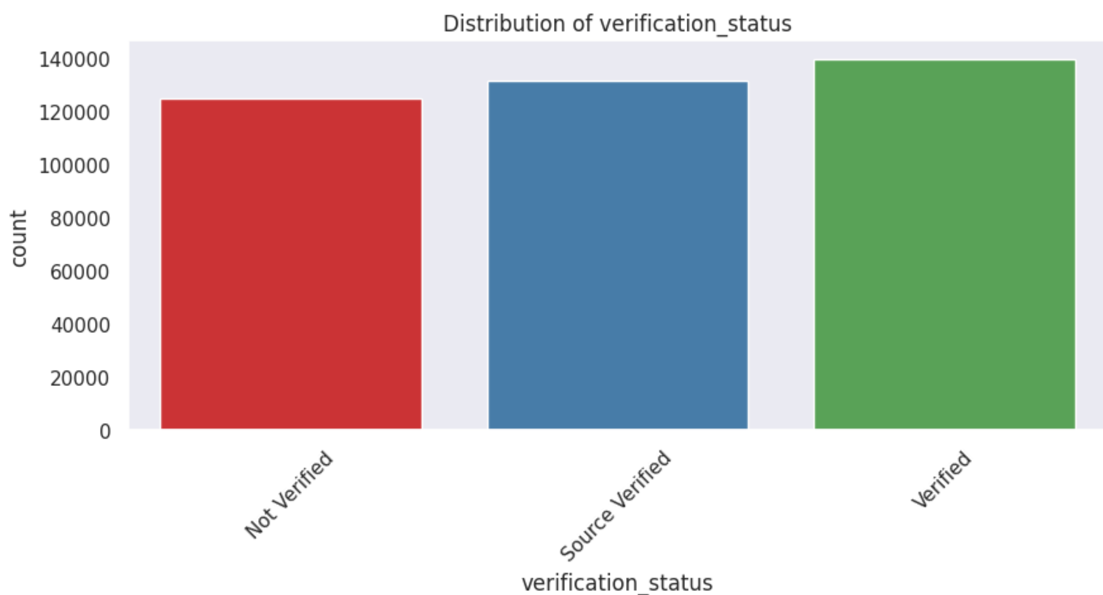


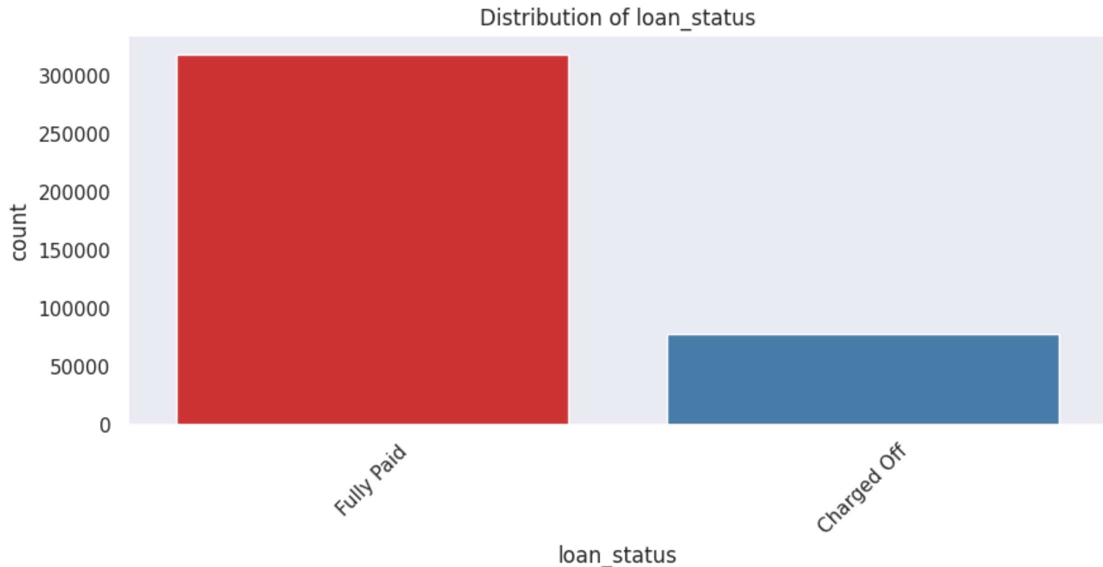


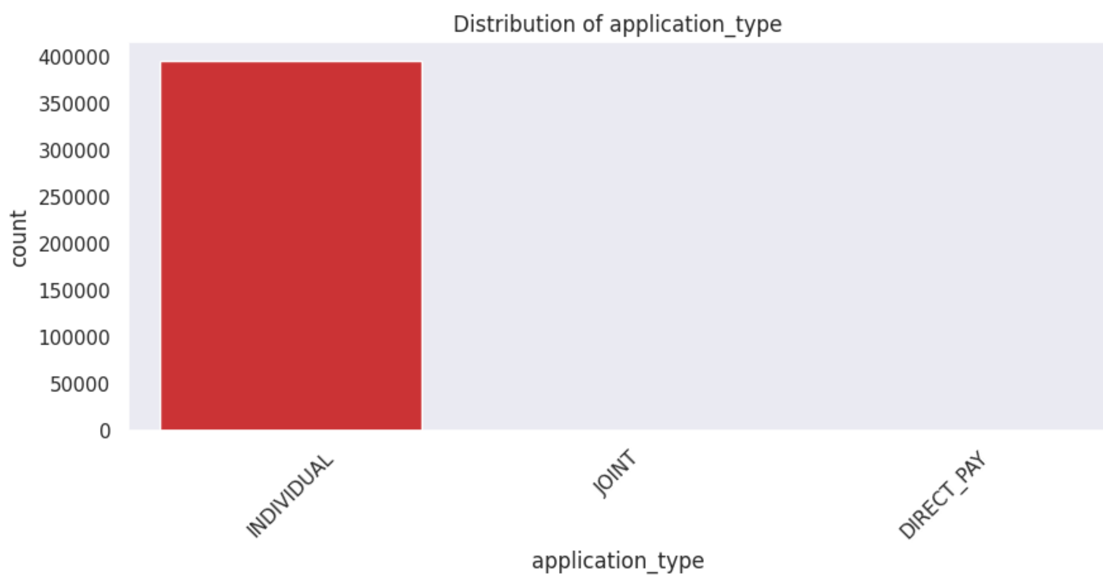
[121]:
```
c_columns = ['home_ownership', 'verification_status', 'loan_status',␣
↪'application_type', 'grade', 'sub_grade', 'term']
```

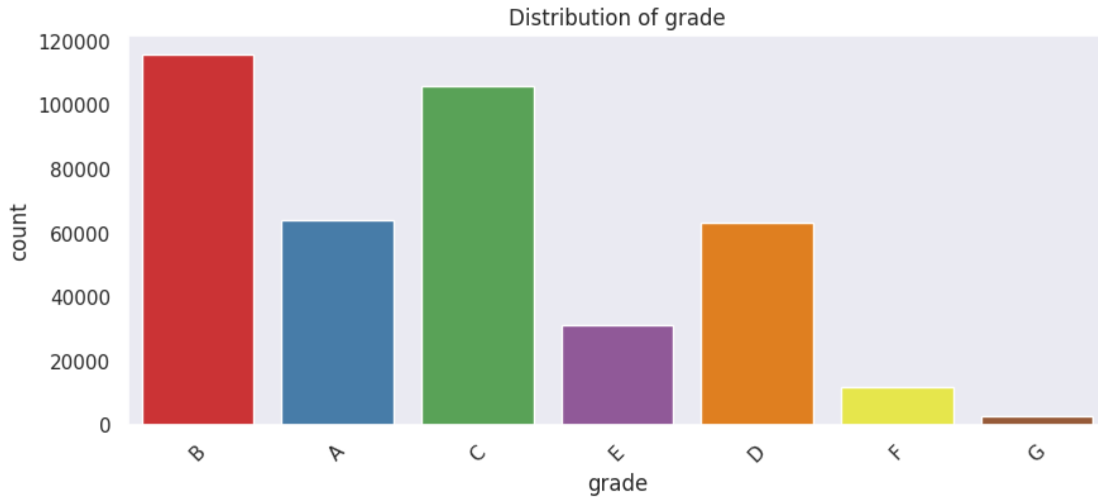
[122]:
```
custom_palette = sns.color_palette("Set1", 8)

for i in c_columns:
    plt.figure(figsize=(10, 4))
    sns.set(style="dark")
    plt.title(f'Distribution of {i}')
    sns.countplot(data=df, x=i, palette=custom_palette)
    plt.xticks(rotation=45)
    plt.show()
```

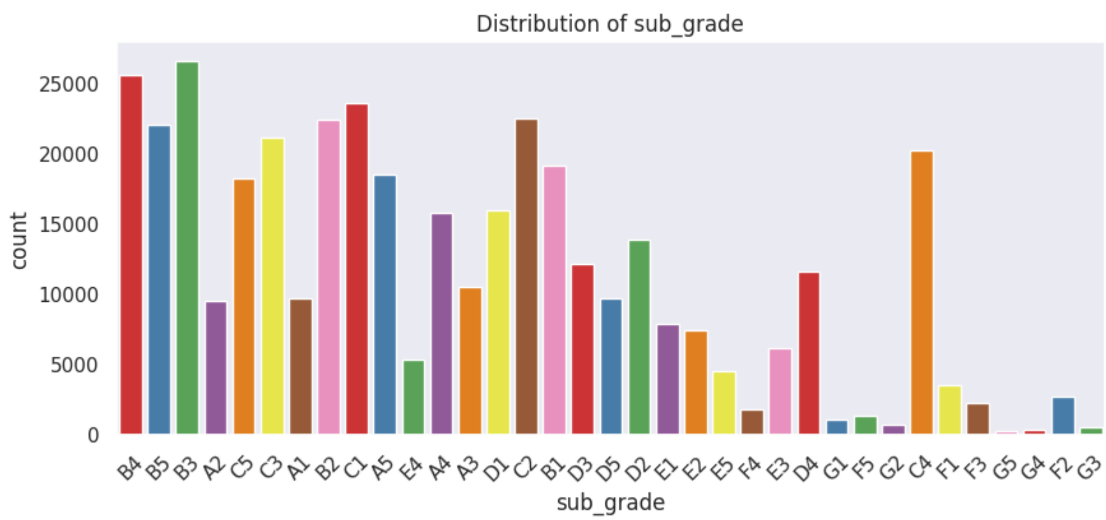

Distribution of home_ownership
200000
175000
150000
125000
100000
75000
50000
25000
0
count
RENT
MORTGAGE
OWN
OTHER
NONE
ANY
home_ownership

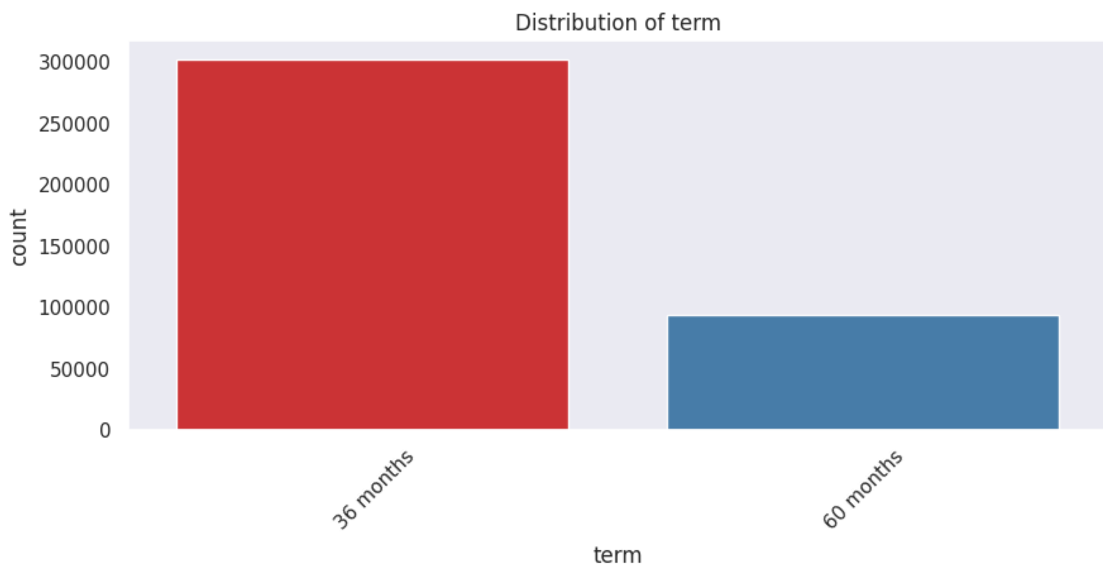

Distribution of verification_status
140000
120000
100000
80000
60000
40000
20000
0
count
Not Verified
Source Verified
Verified
verification_status


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


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

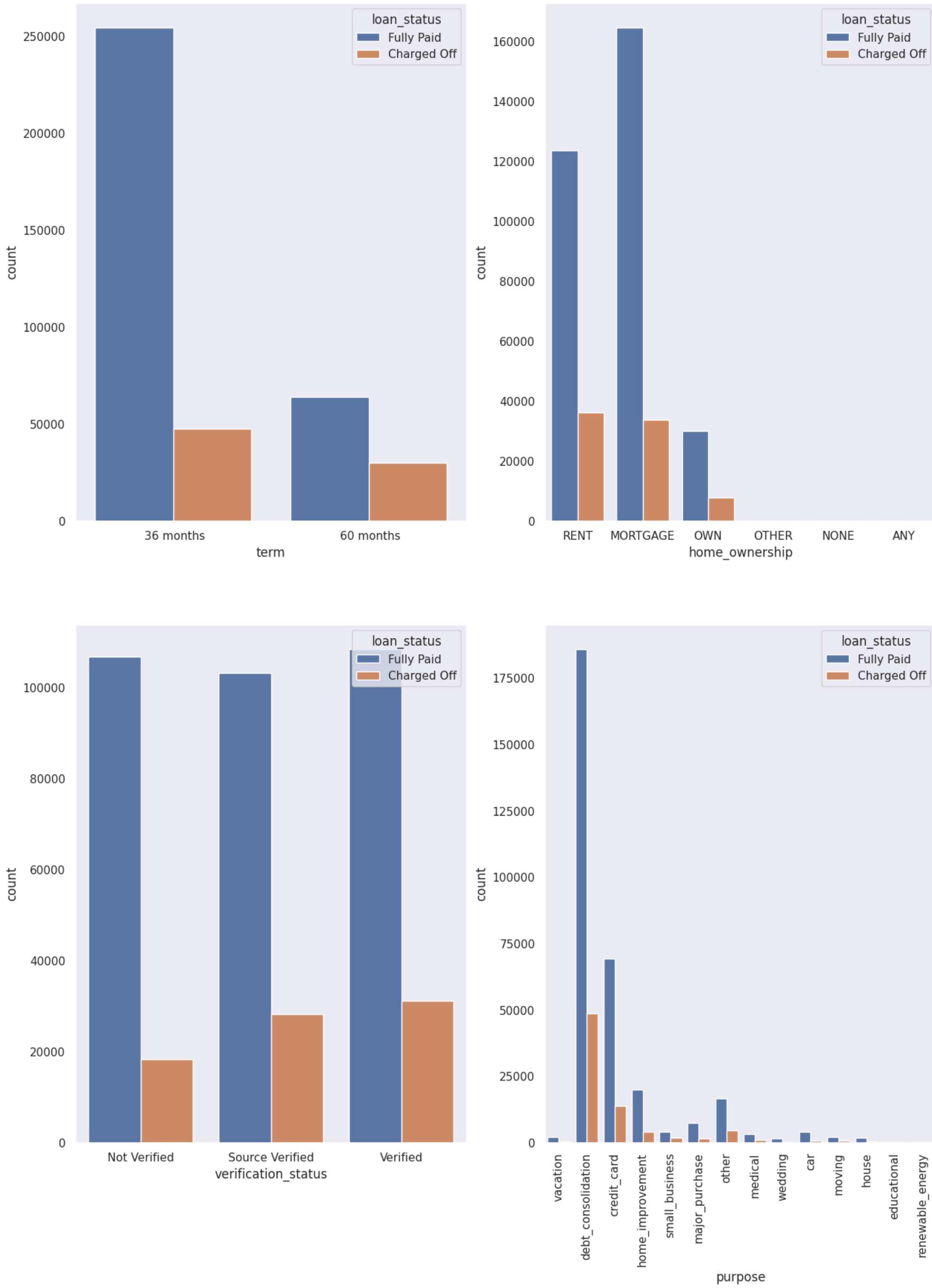

Distribution of grade
120000
100000
80000
60000
40000
20000
0
count
B
A
C
E
D
F
G
grade


