loantap-logistic-regression

January 7, 2024

[153]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib as mpl
import scipy.stats as spy
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from imblearn.over_sampling import SMOTE
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_score
from sklearn.metrics import roc_curve
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,Ridge,Lasso
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import r2_score
from statsmodels.stats.outliers_influence import variance_inflation_factor
from scipy import stats
```

##Structure and Characteristics of the dataset

[68]:
```python
df=pd.read_csv('logistic_regression.csv')
df.head(2)
```

[68]:
```
   loan_amnt        term  int_rate  installment grade sub_grade  \
0      10000   36 months     11.44       329.48     B        B4
1       8000   36 months     11.99       265.68     B        B5

         emp_title emp_length home_ownership  annual_inc  … open_acc  \
0        Marketing  10+ years           RENT    117000.0  …       16
1   Credit analyst    4 years       MORTGAGE     65000.0  …       17
```

```
  pub_rec revol_bal revol_util total_acc  initial_list_status  \
0       0     36369       41.8        25                    w
1       0     20131       53.3        27                    f

  application_type  mort_acc  pub_rec_bankruptcies  \
0       INDIVIDUAL       0.0                   0.0
1       INDIVIDUAL       3.0                   0.0

                                        address
0     0174 Michelle Gateway\nMendozaberg, OK 22690
1  1076 Carney Fort Apt. 347\nLoganmouth, SD 05113

[2 rows x 27 columns]
```

[69]:
```
df.shape
```

[69]:
```
(396030, 27)
```

[70]:
```
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  int64
 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  int64
 18  pub_rec               396030 non-null  int64
 19  revol_bal             396030 non-null  int64
 20  revol_util            395754 non-null  float64
```

```
 21  total_acc             396030 non-null  int64
 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(7), int64(5), object(15)
memory usage: 81.6+ MB
```

[71]:
```
df['application_type'].value_counts()
```

[71]:
```
INDIVIDUAL    395319
JOINT            425
DIRECT_PAY       286
Name: application_type, dtype: int64
```

[72]:
```
df.describe()
```

[72]:
```
           loan_amnt       int_rate    installment    annual_inc  \
count  396030.000000  396030.000000  396030.000000  3.960300e+05
mean    14113.888089      13.639400     431.849698  7.420318e+04
std      8357.441341       4.472157     250.727790  6.163762e+04
min       500.000000       5.320000      16.080000  0.000000e+00
25%      8000.000000      10.490000     250.330000  4.500000e+04
50%     12000.000000      13.330000     375.430000  6.400000e+04
75%     20000.000000      16.490000     567.300000  9.000000e+04
max     40000.000000      30.990000    1533.810000  8.706582e+06

                 dti       open_acc        pub_rec     revol_bal  \
count  396030.000000  396030.000000  396030.000000  3.960300e+05
mean       17.379514      11.311153       0.178191  1.584454e+04
std        18.019092       5.137649       0.530671  2.059184e+04
min         0.000000       0.000000       0.000000  0.000000e+00
25%        11.280000       8.000000       0.000000  6.025000e+03
50%        16.910000      10.000000       0.000000  1.118100e+04
75%        22.980000      14.000000       0.000000  1.962000e+04
max      9999.000000      90.000000      86.000000  1.743266e+06

          revol_util      total_acc       mort_acc  pub_rec_bankruptcies
count  395754.000000  396030.000000  358235.000000         395495.000000
mean       53.791749      25.414744       1.813991              0.121648
std        24.452193      11.886991       2.147930              0.356174
min         0.000000       2.000000       0.000000              0.000000
25%        35.800000      17.000000       0.000000              0.000000
50%        54.800000      24.000000       1.000000              0.000000
75%        72.900000      32.000000       3.000000              0.000000
max       892.300000     151.000000      34.000000              8.000000
```

From the above data, we can get the statistical values of the dataset like Mean, Minimum, Maximum, Count and so on.

[73]:
```
df.describe(include=object)
```

[73]:
```
              term   grade sub_grade emp_title emp_length home_ownership  \
count       396030  396030    396030    373103     377729         396030
unique           2       7        35    173103         11              6
top      36 months       B        B3   Teacher  10+ years       MORTGAGE
freq        302005  116018     26655      4389     126041         198348

       verification_status issue_d loan_status             purpose  \
count               396030  396030      396030              396030
unique                   3     115           2                  14
top               Verified  Oct-14  Fully Paid  debt_consolidation
freq                139563   14846      318357              234507

