```python
import pandas as pd
import numpy as np
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
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
```

```python
data = pd.read_csv('logistic_regression.csv')
data.columns
```

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

```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object
 5   sub_grade             396030 non-null  object
 6   emp_title             373103 non-null  object
 7   emp_length            377729 non-null  object
 8   home_ownership        396030 non-null  object
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object
 11  issue_d               396030 non-null  object
```

```
 12  loan_status           396030 non-null  object
 13  purpose               396030 non-null  object
 14  title                 394275 non-null  object
 15  dti                   396030 non-null  float64
 16  earliest_cr_line      396030 non-null  object
 17  open_acc              396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object
 23  application_type      396030 non-null  object
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

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

data.isnull().sum()/len(data)*100
```

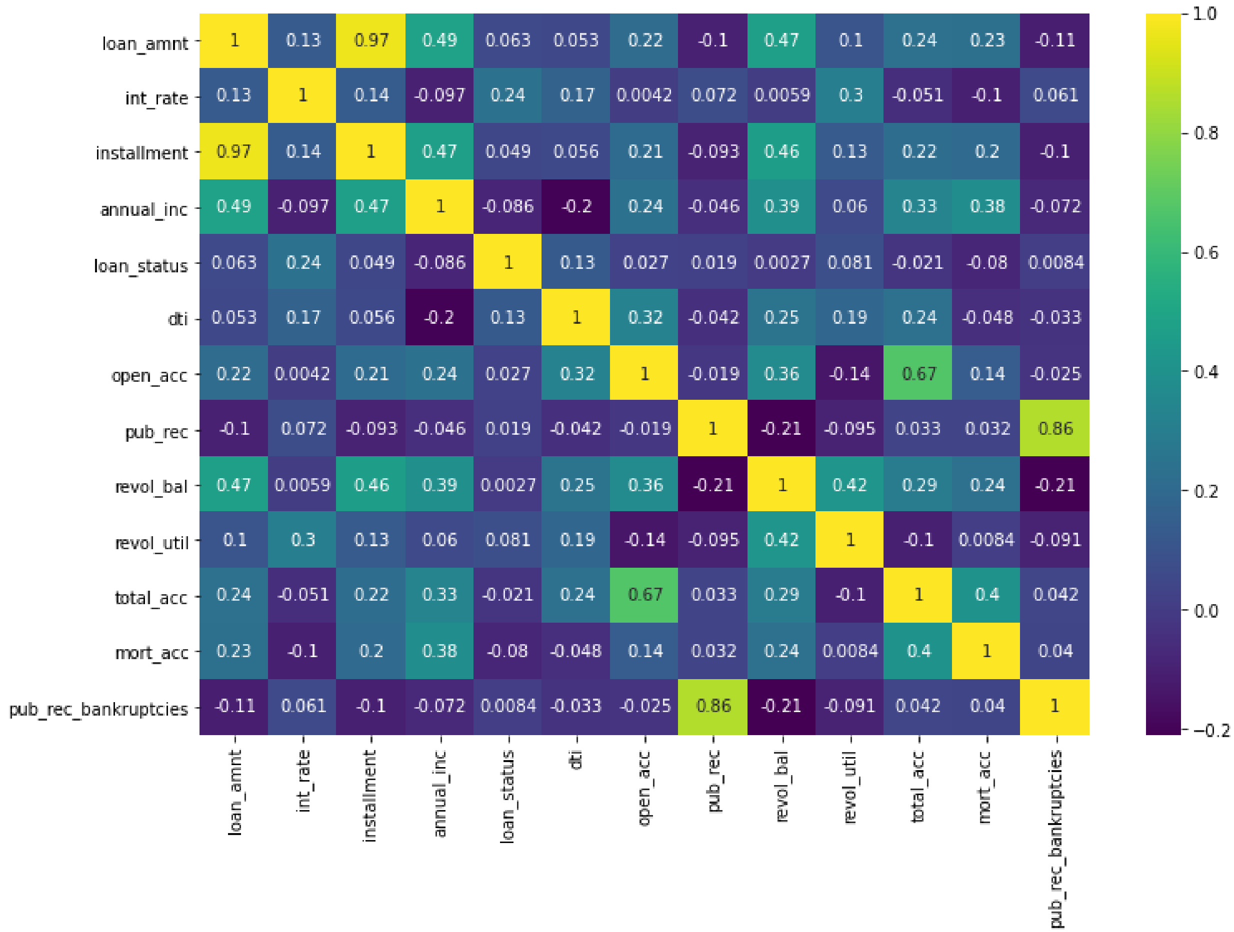
```
loan_amnt               0.000000
term                    0.000000
int_rate                0.000000
installment             0.000000
grade                   0.000000
sub_grade               0.000000
emp_title               5.789208
emp_length              4.621115
home_ownership          0.000000
annual_inc              0.000000
verification_status     0.000000
issue_d                 0.000000
loan_status             0.000000
purpose                 0.000000
title                   0.443148
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

# Checking the distribution of outcome labels -
data.loan_status.value_counts(normalize=True)*100

0    80.387092
1    19.612908
Name: loan_status, dtype: float64

#the data seems imbalanced we will be needing to oversample it later

# Statistical summary of the dataset -
data.describe(include='all')

            loan_amnt        term       int_rate    installment
grade  \
count   396030.000000      396030  396030.000000  396030.000000
396030
unique            NaN           2            NaN            NaN
7
top               NaN   36 months            NaN            NaN
B
freq              NaN      302005            NaN            NaN
116018
mean     14113.888089         NaN      13.639400     431.849698
NaN
std       8357.441341         NaN       4.472157     250.727790
NaN
min        500.000000         NaN       5.320000      16.080000
NaN
25%       8000.000000         NaN      10.490000     250.330000
NaN
50%      12000.000000         NaN      13.330000     375.430000
NaN
75%      20000.000000         NaN      16.490000     567.300000
NaN
max      40000.000000         NaN      30.990000    1533.810000
NaN

       sub_grade emp_title emp_length home_ownership
annual_inc  ...  \
count     396030    373103     377729         396030
3.960300e+05  ...
unique        35    173105         11              6
NaN  ...
top           B3   Teacher  10+ years       MORTGAGE
NaN  ...
freq       26655      4389     126041         198348
NaN  ...
```

```
mean          NaN       NaN        NaN            NaN
7.420318e+04  ...
std           NaN       NaN        NaN            NaN
6.163762e+04  ...
min           NaN       NaN        NaN            NaN
0.000000e+00  ...
25%           NaN       NaN        NaN            NaN
4.500000e+04  ...
50%           NaN       NaN        NaN            NaN
6.400000e+04  ...
75%           NaN       NaN        NaN            NaN
9.000000e+04  ...
max           NaN       NaN        NaN            NaN
8.706582e+06  ...

             open_acc        pub_rec     revol_bal     revol_util  \
count   396030.000000  396030.000000  3.960300e+05  395754.000000
unique            NaN            NaN           NaN            NaN
top               NaN            NaN           NaN            NaN
freq              NaN            NaN           NaN            NaN
mean        11.311153       0.178191  1.584454e+04      53.791749
std          5.137649       0.530671  2.059184e+04      24.452193
min          0.000000       0.000000  0.000000e+00       0.000000
25%          8.000000       0.000000  6.025000e+03      35.800000
50%         10.000000       0.000000  1.118100e+04      54.800000
75%         14.000000       0.000000  1.962000e+04      72.900000
max         90.000000      86.000000  1.743266e+06     892.300000

            total_acc  initial_list_status application_type
mort_acc  \
count   396030.000000               396030           396030
358235.000000
unique            NaN                    2                3
NaN
top               NaN                    f       INDIVIDUAL
NaN
freq              NaN               238066           395319
NaN
mean        25.414744                  NaN              NaN
1.813991
std         11.886991                  NaN              NaN
2.147930
min          2.000000                  NaN              NaN
0.000000
25%         17.000000                  NaN              NaN
0.000000
50%         24.000000                  NaN              NaN
1.000000
75%         32.000000                  NaN              NaN
```

```
3.000000
max         151.000000                  NaN              NaN
34.000000

        pub_rec_bankruptcies                    address
count            395495.000000                   396030
unique                     NaN                   393700
top                        NaN  USS Smith\r\nFPO AP 70466
freq                       NaN                        8
mean                  0.121648                      NaN
std                   0.356174                      NaN
min                   0.000000                      NaN
25%                   0.000000                      NaN
50%                   0.000000                      NaN
75%                   0.000000                      NaN
max                   8.000000                      NaN

[11 rows x 27 columns]
```

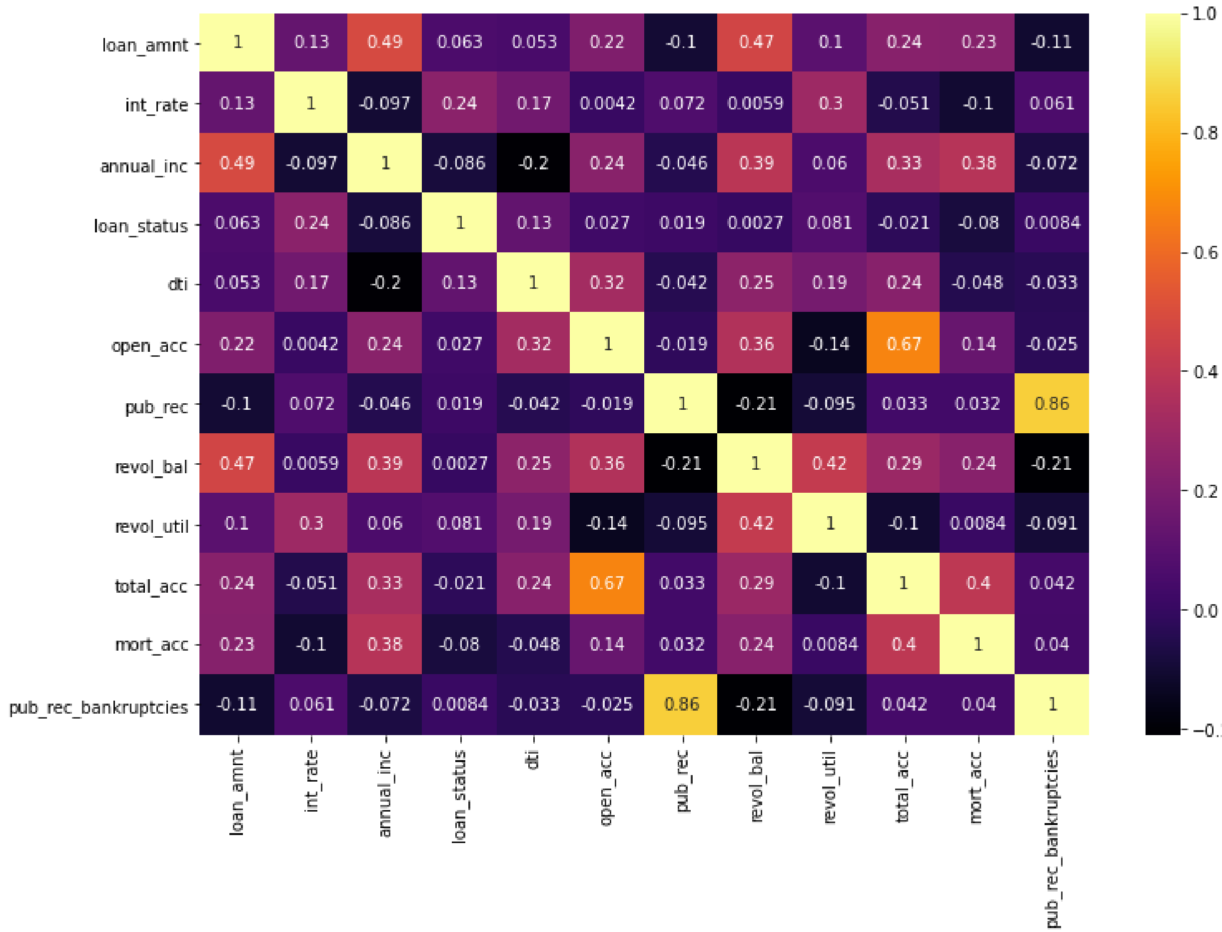
```
plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(method='spearman'), annot=True, cmap='viridis')
plt.show()
```



#Comment about the correlation between Loan Amount and Installment features.