Distribution of sub_grade
25000
20000
15000
10000
5000
0
count
B4 B5 B3 A2 C5 C3 A1 B2 C1 A5 E4 A4 A3 D1 C2 B1 D3 D5 D2 E1 E2 E5 F4 E3 D4 G1 F5 G2 C4 F1 F3 G5 G4 F2 G3
sub_grade

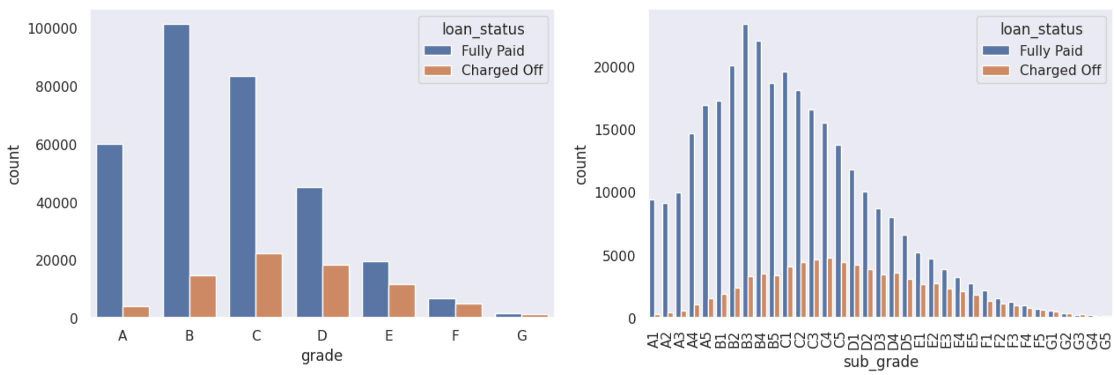


1.0.3 Bivariate Analysis

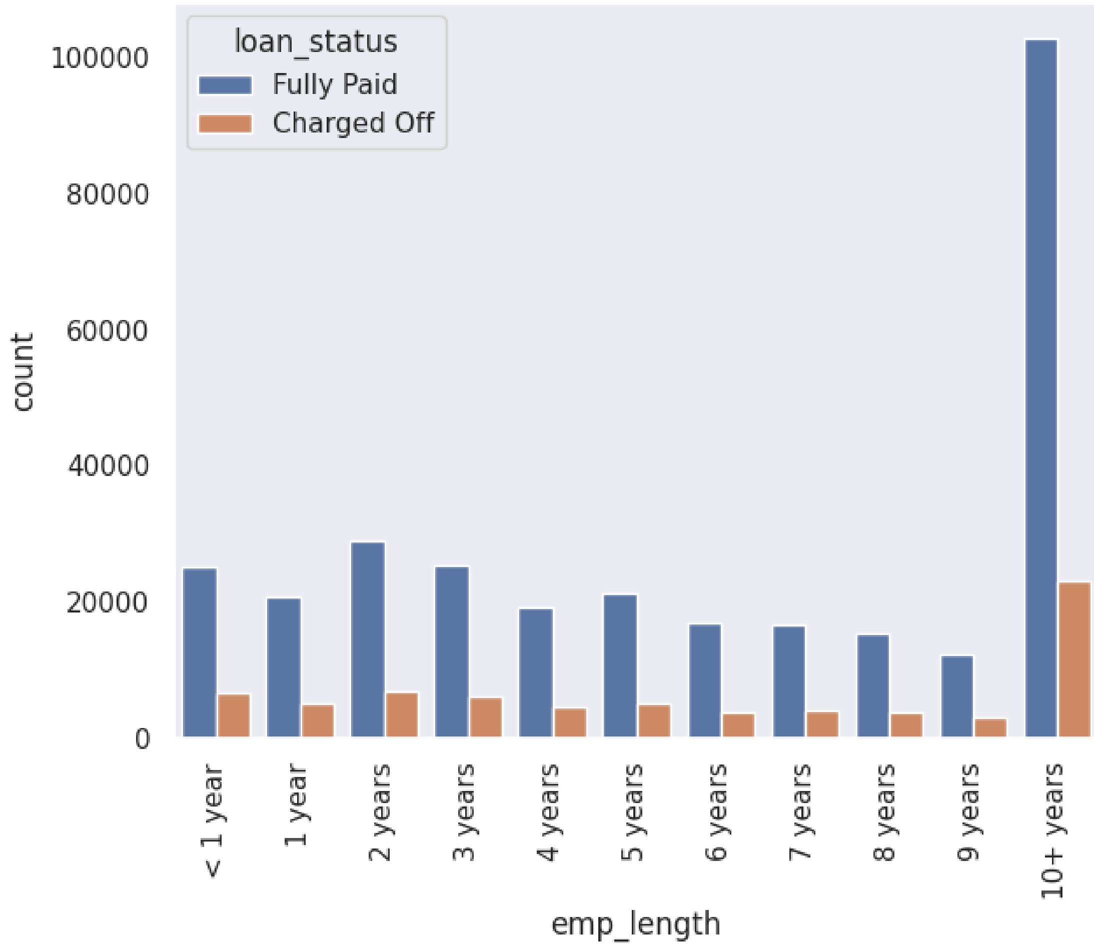
[123]:
```
plt.figure(figsize=(15,20))

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

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

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

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

plt.show()
```


loan_status
Fully Paid
Charged Off
count
term
36 months
60 months
home_ownership
RENT
MORTGAGE
OWN
OTHER
NONE
ANY
verification_status
Not Verified
Source Verified
Verified
purpose
vacation
debt_consolidation
credit_card
home_improvement
small_business
major_purchase
other
medical
wedding
car
moving
house
educational
renewable_energy

[124]:
```
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)
grade = sorted(loantap.grade.unique().tolist())
sns.countplot(x='grade', data=df, 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=df, hue='loan_status', order=sub_grade)
g.set_xticklabels(g.get_xticklabels(), rotation=90)

plt.show()
```



[125]:
```
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=df,hue='loan_status',order=order)
g.set_xticklabels(g.get_xticklabels(),rotation=90)

plt.show()
```



Insights **Loan Terms:** The most popular loan term is 36 months, with a high completion rate.

Loan Types: Mortgages and rental loans are the most common loan types. Debt consolidation loans are also frequently used.

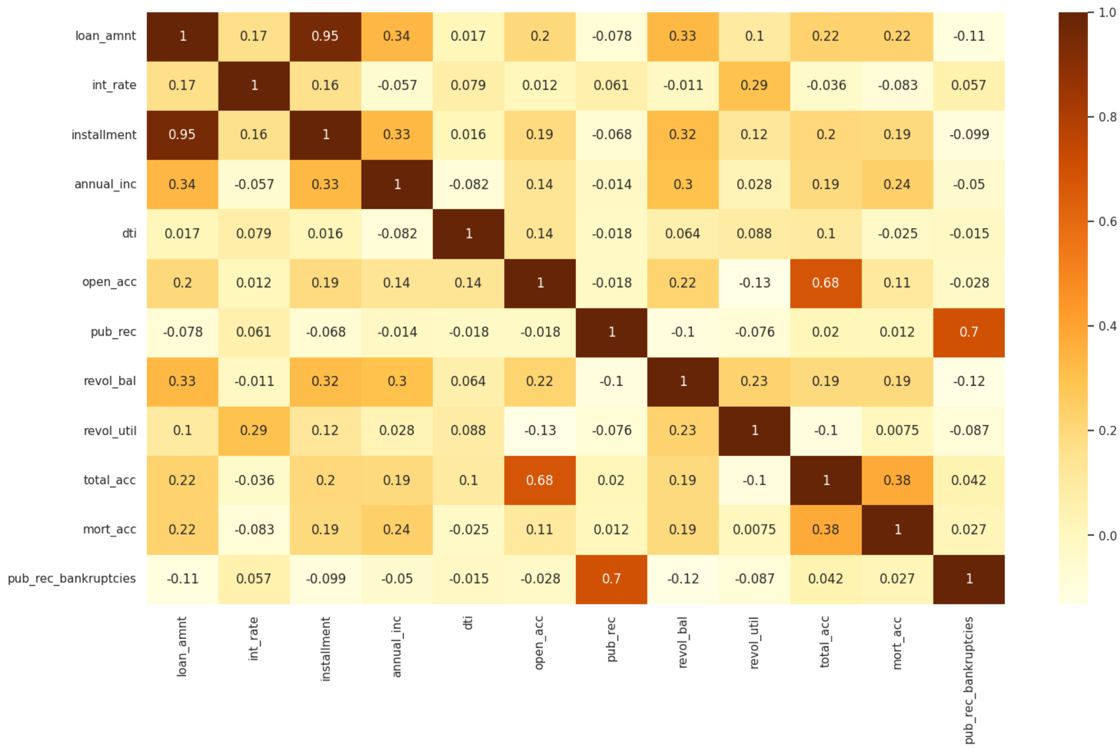
Creditworthiness: Borrowers with a credit grade of "B" and a subgrade of "B3" tend to have the highest repayment rates.

Occupations: Managers and teachers are the professions with the highest loan approval rates.

Repayment: Individuals employed for over 10 years demonstrate a strong track record of loan repayment.

1.0.4 Correlation Analysis

[126]:
```
plt.figure(figsize=(18,10))
sns.heatmap(df.corr(numeric_only=True), cmap = 'YlOrBr', annot = True)
plt.show()
```



[127]:
```python
df.corr(numeric_only=True)
```

[127]:
```
                      loan_amnt  int_rate  installment  annual_inc       dti  \
loan_amnt              1.000000  0.168921     0.953929    0.336887  0.016636
int_rate               0.168921  1.000000     0.162758   -0.056771  0.079038
installment            0.953929  0.162758     1.000000    0.330381  0.015786
annual_inc             0.336887 -0.056771     0.330381    1.000000 -0.081685
dti                    0.016636  0.079038     0.015786   -0.081685  1.000000
open_acc               0.198556  0.011649     0.188973    0.136150  0.136181
pub_rec               -0.077779  0.060986    -0.067892   -0.013720 -0.017639
revol_bal              0.328320 -0.011280     0.316455    0.299773  0.063571
revol_util             0.099911  0.293659     0.123915    0.027871  0.088375
total_acc              0.223886 -0.036404     0.202430    0.193023  0.102128
mort_acc               0.222315 -0.082583     0.193694    0.236320 -0.025439
pub_rec_bankruptcies  -0.106539  0.057450    -0.098628   -0.050162 -0.014558

                      open_acc   pub_rec  revol_bal  revol_util  total_acc  \
loan_amnt             0.198556 -0.077779   0.328320    0.099911   0.223886
int_rate              0.011649  0.060986  -0.011280    0.293659  -0.036404
installment           0.188973 -0.067892   0.316455    0.123915   0.202430
annual_inc            0.136150 -0.013720   0.299773    0.027871   0.193023
dti                   0.136181 -0.017639   0.063571    0.088375   0.102128
open_acc              1.000000 -0.018392   0.221192   -0.131420   0.680728
```

```
pub_rec              -0.018392  1.000000  -0.101664   -0.075910   0.019723
revol_bal             0.221192 -0.101664   1.000000    0.226346   0.191616
revol_util           -0.131420 -0.075910   0.226346    1.000000  -0.104273
total_acc             0.680728  0.019723   0.191616   -0.104273   1.000000
mort_acc              0.109205  0.011552   0.194925    0.007514   0.381072
pub_rec_bankruptcies -0.027732  0.699408  -0.124532   -0.086751   0.042035

                      mort_acc  pub_rec_bankruptcies
loan_amnt             0.222315             -0.106539
int_rate             -0.082583              0.057450
installment           0.193694             -0.098628
annual_inc            0.236320             -0.050162
dti                  -0.025439             -0.014558
open_acc              0.109205             -0.027732
pub_rec               0.011552              0.699408
revol_bal             0.194925             -0.124532
revol_util            0.007514             -0.086751
total_acc             0.381072              0.042035
mort_acc              1.000000              0.027239
pub_rec_bankruptcies  0.027239              1.000000
```

