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

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

```
df=pd.read_csv('logistic_regression.csv')
```

```
df.head(5)
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_owr
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	MOR
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	MOR

5 rows × 27 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235886 entries, 0 to 235885
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype  
---  ------                --------------   -----  
 0   loan_amnt             235886 non-null  float64
 1   term                  235886 non-null  object 
 2   int_rate              235886 non-null  float64
 3   installment           235886 non-null  float64
 4   grade                 235886 non-null  object 
 5   sub_grade             235886 non-null  object 
 6   emp_title             222209 non-null  object 
 7   emp_length            224948 non-null  object 
 8   home_ownership        235886 non-null  object 
 9   annual_inc            235886 non-null  float64
 10  verification_status   235886 non-null  object 
 11  issue_d               235886 non-null  object 
 12  loan_status           235886 non-null  object 
 13  purpose               235886 non-null  object 
 14  title                 234848 non-null  object 
 15  dti                   235886 non-null  float64
 16  earliest_cr_line      235886 non-null  object 
 17  open_acc              235886 non-null  float64
 18  pub_rec               235886 non-null  float64
 19  revol_bal             235886 non-null  float64
 20  revol_util            235715 non-null  float64
 21  total_acc             235886 non-null  float64
 22  initial_list_status   235886 non-null  object 
 23  application_type      235886 non-null  object 
 24  mort_acc              213302 non-null  float64
 25  pub_rec_bankruptcies  235557 non-null  float64
 26  address               235885 non-null  object 
dtypes: float64(12), object(15)
memory usage: 48.6+ MB
```

```
df.describe()
```

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_util	total_a
count	235886.000000	235886.000000	235886.000000	2.358860e+05	235886.000000	235886.000000	235886.000000	2.358860e+05	235715.000000	235886.0000
mean	14104.732053	13.643355	431.524698	7.427901e+04	17.325831	11.306932	0.179167	1.581559e+04	53.777674	25.4245
std	8354.907949	4.467399	250.662467	6.006704e+04	8.130635	5.136844	0.544709	2.045974e+04	24.502087	11.8985
min	500.000000	5.320000	16.250000	2.500000e+03	0.000000	0.000000	0.000000	0.000000e+00	0.000000	2.0000
25%	8000.000000	10.490000	250.330000	4.500000e+04	11.260000	8.000000	0.000000	6.011000e+03	35.800000	17.0000
50%	12000.000000	13.330000	375.370000	6.400000e+04	16.880000	10.000000	0.000000	1.116300e+04	54.800000	24.0000
75%	20000.000000	16.490000	567.010000	9.000000e+04	22.960000	14.000000	0.000000	1.960375e+04	72.900000	32.0000
max	40000.000000	30.990000	1533.810000	7.446395e+06	189.900000	90.000000	86.000000	1.743266e+06	892.300000	151.0000

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

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

```
#checking for Non-Numeric Columns
cat_col=[col for col in df.columns if df[col].dtype=='O']
cat_col
```

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

```
#Number of Unique values from all non_numeric columns
for col in cat_col:

  print(f"No. of Unique values {col}: {df[col].nunique()}")
```

```
No. of Unique values term: 2
No. of Unique values grade: 7
No. of Unique values sub_grade: 35
No. of Unique values emp_title: 111427
No. of Unique values emp_length: 11
No. of Unique values home_ownership: 6
No. of Unique values verification_status: 3
No. of Unique values issue_d: 115
No. of Unique values loan_status: 2
No. of Unique values purpose: 14
No. of Unique values title: 31403
No. of Unique values earliest_cr_line: 664
No. of Unique values initial_list_status: 2
No. of Unique values application_type: 3
No. of Unique values address: 234999
```

```
#convert string data types into datetime format
df["earliest_cr_line"]=pd.to_datetime(df["earliest_cr_line"])
df['issue_d']=pd.to_datetime(df['issue_d'])
```

```
<ipython-input-8-a65e86cf7087>:2: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To
  df["earliest_cr_line"]=pd.to_datetime(df["earliest_cr_line"])
<ipython-input-8-a65e86cf7087>:3: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `dateutil`. To
  df['issue_d']=pd.to_datetime(df['issue_d'])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235886 entries, 0 to 235885
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             235886 non-null  float64
 1   term                  235886 non-null  object
 2   int_rate              235886 non-null  float64
 3   installment           235886 non-null  float64
 4   grade                 235886 non-null  object
```

```
 5   sub_grade             235886 non-null  object
 6   emp_title             222209 non-null  object
 7   emp_length            224948 non-null  object
 8   home_ownership        235886 non-null  object
 9   annual_inc            235886 non-null  float64
 10  verification_status   235886 non-null  object
 11  issue_d               235886 non-null  datetime64[ns]
 12  loan_status           235886 non-null  object
 13  purpose               235886 non-null  object
 14  title                 234848 non-null  object
 15  dti                   235886 non-null  float64
 16  earliest_cr_line      235886 non-null  datetime64[ns]
 17  open_acc              235886 non-null  float64
 18  pub_rec               235886 non-null  float64
 19  revol_bal             235886 non-null  float64
 20  revol_util            235715 non-null  float64
 21  total_acc             235886 non-null  float64
 22  initial_list_status   235886 non-null  object
 23  application_type      235886 non-null  object
 24  mort_acc              213302 non-null  float64
 25  pub_rec_bankruptcies  235557 non-null  float64
 26  address               235885 non-null  object
dtypes: datetime64[ns](2), float64(12), object(13)
memory usage: 48.6+ MB
```

```
df.dtypes
```

```
loan_amnt                      float64
term                            object
int_rate                       float64
installment                    float64
grade                           object
sub_grade                       object
emp_title                       object
emp_length                      object
home_ownership                  object
annual_inc                     float64
verification_status             object
issue_d                 datetime64[ns]
loan_status                     object
purpose                         object
title                           object
dti                            float64
earliest_cr_line        datetime64[ns]
open_acc                       float64
pub_rec                        float64
revol_bal                      float64
revol_util                     float64
total_acc                      float64
initial_list_status             object
application_type                object
mort_acc                       float64
pub_rec_bankruptcies           float64
address                         object
dtype: object
```

```
df.duplicated().sum()
```

```
0
```

```
df.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               13677
emp_length              10938
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1038
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                171
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                22584
pub_rec_bankruptcies      329
address                     1
dtype: int64
```

We have bunch of missing value attributes.

```
df.describe(include = 'object')
```

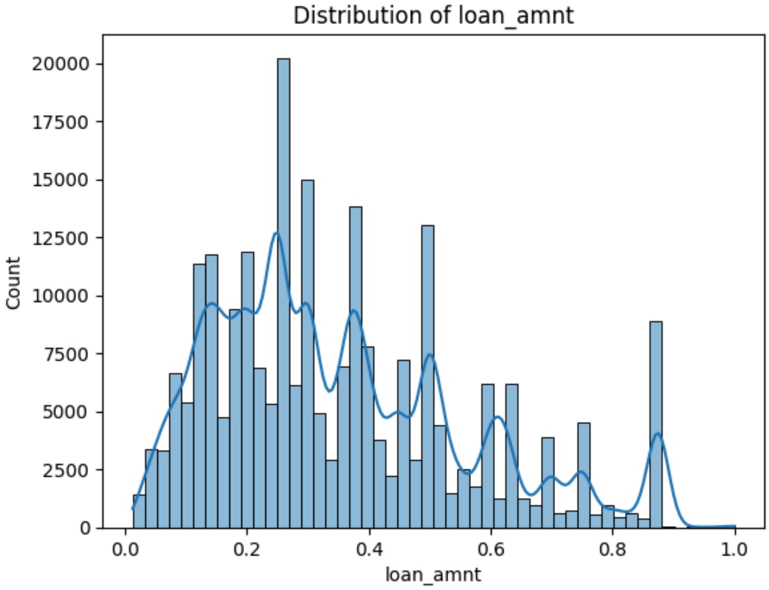
	term	grade	sub_grade	emp_title	emp_length	home_ownership	verification_status	loan_status	purpose	title	initial_l
count	235886	235886	235886	222209	224948	235886	235886	235886	235886	234848	
unique	2	7	35	111427	11	6	3	2	14	31403	
top	36 months	B	B3	Teacher	10+ years	MORTGAGE	Verified	Fully Paid	debt_consolidation	Debt consolidation	
freq	179776	69059	15801	2607	75166	118154	82951	189716	139631	90798	

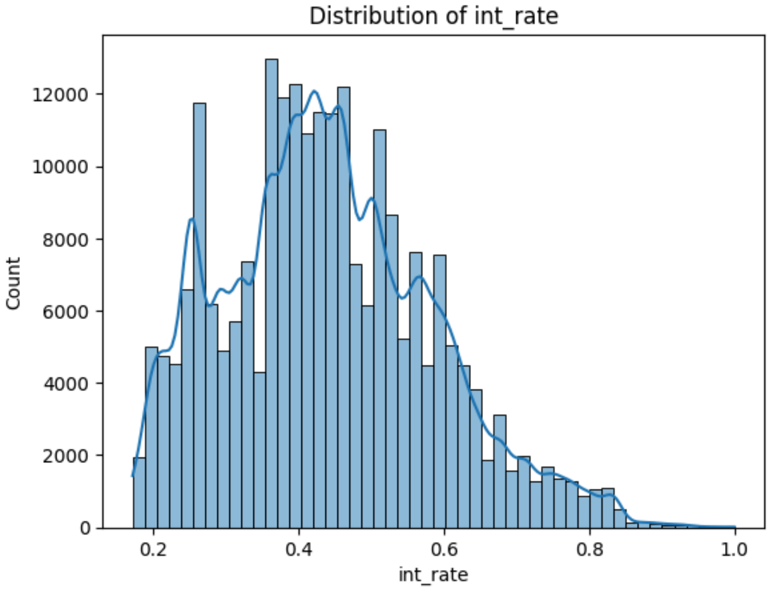
- Most of the loan disburesed for the 36 months period
- Most of the loan applicant have mortgage the home
- Majority of loans been fully paid off
- Majorily the loans been disbursed for the purpose of debt consolidation
- Most of the applicant is Individual

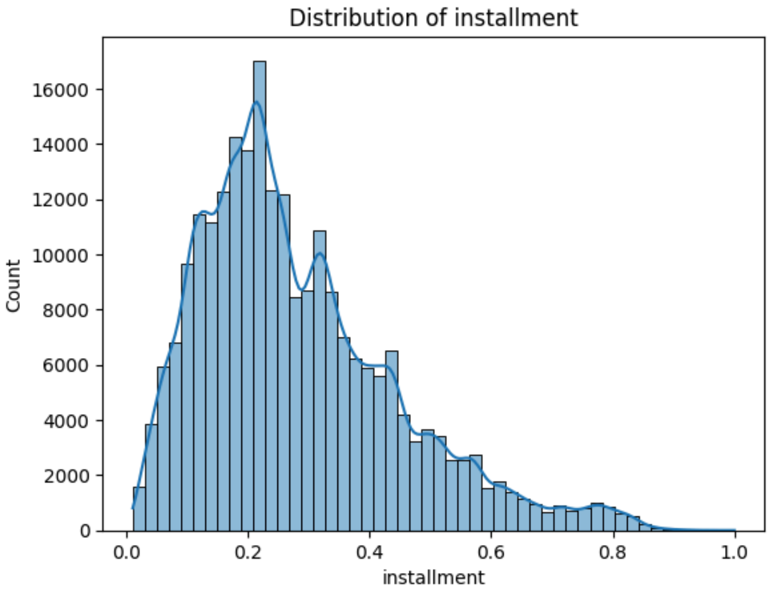
Univariate Analysis

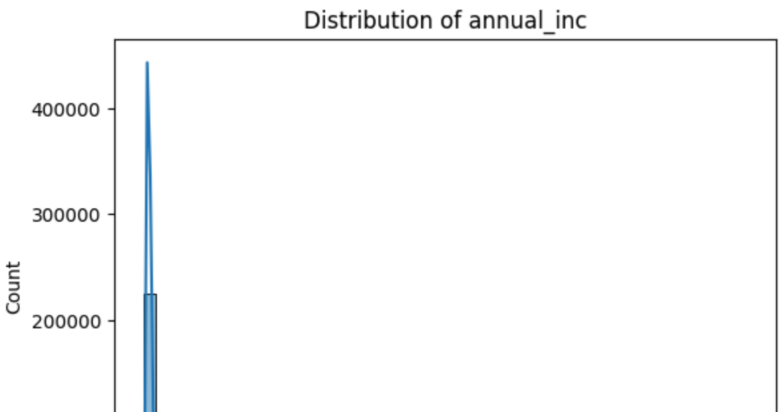
```
num_vars = df.select_dtypes('float64').columns.tolist()
```

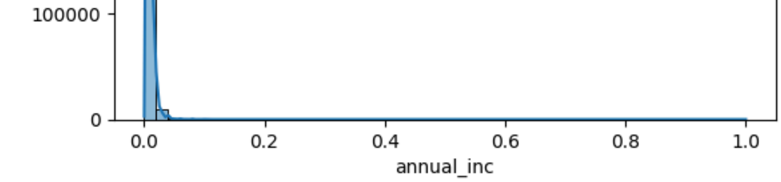
```
for i in num_vars:
    plt.title("Distribution of {}".format(i))
    sns.histplot(df[i]/df[i].max(), kde=True, bins=50)
    plt.show()
```