                     title earliest_cr_line initial_list_status  \
count               394275           396030              396030
unique               48805              684                   2
top     Debt consolidation           Oct-00                   f
freq                152472             3017              238066

       application_type                      address
count            396030                       396030
unique                3                       393700
top          INDIVIDUAL  USCGC Smith\nFPO AE 70466
freq             395319                            8
```

[74]:
```
df.isnull().sum()
```

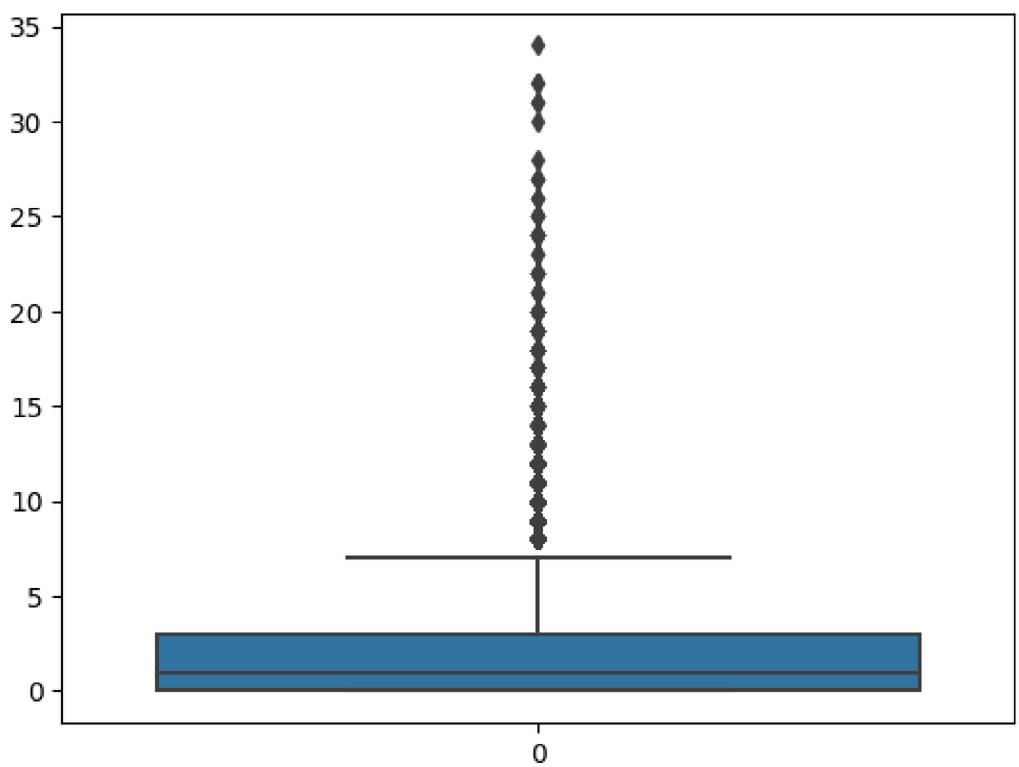
[74]:
```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1755
dti                         0
```

```
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

Here, we can see that columns like emp_title, emp_length, mort_acc have significant null values in them compared to all other columns, therefore, we can drop these columns. We will fill up the missing values in these columns and rest columns with small number of null values can be dropped.

[75]:
```
sns.boxplot(df['mort_acc'])
```

[75]:
```
<Axes: >
```



[76]:
```
df['mort_acc'].fillna(df['mort_acc'].median(),inplace=True)
```

[77]:
```
df.isnull().sum()
```

[77]:
```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1755
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                    0
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

Here, we can see that there are 0 values in mort_acc, we have filled the 37795 null values with median values of mort_acc. Now, we will also try to fill the other two columns emp_length and emp_title with median values.

[78]:
```
df['emp_length'].fillna('5 years',inplace=True)
```

[79]:
```
df['emp_title'].fillna('unknown_title',inplace=True)
```

We have filled the null values for emp_title and emp_length and for remaining columns where the null values were less in number, we will drop them.

[80]:
```
df=df.dropna()
```

[81]:
```
df.isnull().sum()
```

[81]:
```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
emp_title               0
emp_length              0
home_ownership          0
annual_inc              0
verification_status     0
issue_d                 0
loan_status             0
purpose                 0
title                   0
dti                     0
earliest_cr_line        0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util              0
total_acc               0
initial_list_status     0
application_type        0
mort_acc                0
pub_rec_bankruptcies    0
address                 0
dtype: int64
```

##Data Cleaning

[82]:
```
df['emp_length']=df['emp_length'].replace(['< 1 year'],'0 year')
df['emp_length']=df['emp_length'].replace(['10+ years'],'10 years')
df[['employment_age','redundent_data']]=df['emp_length'].str.split(' ',
 ↪expand=True)
df.drop(['emp_length','redundent_data'],axis='columns',inplace=True)
```

[83]:
```
df['employment_age'].value_counts()
```

[83]:
```
10    125270
5      44429
2      35597
0      31489
3      31469
1      25637
4      23811
6      20750
```