1.0.5 Insights:

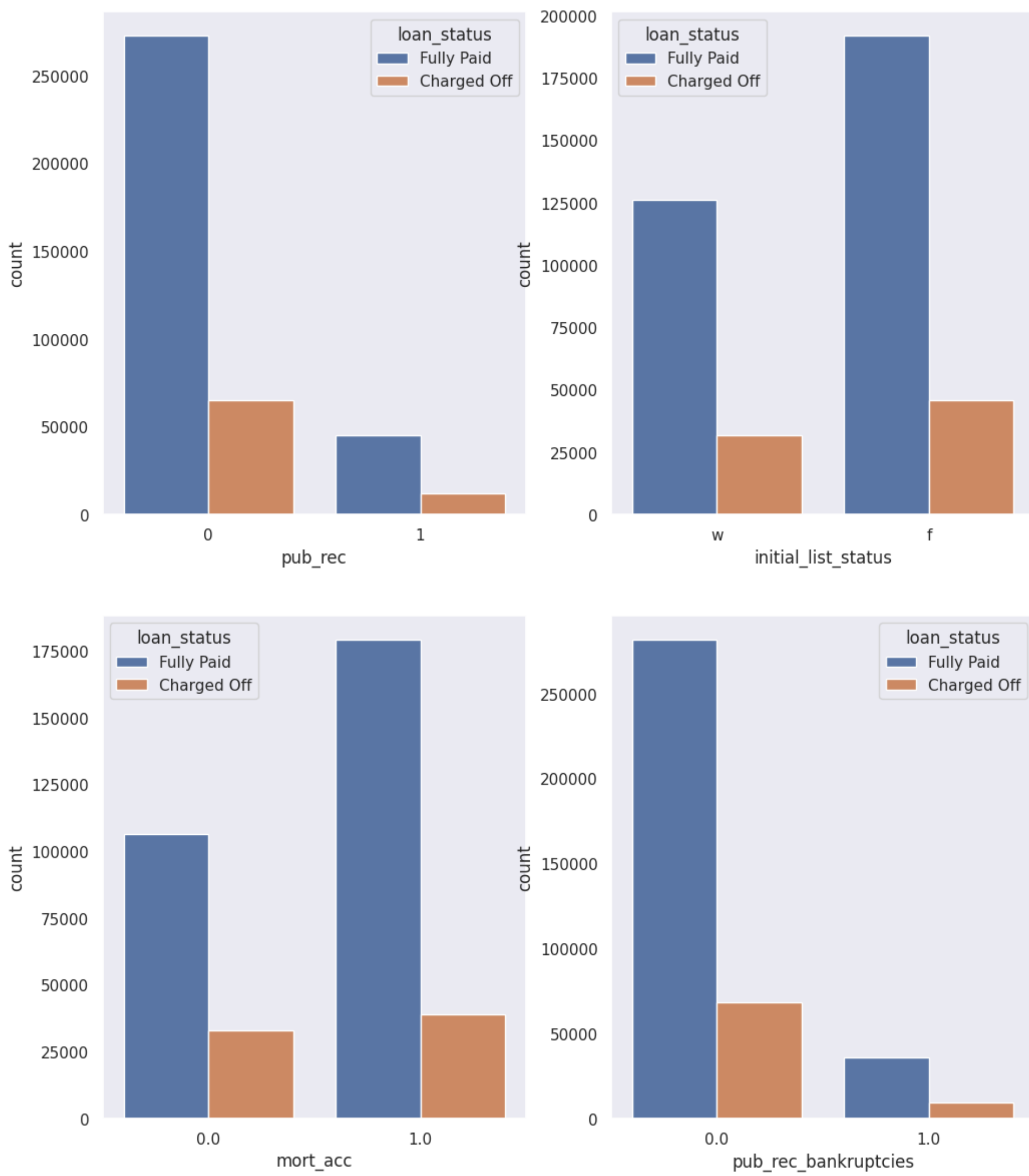
- Positive correlation with annual income (annual_inc) - Higher income allows for larger loan applications.
- Weak positive correlation with installment amount (installment) - Makes sense as larger loans will typically have higher installments.
- Weak positive correlation with total accounts (total_acc) and mortgage accounts (mort_acc) - Borrowers with more established credit lines may be eligible for higher loan amounts.
- Weak negative correlation with annual income (annual_inc) - Generally, borrowers with higher income qualify for lower interest rates.
- Weak positive correlation with total accounts (total_acc) and mortgage accounts (mort_acc) - People with a higher income may tend to have more credit accounts.
- Positive correlation between revolving balance (revol_bal) and credit line utilization (revol_util) - This indicates that people with higher credit balances also tend to have a higher utilization ratio.
- Weak positive correlation between number of open accounts (open_acc) and total accounts (total_acc) - As expected, people with more open accounts tend to have more total accounts.

1.0.6 Data Preprocessing using Feautre Engineering

[128]:
```
def pub_rec(number):
    if number == 0.0:
        return 0
    else:
        return 1

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


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

[129]:
```
df['pub_rec']=df.pub_rec.apply(pub_rec)

df['mort_acc']=df.mort_acc.apply(mort_acc)

df['pub_rec_bankruptcies']=df.pub_rec_bankruptcies.apply(pub_rec_bankruptcies)
```

[130]:
```
plt.figure(figsize=(12,30))

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

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

plt.subplot(4,2,3)
sns.countplot(x='mort_acc',data=df,hue='loan_status')

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

plt.show()
```



Most the loan disbursed to the people who do not hold bankrupties record have successfully paid loan

1.0.7 Duplicate checks

[131]:
```
df.duplicated().sum()
```

[131]: 0

1.0.8 Missing values

[132]:
```
df.isnull().sum()
```

[132]:
```
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                    1756
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
```

[133]:
```
numeric_columns = df.select_dtypes(include=['float64', 'int64'])

total_acc_avg = numeric_columns.groupby('total_acc')['mort_acc'].mean()

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

df['mort_acc'] = df.apply(lambda x: fill_mort_acc(x['total_acc'],
 ↪x['mort_acc']), axis=1)
```

[134]:
```
df.isnull().sum()
```

[134]:
```
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                    1756
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                    0
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

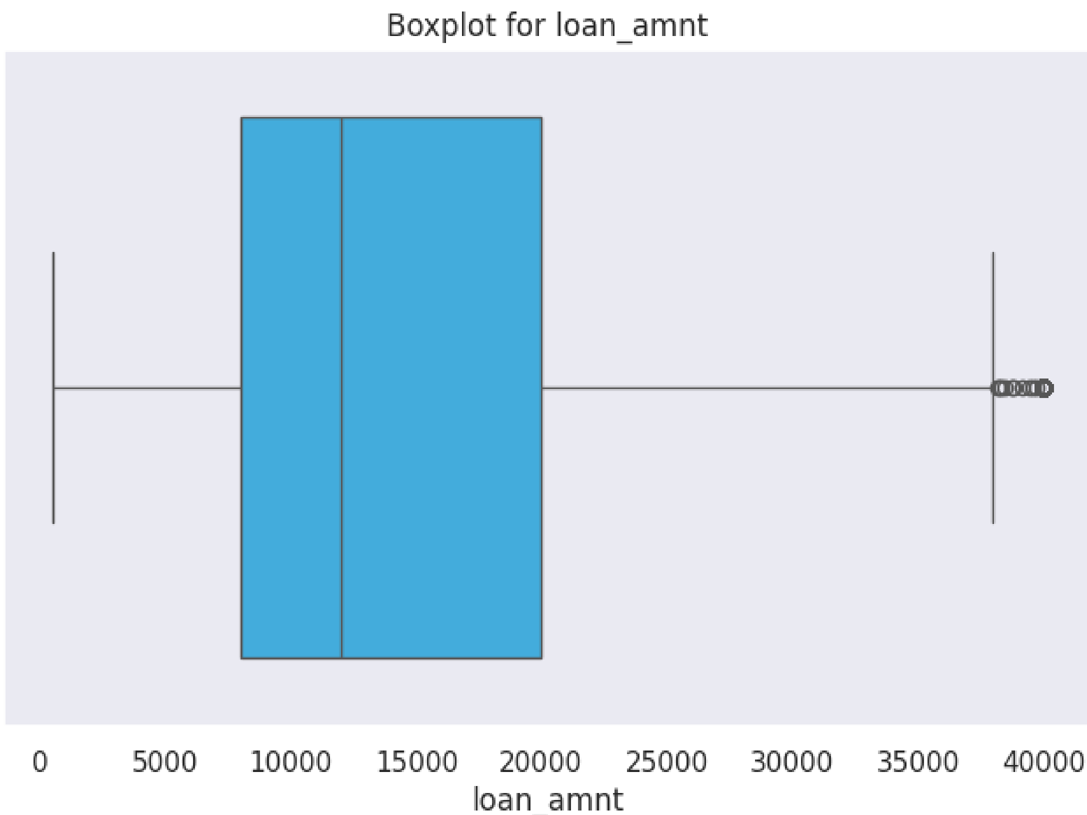
[135]:
```
# droping remaining null values

df.dropna(inplace=True)

df.shape
```

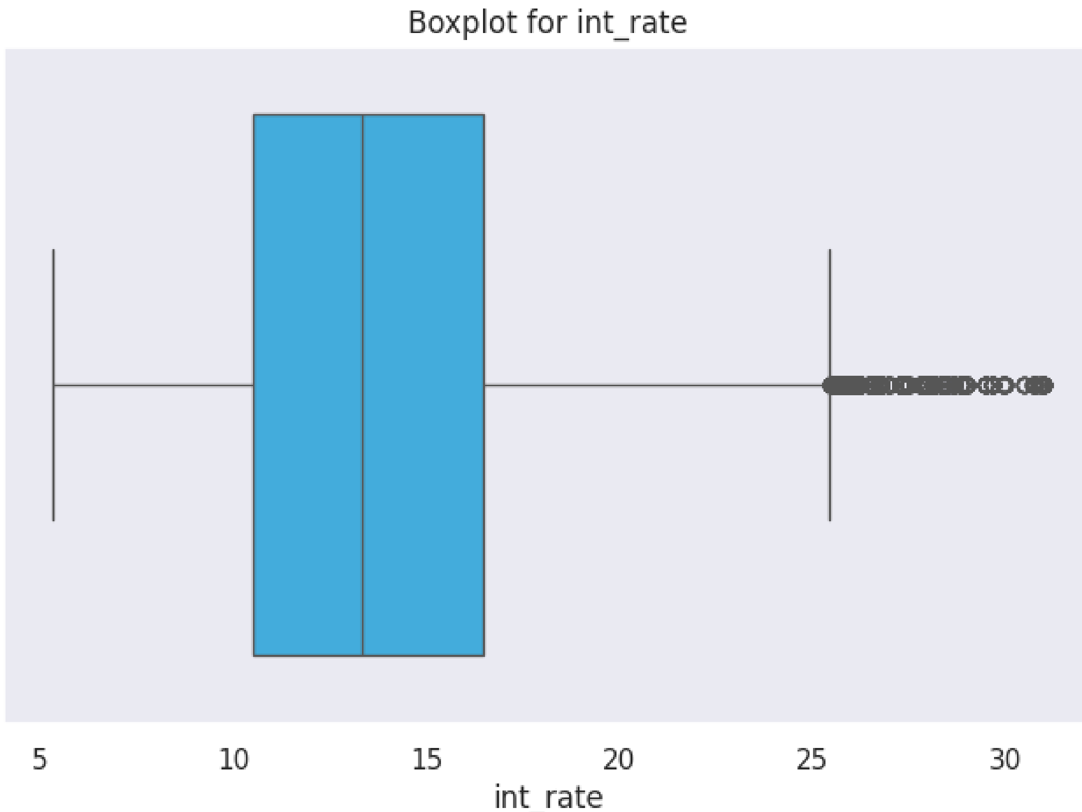
[135]:
```
(370621, 27)
```

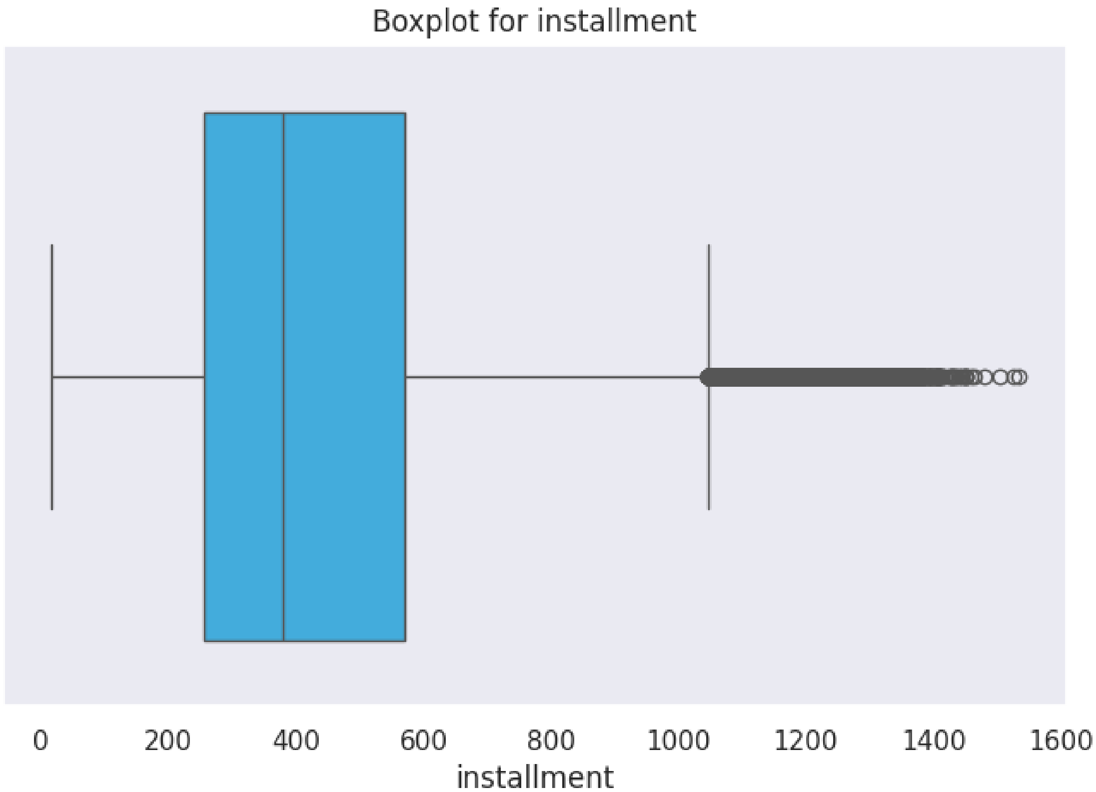
1.0.9 Outlier Detection

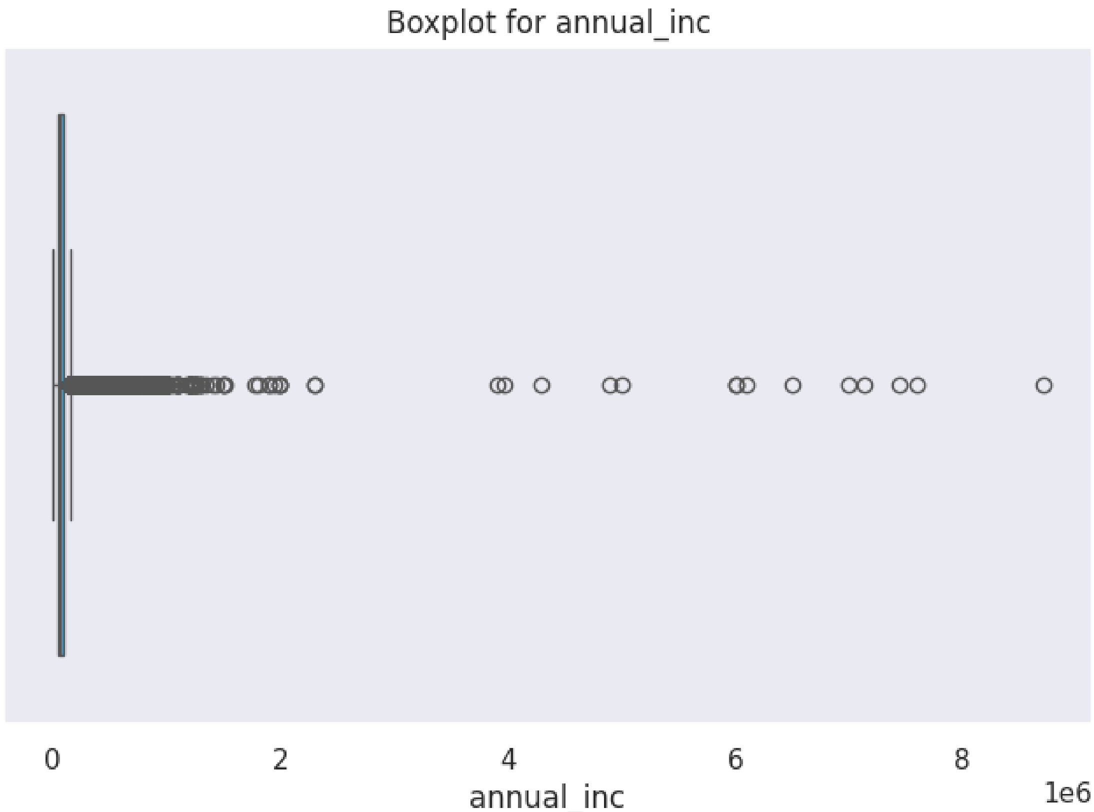
[136]:
```
def box_plot(col):
    if col in df.columns:
        plt.figure(figsize=(8, 5))
        sns.boxplot(x=df[col],color="#29B6F6")
        plt.title('Boxplot for {}'.format(col))
        plt.show()
    else:
        print(f"Column '{col}' not found in the DataFrame.")
```