Distribution of loan_amnt
Count
loan_amnt

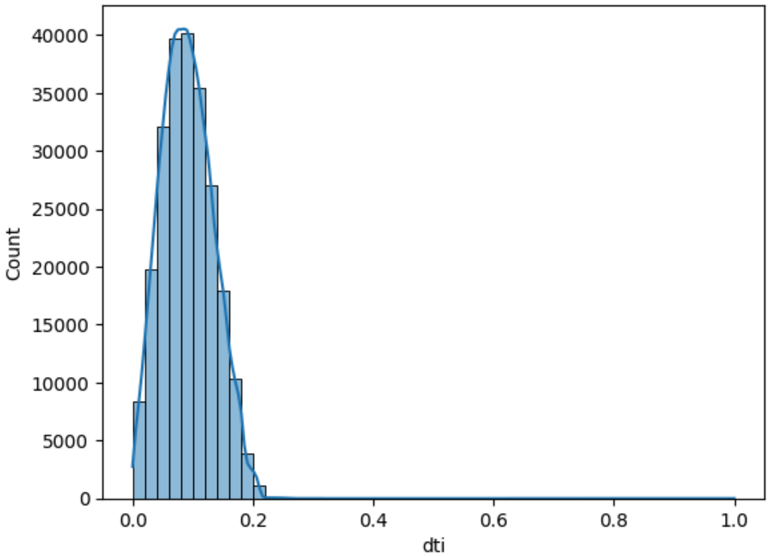

Distribution of int_rate
Count
int_rate


Distribution of installment
Count
installment

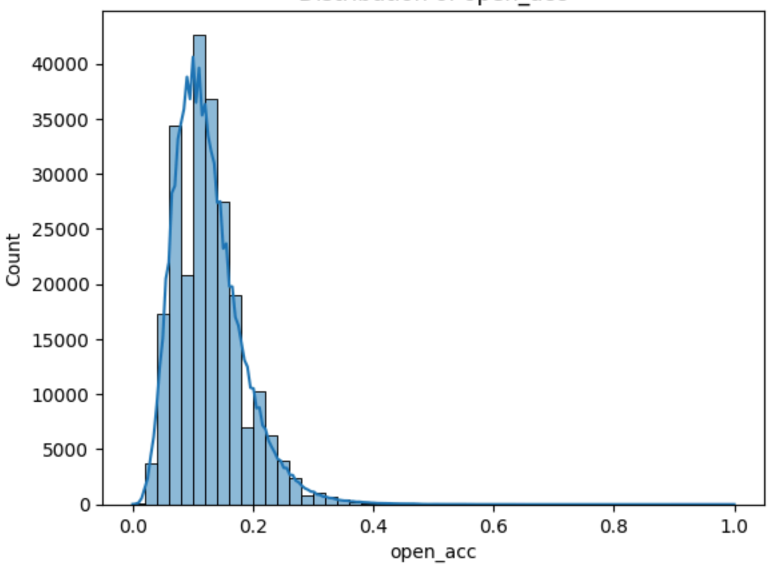

Distribution of annual_inc
Count



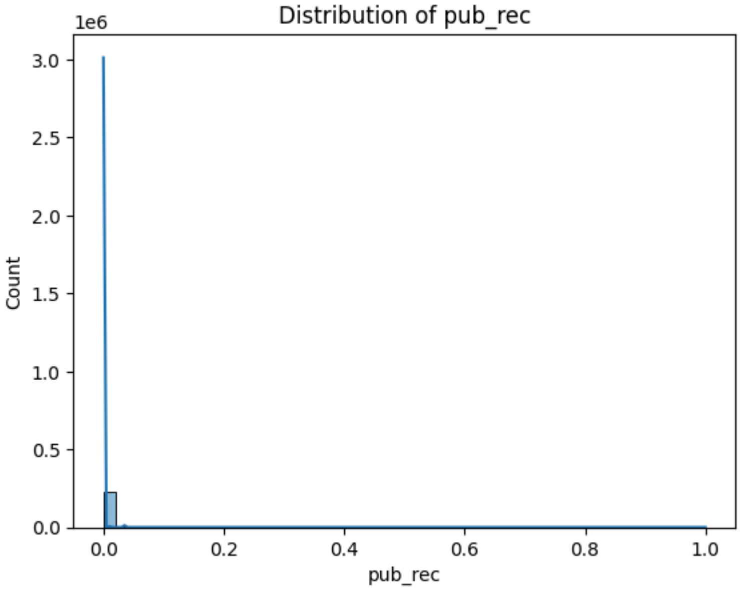
Distribution of dti



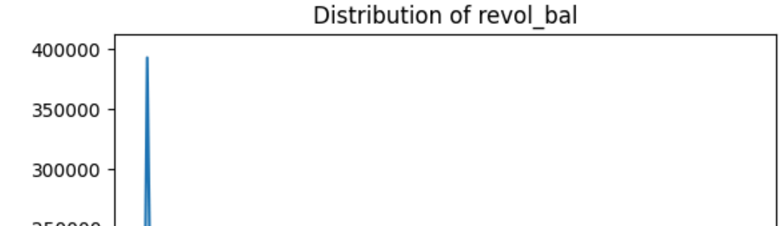
Distribution of open_acc

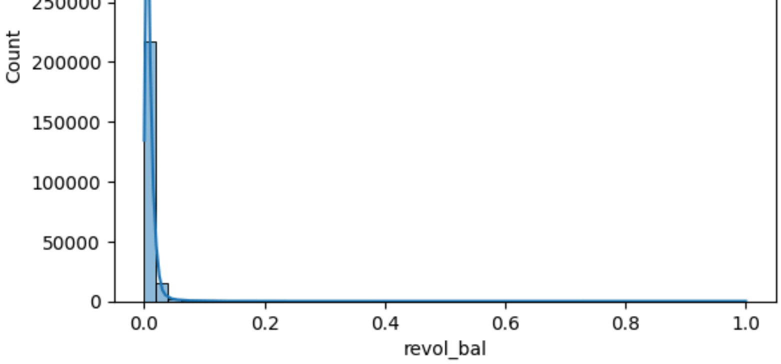


Distribution of pub_rec

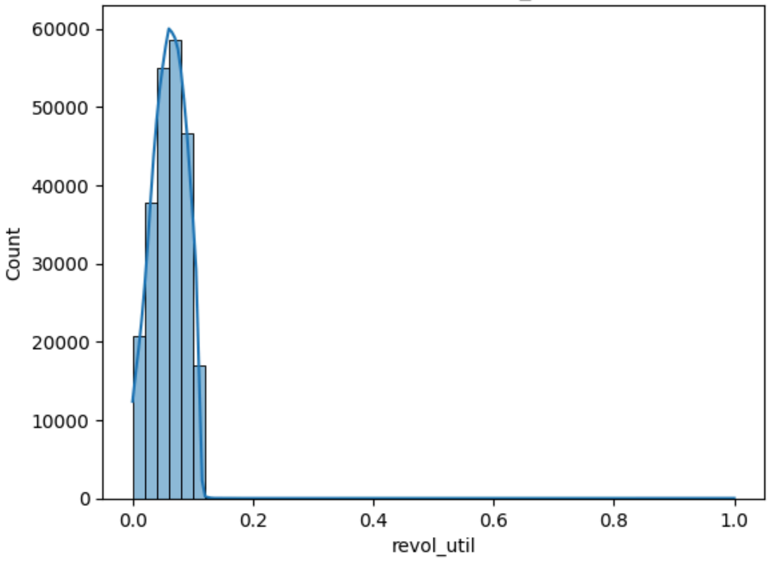


Distribution of revol_bal

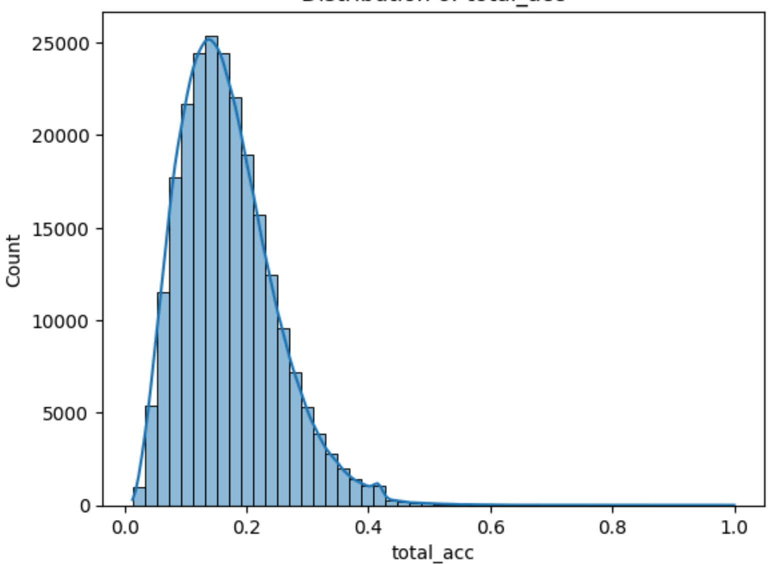




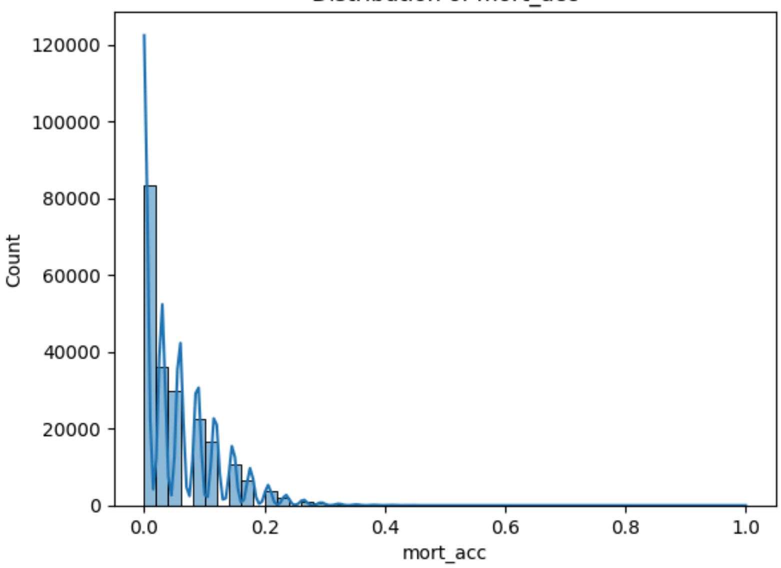
Distribution of revol_util



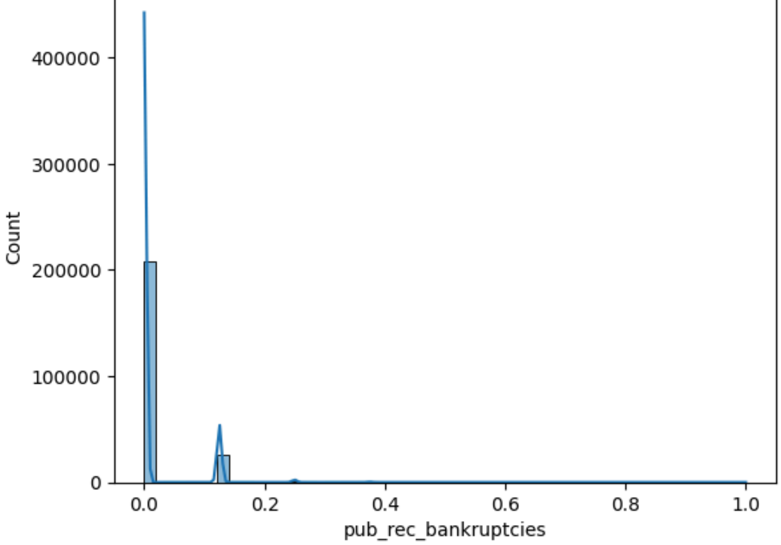
Distribution of total_acc



Distribution of mort_acc

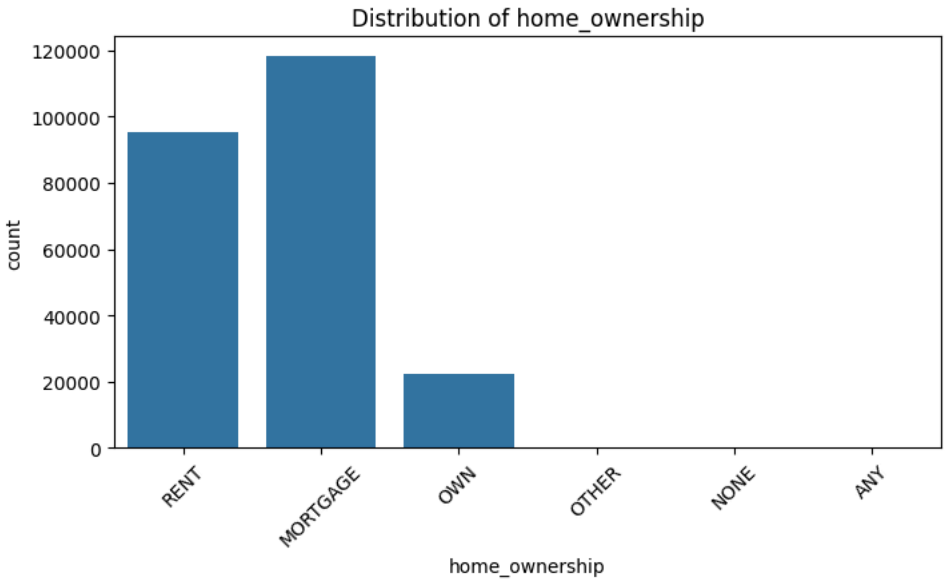


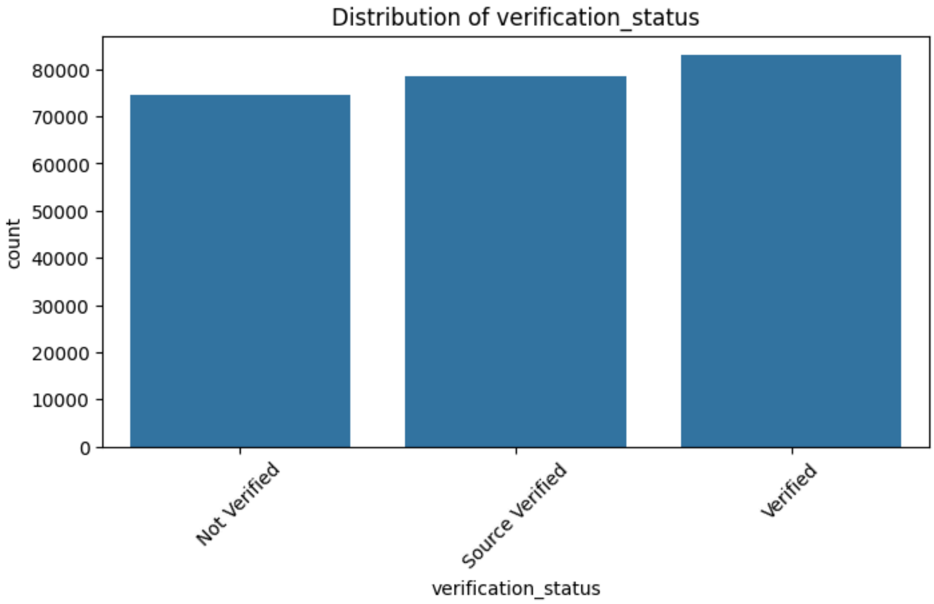
Distribution of pub_rec_bankruptcies

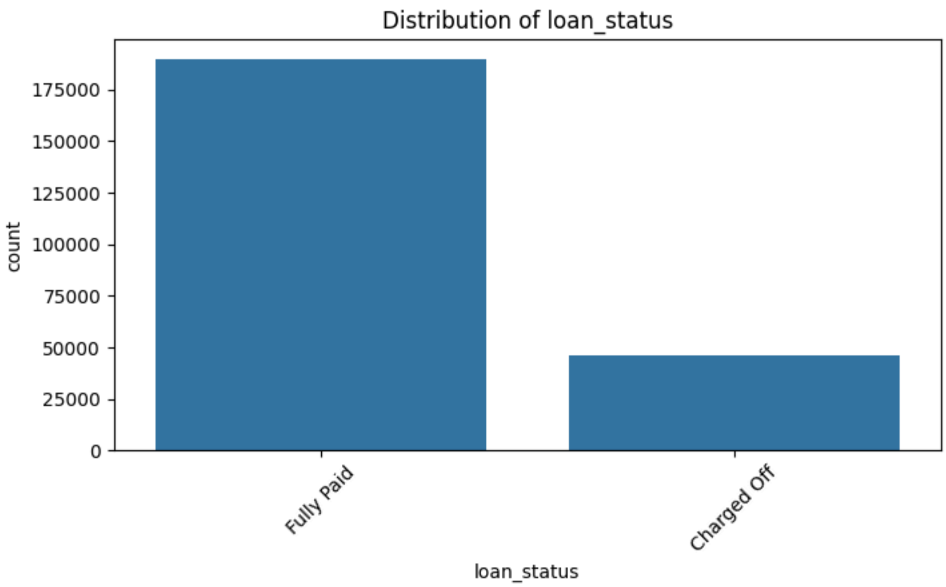


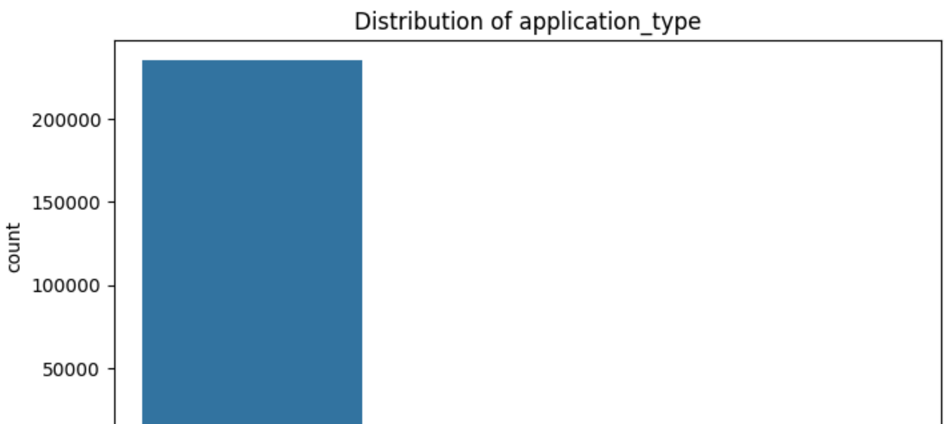