```
7      20727
8      19071
9      15215
Name: employment_age, dtype: int64
```

[84]:
```
df['employment_age']=df['employment_age'].astype(int)
```

[85]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 393465 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             393465 non-null  int64
 1   term                  393465 non-null  object
 2   int_rate              393465 non-null  float64
 3   installment           393465 non-null  float64
 4   grade                 393465 non-null  object
 5   sub_grade             393465 non-null  object
 6   emp_title             393465 non-null  object
 7   home_ownership        393465 non-null  object
 8   annual_inc            393465 non-null  float64
 9   verification_status   393465 non-null  object
 10  issue_d               393465 non-null  object
 11  loan_status           393465 non-null  object
 12  purpose               393465 non-null  object
 13  title                 393465 non-null  object
 14  dti                   393465 non-null  float64
 15  earliest_cr_line      393465 non-null  object
 16  open_acc              393465 non-null  int64
 17  pub_rec               393465 non-null  int64
 18  revol_bal             393465 non-null  int64
 19  revol_util            393465 non-null  float64
 20  total_acc             393465 non-null  int64
 21  initial_list_status   393465 non-null  object
 22  application_type      393465 non-null  object
 23  mort_acc              393465 non-null  float64
 24  pub_rec_bankruptcies  393465 non-null  float64
 25  address               393465 non-null  object
 26  employment_age        393465 non-null  int64
dtypes: float64(7), int64(6), object(14)
memory usage: 84.1+ MB
```

[86]:
```
df.describe(include=object)
```

```
[86]:                 term   grade sub_grade      emp_title home_ownership  \
count          393465  393465    393465         393465         393465
unique              2       7        35         172225              6
top         36 months       B        B3  unknown_title       MORTGAGE
freq           300024  115395     26518          22668         197110

       verification_status issue_d loan_status             purpose  \
count               393465  393465      393465              393465
unique                   3     112           2                  14
top               Verified  Oct-14  Fully Paid  debt_consolidation
freq                138867   14838      316271              233108

                     title earliest_cr_line initial_list_status  \
count               393465           393465              393465
unique               48460              683                   2
top     Debt consolidation           Oct-00                   f
freq                152392             2999              236947

       application_type                 address
count            393465                  393465
unique                3                  391162
top          INDIVIDUAL  USS Smith\nFPO AP 70466
freq             392844                       8
```