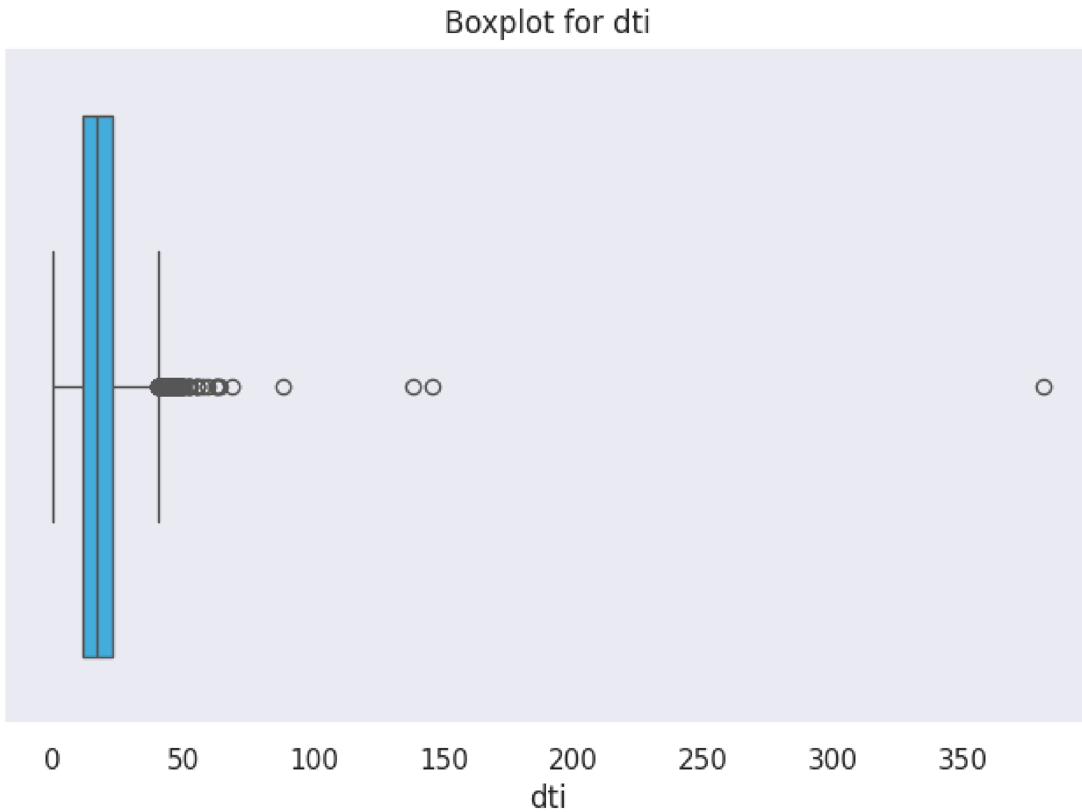
```
for col in n_columns:
    box_plot(col)
```