- Most of the distribution is highly skewed towerds the left side which tells us that they might contain outliers
- Almost all the continuous features have outliers present in the dataset.

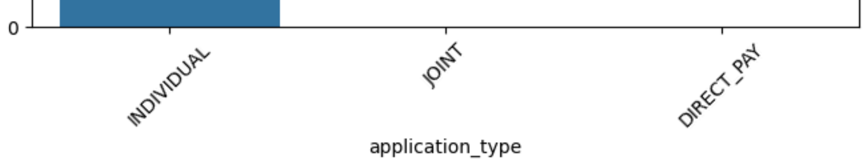
```
cat_vars = ['home_ownership', 'verification_status', 'loan_status', 'application_type', 'grade', 'sub_grade', 'term']
for i in cat_vars:
    plt.figure(figsize=(8, 4))
    plt.title(f'Distribution of {i}')
    sns.countplot(data=df, x=i)
    plt.xticks(rotation = 45)
    plt.show()
```

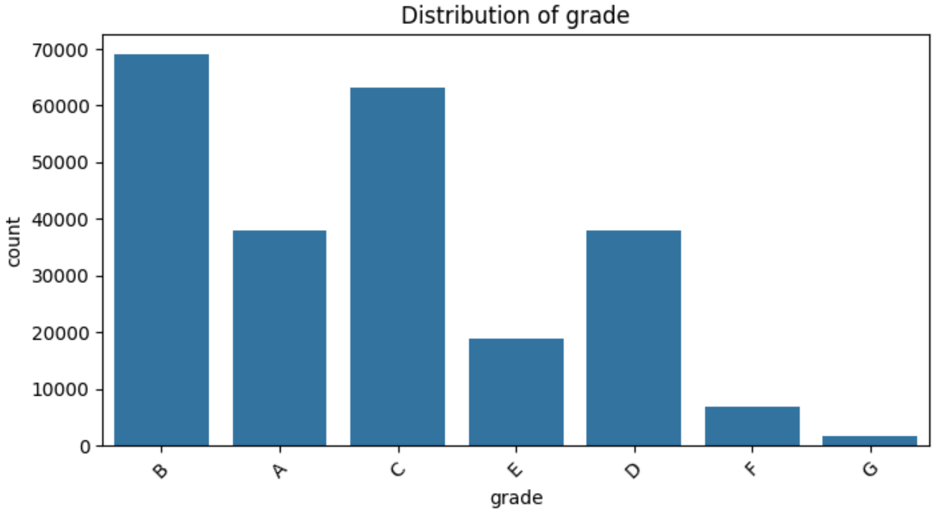

Distribution of home_ownership
count
0
20000
40000
60000
80000
100000
120000
RENT
MORTGAGE
OWN
OTHER
NONE
ANY
home_ownership

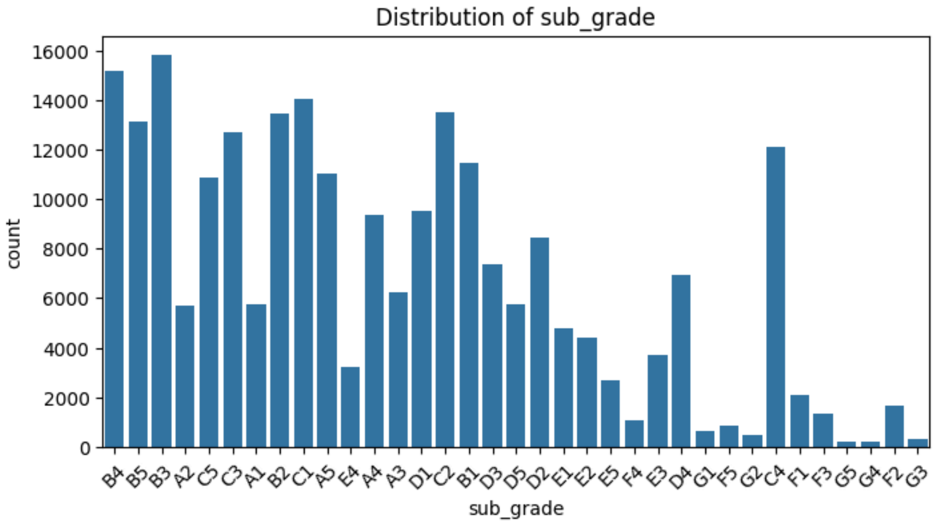

Distribution of verification_status
count
0
10000
20000
30000
40000
50000
60000
70000
80000
Not Verified
Source Verified
Verified
verification_status

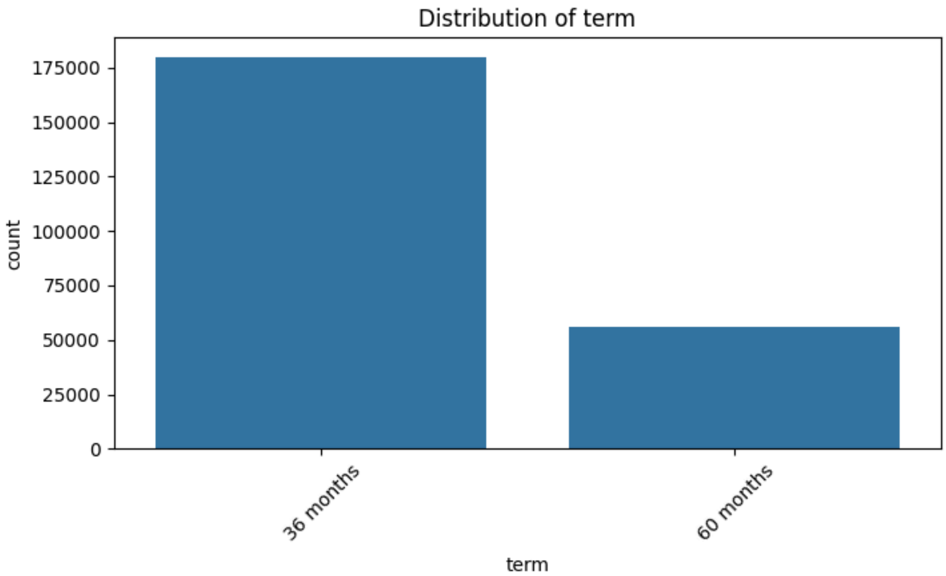

Distribution of loan_status
count
0
25000
50000
75000
100000
125000
150000
175000
Fully Paid
Charged Off
loan_status