Both columns named earliest_cr_line and issue_d are dates, so to make it useful for our analysis, we will try to convert these string dates to numerical form.

```
[88]: df['issue_d'] = pd.to_datetime(df['issue_d'])
```

```
---------------------------------------------------------------------------
OutOfBoundsDatetime                       Traceback (most recent call last)
<ipython-input-88-e248311e73c9> in <cell line: 1>()
----> 1 df['issue_d'] = pd.to_datetime(df['issue_d'])

/usr/local/lib/python3.10/dist-packages/pandas/core/tools/datetimes.py in␣
 ↪to_datetime(arg, errors, dayfirst, yearfirst, utc, format, exact, unit,␣
 ↪infer_datetime_format, origin, cache)
   1062                     result = arg.tz_localize(tz)
   1063     elif isinstance(arg, ABCSeries):
-> 1064         cache_array = _maybe_cache(arg, format, cache, convert_listlike
   1065         if not cache_array.empty:
   1066             result = arg.map(cache_array)

/usr/local/lib/python3.10/dist-packages/pandas/core/tools/datetimes.py in␣
 ↪_maybe_cache(arg, format, cache, convert_listlike)
    227         unique_dates = unique(arg)
    228         if len(unique_dates) < len(arg):
--> 229             cache_dates = convert_listlike(unique_dates, format)
```

```
    230             # GH#45319
    231             try:

/usr/local/lib/python3.10/dist-packages/pandas/core/tools/datetimes.py in␣
 ↪_convert_listlike_datetimes(arg, format, name, tz, unit, errors,␣
 ↪infer_datetime_format, dayfirst, yearfirst, exact)
    436     assert format is None or infer_datetime_format
    437     utc = tz == "utc"
--> 438     result, tz_parsed = objects_to_datetime64ns(
    439         arg,
    440         dayfirst=dayfirst,

/usr/local/lib/python3.10/dist-packages/pandas/core/arrays/datetimes.py in␣
 ↪objects_to_datetime64ns(data, dayfirst, yearfirst, utc, errors,␣
 ↪require_iso8601, allow_object, allow_mixed)
   2175     order: Literal["F", "C"] = "F" if flags.f_contiguous else "C"
   2176     try:
-> 2177         result, tz_parsed = tslib.array_to_datetime(
   2178             data.ravel("K"),
   2179             errors=errors,

/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslib.pyx in pandas._libs.
 ↪tslib.array_to_datetime()

/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslib.pyx in pandas._libs.
 ↪tslib.array_to_datetime()

/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslib.pyx in pandas._libs.
 ↪tslib.array_to_datetime()

/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslib.pyx in pandas._libs.
 ↪tslib.array_to_datetime()

/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslibs/conversion.pyx in␣
 ↪pandas._libs.tslibs.conversion.convert_datetime_to_tsobject()

/usr/local/lib/python3.10/dist-packages/pandas/_libs/tslibs/np_datetime.pyx in␣
 ↪pandas._libs.tslibs.np_datetime.check_dts_bounds()

OutOfBoundsDatetime: Out of bounds nanosecond timestamp: 1-01-15 00:00:00␣
 ↪present at position 0
```