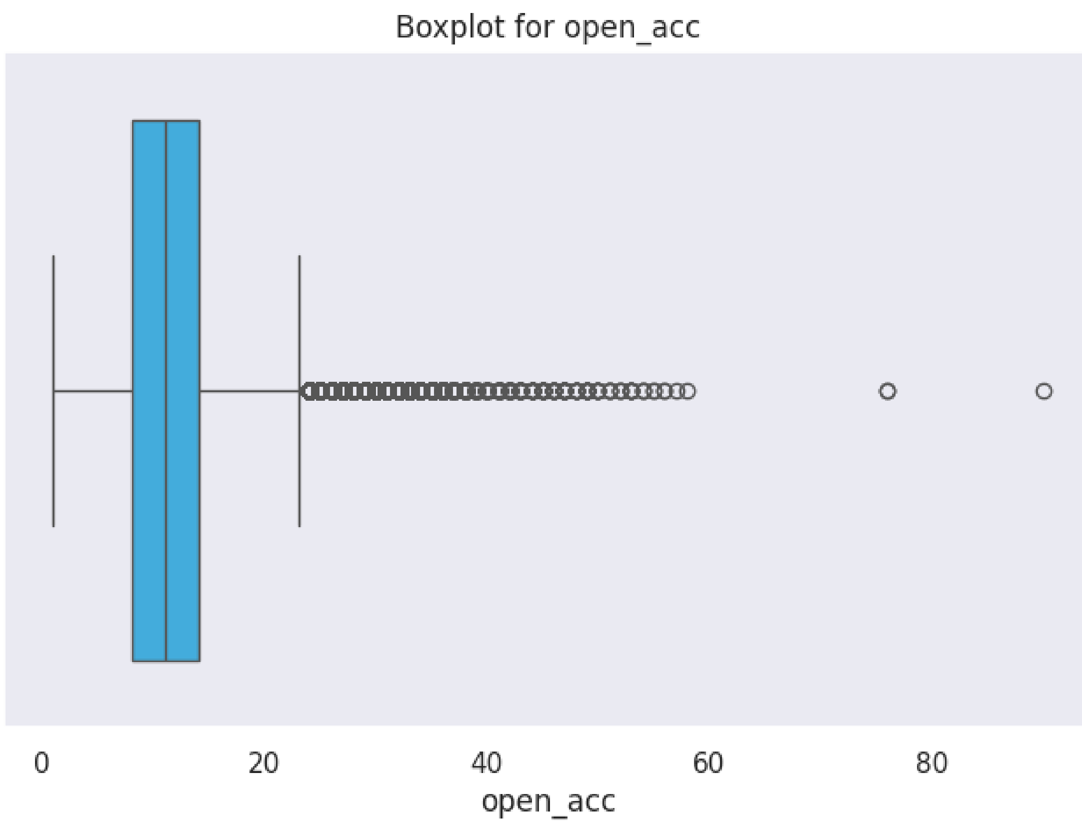


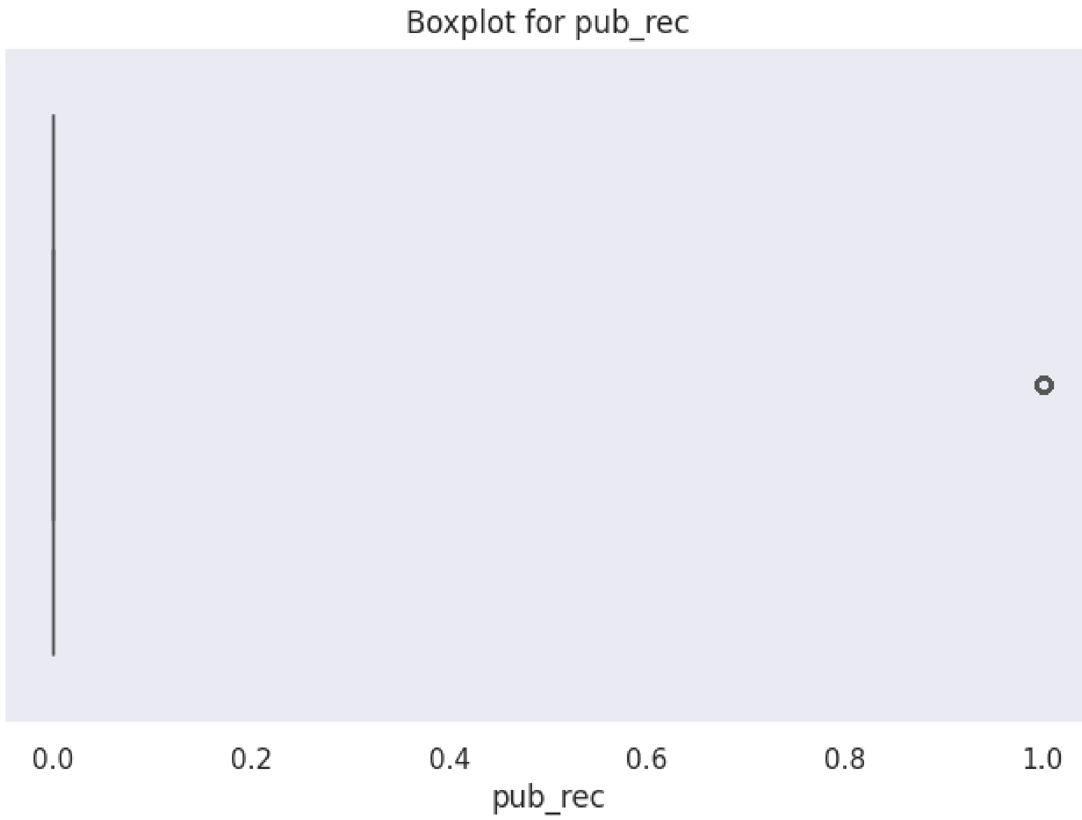

Boxplot for int_rate
5
10
15
20
25
30
int_rate

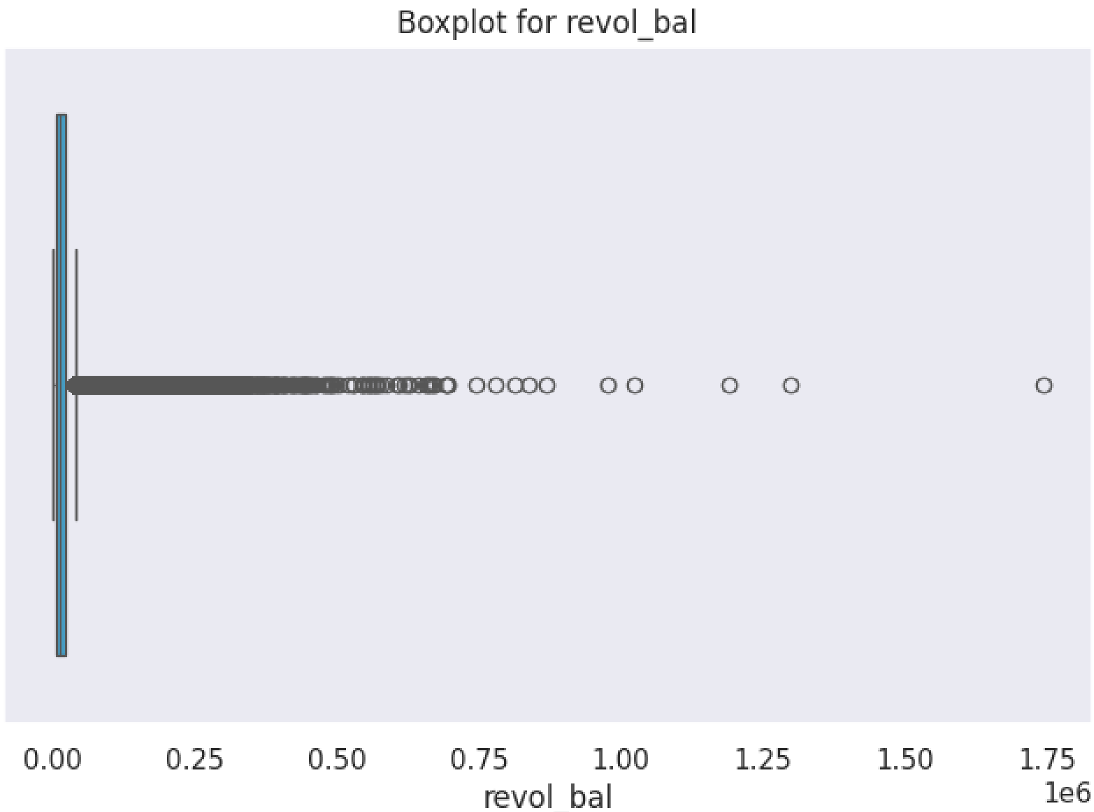

Boxplot for installment
0
200
400
600
800
1000
1200
1400
1600
installment

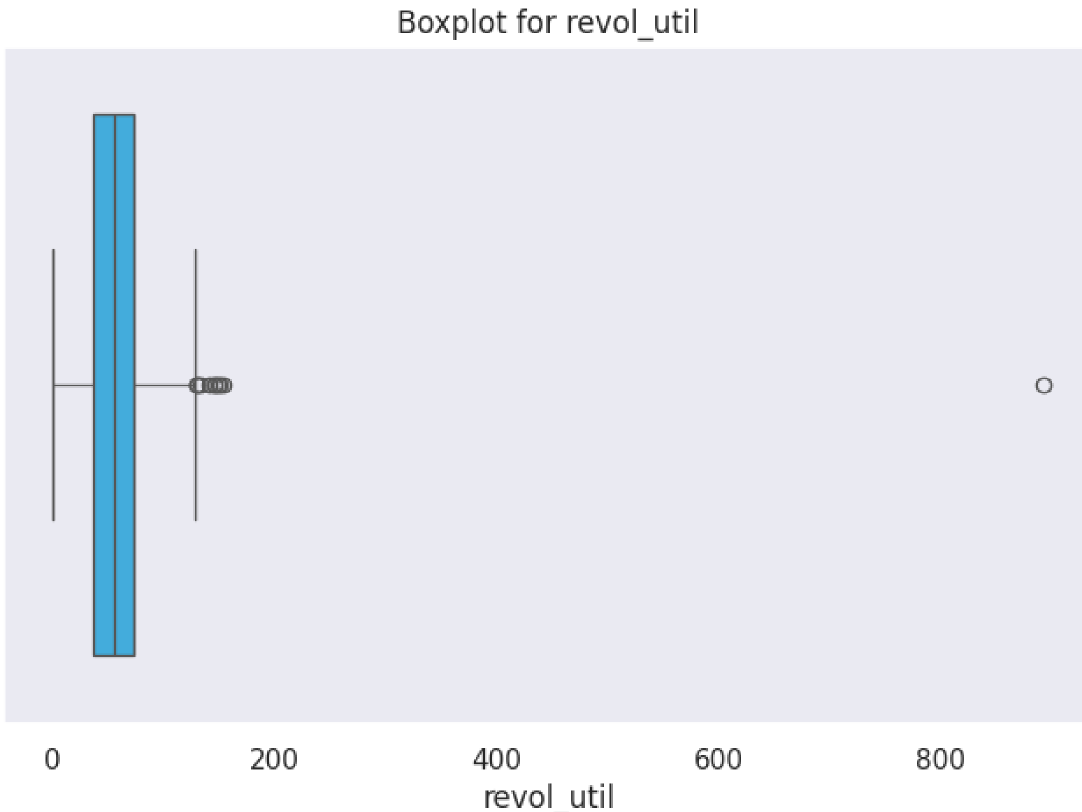

Boxplot for annual_inc
0
2
4
6
8
1e6
annual_inc

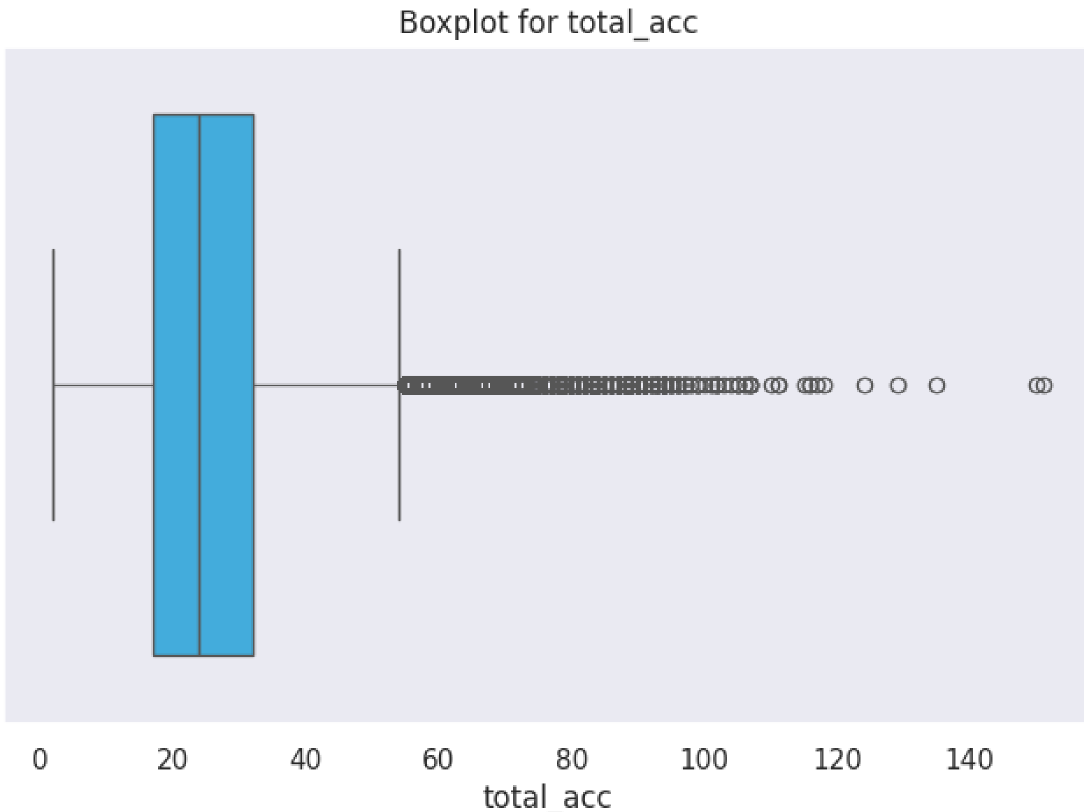

Boxplot for dti
0
50
100
150
200
250
300
350
dti

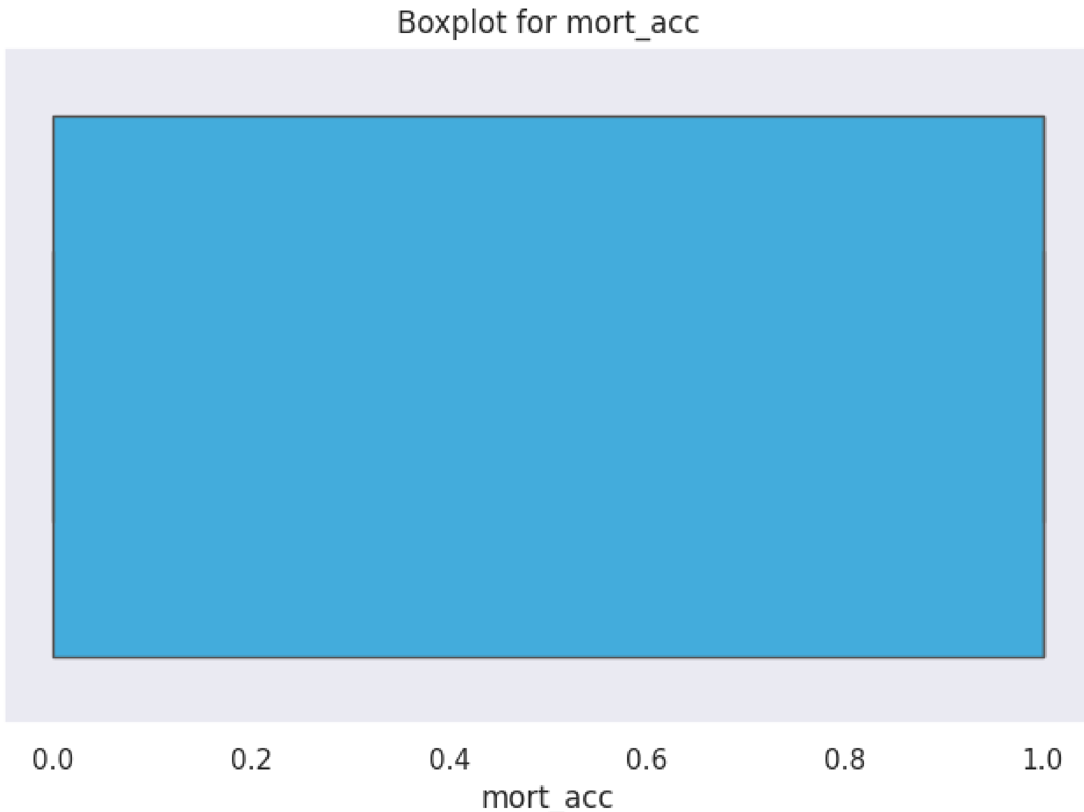

Boxplot for open_acc
0
20
40
60
80
open_acc

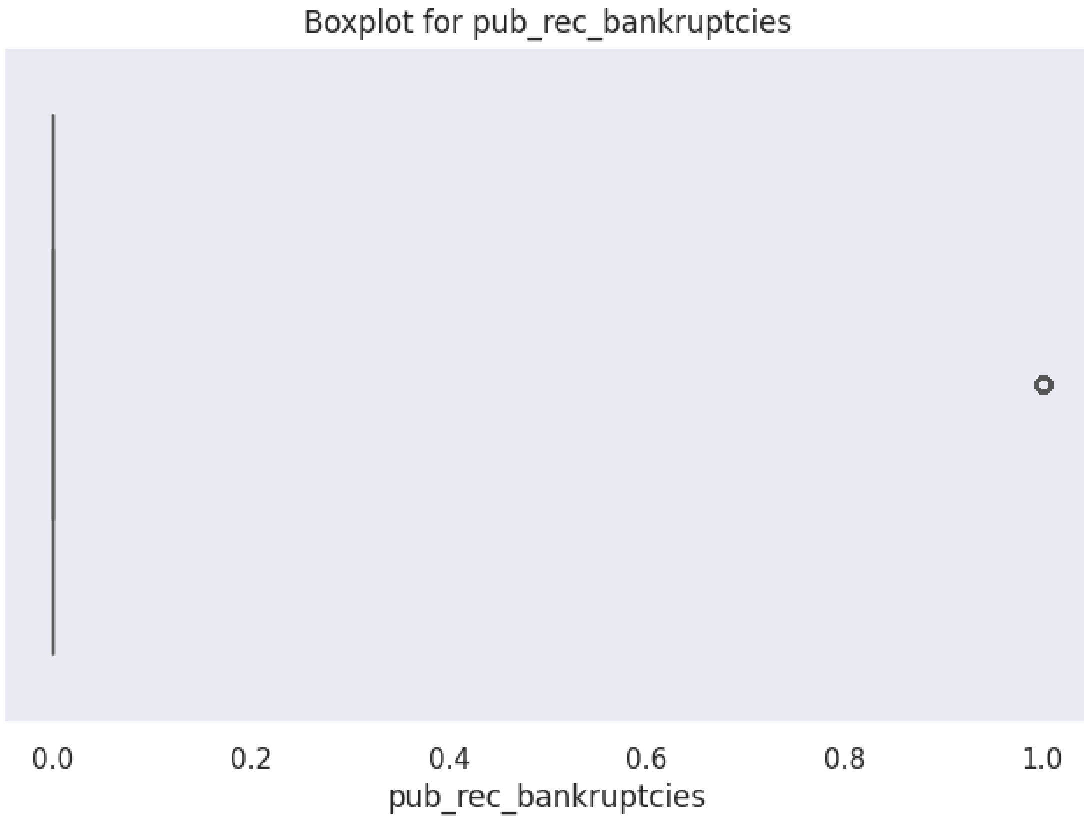

Boxplot for pub_rec
0.0
0.2
0.4
0.6
0.8
1.0
pub_rec


Boxplot for revol_bal
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
1e6
revol_bal