Distribution of application_type
count
50000
100000
150000
200000







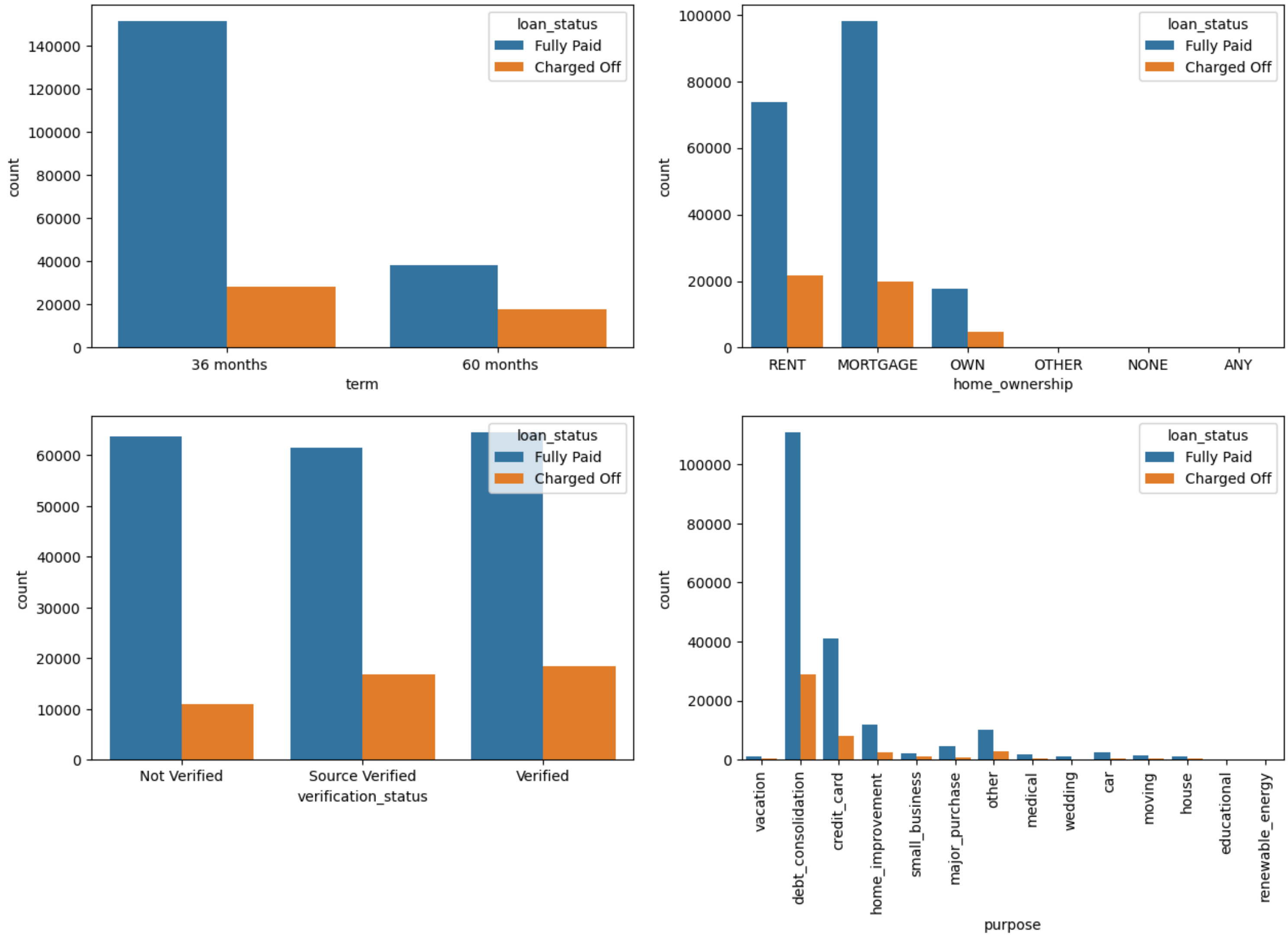


- All the application type is Individual
- Most of the loan tenure is disbursed for 36 months
- The grade of majority of people those who have took the loan is 'B' and have subgrade 'B3'.
- So from that we can infer that people with grade 'B' and subgrade 'B3' are more likely to fully pay the loan.

Bivariate Analysis

```
plt.figure(figsize=(15,20))

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

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

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

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

plt.show()
```

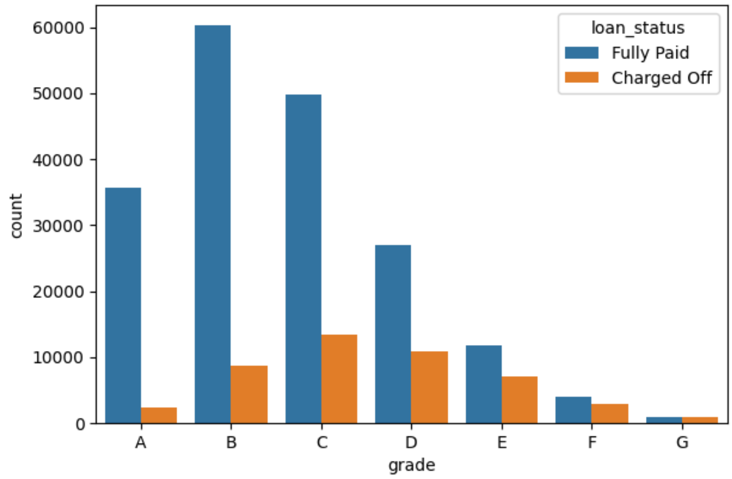
```
<ipython-input-17-cafc6a589ab6>:14: UserWarning: FixedFormatter should only be used together with FixedLocator
  g.set_xticklabels(g.get_xticklabels(),rotation=90)
```

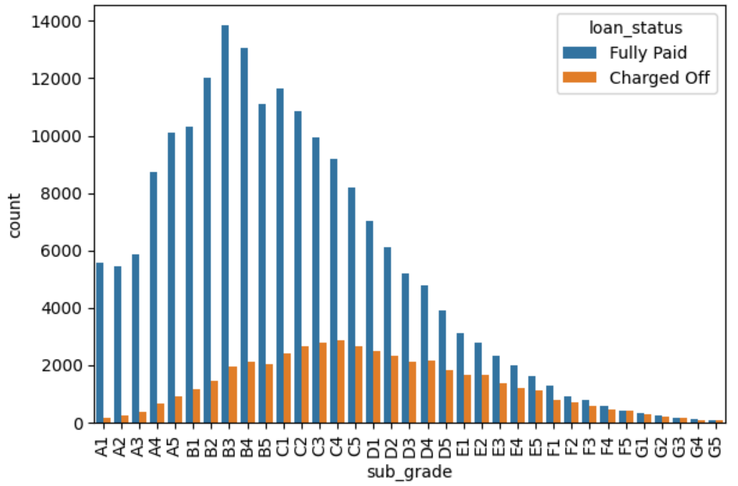


- Most of the people took loan term for 36 months and full paid on time
- Most of people have home ownership as mortgage and rent
- Most of the people took loan for debt consolidations

```
plt.figure(figsize=(15, 10))

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

plt.subplot(2, 2, 2)
sub_grade = sorted(df.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()
```

```
<ipython-input-18-4fc383fe14dc>:10: UserWarning: FixedFormatter should only be used together with FixedLocator
  g.set_xticklabels(g.get_xticklabels(), rotation=90)
```

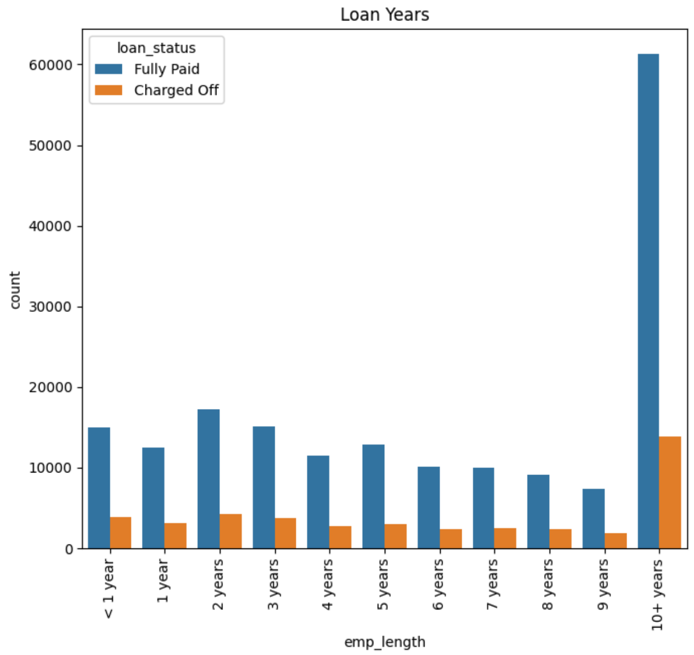


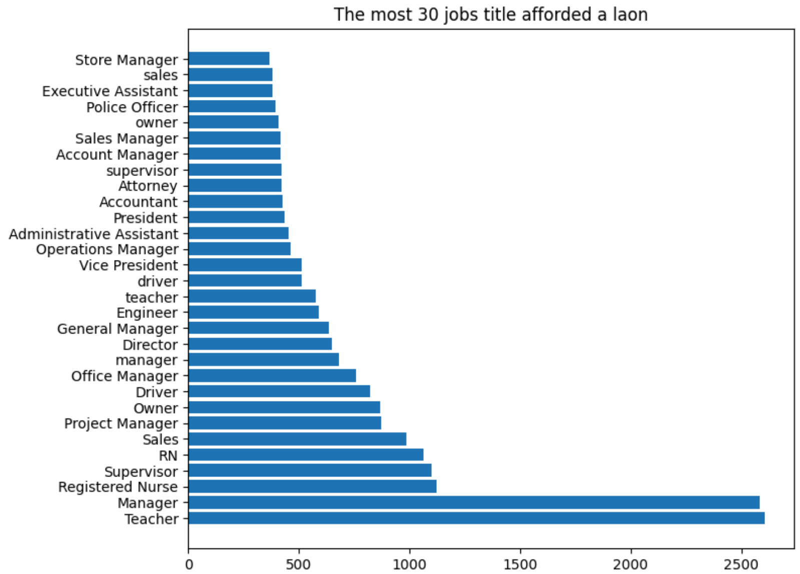


- The grade of majority of people those who have fully paid the loan is 'B' and have subgrade 'B3'.
- So from that we can infer that people with grade 'B' and subgrade 'B3' are more likely to fully pay the loan.

```
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.title("Loan Years")

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

plt.show()
```

```
<ipython-input-19-d54df4ff7f6f>:7: UserWarning: FixedFormatter should only be used together with FixedLocator
  g.set_xticklabels(g.get_xticklabels(),rotation=90)
```

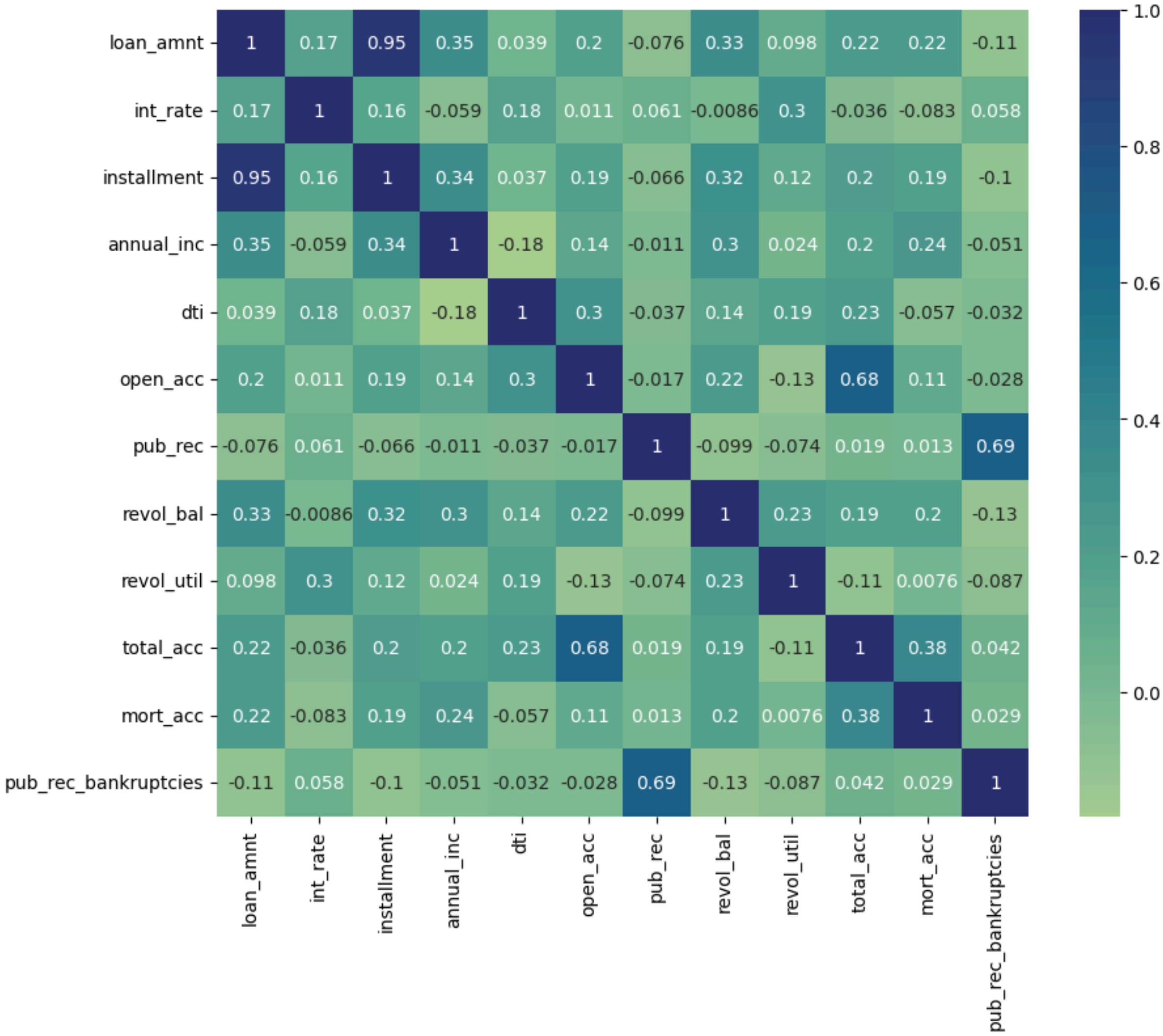




- Person who employed for more than 10 years has successfully paid of the loan
- Manager and Teacher are the most afforded loan on titles

Correlatio Analysis

```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(numeric_only=True), cmap = 'crest', annot = True)

plt.show()
```



- We can see that almost perfect correlation between "loan_amnt" the "installment" feature.
- installment: The monthly payment owed by the borrower if the loan originates.
- loan_amnt: The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.

```
#drop column Installment
#  df.drop(columns=['installment'],axis=1,inplace=True)
```