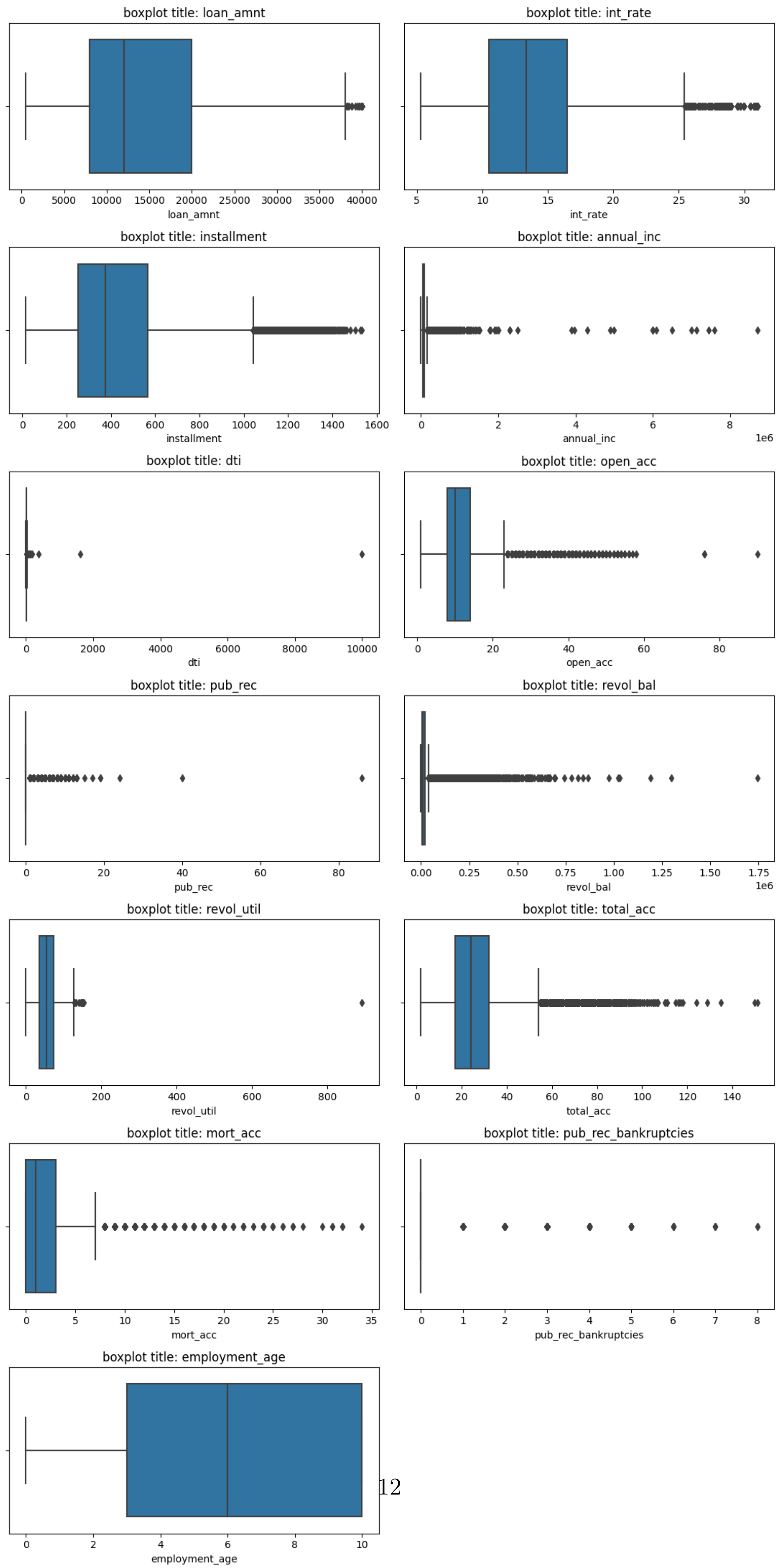
##Outlier Treatment

[87]:
```
number_cols=df.select_dtypes(include='number').columns
print(number_cols)
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'mort_acc',
       'pub_rec_bankruptcies', 'employment_age'],
      dtype='object')
```

[89]:

```
figure=plt.figure(figsize=(11,22))
k=1
for col in number_cols:
  ax=plt.subplot(7,2,k)
  sns.boxplot(x=df[col])
  plt.title(f'boxplot title: {col}')
  k=k+1
plt.tight_layout()
plt.show()
```


boxplot title: loan_amnt
loan_amnt
boxplot title: int_rate
int_rate
boxplot title: installment
installment
boxplot title: annual_inc
annual_inc
1e6
boxplot title: dti
dti
boxplot title: open_acc
open_acc
boxplot title: pub_rec
pub_rec
boxplot title: revol_bal
revol_bal
1e6
boxplot title: revol_util
revol_util
boxplot title: total_acc
total_acc
boxplot title: mort_acc
mort_acc
boxplot title: pub_rec_bankruptcies
pub_rec_bankruptcies
boxplot title: employment_age
employment_age

From the boxplots, we can see that many columns have outliers in them, so now let's remove the outliers using standard deviation method(that is 99% of data within the first three standard deviation in case of normally distributed data) for pub_rec, and for pub_rec_bankruptcies, we can apply 0 or 1 approach.