Boxplot for revol_util
0
200
400
600
800
revol_util

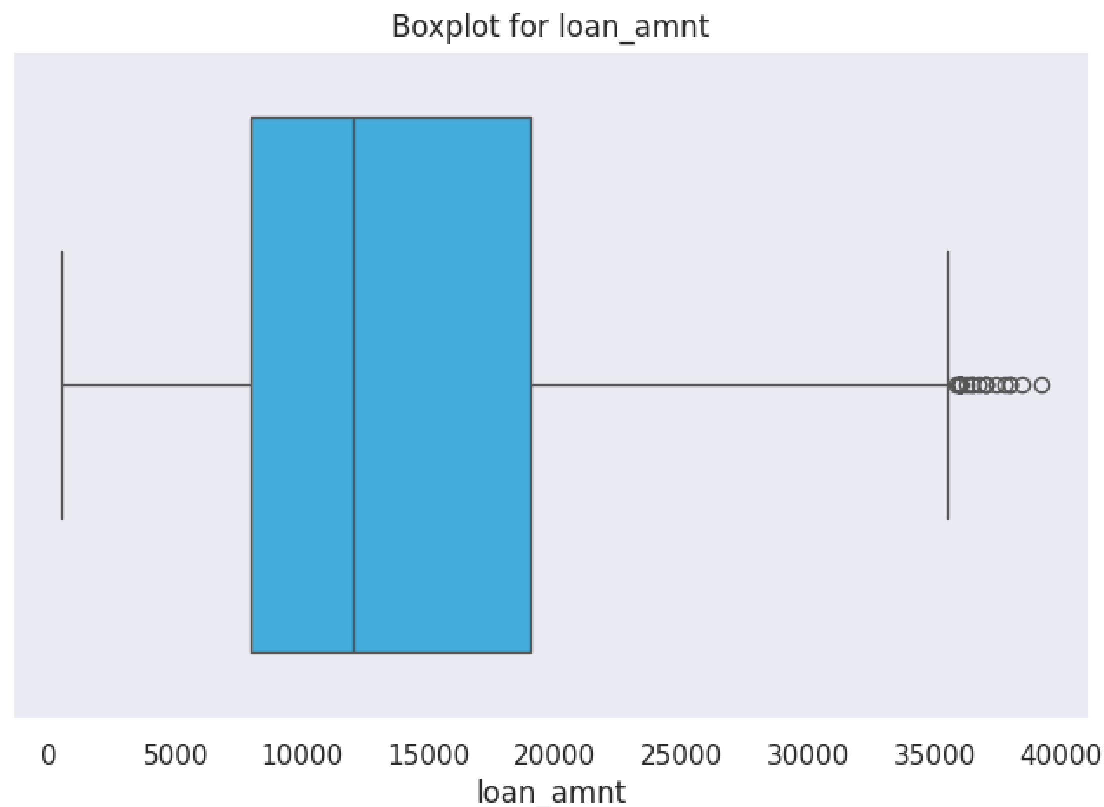

Boxplot for total_acc
0
20
40
60
80
100
120
140
total_acc

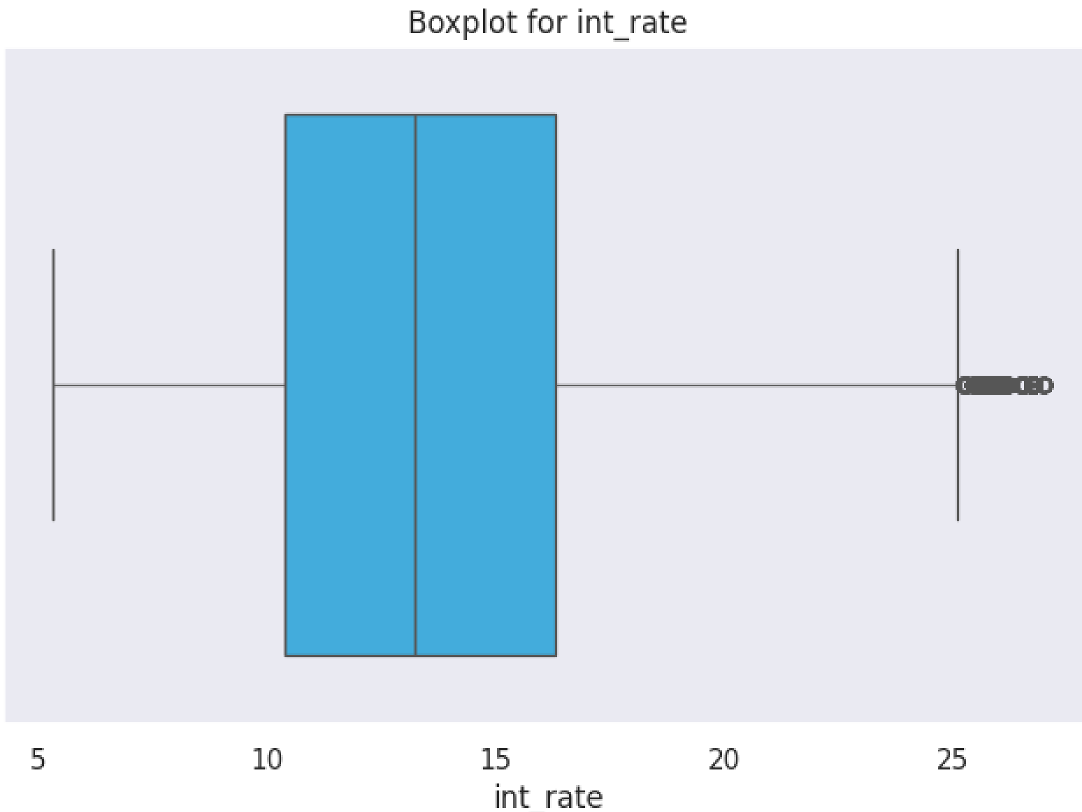

Boxplot for mort_acc
0.0
0.2
0.4
0.6
0.8
1.0
mort_acc

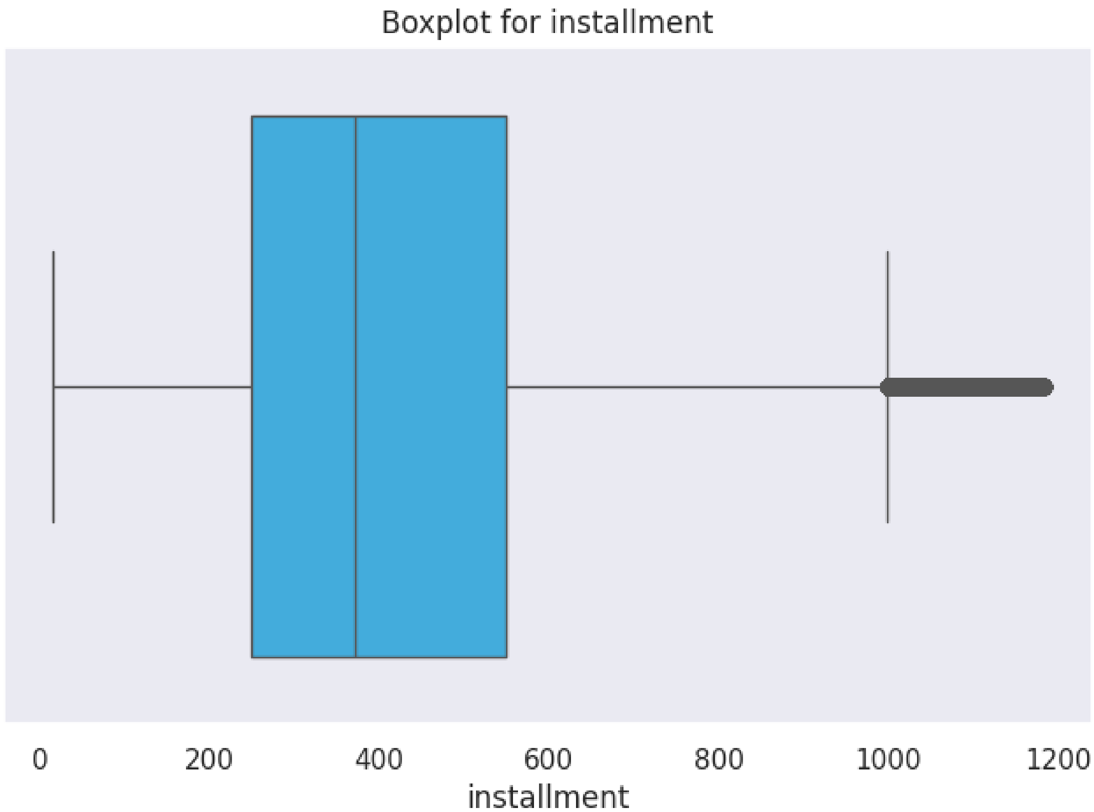


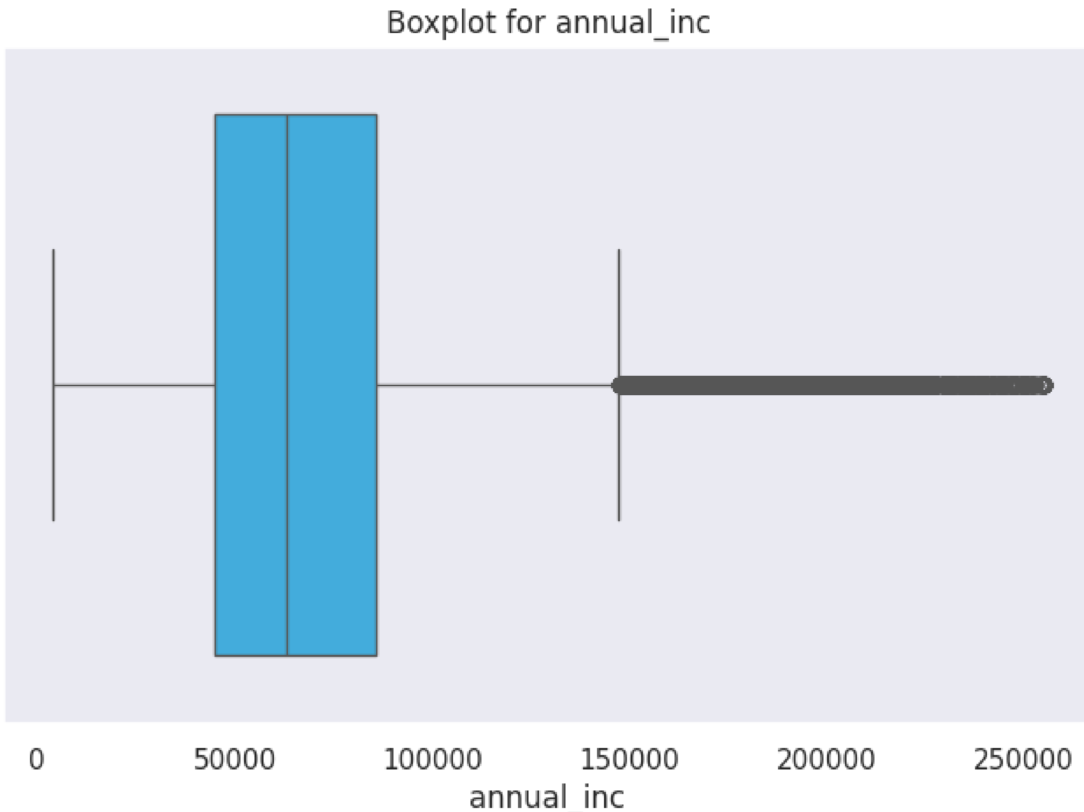
[137]:
```
# Outlier treatment

for col in n_columns:
    if col in df.columns:
        mean = df[col].mean()
        std = df[col].std()
        upper_limit = mean + 3 * std
        lower_limit = mean - 3 * std
        df = df[(df[col] < upper_limit) & (df[col] > lower_limit)]
```

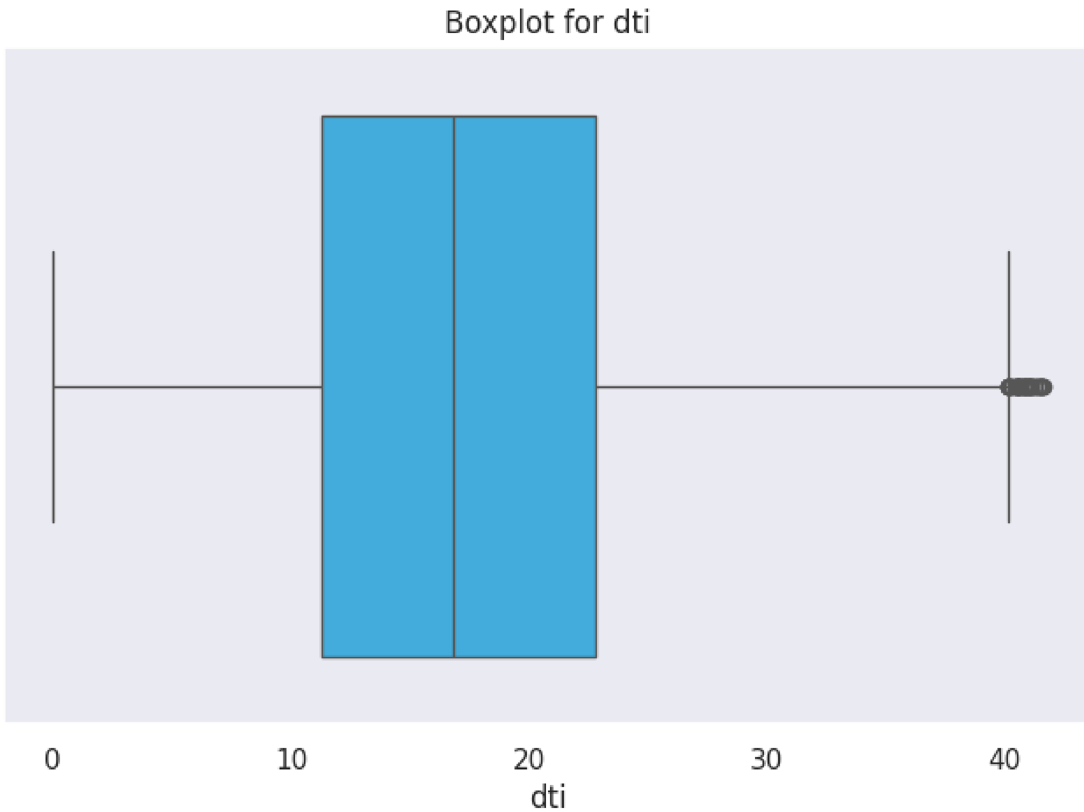
[138]:
```
def box_plot(col):
    if col in df.columns:
        plt.figure(figsize=(8, 5))
        sns.boxplot(x=df[col],color="#29B6F6")
        plt.title('Boxplot for {}'.format(col))
        plt.show()
    else:
        print(f"Column '{col}' not found in the DataFrame.")

for col in n_columns:
    box_plot(col)
```