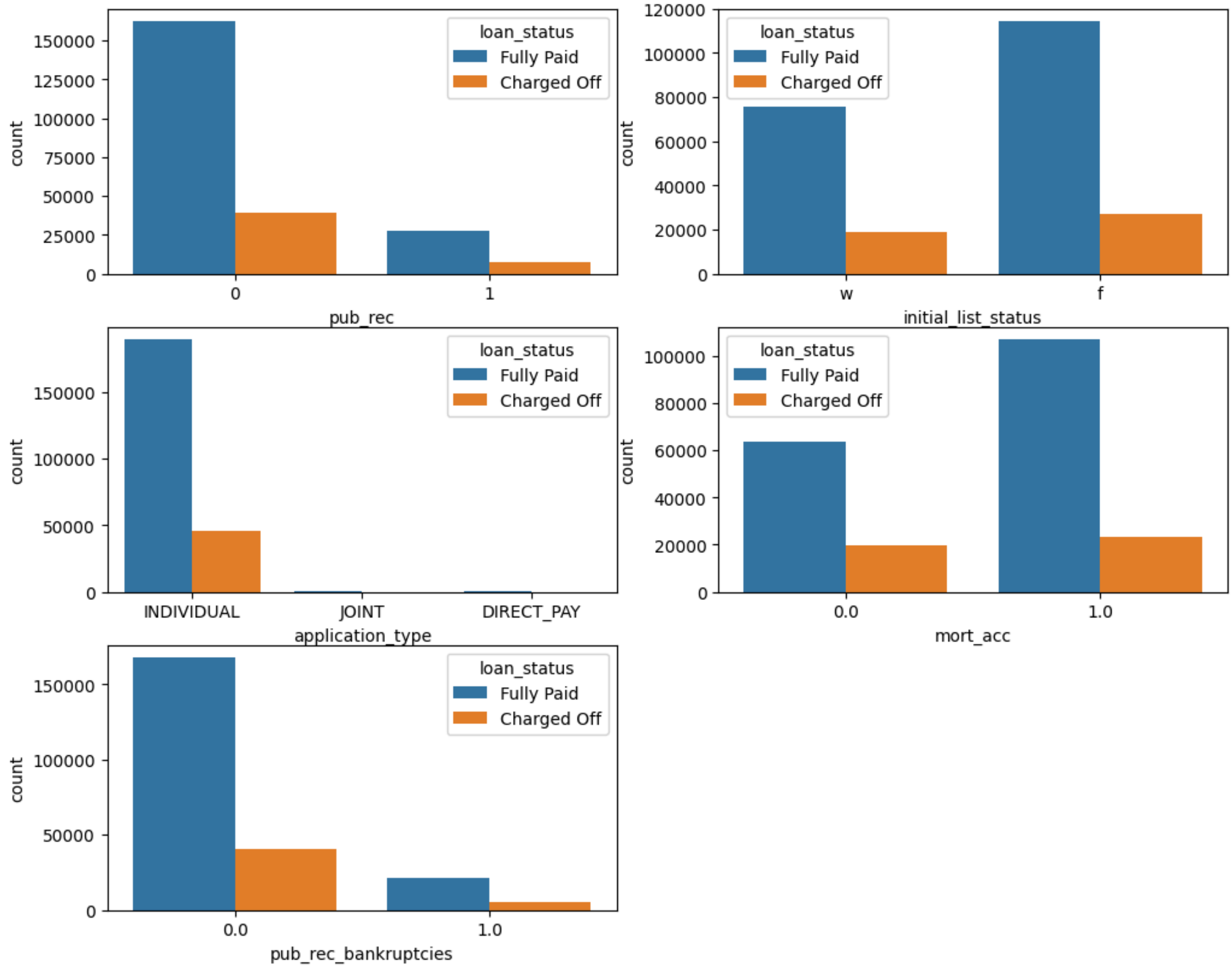
Feature Engineering

```
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
```

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

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

```
plt.figure(figsize=(12,20))

plt.subplot(6,2,1)
sns.countplot(x='pub_rec',data=df,hue='loan_status')

plt.subplot(6,2,2)
sns.countplot(x='initial_list_status',data=df,hue='loan_status')

plt.subplot(6,2,3)
sns.countplot(x='application_type',data=df,hue='loan_status')

plt.subplot(6,2,4)
sns.countplot(x='mort_acc',data=df,hue='loan_status')

plt.subplot(6,2,5)
sns.countplot(x='pub_rec_bankruptcies',data=df,hue='loan_status')

plt.show()
```



- As we can see that Most the loan disbursed to the people whose do not hold bankrupties record have successfully paid loan

```
df.describe()
```

	loan_amnt	int_rate	installment	annual_inc	issue_d	dti	earliest_cr_line	open_acc	pub_rec
count	235886.000000	235886.000000	235886.000000	2.358860e+05	235886	235886.000000	235886	235886.000000	235886.000000
mean	14104.732053	13.643355	431.524698	7.427901e+04	2014-02-02 12:26:02.544618752	17.325831	1998-04-30 06:31:22.122720512	11.306932	0.146579
min	500.000000	5.320000	16.250000	2.500000e+03	2007-06-01 00:00:00	0.000000	1944-01-01 00:00:00	0.000000	0.000000
25%	8000.000000	10.490000	250.330000	4.500000e+04	2013-05-01 00:00:00	11.260000	1994-10-01 00:00:00	8.000000	0.000000
50%	12000.000000	13.330000	375.370000	6.400000e+04	2014-04-01 00:00:00	16.880000	1999-09-01 00:00:00	10.000000	0.000000
75%	20000.000000	16.490000	567.010000	9.000000e+04	2015-03-01 00:00:00	22.960000	2003-04-01 00:00:00	14.000000	0.000000
max	40000.000000	30.990000	1533.810000	7.446395e+06	2016-12-01 00:00:00	189.900000	2013-10-01 00:00:00	90.000000	1.000000
std	8354.907949	4.467399	250.662467	6.006704e+04	NaN	8.130635	NaN	5.136844	0.353687

Default title text

```
# @title Default title text
```

```
df['total_acc'] = pd.to_numeric(df['total_acc'], errors='coerce')
```

```
df.head(5)
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_rec	revol_ba
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	0	36369.
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	0	20131.
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	0	11987.
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	6.0	0	5472.
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	0	24584.

5 rows × 27 columns

```
df.isnull().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               13677
emp_length              10938
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1038
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                171
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                22584
pub_rec_bankruptcies      329
address                     1
dtype: int64
```

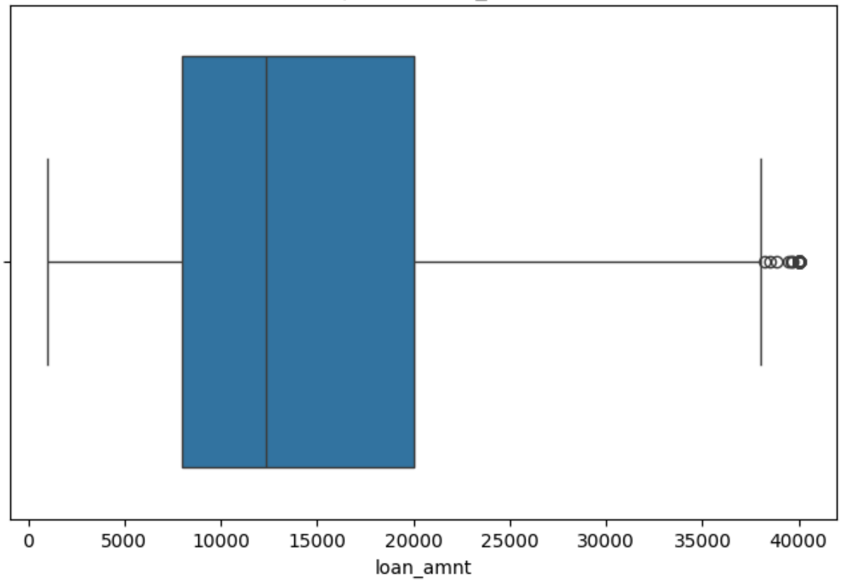
```
# Dropping rows with null values
df.dropna(inplace=True)
```

```
# Remaining no. of rows
df.shape
```

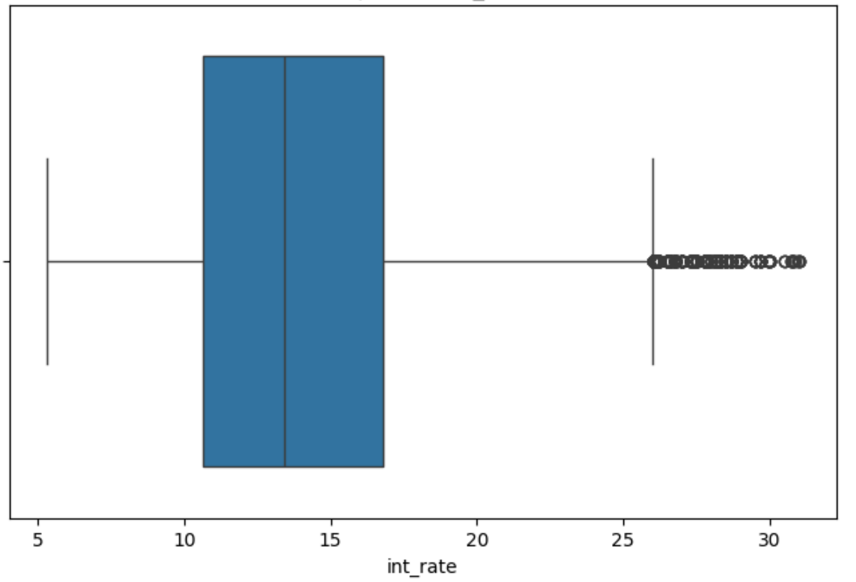
```
(199942, 27)
```

```
def box_plot(col):
    plt.figure(figsize=(8,5))
    sns.boxplot(x=df[col])
    plt.title('Boxplot for {}'.format(col))
    plt.show()