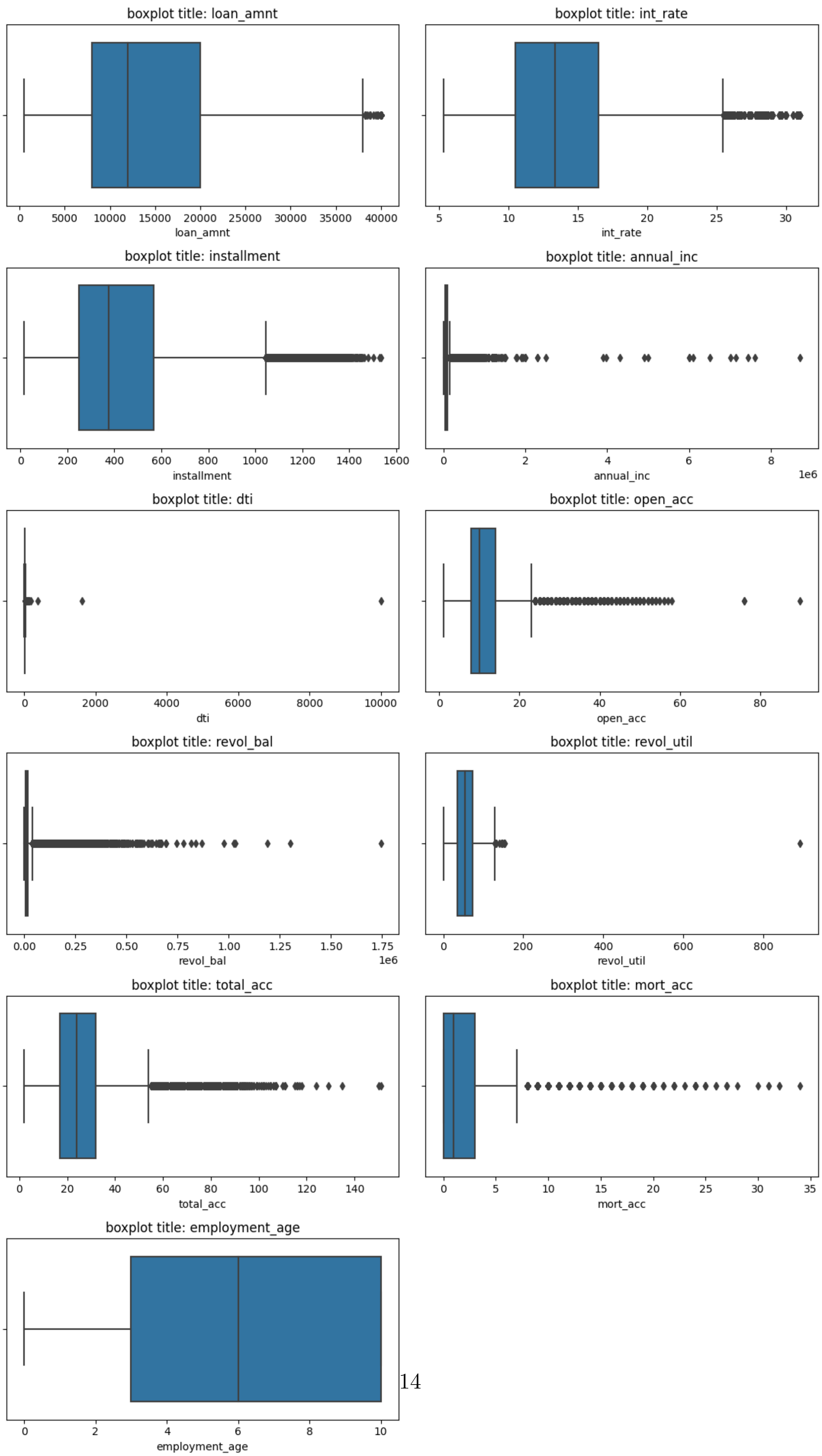
[90]:
```
##converting pub_rec_bankruptcies and pub_rec to categorical variables
df['pub_rec']=np.where(df['pub_rec']>0,'yes','no')
df['pub_rec_bankruptcies']=np.where(df['pub_rec_bankruptcies']>0,'yes','no')
df[['pub_rec','pub_rec_bankruptcies']]=df[['pub_rec','pub_rec_bankruptcies']].
 ↪astype('category')
```

[91]:
```
##after converting public records to category, let's see the numerical columns
number_cols=df.select_dtypes(include='number').columns
print(number_cols)
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti', 'open_acc',
       'revol_bal', 'revol_util', 'total_acc', 'mort_acc', 'employment_age'],
      dtype='object')
```

[92]:
```
##now removing the outliers from the numerical columns using standard deviation␣
 ↪method
k=1
for col in number_cols:
  mean=df[col].mean()
  sd=df[col].std()
  upper_sd=mean+(3*sd)
df=df[~(df[col]>upper_sd)]
```

[93]:
```
figure=plt.figure(figsize=(11,22))
k=1
for col in number_cols:
  ax=plt.subplot(7,2,k)
  sns.boxplot(x=df[col])
  plt.title(f'boxplot title: {col}')
  k=k+1
plt.tight_layout()
plt.show()
```


boxplot title: loan_amnt
loan_amnt
boxplot title: int_rate
int_rate
boxplot title: installment
installment
boxplot title: annual_inc
annual_inc
1e6
boxplot title: dti
dti
boxplot title: open_acc
open_acc
boxplot title: revol_bal
revol_bal
1e6
boxplot title: revol_util
revol_util
boxplot title: total_acc
total_acc
boxplot title: mort_acc
mort_acc
boxplot title: employment_age
employment_age

[94]:
```
df.shape
```

[94]: (393465, 27)

[95]:
```
df['address'].sample(5)
df[['state','zip_code']]=df['address'].apply(lambda y: pd.Series([y[-8:-6],␣
 ↪y[-5:]]))
```