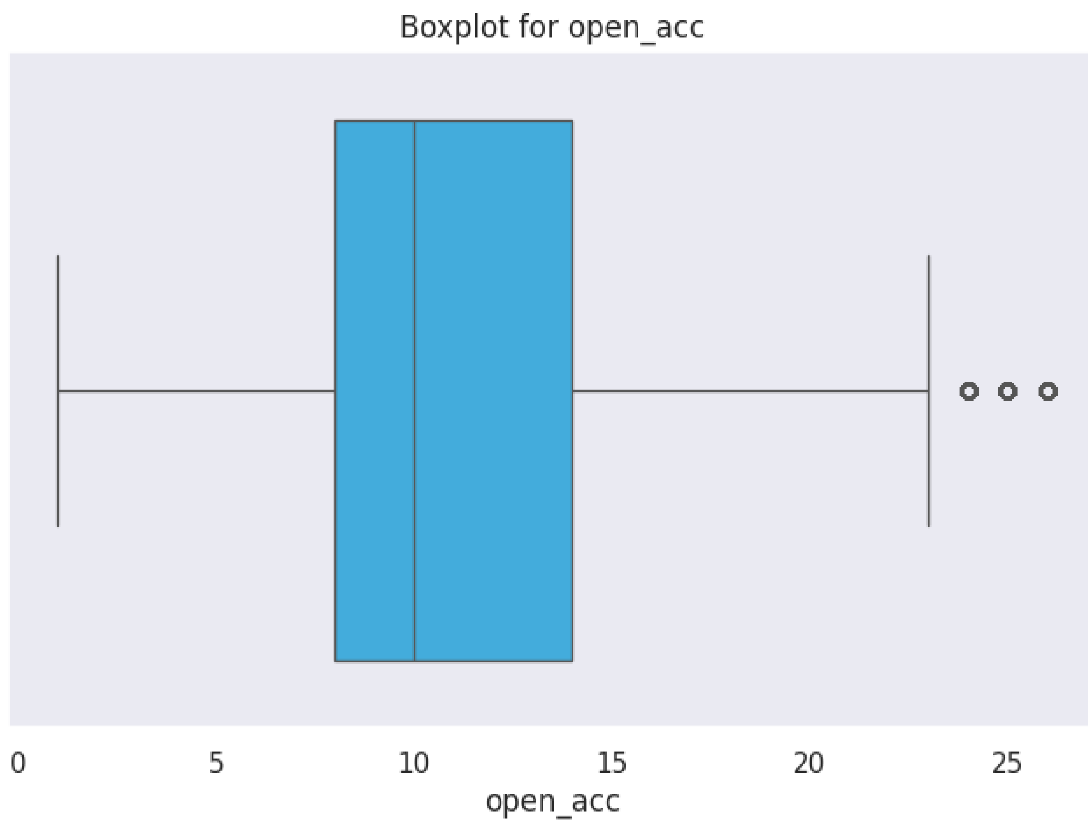

Boxplot for loan_amnt
0
5000
10000
15000
20000
25000
30000
35000
40000
loan_amnt

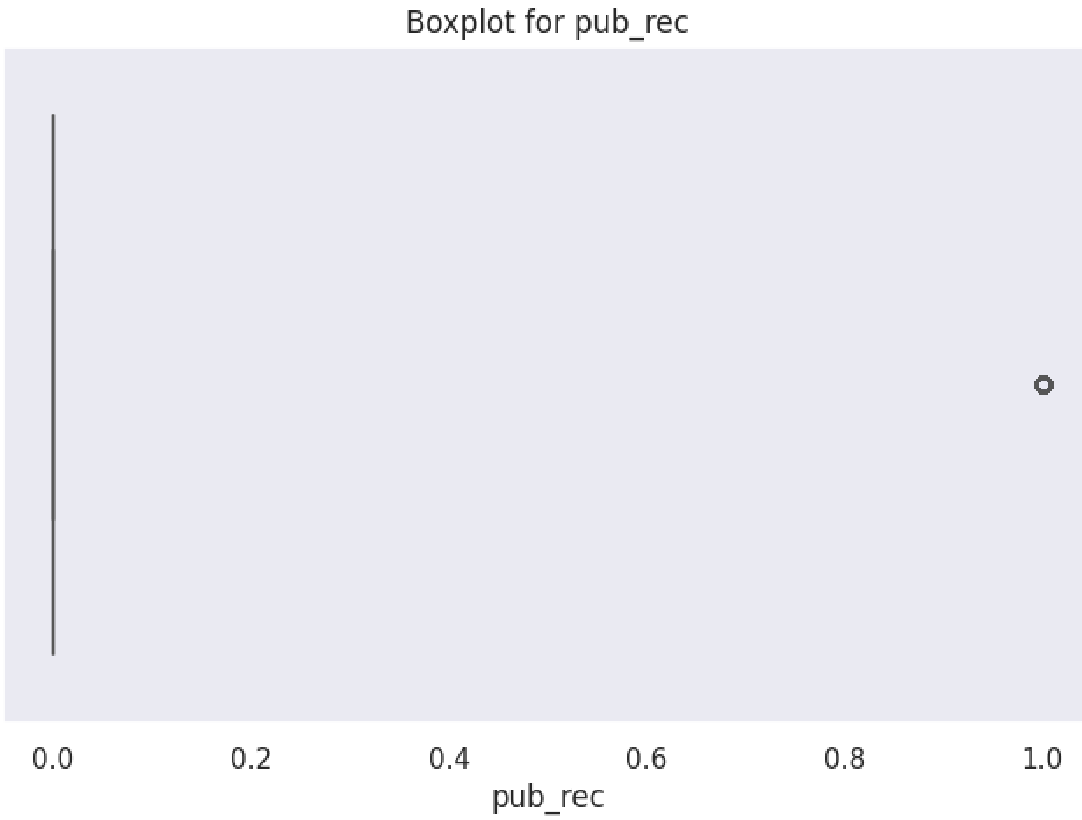

Boxplot for int_rate
5
10
15
20
25
int_rate


Boxplot for installment
0
200
400
600
800
1000
1200
installment

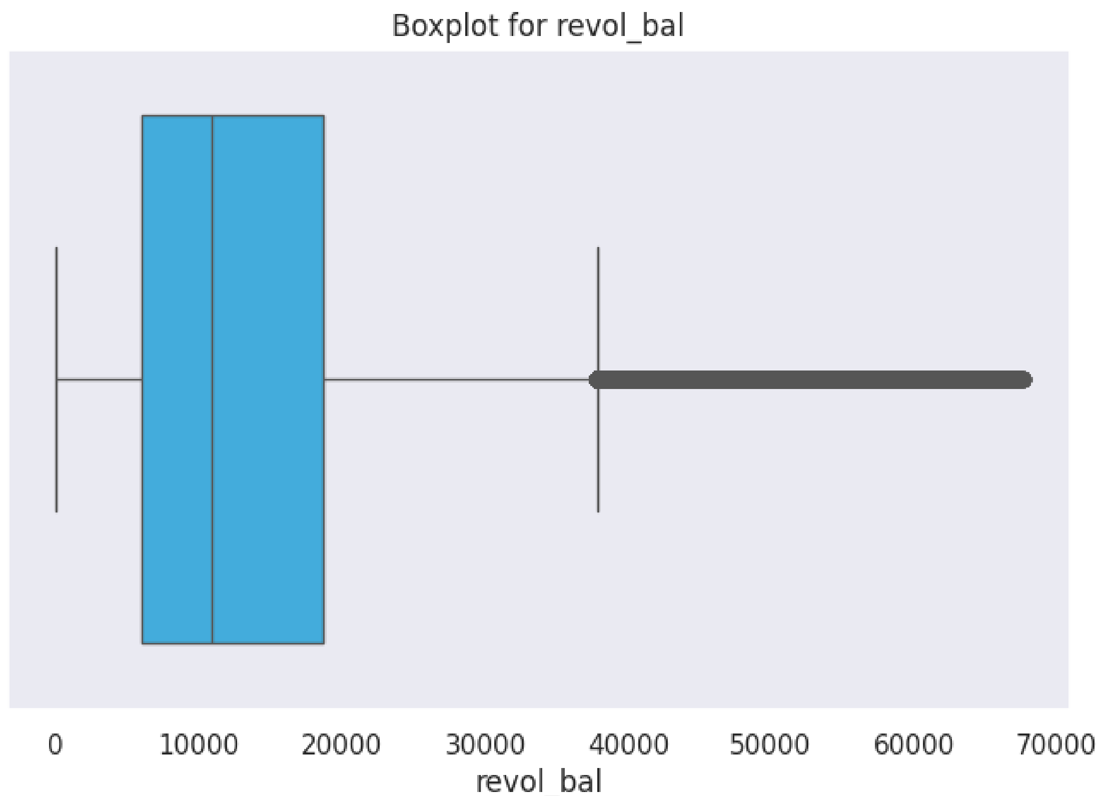

Boxplot for annual_inc
0
50000
100000
150000
200000
250000
annual_inc

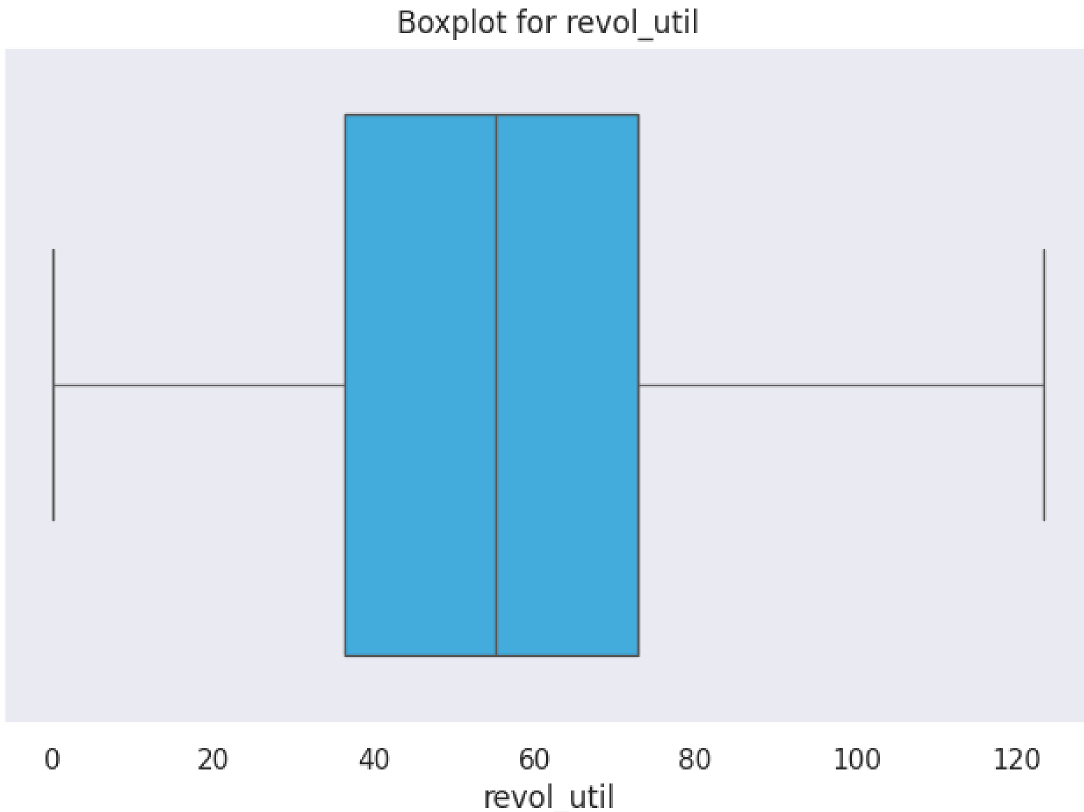

Boxplot for dti
0
10
20
30
40
dti

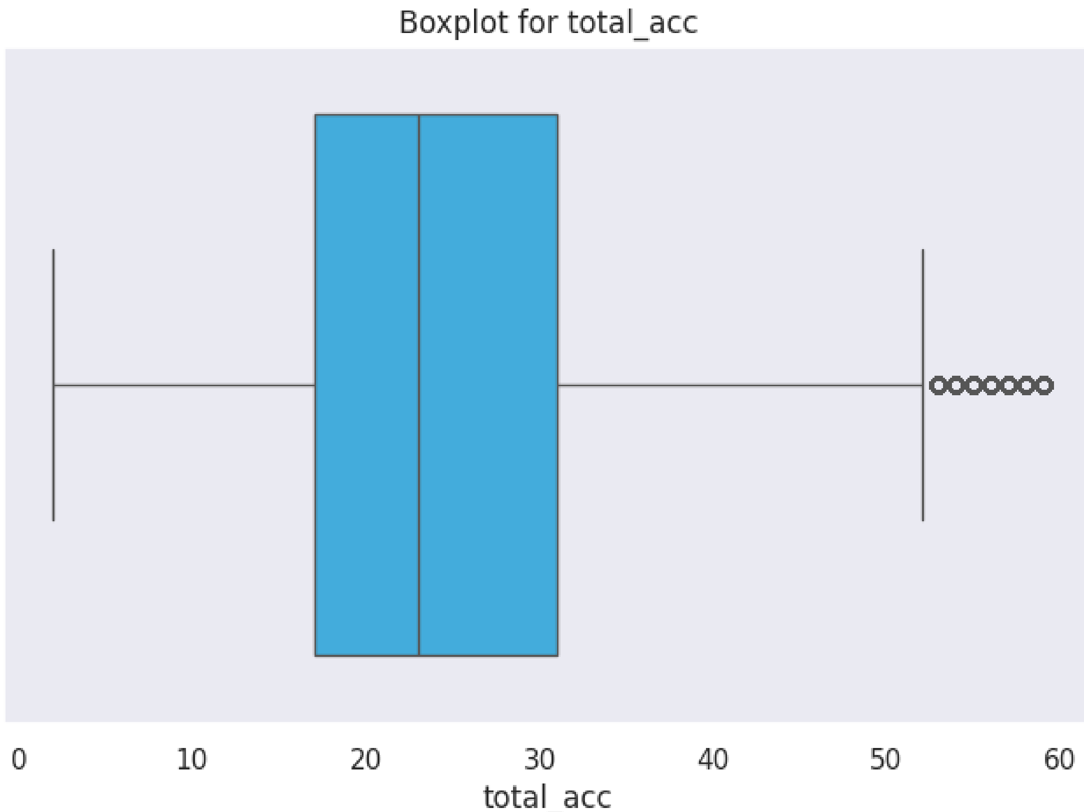

Boxplot for open_acc
0
5
10
15
20
25
open_acc



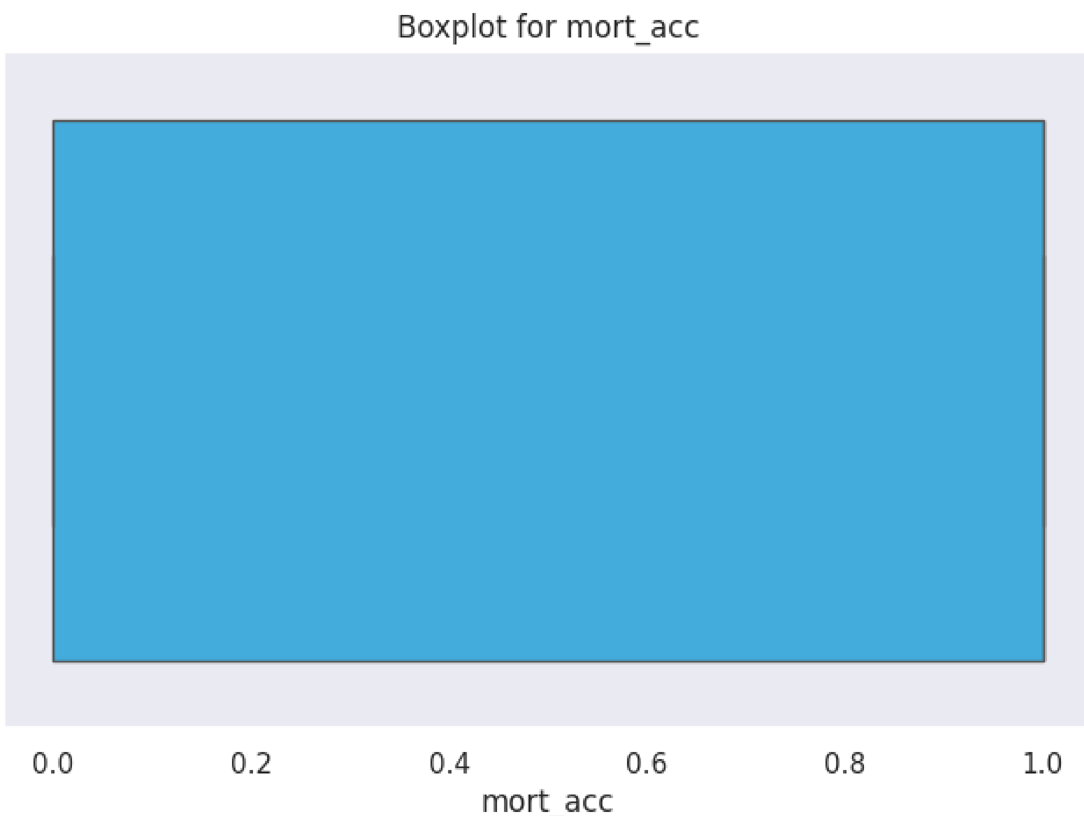
Boxplot for pub_rec
0.0
0.2
0.4
0.6
0.8
1.0
pub_rec

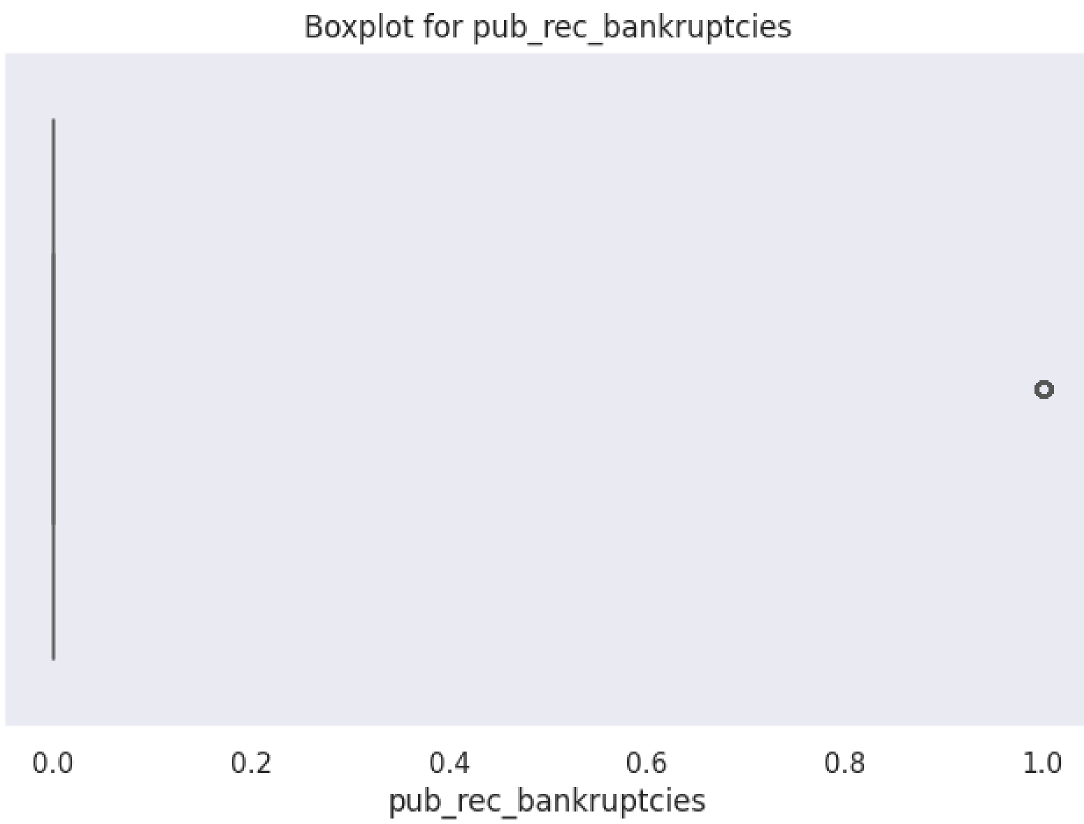

Boxplot for revol_bal
0
10000
20000
30000
40000
50000
60000
70000
revol_bal


Boxplot for revol_util
0
20
40
60
80
100
120
revol_util



Boxplot for total_acc
0
10
20
30
40
50
60
total_acc


Boxplot for mort_acc
0.0
0.2
0.4
0.6
0.8
1.0
mort_acc



[139]:
```
term_values = {' 36 months': 36, ' 60 months': 60}

df['term'] = df['term'].map(term_values)

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

list_status = {'w': 0, 'f': 1}

df['initial_list_status'] = df['initial_list_status'].map(list_status)

df['zip_code'] = df['address'].apply(lambda x: x[-5:])

df['zip_code'].value_counts(normalize=True) * 100
```