We noticed 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. So, we can drop either one of those columns.

```
data.drop(columns=['installment'], axis=1, inplace=True)

plt.figure(figsize=(12, 8))
sns.heatmap(data.corr(method='spearman'), annot=True, cmap='inferno')
plt.show()
```



```
data.loan_status.value_counts()

0    318357
1     77673
Name: loan_status, dtype: int64
```

Data Exploration -

1. The no of people those who have fully paid are 318357 and that of Charged Off are 77673.

```
data.groupby(by='loan_status')['loan_amnt'].describe()

                count          mean          std     min     25%
50%  \
loan_status

0            318357.0  13866.878771  8302.319699   500.0  7500.0
12000.0
1             77673.0  15126.300967  8505.090557  1000.0  8525.0
14000.0
```

```
                  75%      max
loan_status
0            19225.0  40000.0
1            20000.0  40000.0

#2. The majority of people have home ownership as Mortgage and Rent
data['home_ownership'].value_counts()

MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          112
NONE            31
ANY              3
Name: home_ownership, dtype: int64

 #Combining the minority classes as 'OTHER'.
data.loc[(data.home_ownership == 'ANY') | (data.home_ownership ==
'NONE'), 'home_ownership'] = 'OTHER'
data.home_ownership.value_counts()

MORTGAGE    198348
RENT        159790
OWN          37746
OTHER          146
Name: home_ownership, dtype: int64

# Checking the distribution of 'Other' -
data.loc[data['home_ownership']=='OTHER',
'loan_status'].value_counts()

0    123
1     23
Name: loan_status, dtype: int64

data['issue_d'] = pd.to_datetime(data['issue_d'])
data['earliest_cr_line'] = pd.to_datetime(data['earliest_cr_line'])

#Saw some issues in title (Looks like it was filled manually and needs
some fixing).
data['title'].value_counts()[:20]

Debt consolidation           152472
Credit card refinancing       51487
Home improvement              15264
Other                         12930
Debt Consolidation            11608
Major purchase                 4769
Consolidation                  3852
debt consolidation             3547
Business                       2949
```

```
Debt Consolidation Loan          2864
Medical expenses                 2742
Car financing                    2139
Credit Card Consolidation        1775
Vacation                         1717
Moving and relocation            1689
consolidation                    1595
Personal Loan                    1591
Consolidation Loan               1299
Home Improvement                 1268
Home buying                      1183
Name: title, dtype: int64
```

```
data['title'] = data.title.str.lower()
```

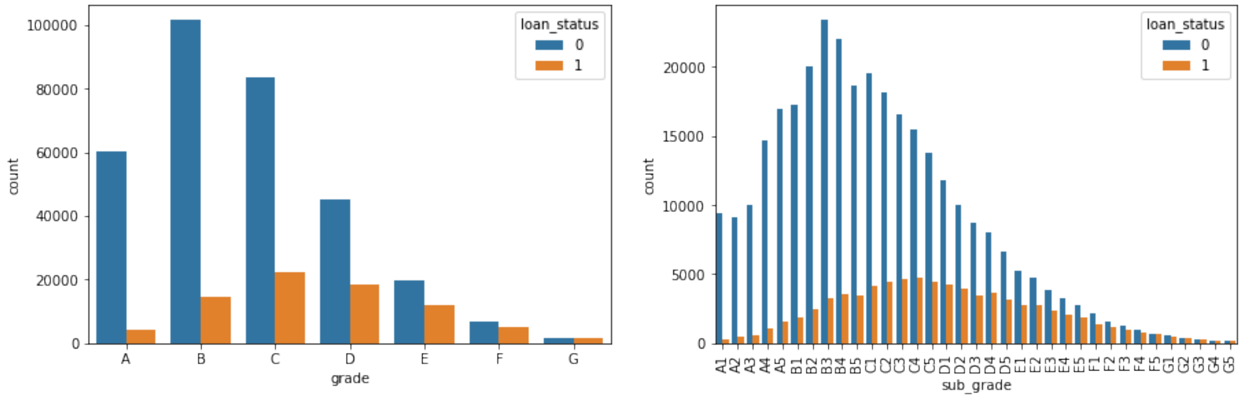
```
data.title.value_counts()[:10]
```

```
debt consolidation             168108
credit card refinancing         51781
home improvement                17117
other                           12993
consolidation                    5583
major purchase                   4998
debt consolidation loan          3513
business                         3017
medical expenses                 2820
credit card consolidation        2638
Name: title, dtype: int64
```

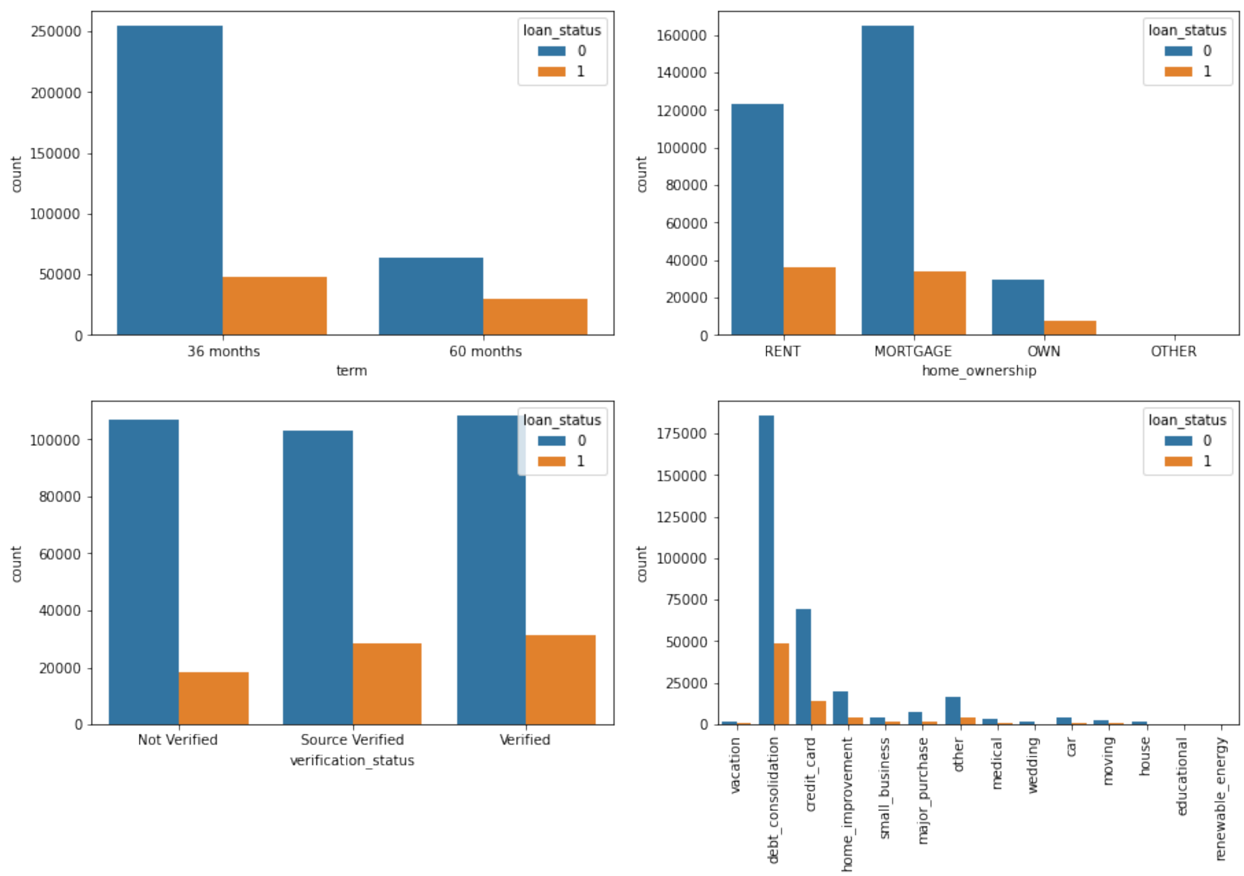
Visualization - The grade of majority of people those who have fully paid the loan is 'B' and have subgrade 'B3'. So from where we can infer that people with grade 'B' and subgrade 'B3' are more likely to fully pay the loan.