[96]:
```
df.drop(['address'], axis='columns', inplace=True)
```

[97]:
```
df['zip_code']=df['zip_code'].astype('category')
```

[98]:
```
df.describe(include=object).T
```

[98]:

	count	unique	top	freq
term	393465	2	36 months	300024
grade	393465	7	B	115395
sub_grade	393465	35	B3	26518
emp_title	393465	172225	unknown_title	22668
home_ownership	393465	6	MORTGAGE	197110
verification_status	393465	3	Verified	138867
issue_d	393465	112	Oct-14	14838
loan_status	393465	2	Fully Paid	316271
purpose	393465	14	debt_consolidation	233108
title	393465	48460	Debt consolidation	152392
earliest_cr_line	393465	683	Oct-00	2999
initial_list_status	393465	2	f	236947
application_type	393465	3	INDIVIDUAL	392844
state	393465	54	AP	14199

From above describe, we can see that emp_title and title are of no use as they have very large number of unique values in them. So, we will try to remove the number of unique values by target encoding or either we will drop the whole column. So, here we will do the target encoding for emp_title and we will drop the title column.

[99]:
```
df.drop(['title'],axis='columns',inplace=True)
```

[100]:
```
cat_features=list(df.select_dtypes('object').columns)
for i in cat_features:
  print('The unique values in {0} are {1}'.format(i,df[i].unique()))
```

```
The unique values in term are [' 36 months' ' 60 months']
The unique values in grade are ['B' 'A' 'C' 'E' 'D' 'F' 'G']
The unique values in sub_grade are ['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']
The unique values in emp_title are ['Marketing' 'Credit analyst ' 'Statistician'
…
 "Michael's Arts & Crafts" 'licensed bankere' 'Gracon Services, Inc']
The unique values in home_ownership are ['RENT' 'MORTGAGE' 'OWN' 'OTHER' 'ANY'
'NONE']
The unique values in verification_status are ['Not Verified' 'Source Verified'
'Verified']
The unique values in issue_d are ['Jan-15' 'Nov-14' 'Apr-13' 'Sep-15' 'Sep-12'
'Oct-14' 'Apr-12' 'Jun-13'
 'May-14' 'Dec-15' 'Apr-15' 'Oct-12' 'Jul-14' 'Feb-13' 'Oct-15' 'Jan-14'
 'Mar-16' 'Apr-14' 'Jun-11' 'Apr-10' 'Jun-14' 'Oct-13' 'May-13' 'Feb-15'
 'Oct-11' 'Jun-15' 'Aug-13' 'Feb-14' 'Dec-11' 'Mar-13' 'Jun-16' 'Mar-14'
 'Nov-13' 'Dec-14' 'Apr-16' 'Sep-13' 'May-16' 'Jul-15' 'Jul-13' 'Aug-14'
 'May-08' 'Mar-10' 'Dec-13' 'Mar-12' 'Mar-15' 'Sep-11' 'Jul-12' 'Dec-12'
 'Sep-14' 'Nov-12' 'Nov-15' 'Jan-11' 'May-12' 'Feb-16' 'Jun-12' 'Aug-12'
 'Jan-16' 'May-15' 'Oct-16' 'Aug-15' 'Jul-16' 'May-09' 'Aug-16' 'Jan-12'
 'Jan-13' 'Nov-10' 'Jul-11' 'Mar-11' 'Feb-12' 'May-11' 'Aug-10' 'Nov-16'
 'Jul-10' 'Sep-10' 'Dec-10' 'Feb-11' 'Jun-09' 'Aug-11' 'Dec-16' 'Mar-09'
 'Jun-10' 'May-10' 'Nov-11' 'Sep-16' 'Oct-09' 'Nov-08' 'Dec-09' 'Oct-10'