[139]:
```
zip_code
70466    14.375337
30723    14.289710
22690    14.272299
48052    14.127019
00813    11.605591
29597    11.548792
```

```
05113    11.519108
93700     2.768605
11650     2.762896
86630     2.730643
Name: proportion, dtype: float64
```

[140]:
```
# Dropping of unnecessary columns

unnecessary_columns=['issue_d', 'emp_title', 'title', 'sub_grade','address',␣
↪'earliest_cr_line', 'emp_length']

df.drop(unnecessary_columns,axis=1, inplace=True)
```

1.0.10 One hot encoding

[141]:
```
dummies=['purpose', 'zip_code', 'grade', 'verification_status',␣
↪'application_type', 'home_ownership']

data=pd.get_dummies(df,columns=dummies,drop_first=True)

pd.set_option('display.max_columns',None)

pd.set_option('display.max_rows',None)
```

[142]:
```
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=y,random_state=42)
print(X_train.shape)
print(X_test.shape)
```

```
(245249, 51)
(105108, 51)
```

1.0.11 Model Building

[143]:
```
# Importing stats libraries

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
```

[144]:
```
scaler = MinMaxScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[145]:
```
logreg=LogisticRegression(max_iter=1000)

logreg.fit(X_train,y_train)
```

[145]:
```
LogisticRegression(max_iter=1000)
```

[146]:
```
y_pred = logreg.predict(X_test)

print('Accuracy of Logistic Regression Classifier on test set: {:.3f}'.
 ↪format(logreg.score(X_test, y_test)))
```

```
Accuracy of Logistic Regression Classifier on test set: 0.891
```

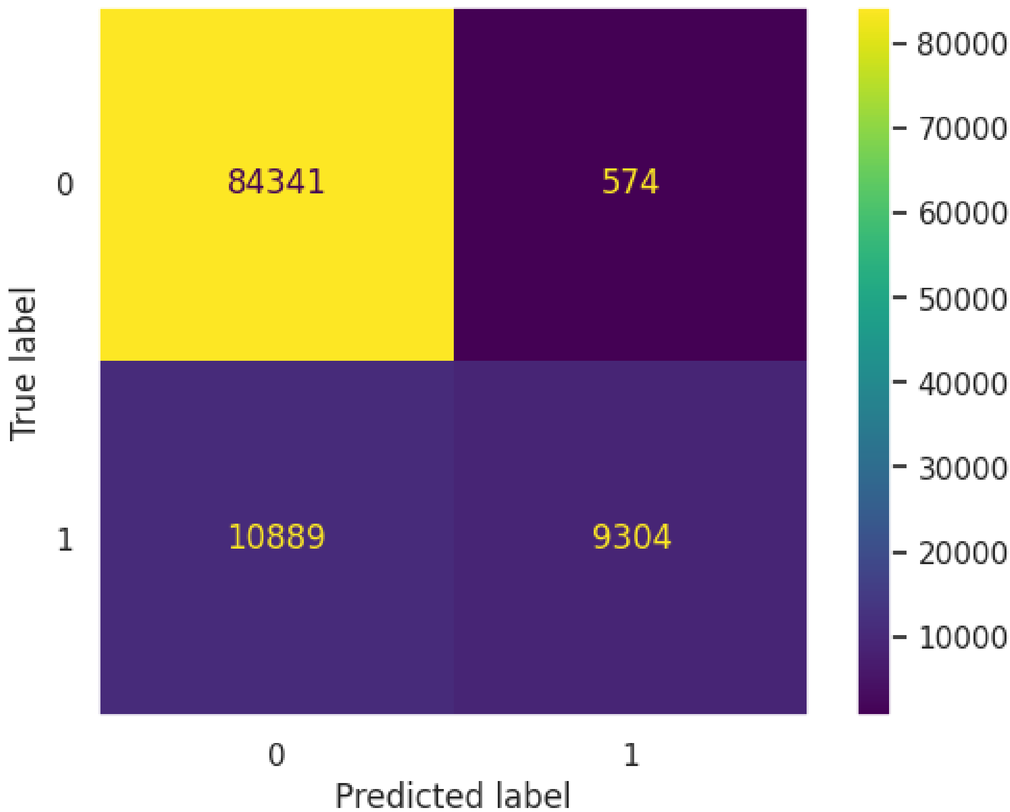
[147]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     84915
           1       0.94      0.46      0.62     20193

    accuracy                           0.89    105108
   macro avg       0.91      0.73      0.78    105108
weighted avg       0.90      0.89      0.88    105108
```

[148]:
```
#Plot confusion Matrix

confusion_matrix=confusion_matrix(y_test,y_pred)

print(confusion_matrix)

ConfusionMatrixDisplay(confusion_matrix=confusion_matrix, display_labels=logreg.
 ↪classes_).plot()
```

```
[[84341   574]
 [10889  9304]]
```

[148]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x79b6259daa10>



ROC Curve - An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

- TPR=(TP)/(TP+FN)

False Positive Rate (FPR) is defined as follows:

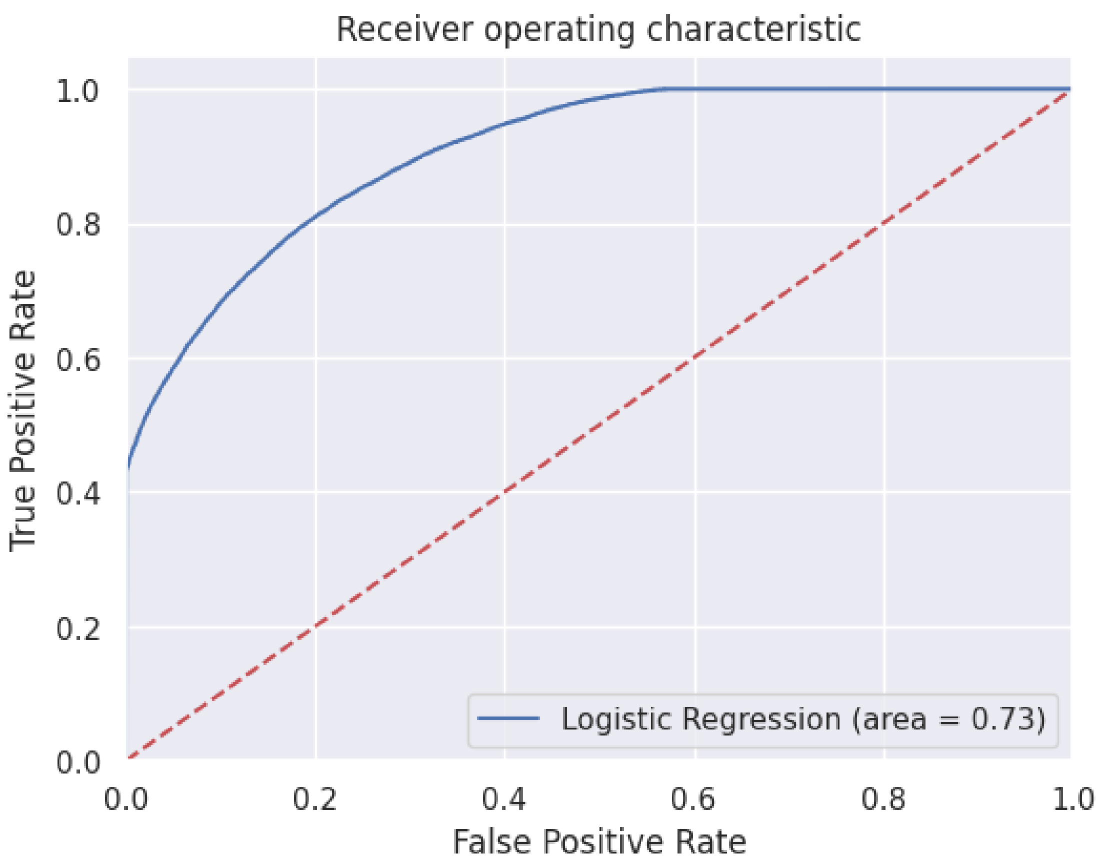
- FPR=(FP)/(FP+TN)

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

AUC (Area under the ROC Curve) - AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:

[149]:
```
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))

fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])



plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.grid()
plt.show()
```



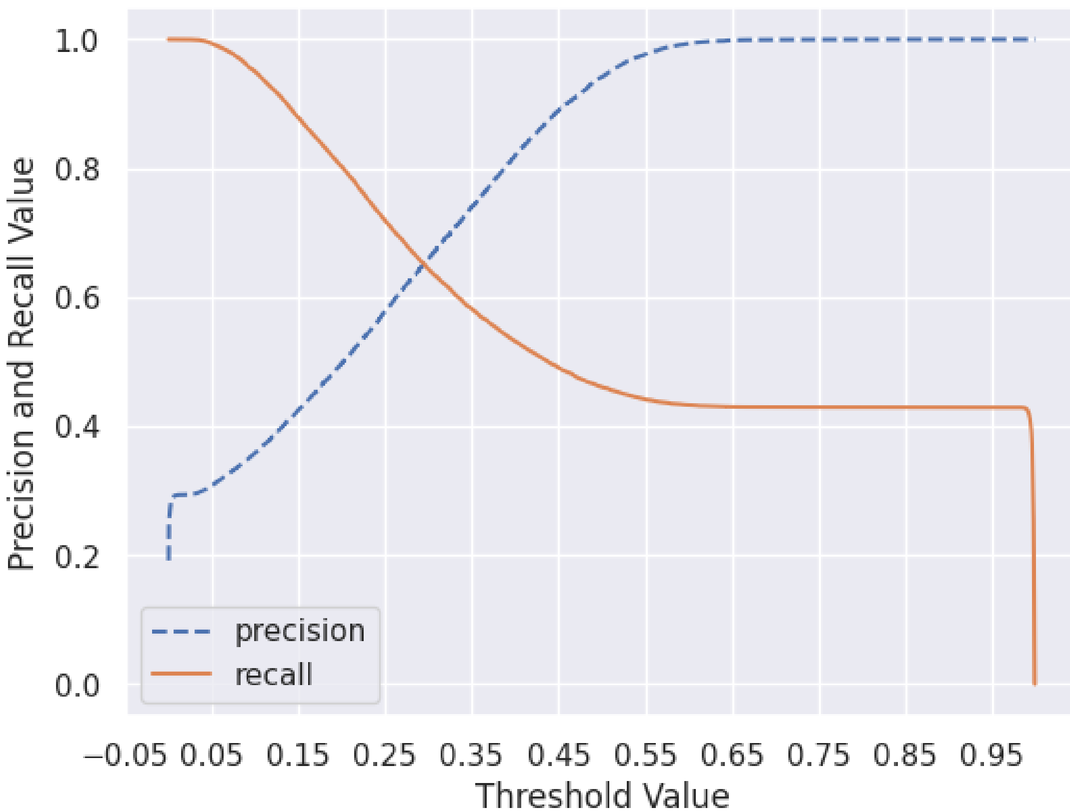
Insights:

- ROC-AUC curve is grossing the area near about 0.73 which indicates that model is performing well.
- There is still room for some model improvement
- By collecting more data, using a more complex model, or tuning the hyperparameters, it is possible to improve the model's performance.

1.0.12 Precision-Recall Curve

[150]:
```
precisions, recalls, thresholds = precision_recall_curve(y_test, logreg.
    ↪predict_proba(X_test)[:, 1])

threshold_boundary = thresholds.shape[0]

# Plot precision
plt.plot(thresholds, precisions[0:threshold_boundary], linestyle='--',␣
    ↪label='precision')

# Plot recall
```

```
plt.plot(thresholds, recalls[0:threshold_boundary], label='recall')

start, end = plt.xlim()
plt.xticks(np.round(np.arange(start, end, 0.1), 2))

plt.xlabel('Threshold Value')
plt.ylabel('Precision and Recall Value')
plt.legend()
plt.grid()
plt.show()
```



Insights

- Precision score is highest at 0.55 threshold. High precision value indicates that model is positevly predicating the charged off loan status which helps business to take more stable decision.
- Recall score is higher on smaller threshold but after 0.55 the recall value is constant. Model is correctly classifying the actual predicated values as instances.

Actional Insights and Recommendations

1. 80% of the customers have paid the loan fully.

2. 20% of the customers are the defaulters.
3. The organization can the trained model to make prediction for whether a person will likely to pay the loan amount or he will be a defaulter.
4. Model achieves the 94% f1-score for the negative class (Fully Paid).
5. Model achieves the 62% f1-score for the positive class (Charged off).
6. Cross Validation accuracy and testing accuracy is almost same which infers model is performing the decent job. We can trust this model for unseen data
7. By collecting more data, using a more complex model, or tuning the hyperparameters, it is possible to improve the model's performance.
8. ROC AUC curve area of 0.73, the model is correctly classifying about 73% of the instances. This is a good performance, but there is still room for improvement.
9. The precision-recall curve allows us to see how the precision and recall trade-off as we vary the threshold. A higher threshold will result in higher precision, but lower recall, and vice versa. The ideal point on the curve is the one that best meets the needs of the specific application.
10. After balancing the dataset, there is significant change observed in the precion and recall score for both of the classes.
11. Accuracy of Logistic Regression Classifier on test set: 0.891 which is decent and not by chance