People with grades ‘A’ are more likely to fully pay their loan. (T/F) so this is false

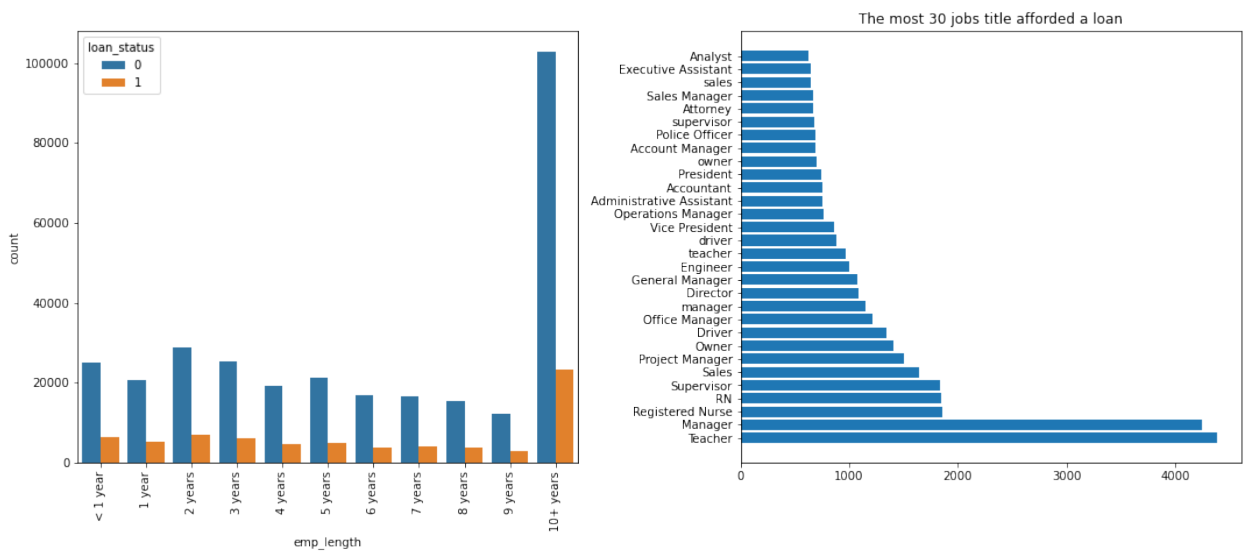
```
plt.figure(figsize=(15, 10))
plt.subplot(2, 2, 1)
grade = sorted(data.grade.unique().tolist())
sns.countplot(x='grade', data=data, hue='loan_status', order=grade)
plt.subplot(2, 2, 2)
sub_grade = sorted(data.sub_grade.unique().tolist())
g = sns.countplot(x='sub_grade', data=data, hue='loan_status',
order=sub_grade)
g.set_xticklabels(g.get_xticklabels(), rotation=90);
```




```
plt.figure(figsize=(15, 20))
plt.subplot(4, 2, 1)
sns.countplot(x='term', data=data, hue='loan_status')
plt.subplot(4, 2, 2)
sns.countplot(x='home_ownership', data=data, hue='loan_status')
plt.subplot(4, 2, 3)
sns.countplot(x='verification_status', data=data, hue='loan_status')
plt.subplot(4, 2, 4)
g = sns.countplot(x='purpose', data=data, hue='loan_status')
g.set_xticklabels(g.get_xticklabels(), rotation=90);
```

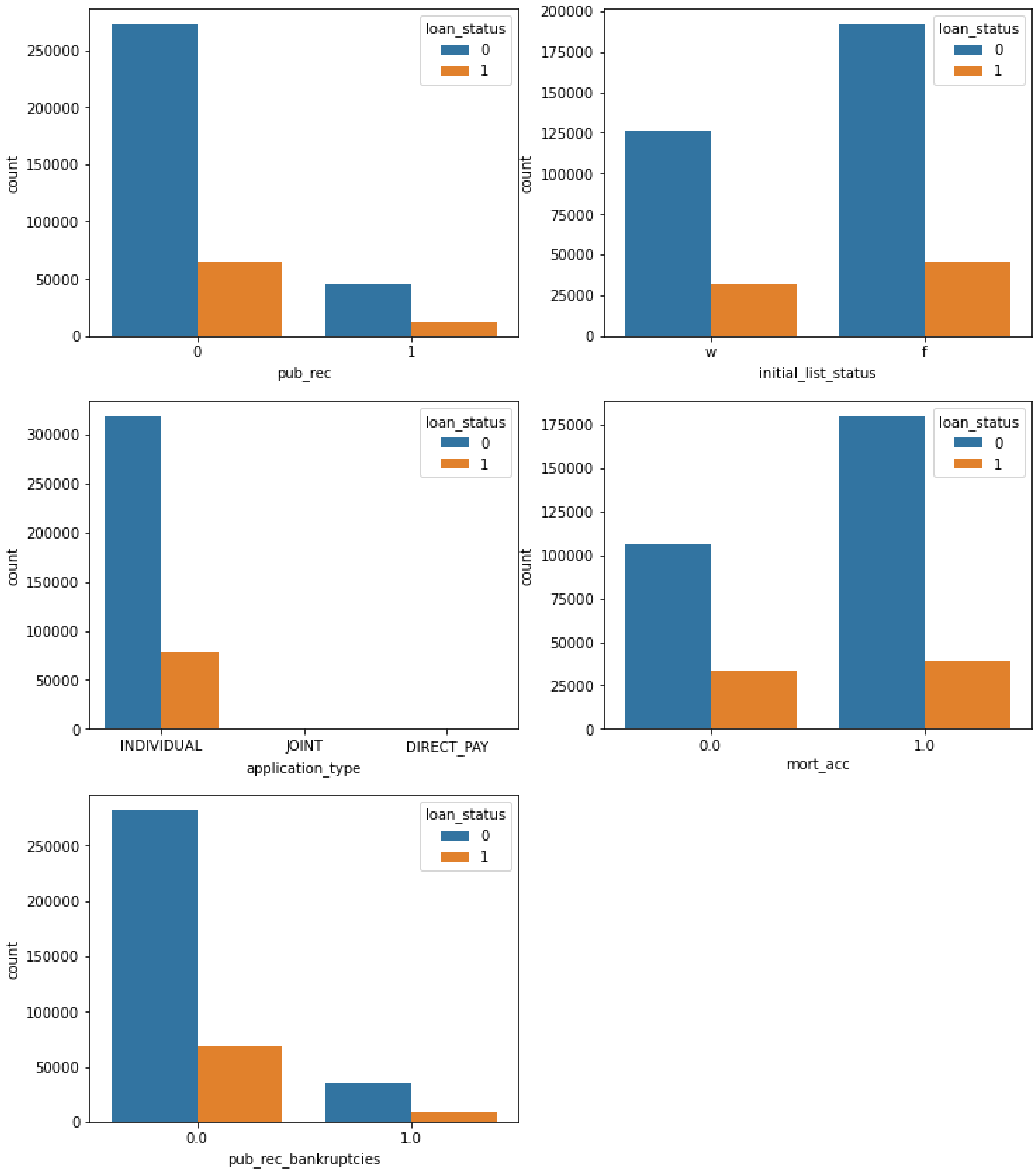


```
#Name the top 2 afforded job titles.
#Manager and Teacher are the most afforded loan job titles

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=data, hue='loan_status', order=order)
g.set_xticklabels(g.get_xticklabels(), rotation=90);
plt.subplot(2, 2, 2)
plt.barh(data.emp_title.value_counts()[:30].index, data.emp_title.value_counts()[:30])
plt.title("The most 30 jobs title afforded a loan")
plt.tight_layout()
```




```
#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
```

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

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

plt.figure(figsize=(12, 30))
plt.subplot(6, 2, 1)
sns.countplot(x='pub_rec', data=data, hue='loan_status')
plt.subplot(6, 2, 2)
sns.countplot(x='initial_list_status', data=data, hue='loan_status')
plt.subplot(6, 2, 3)
sns.countplot(x='application_type', data=data, hue='loan_status')
plt.subplot(6, 2, 4)
sns.countplot(x='mort_acc', data=data, hue='loan_status')
plt.subplot(6, 2, 5)
sns.countplot(x='pub_rec_bankruptcies', data=data, hue='loan_status')
plt.show()
```




```
#lets impute the null value of mort_acc with the help of total_acc
data.groupby(by='total_acc')['mort_acc'].median()

total_acc
2.0       0.0
3.0       0.0
4.0       0.0
```

```
5.0      0.0
6.0      0.0
        ...
124.0    1.0
129.0    1.0
135.0    1.0
150.0    1.0
151.0    0.0
Name: mort_acc, Length: 118, dtype: float64
```

```
total_acc_avg = data.groupby(by='total_acc').median().mort_acc
```

```
total_acc_avg
```

```
total_acc
2.0      0.0
3.0      0.0
4.0      0.0
5.0      0.0
6.0      0.0
        ...
124.0    1.0
129.0    1.0
135.0    1.0
150.0    1.0
151.0    0.0
Name: mort_acc, Length: 118, dtype: float64
```

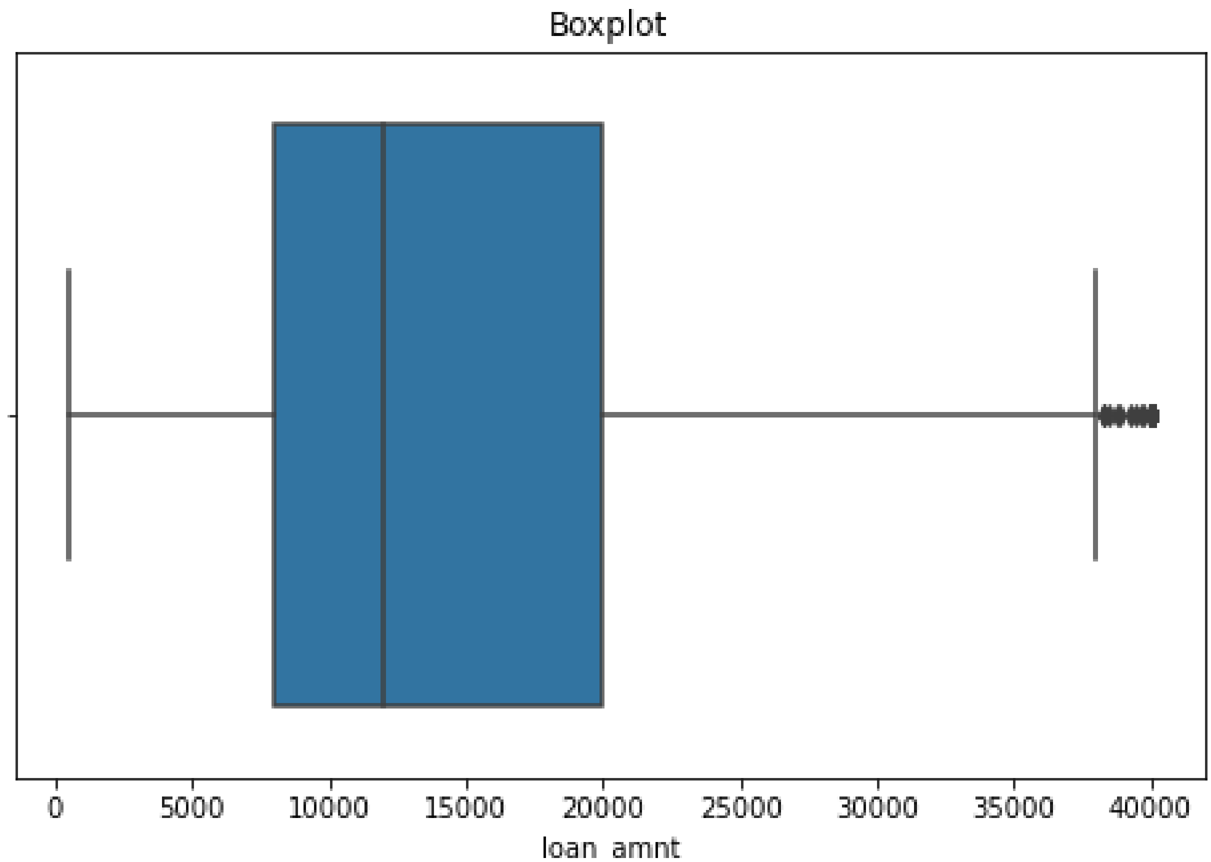
```
def fill_mort_acc(x):
    total_acc = x['total_acc']
    mort_acc = x['mort_acc']