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

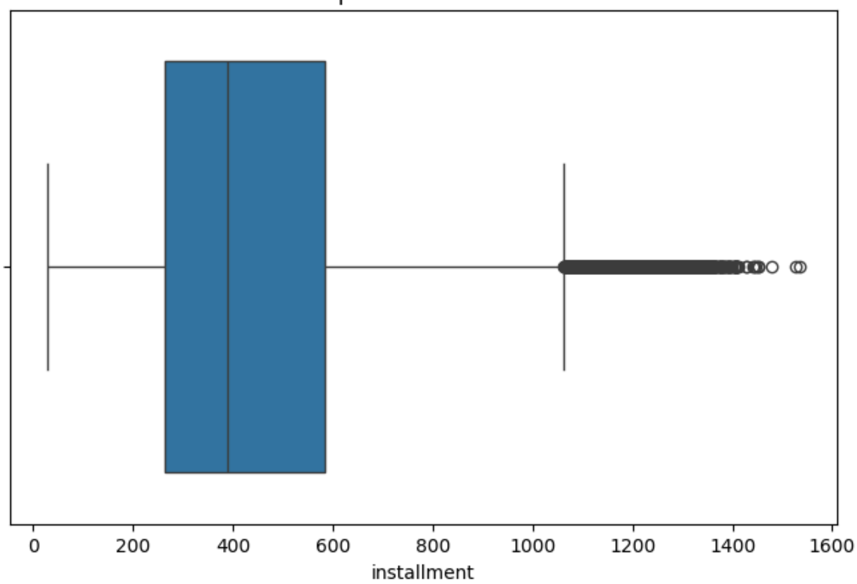
Boxplot for loan_amnt



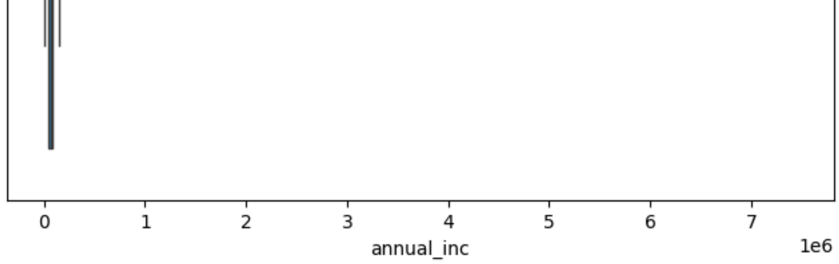
Boxplot for int_rate



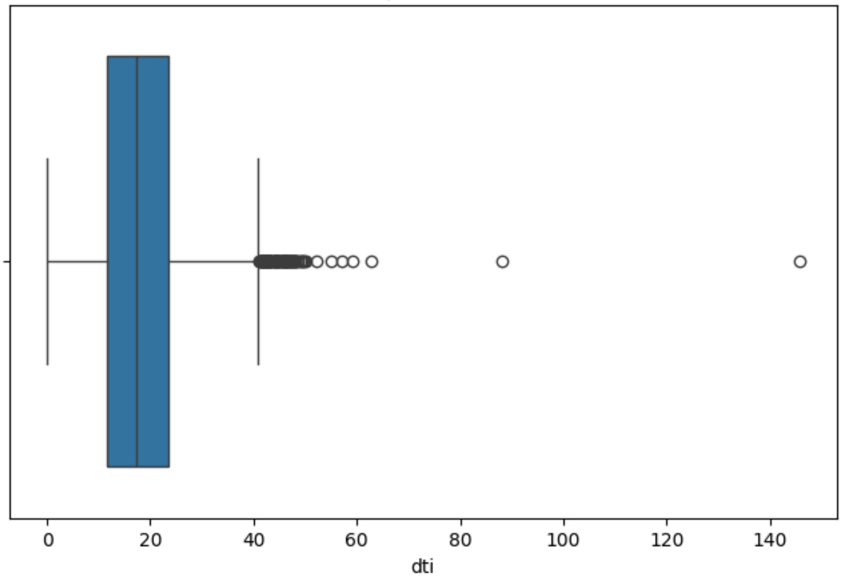
Boxplot for installment



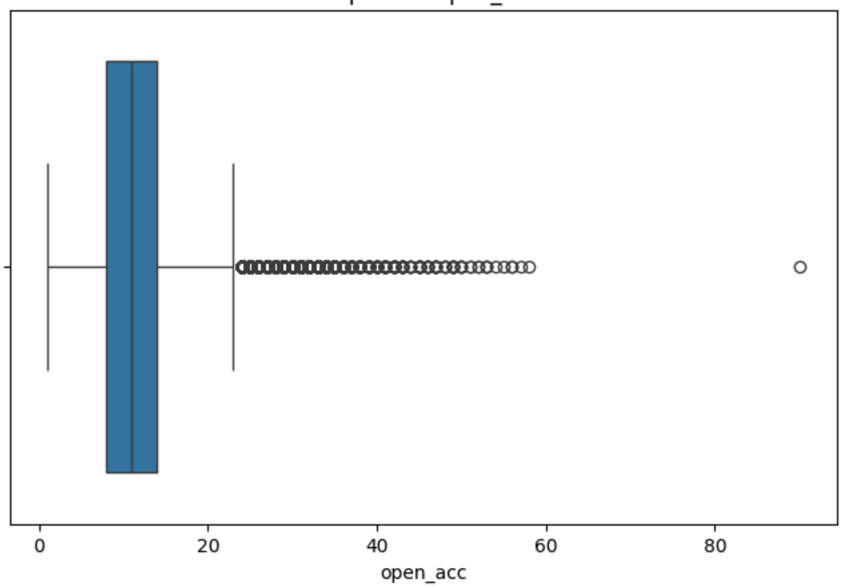
Boxplot for annual_inc



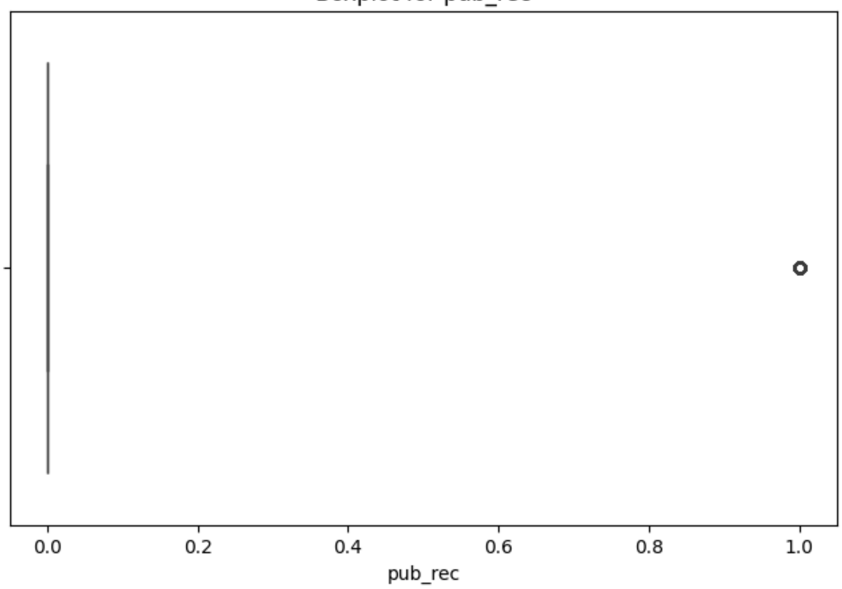
Boxplot for dti



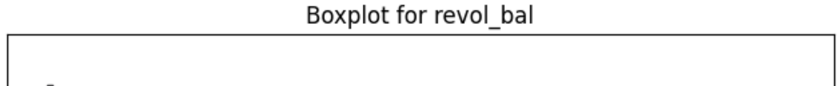
Boxplot for open_acc

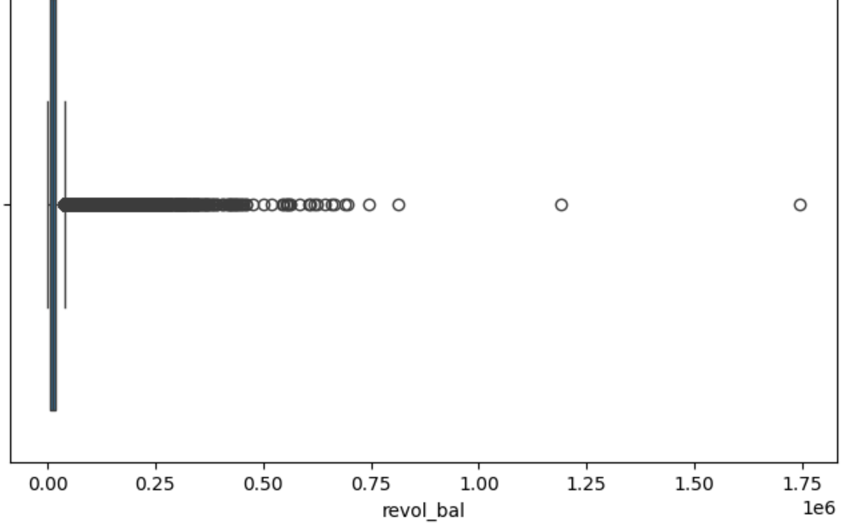


Boxplot for pub_rec

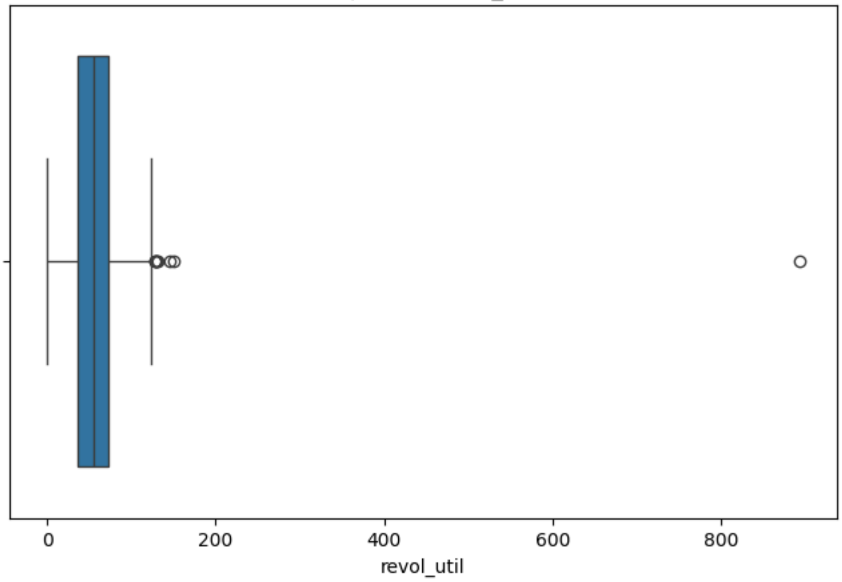


Boxplot for revol_bal

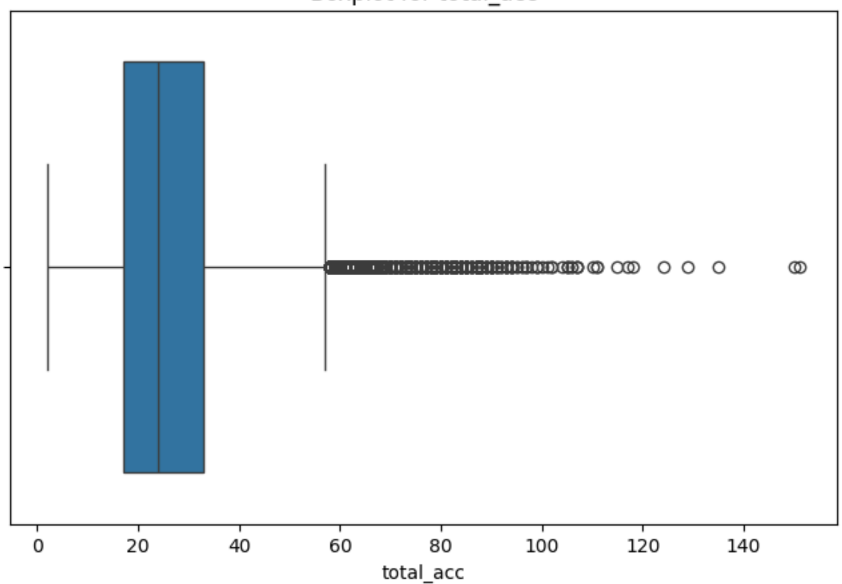




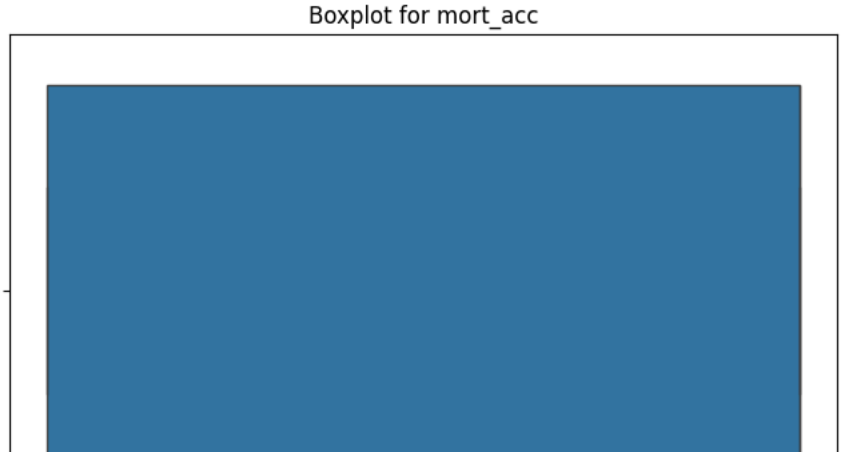
Boxplot for revol_util

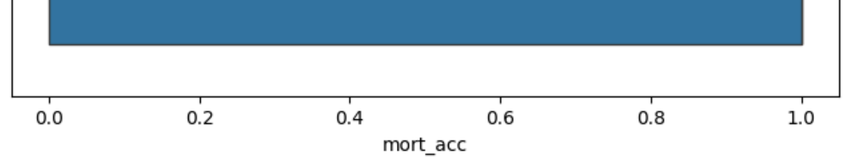


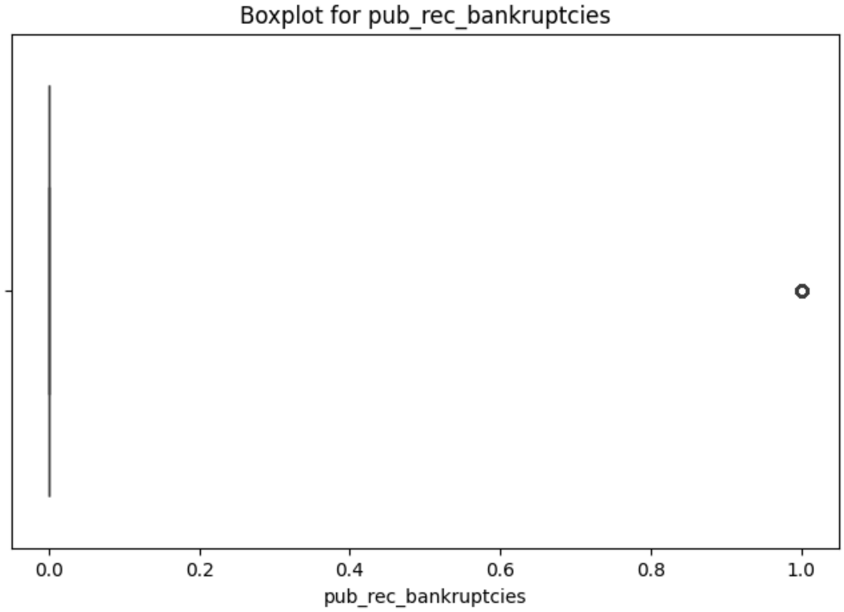
Boxplot for total_acc



Boxplot for mort_acc



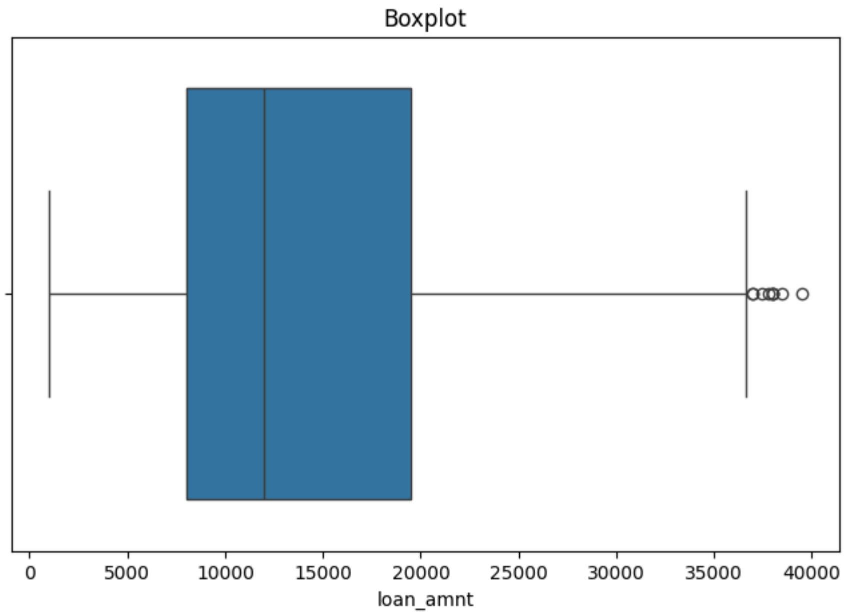


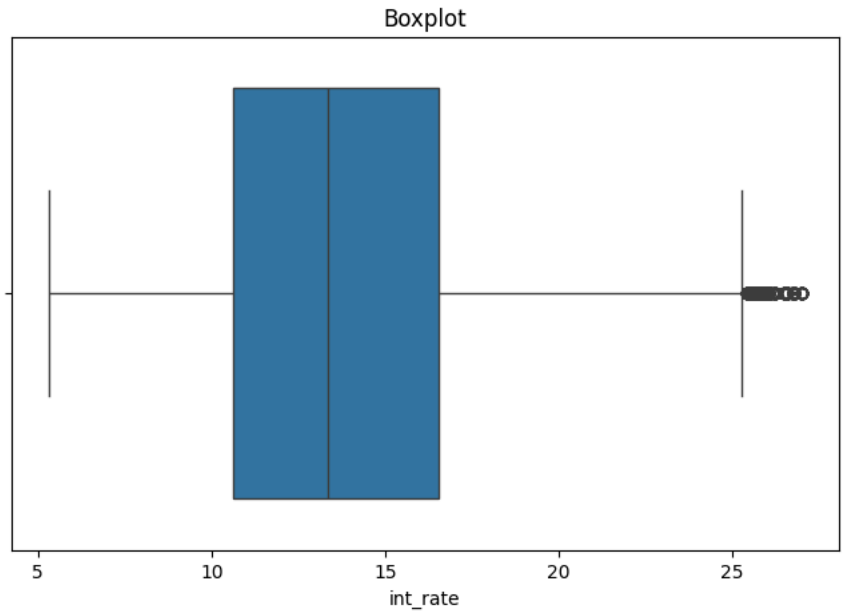


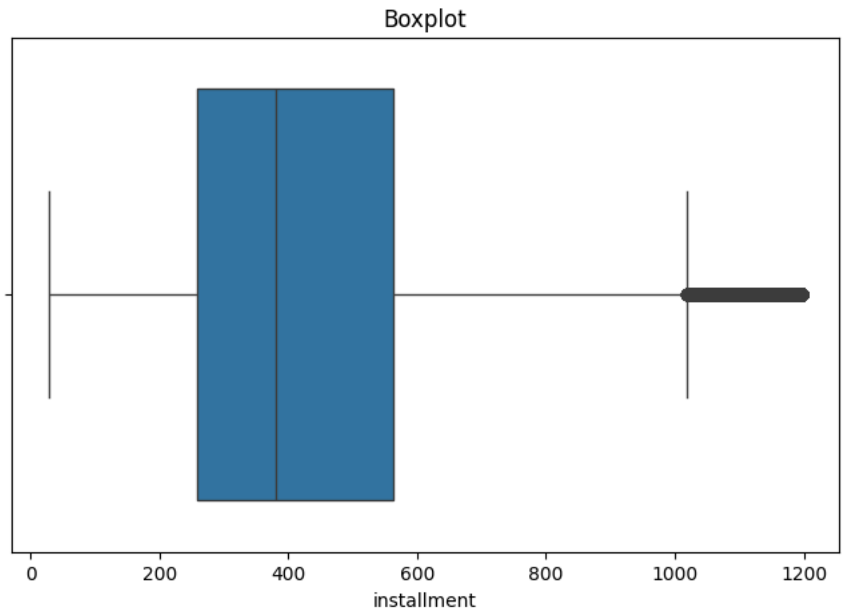

```
for col in num_vars:
    mean=df[col].mean()
    std=df[col].std()

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

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

df.shape
```

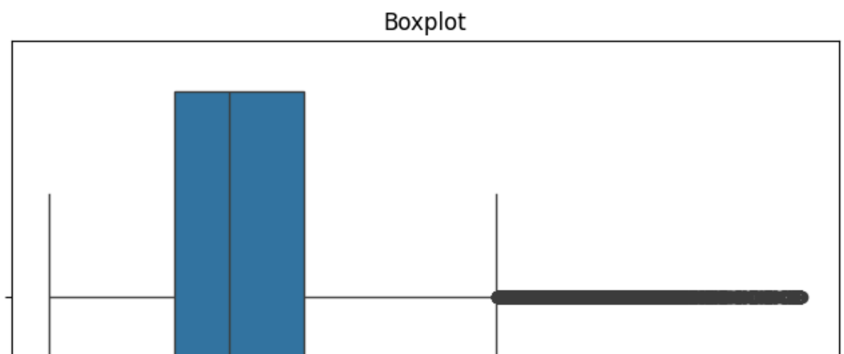
```
(189438, 27)
```