    if np.isnan(mort_acc):
        return total_acc_avg[total_acc].round()
    else:
        return mort_acc
```

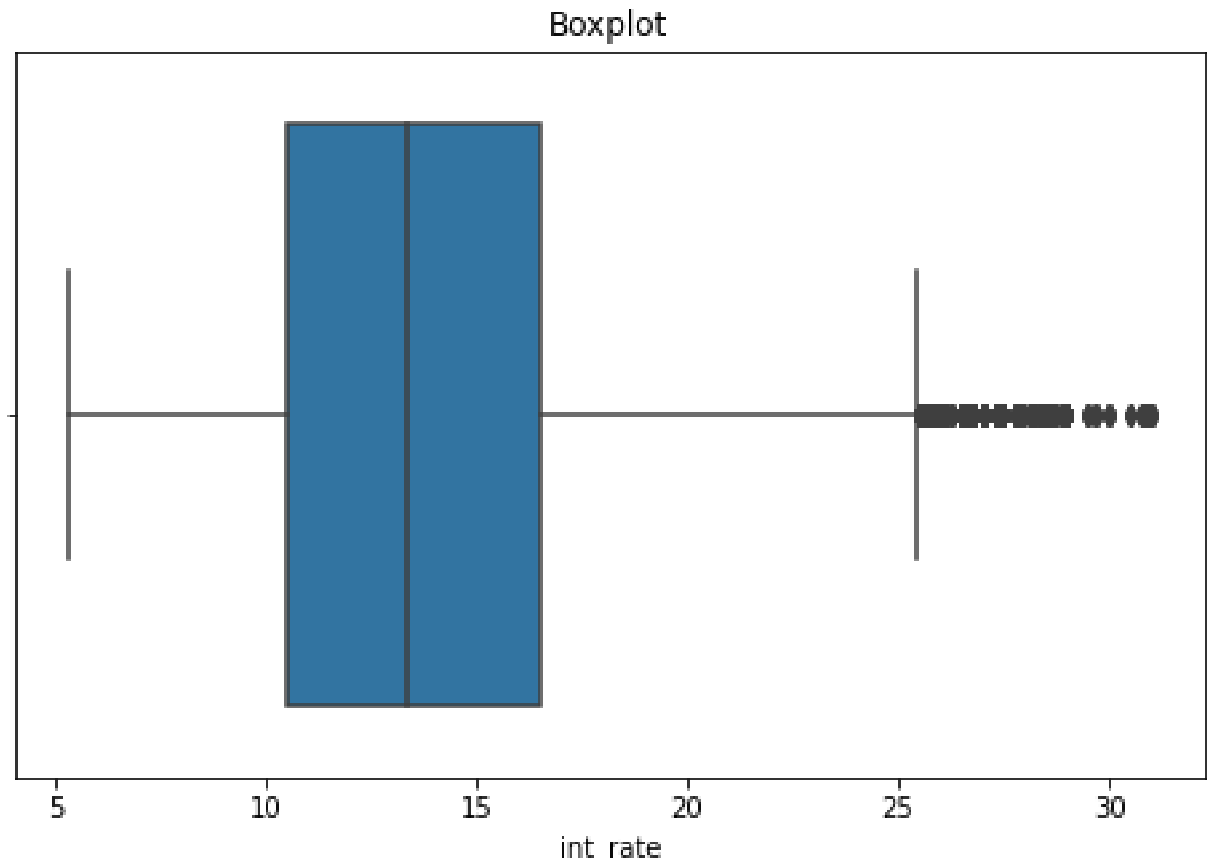
```
data['mort_acc'] = data.apply(fill_mort_acc, axis=1)
```

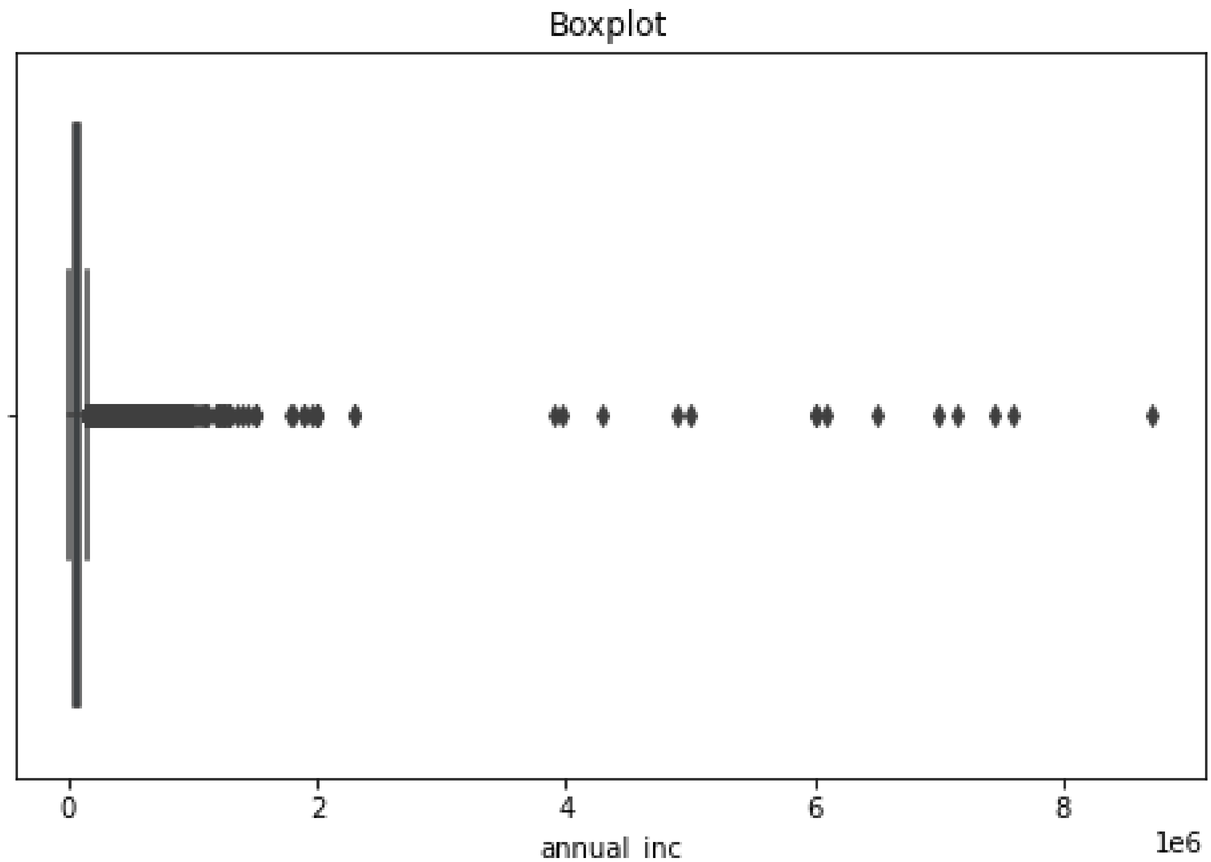
```
data.isnull().sum()
```

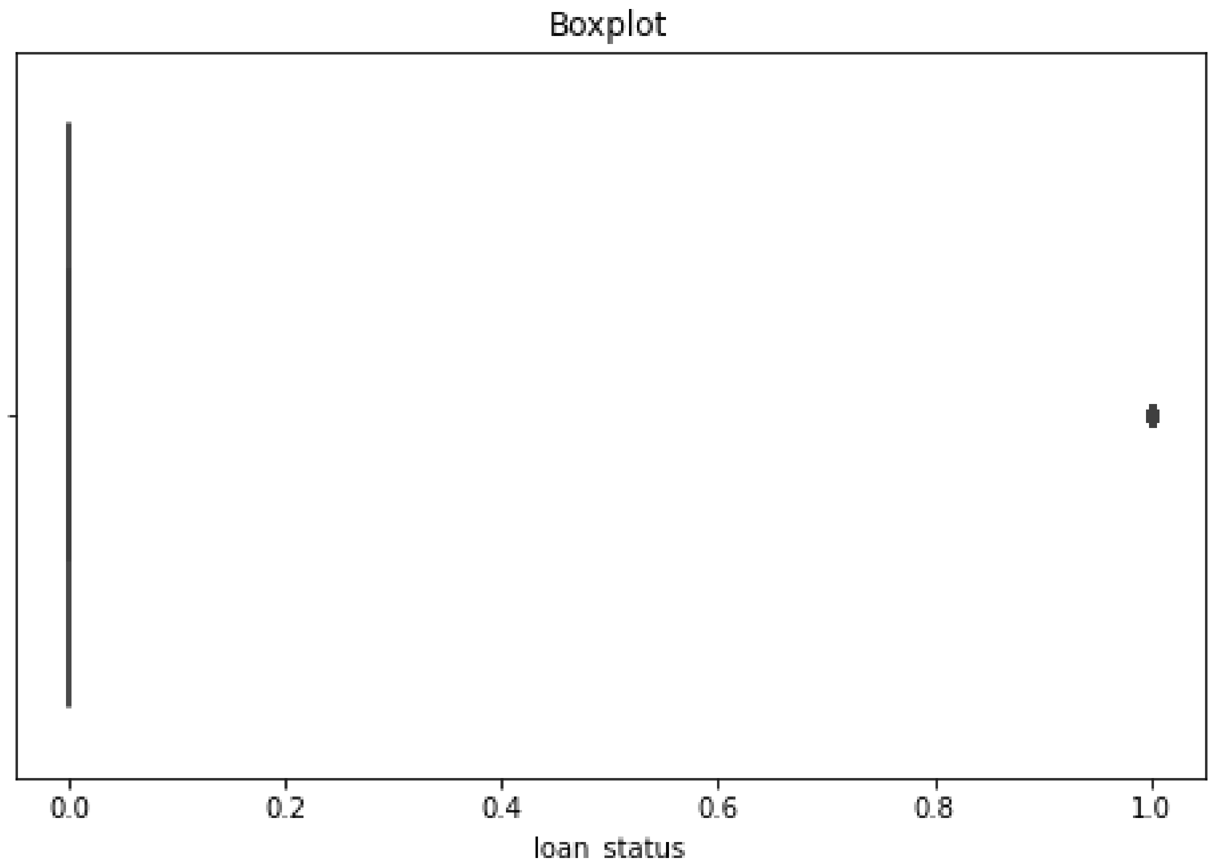
```
loan_amnt                   0
term                        0
int_rate                    0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
```

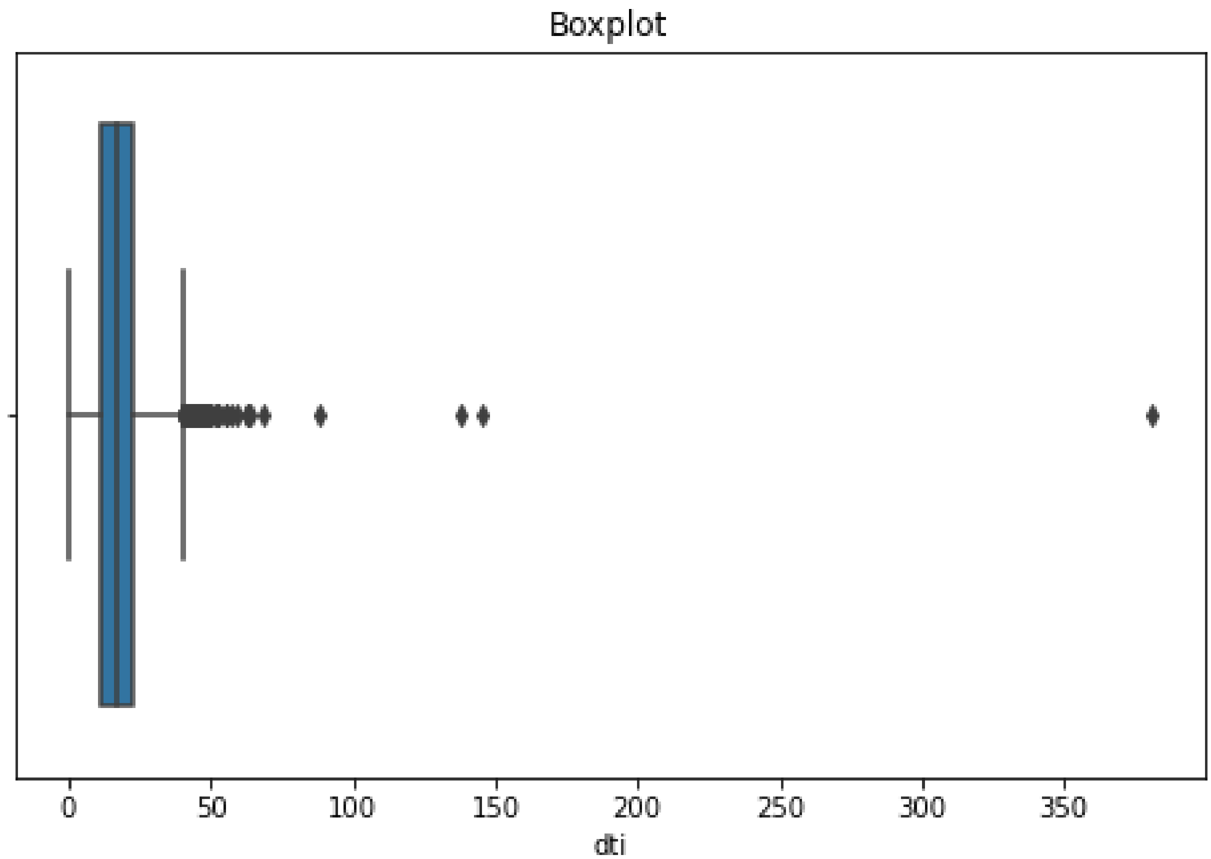
```
issue_d                        0
loan_status                    0
purpose                        0
title                       1755
dti                            0
earliest_cr_line               0
open_acc                       0
pub_rec                        0
revol_bal                      0
revol_util                   276
total_acc                      0
initial_list_status            0
application_type               0
mort_acc                       0
pub_rec_bankruptcies         535
address                        0
dtype: int64

data.shape

(396030, 26)

data.dropna(inplace=True)
data.shape

(370622, 26)

#Outlier Detection & Treatment -

numerical_data = data.select_dtypes(include='number')
num_cols = numerical_data.columns
len(num_cols)

12

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

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


Boxplot
0
5000
10000
15000
20000
25000
30000
35000
40000
loan_amnt

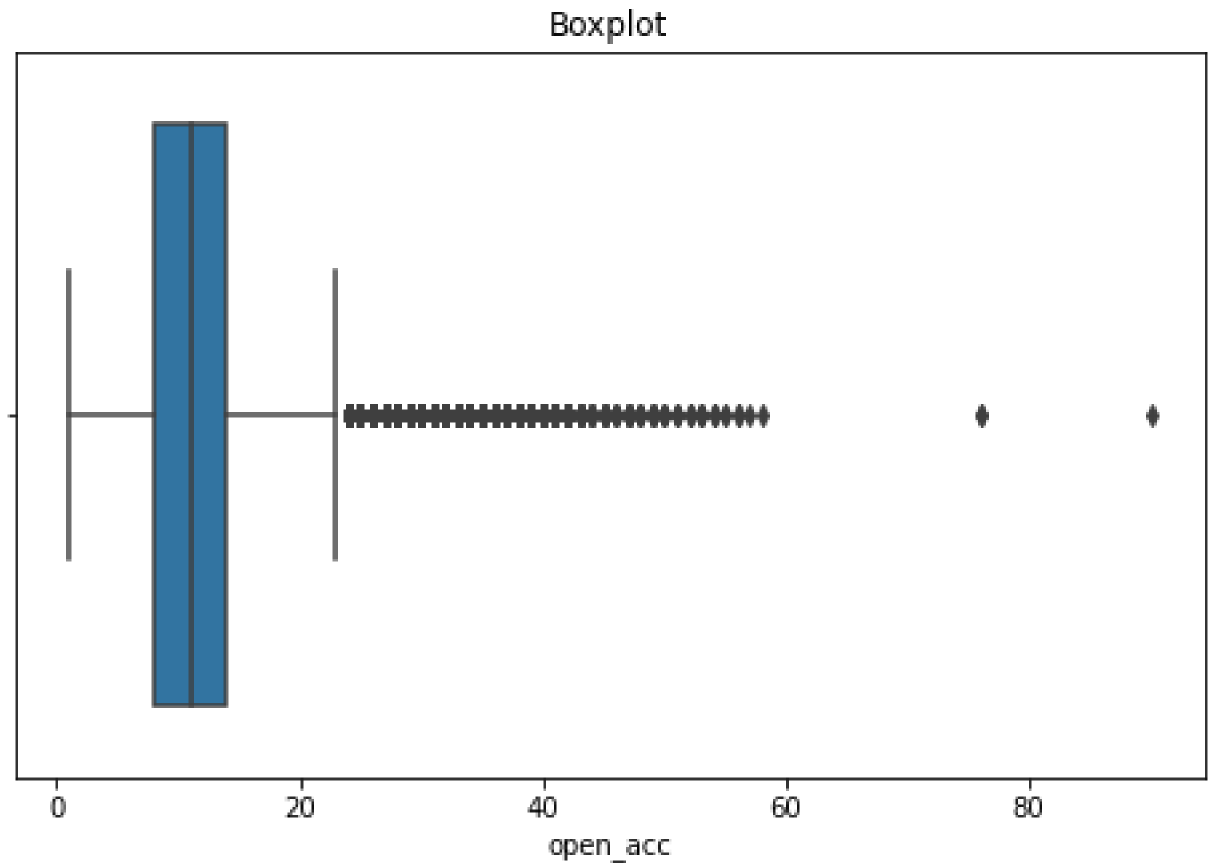

Boxplot
5
10
15
20
25
30
int_rate

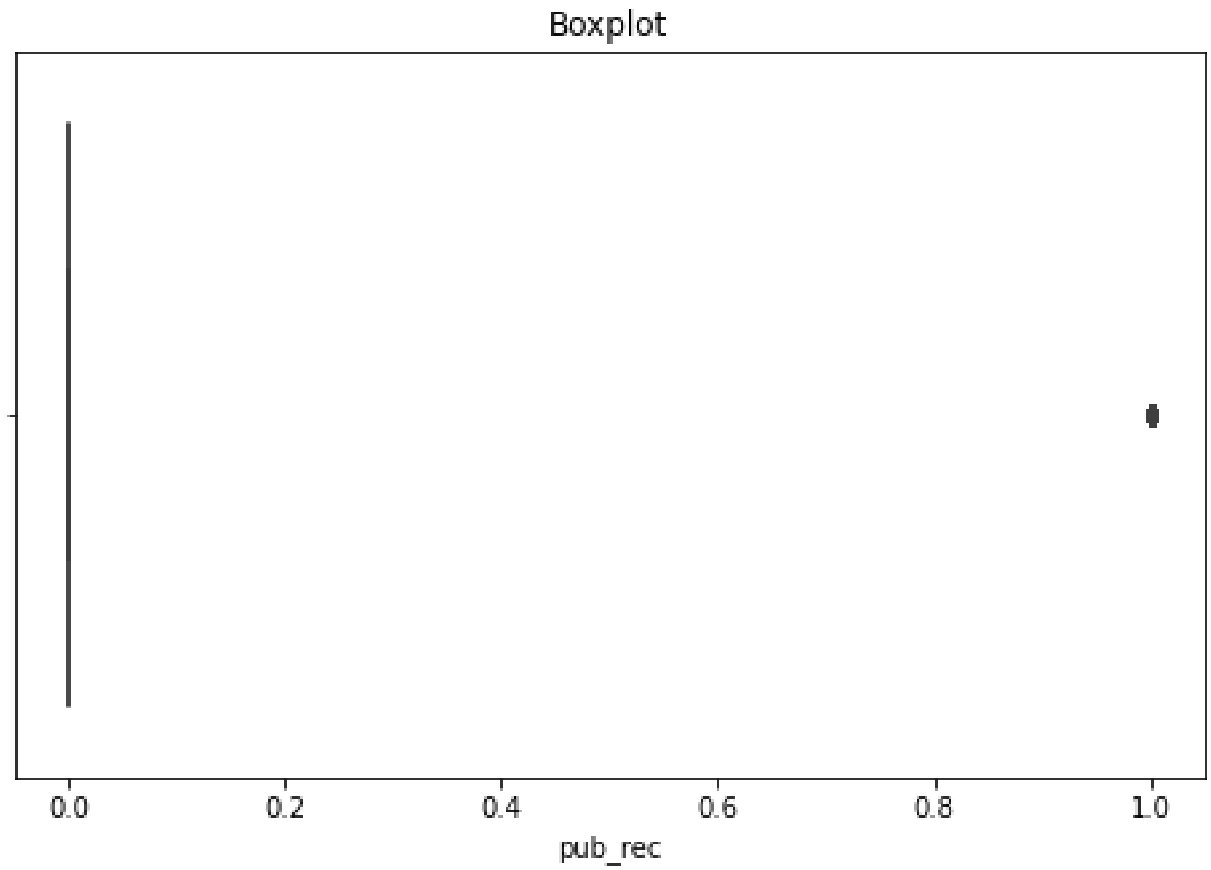

Boxplot
0
2
4
6
8
1e6
annual_inc

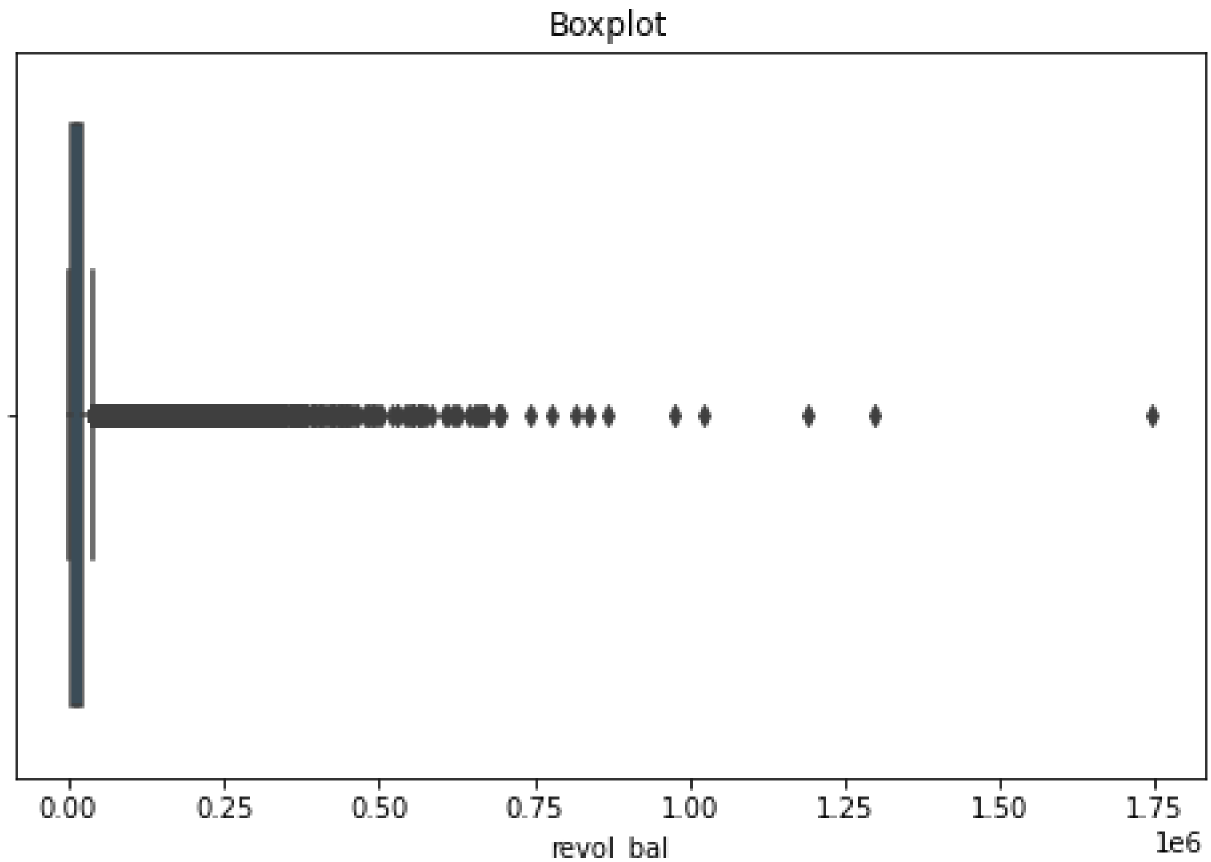

Boxplot
0.0
0.2
0.4
0.6
0.8
1.0
loan_status

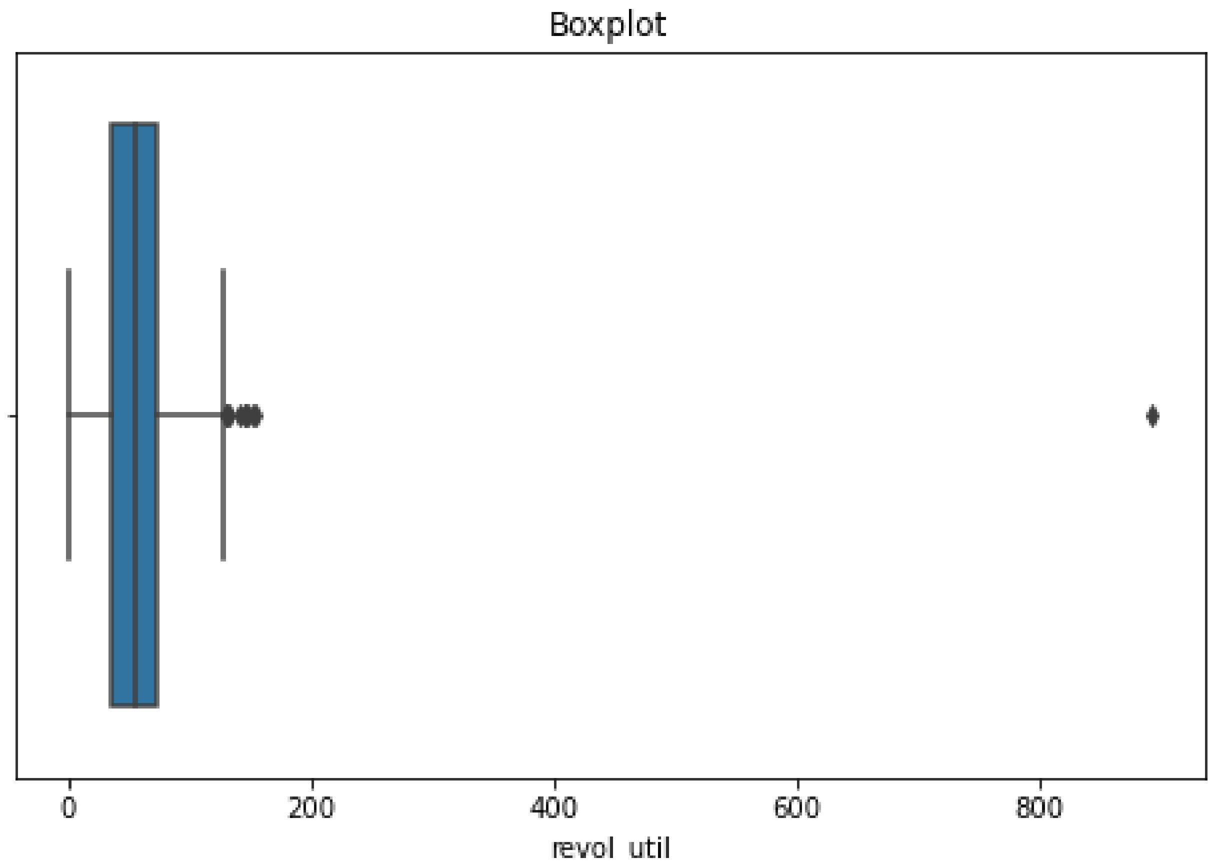


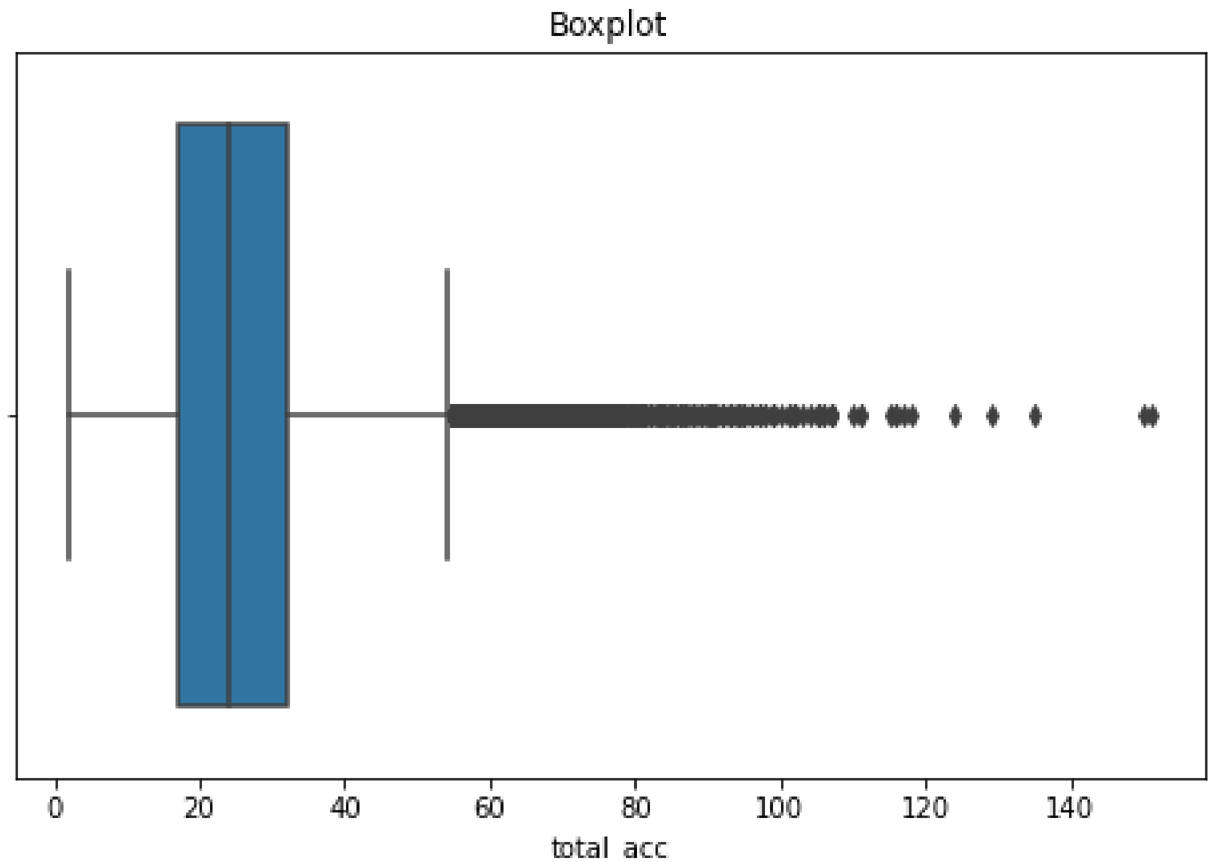
Boxplot
0
50
100
150
200
250
300
350
dti