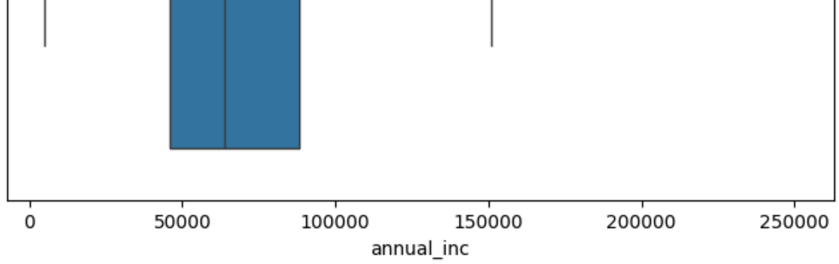
```
def box_plot(col):
    plt.figure(figsize=(8,5))
    sns.boxplot(x=df[col])
    plt.title('Boxplot')
    plt.show()

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

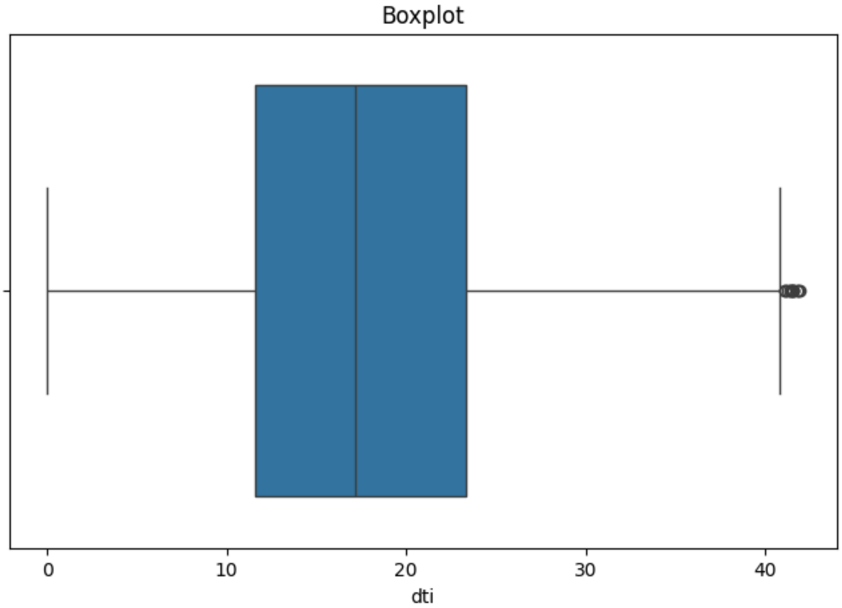

Boxplot
0
5000
10000
15000
20000
25000
30000
35000
40000
loan_amnt

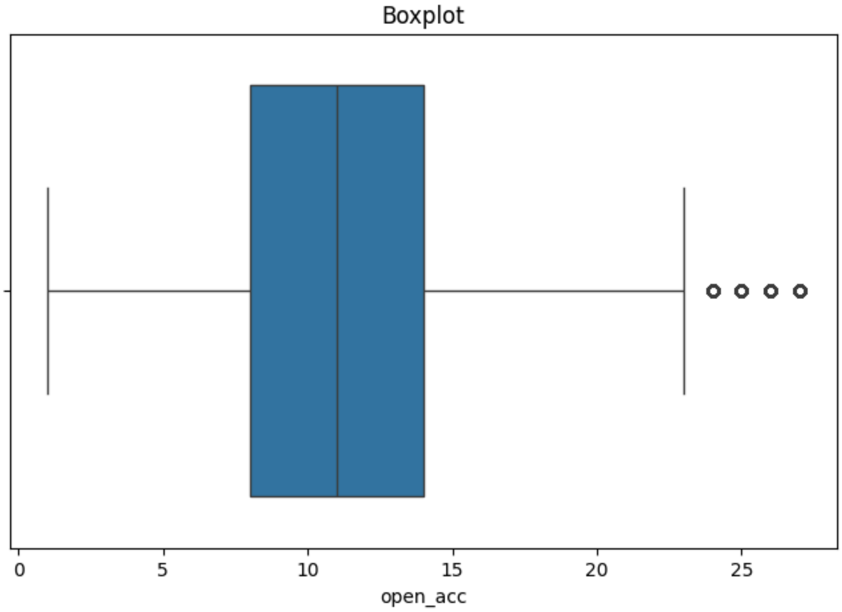

Boxplot
5
10
15
20
25
int_rate

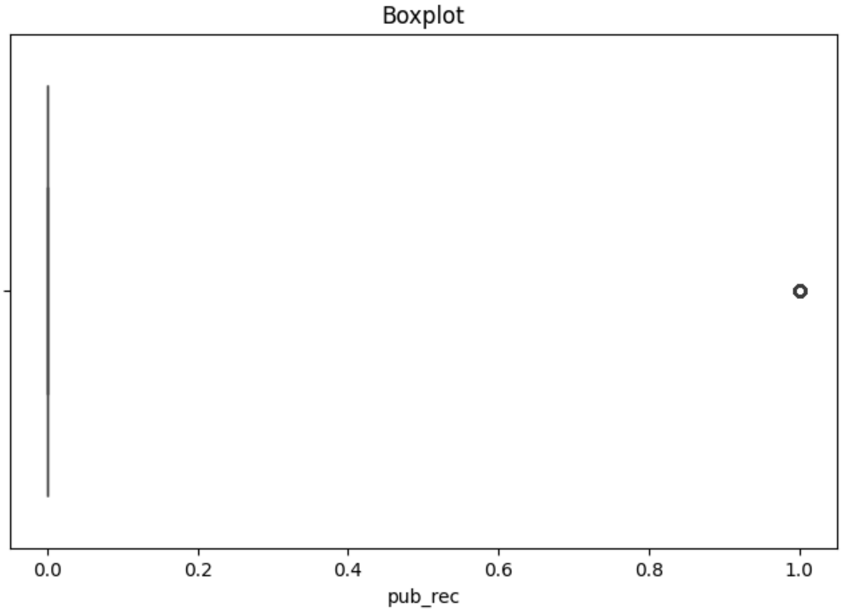

Boxplot
0
200
400
600
800
1000
1200
installment

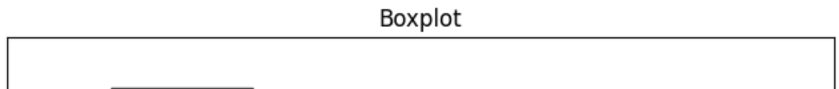

Boxplot

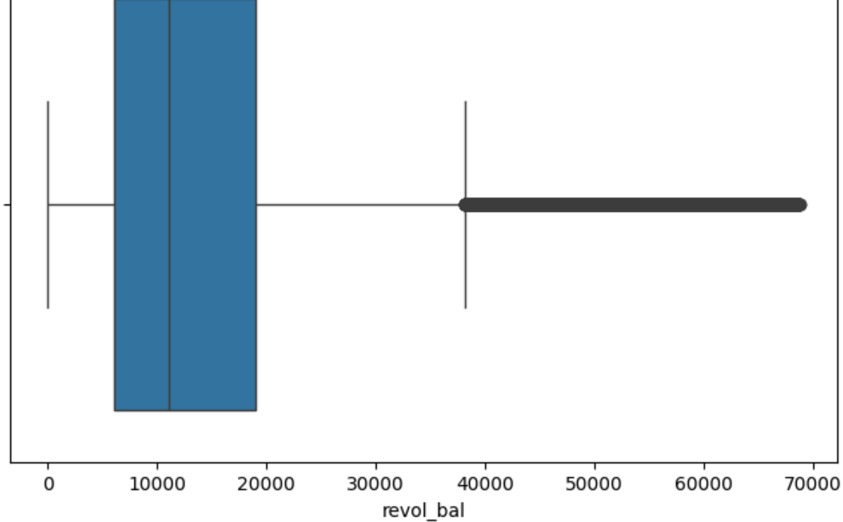

0
50000
100000
150000
200000
250000
annual_inc


Boxplot
0
10
20
30
40
dti

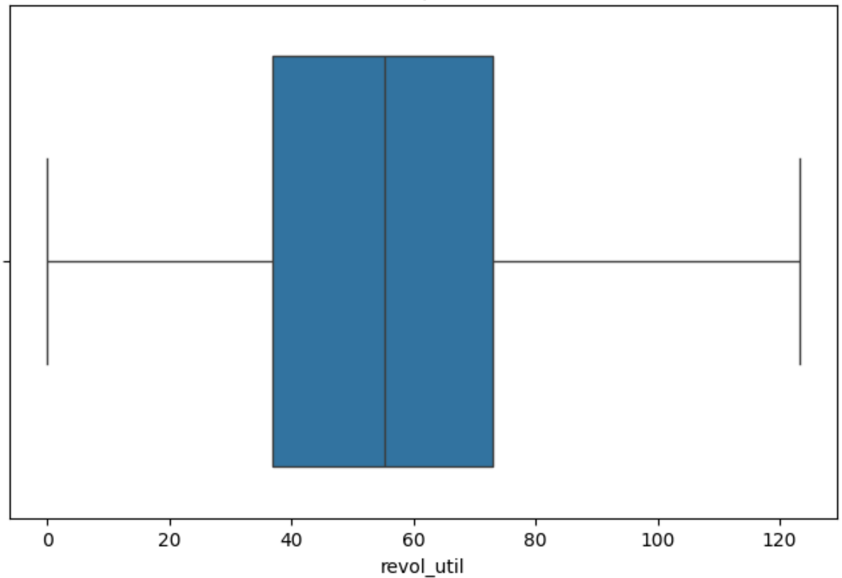

Boxplot
0
5
10
15
20
25
open_acc


Boxplot
0.0
0.2
0.4
0.6
0.8
1.0
pub_rec

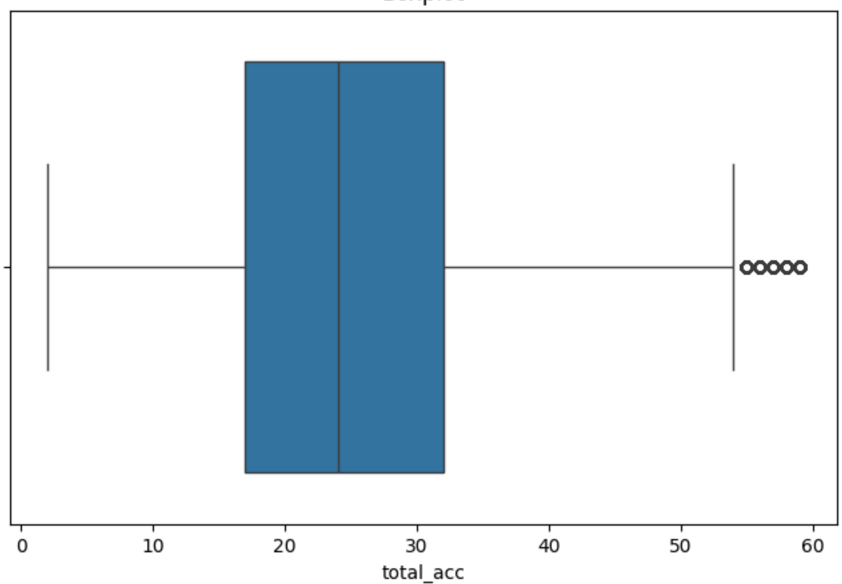

Boxplot



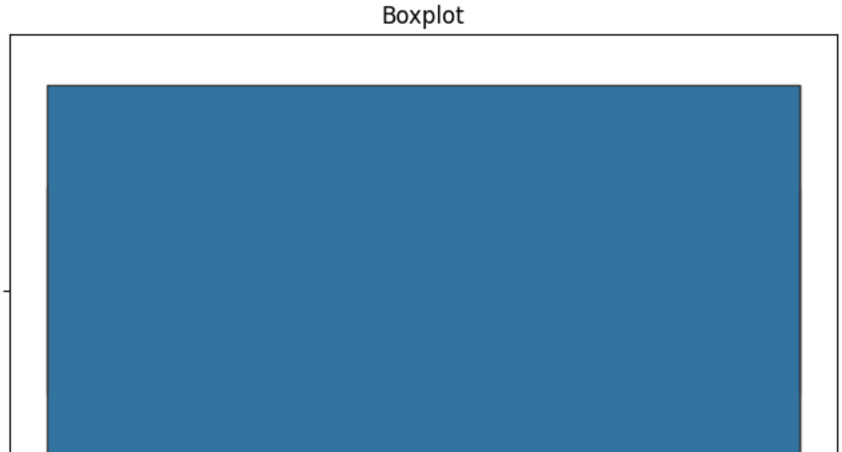
Boxplot

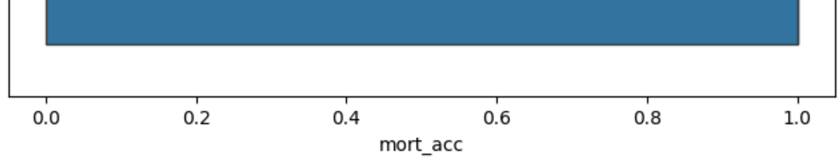


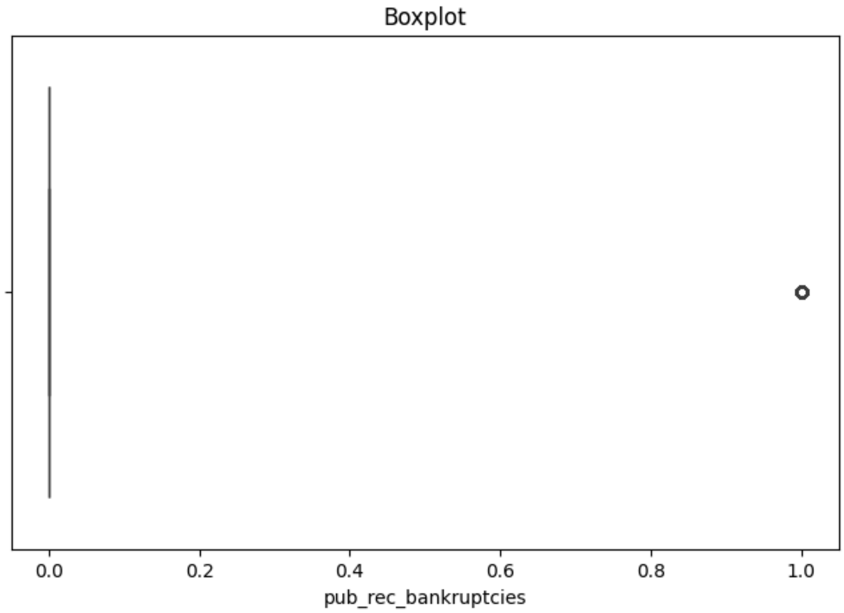
Boxplot



Boxplot








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

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

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

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