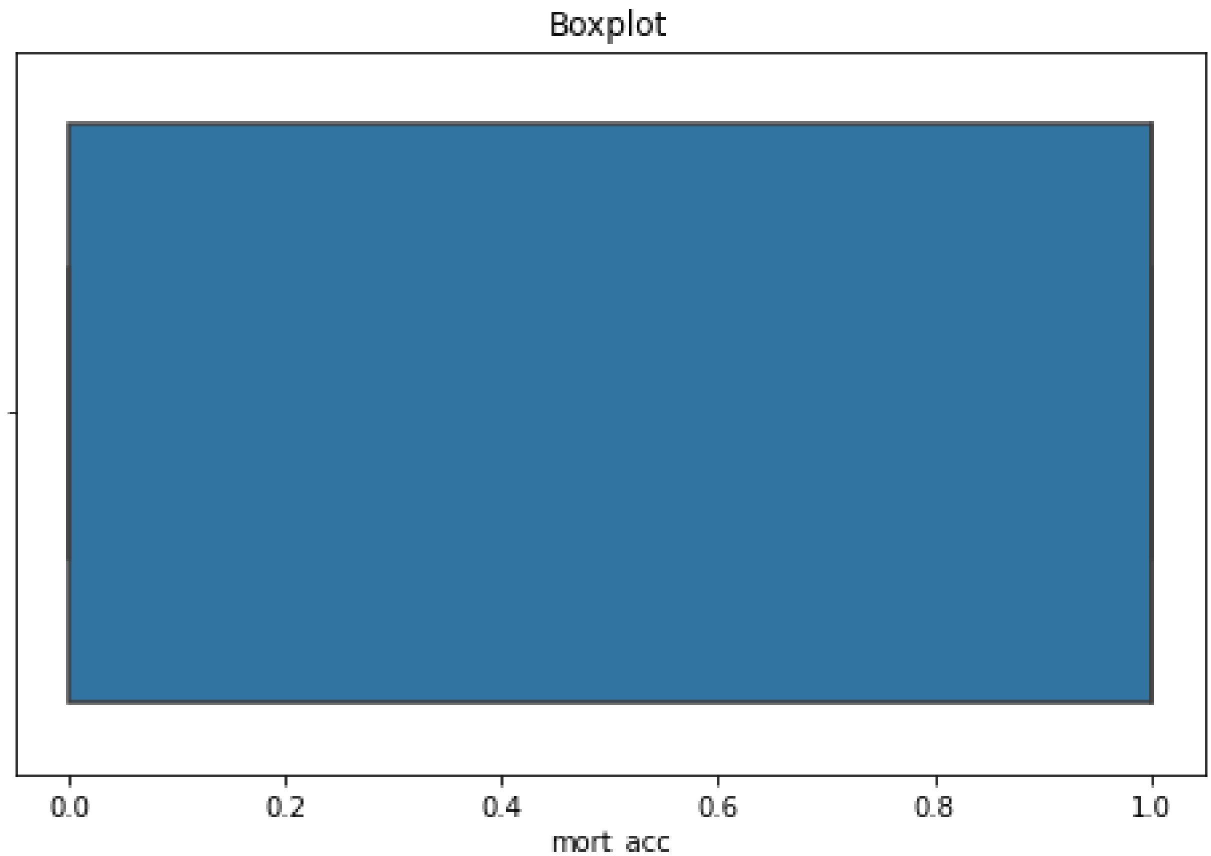

Boxplot
0
20
40
60
80
open_acc

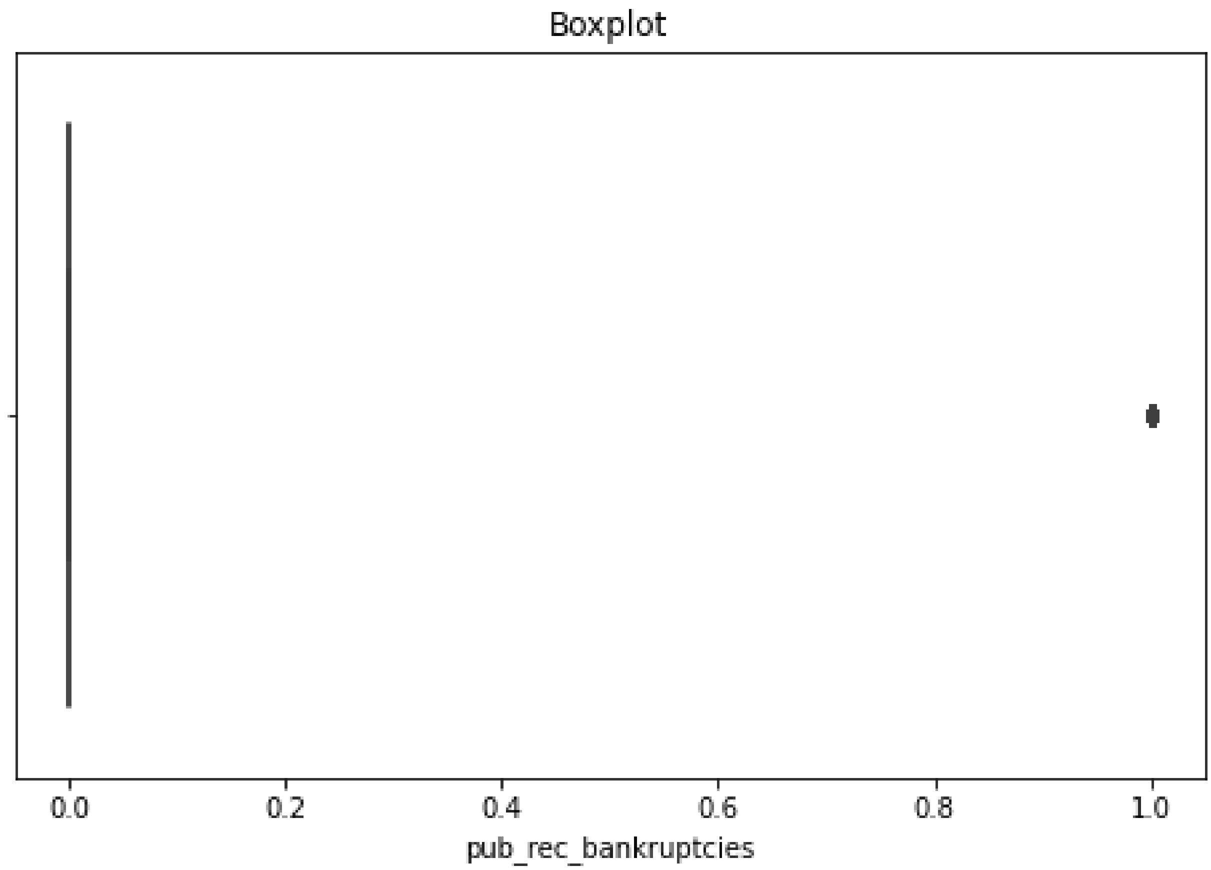

Boxplot
0.0
0.2
0.4
0.6
0.8
1.0
pub_rec


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


Boxplot
0
200
400
600
800
revol_util


Boxplot
0
20
40
60
80
100
120
140
total_acc


Boxplot
0.0
0.2
0.4
0.6
0.8
1.0
mort_acc




```
for col in num_cols:
    mean = data[col].mean()
    std = data[col].std()
    upper_limit = mean+3*std
    lower_limit = mean-3*std
    data = data[(data[col]<upper_limit) & (data[col]>lower_limit)]
data.shape
```

```
(354519, 26)
```

```
#Data Preprocessing -

term_values = {' 36 months': 36, ' 60 months': 60}
data['term'] = data.term.map(term_values)
list_status = {'w': 0, 'f': 1}
data['initial_list_status'] =
data.initial_list_status.map(list_status)
data['zip_code'] = data.address.apply(lambda x: x[-5:])

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

```
70466    14.382022
30723    14.277373
22690    14.268347
```

```
48052    14.127028
00813    11.610097
29597    11.537322
05113    11.516731
93700     2.774746
11650     2.772771
86630     2.733563
Name: zip_code, dtype: float64
```

```
#dropiing the columns which are of less significance
data.drop(columns=['issue_d', 'emp_title', 'title', 'sub_grade',
 'address', 'earliest_cr_line', 'emp_length'],
 axis=1, inplace=True)
```

```
data.shape
```

```
(354519, 20)
```

```
data.columns
```

```
Index(['loan_amnt', 'term', 'int_rate', 'grade', 'home_ownership',
       'annual_inc', 'verification_status', 'loan_status', 'purpose',
'dti',
       'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc',
       'initial_list_status', 'application_type', 'mort_acc',
       'pub_rec_bankruptcies', 'zip_code'],
      dtype='object')
```

```
data
```

```
        loan_amnt  term  int_rate grade home_ownership  annual_inc  \
0         10000.0    36     11.44     B           RENT    117000.0
1          8000.0    36     11.99     B       MORTGAGE     65000.0
2         15600.0    36     10.49     B           RENT     43057.0
3          7200.0    36      6.49     A           RENT     54000.0
4         24375.0    60     17.27     C       MORTGAGE     55000.0
...           ...   ...       ...   ...            ...         ...
396025    10000.0    60     10.99     B           RENT     40000.0
396026    21000.0    36     12.29     C       MORTGAGE    110000.0
396027     5000.0    36      9.99     B           RENT     56500.0
396028    21000.0    60     15.31     C       MORTGAGE     64000.0
396029     2000.0    36     13.61     C           RENT     42996.0

       verification_status  loan_status             purpose    dti
open_acc  \
0             Not Verified            0            vacation  26.24
16.0
1             Not Verified            0  debt_consolidation  22.05
17.0
2          Source Verified            0         credit_card  12.79
13.0
```

```
3             Not Verified            0         credit_card   2.60
6.0
4                 Verified            1         credit_card  33.95
13.0
...                    ...          ...                 ...    ...
...
396025     Source Verified            0  debt_consolidation  15.63
6.0
396026     Source Verified            0  debt_consolidation  21.45
6.0
396027            Verified            0  debt_consolidation  17.56
15.0
396028            Verified            0  debt_consolidation  15.88
9.0
396029            Verified            0  debt_consolidation   8.32
3.0

        pub_rec  revol_bal  revol_util  total_acc  initial_list_status
\
0             0    36369.0        41.8       25.0                    0

1             0    20131.0        53.3       27.0                    1

2             0    11987.0        92.2       26.0                    1

3             0     5472.0        21.5       13.0                    1

4             0    24584.0        69.8       43.0                    1

...         ...        ...         ...        ...                  ...

396025        0     1990.0        34.3       23.0                    0

396026        0    43263.0        95.7        8.0                    1

396027        0    32704.0        66.9       23.0                    1

396028        0    15704.0        53.8       20.0                    1

396029        0     4292.0        91.3       19.0                    1


       application_type  mort_acc  pub_rec_bankruptcies zip_code
0            INDIVIDUAL       0.0                   0.0    22690
1            INDIVIDUAL       1.0                   0.0    05113
2            INDIVIDUAL       0.0                   0.0    05113
3            INDIVIDUAL       0.0                   0.0    00813
4            INDIVIDUAL       1.0                   0.0    11650
...                 ...       ...                   ...      ...
396025       INDIVIDUAL       0.0                   0.0    30723
```

```
396026       INDIVIDUAL       1.0                   0.0    05113
396027       INDIVIDUAL       0.0                   0.0    70466
396028       INDIVIDUAL       1.0                   0.0    29597
396029       INDIVIDUAL       1.0                   0.0    48052

[354519 rows x 20 columns]
```

```
#--one hot encoding

columns_to_encode = ['purpose', 'zip_code', 'grade',
'verification_status', 'application_type', 'home_ownership']
# Perform one-hot encoding
data = pd.get_dummies(data, columns=columns_to_encode,
drop_first=True)
data.shape
```

```
(354519, 49)
```

```
#Data Preparation for Modeling -

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)

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

logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)
```

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

```
y_pred = logreg.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy of Logistic Regression Classifier on test set:
{:.3f}'.format(accuracy))
```

```
Accuracy of Logistic Regression Classifier on test set: 0.890
```

```
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)
```

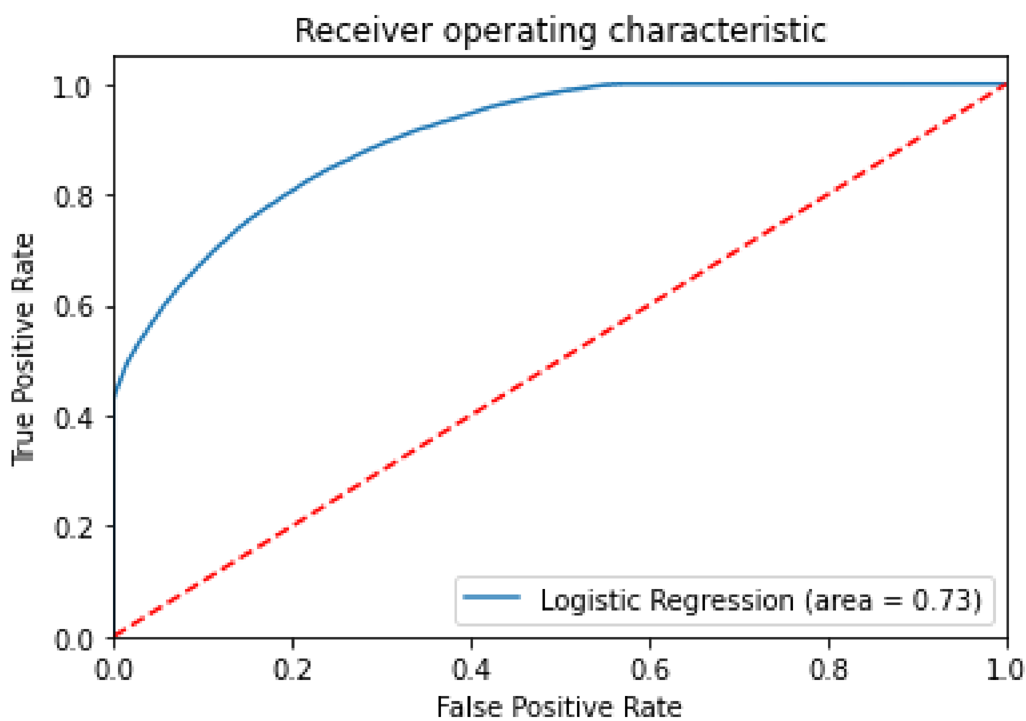
```
[[85365   523]
 [11131  9337]]
```

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

```
              precision    recall  f1-score   support
```

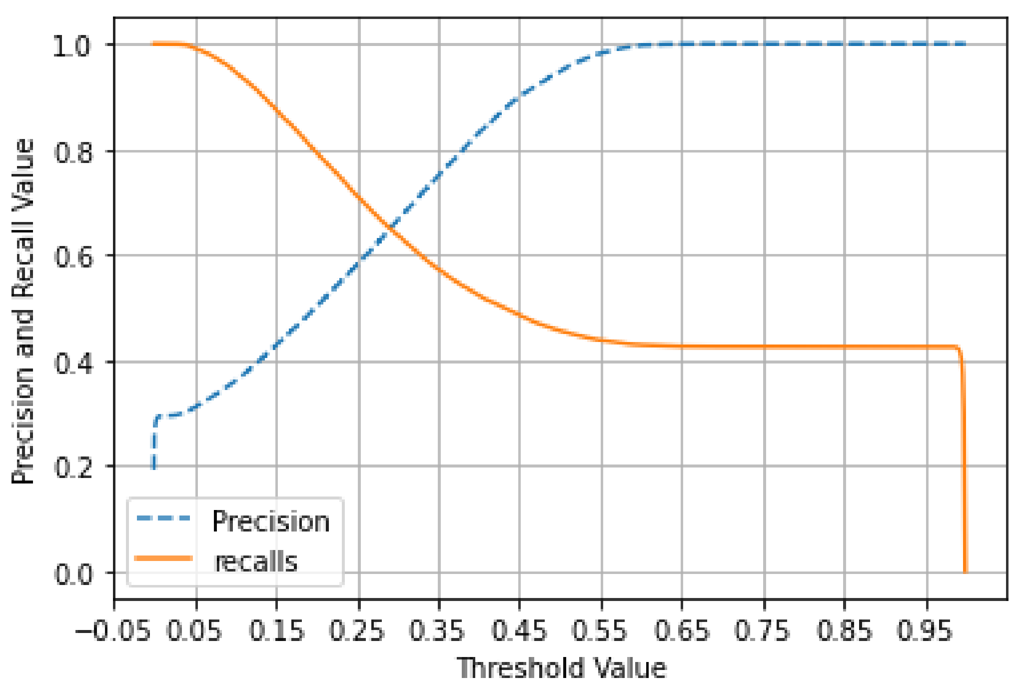
```
              0       0.88      0.99      0.94     85888
              1       0.95      0.46      0.62     20468

       accuracy                           0.89    106356
      macro avg       0.92      0.73      0.78    106356
   weighted avg       0.90      0.89      0.87    106356
```

```
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.savefig('Log_ROC')
plt.show()
```




```
def precision_recall_curve_plot(y_test, pred_proba_c1):
    precisions, recalls, thresholds = precision_recall_curve(y_test,
pred_proba_c1)
```

```
    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='recalls')

    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()

precision_recall_curve_plot(y_test, logreg.predict_proba(X_test)[:,1])
```




```
from statsmodels.stats.outliers_influence import
variance_inflation_factor
def calc_vif(X):
    # Calculating the VIF
    vif = pd.DataFrame()
    vif['Feature'] = X.columns
    vif['VIF'] = [variance_inflation_factor(X.values, i) for i in
range(X.shape[1])]
    vif['VIF'] = round(vif['VIF'], 2)
```

```
    vif = vif.sort_values(by='VIF', ascending = False)
    return vif
#calc_vif(X)[:5]

X.drop(columns=['application_type_INDIVIDUAL'], axis=1, inplace=True)
#calc_vif(X)[:5]

X.drop(columns=['int_rate'], axis=1, inplace=True)
#calc_vif(X)[:5]

X.drop(columns=['term'], axis=1, inplace=True)
#calc_vif(X)[:5]

X.drop(columns=['purpose_debt_consolidation'], axis=1, inplace=True)
#calc_vif(X)[:5]

X.drop(columns=['open_acc'], axis=1, inplace=True)
#calc_vif(X)[:5]

X = scaler.fit_transform(X)
kfold = KFold(n_splits=5)
accuracy = np.mean(cross_val_score(logreg, X, y, cv=kfold,
scoring='accuracy', n_jobs=-1))
print("Cross Validation accuracy: {:.3f}".format(accuracy))
```

```
Cross Validation accuracy: 0.891
```

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print("After OverSampling, counts of label '1':
{}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0':
{}".format(sum(y_train_res == 0)))
```

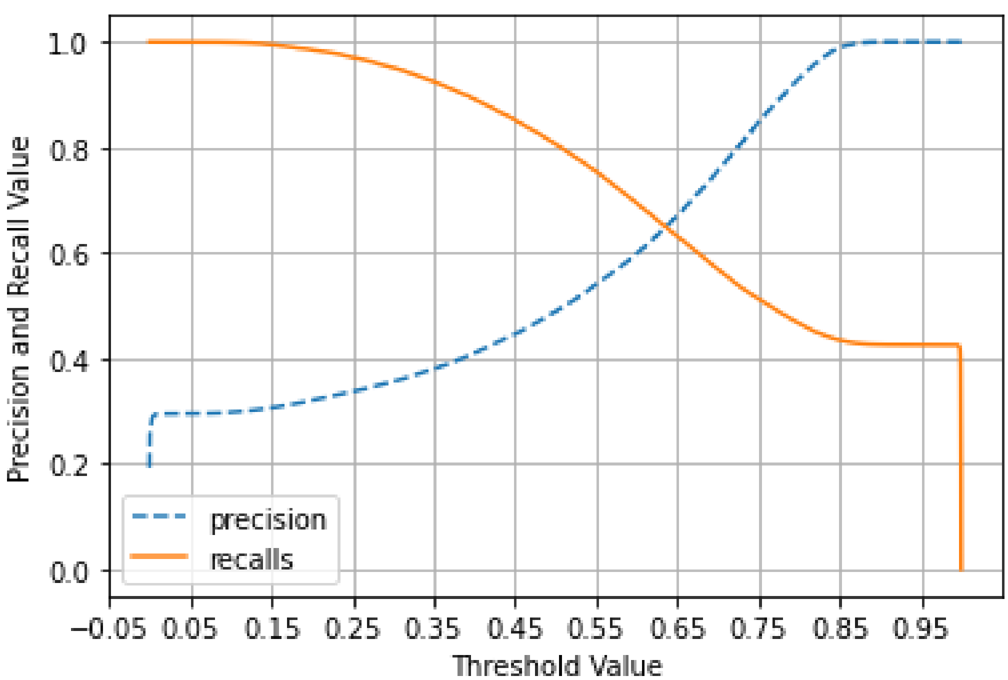
```
After OverSampling, counts of label '1': 200405
After OverSampling, counts of label '0': 200405
```

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

# Classification Report
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.80      0.87     85888
           1       0.49      0.81      0.61     20468

    accuracy                           0.80    106356
```

```
   macro avg       0.72      0.80      0.74    106356
weighted avg       0.86      0.80      0.82    106356
```

```python
def precision_recall_curve_plot(y_test, pred_proba_c1):
    precisions, recalls, thresholds = precision_recall_curve(y_test,
pred_proba_c1)

    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='recalls')

    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()

precision_recall_curve_plot(y_test, lr1.predict_proba(X_test)[:,1])
```




```python
#we should try our best to have improved precision recall both
```

High recall means that the model is effective at capturing all positive instances. This is important in scenarios where missing a positive instance has severe consequences, such as failing to detect a fraudulent transaction.