```
zip_code
70466    14.365650
30723    14.316557
22690    14.229986
48052    14.028864
29597    11.534117
05113    11.523559
00813    11.494526
11650     2.843147
93700     2.836812
86630     2.826782
Name: proportion, dtype: float64
```

```
# Dropping some variables which we can let go for now
df.drop(columns=['issue_d', 'emp_title', 'title', 'sub_grade',
                   'address', 'earliest_cr_line', 'emp_length'],
                   axis=1, inplace=True)
```

One hot encoding

```
dummies=['purpose', 'zip_code', 'grade', 'verification_status', 'application_type', 'home_ownership']
df=pd.get_dummies(df,columns=dummies,drop_first=True)
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)
```

Data processing for modelling

```
from sklearn.model_selection import train_test_split
```

```
X=df.drop('loan_status',axis=1)
y=df['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)
```

```
(132606, 50)
(56832, 50)
```

```
from sklearn.preprocessing import MinMaxScaler
```

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

```
X_train
```

```
array([[0.57104736, 0.        , 0.27226581, ..., 0.        , 0.        ,
        0.        ],
       [0.66711141, 1.        , 0.69173973, ..., 0.        , 0.        ,
        1.        ],
       [0.6490994 , 1.        , 0.11859714, ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.18412275, 0.        , 0.29303184, ..., 0.        , 0.        ,
        1.        ],
       [0.37358239, 0.        , 0.13244116, ..., 0.        , 1.        ,
        0.        ],
       [0.37358239, 0.        , 0.40470697, ..., 0.        , 1.        ,
        0.        ]])
```

```
X_test
```

```
array([[0.18679119, 0.        , 0.28749423, ..., 0.        , 0.        ,
        0.        ],
       [0.4269513 , 0.        , 0.26165205, ..., 0.        , 1.        ,
        0.        ],
       [0.37358239, 0.        , 0.35394555, ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.72048032, 0.        , 0.58467928, ..., 0.        , 0.        ,
        0.        ],
       [0.17344897, 0.        , 0.26165205, ..., 0.        , 0.        ,
        0.        ],
       [0.40026684, 1.        , 0.60775265, ..., 0.        , 0.        ,
        0.        ]])
```

Model Building

```
from sklearn.linear_model import LogisticRegression
```

```
# Fit the LogisticRegression model on the preprocessed data
logreg=LogisticRegression(max_iter=1000)
logreg.fit(X_train,y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000)
```

```
pd.Series((zip(X.columns, logreg.coef_[0])))
```

```
0                    (loan_amnt, -0.04320443805806286)
1                           (term, 0.5588719144348618)
2                      (int_rate, -0.1909295408914164)
3                    (installment, 0.5846791039181174)
4                    (annual_inc, -1.0315790727485319)
5                             (dti, 1.0103079618003479)
6                       (open_acc, 0.7991834956794442)
7                       (pub_rec, 0.13750498220070798)
8                    (revol_bal, -0.39934958560523326)
9                    (revol_util, 0.48983719714913054)
10                    (total_acc, -0.6984359021795911)
11         (initial_list_status, 0.01798726724955526)
12                   (mort_acc, -0.049546237270576714)
13        (pub_rec_bankruptcies, -0.16603105956947073)
14            (purpose_credit_card, 0.286135759732737)
15    (purpose_debt_consolidation, 0.3398576323440972)
16     (purpose_home_improvement, 0.3906316452021177)
17              (purpose_house, 0.24215443894893757)
18        (purpose_major_purchase, 0.3948450037229394)
19            (purpose_medical, 0.47399190105679645)
20              (purpose_moving, 0.3709326989026361)
21               (purpose_other, 0.3017251333476671)
22      (purpose_renewable_energy, 0.3401120689222868)
23        (purpose_small_business, 0.7671526300924224)
24           (purpose_vacation, 0.37073665091031516)
25            (purpose_wedding, -0.5012600259510492)
26              (zip_code_05113, -2.811547183676483)
27              (zip_code_11650, 11.563392239598716)
28              (zip_code_22690, 4.3333578375091015)
```

```
29                 (zip_code_29597, -2.808033859868557)
30                  (zip_code_30723, 4.374741129605297)
31                  (zip_code_48052, 4.420405036331178)
32                 (zip_code_70466, 4.3831363125349405)
33                 (zip_code_86630, 11.547624020548648)
34                 (zip_code_93700, 11.578847363633892)
35                        (grade_B, 0.5495342892516659)
36                        (grade_C, 1.0653674979435734)
37                        (grade_D, 1.3950598653799136)
38                        (grade_E, 1.6288226893735742)
39                         (grade_F, 1.790221103052595)
40                        (grade_G, 1.9088402431029736)
41    (verification_status_Source Verified, 0.190195...
42    (verification_status_Verified, 0.0523628414920...
43    (application_type_INDIVIDUAL, 0.7916536514798871)
44        (application_type_JOINT, -0.1452821665339001)
45       (home_ownership_MORTGAGE, -0.2585992735960012)
46           (home_ownership_NONE, -0.2344243559761461)
47          (home_ownership_OTHER, 0.48659722930795735)
48           (home_ownership_OWN, -0.12966224859926567)
49          (home_ownership_RENT, 0.041645790966338914)
dtype: object
```

```
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: 0.890
```

```
!pip install scikit-plot
!pip install scikitplot
!pip install scikitplot.metrics
```

```
Requirement already satisfied: scikit-plot in /usr/local/lib/python3.10/dist-packages (0.3.7)
Requirement already satisfied: matplotlib>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (3.7.1)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.2.2)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.11.4)
Requirement already satisfied: joblib>=0.10 in /usr/local/lib/python3.10/dist-packages (from scikit-plot) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.4.5)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=1.4.0->scikit-plot) (2.8.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.18->scikit-plot) (3.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib>=1.4.0->scikit-plot) (1.
ERROR: Could not find a version that satisfies the requirement scikitplot (from versions: none)
ERROR: No matching distribution found for scikitplot
ERROR: Could not find a version that satisfies the requirement scikitplot.metrics (from versions: none)
ERROR: No matching distribution found for scikitplot.metrics
```

Classification Report

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.99      0.94     45653
           1       0.93      0.47      0.63     11179

    accuracy                           0.89     56832
   macro avg       0.91      0.73      0.78     56832
weighted avg       0.89      0.89      0.87     56832
```

- Precision score and recall score for full paid status is almost same indicates that model is doing decent job which correctly classified the both of the scenarios
- Precision score for charged off status is more than recall score which is perfect

ROC / AUC

```
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--')
```