 'Sep-09' 'Aug-09' 'Jul-09' 'Nov-09' 'Jan-10' 'Dec-08' 'Feb-09' 'Oct-08'
 'Apr-09' 'Feb-10' 'Apr-11' 'Apr-08' 'Aug-08' 'Jan-09' 'Sep-08' 'Jun-08'
 'Jul-08' 'Mar-08' 'Oct-07' 'Dec-07' 'Feb-08' 'Jan-08' 'Nov-07' 'Aug-07']
The unique values in loan_status are ['Fully Paid' 'Charged Off']
The unique values in purpose are ['vacation' 'debt_consolidation' 'credit_card'
'home_improvement'
 'small_business' 'major_purchase' 'other' 'medical' 'wedding' 'car'
 'moving' 'house' 'educational' 'renewable_energy']
The unique values in earliest_cr_line are ['Jun-90' 'Jul-04' 'Aug-07' 'Sep-06'
'Mar-99' 'Jan-05' 'Aug-05' 'Sep-94'
 'Jun-94' 'Dec-97' 'Dec-90' 'May-84' 'Apr-95' 'Jan-97' 'May-01' 'Mar-82'
 'Sep-96' 'Jan-90' 'Mar-00' 'Jan-06' 'Oct-06' 'Jan-03' 'May-08' 'Oct-03'
 'Jun-04' 'Jan-99' 'Apr-94' 'Apr-98' 'Jul-07' 'Apr-02' 'Oct-07' 'Jun-09'
 'May-97' 'Jul-06' 'Sep-03' 'Aug-92' 'Dec-88' 'Feb-02' 'Jan-92' 'Aug-01'
 'Dec-10' 'Oct-99' 'Sep-04' 'Aug-94' 'Jul-03' 'Apr-00' 'Dec-04' 'Jun-95'
 'Dec-03' 'Jul-94' 'Oct-90' 'Dec-01' 'Apr-99' 'Feb-95' 'May-03' 'Oct-02'
 'Mar-04' 'Aug-03' 'Oct-00' 'Nov-04' 'Mar-10' 'Mar-96' 'May-94' 'Jun-96'
 'Nov-86' 'Jan-01' 'Jan-02' 'Mar-01' 'Sep-12' 'Apr-06' 'May-98' 'Dec-02'
 'Nov-03' 'Oct-05' 'May-90' 'Jun-03' 'Jun-01' 'Jan-98' 'Oct-78' 'Feb-01'
 'Jun-06' 'Aug-93' 'Apr-01' 'Nov-01' 'Feb-03' 'Jun-93' 'Sep-92' 'Nov-92'
 'Jun-83' 'Oct-01' 'Jul-99' 'Sep-97' 'Nov-93' 'Feb-93' 'Apr-07' 'Nov-99'
 'Nov-05' 'Dec-92' 'Mar-86' 'May-89' 'Dec-00' 'Mar-91' 'Mar-05' 'Jun-10'
 'Dec-98' 'Sep-01' 'Nov-00' 'Jan-94' 'Aug-02' 'Jan-11' 'Aug-08' 'Jun-05'
 'Nov-97' 'May-96' 'Apr-10' 'May-93' 'Sep-05' 'Jun-92' 'Apr-86' 'Aug-96'
 'Aug-97' 'Jul-05' 'May-11' 'Sep-02' 'Jan-89' 'Aug-99' 'Feb-92' 'Sep-99'
 'Jul-01' 'May-80' 'Oct-08' 'Nov-07' 'Apr-97' 'Jun-86' 'Sep-98' 'Jun-82'
 'Oct-81' 'Feb-94' 'Dec-84' 'Nov-91' 'Nov-06' 'Aug-00' 'Oct-04' 'Jun-11'
```

```
'Apr-88' 'May-04' 'Aug-88' 'Mar-94' 'Aug-04' 'Dec-06' 'Nov-98' 'Oct-97'
'Mar-89' 'Feb-88' 'Jul-82' 'Nov-95' 'Mar-97' 'Oct-94' 'Jul-98' 'Jun-02'
'May-91' 'Oct-11' 'Sep-07' 'Jan-07' 'Jan-10' 'Mar-87' 'Feb-97' 'Oct-86'
'Mar-02' 'Jul-93' 'Mar-07' 'Aug-89' 'Oct-95' 'May-07' 'Dec-93' 'Jun-89'
'Apr-04' 'Jun-97' 'Apr-96' 'Apr-92' 'Oct-98' 'Mar-83' 'Mar-85' 'Oct-93'
'Feb-00' 'Apr-03' 'Oct-85' 'Jul-85' 'May-78' 'Sep-10' 'Oct-96' 'Sep-09'
'Jun-99' 'Jan-00' 'Sep-87' 'Aug-98' 'Jan-95' 'Jul-88' 'May-00' 'Jun-81'
'Feb-98' 'Nov-96' 'Aug-67' 'Dec-99' 'Aug-06' 'Nov-09' 'Jul-00' 'Mar-88'
'Jul-92' 'Jul-91' 'Mar-90' 'May-86' 'Jun-91' 'Dec-87' 'Jul-96' 'Jul-97'
'Aug-90' 'Jan-88' 'Dec-05' 'Mar-03' 'Feb-99' 'Nov-90' 'Jun-00' 'Dec-96'
'Jan-04' 'May-99' 'Sep-72' 'Jul-81' 'Sep-93' 'Feb-09' 'Nov-02' 'Nov-69'
'Jan-93' 'May-05' 'Sep-82' 'Apr-90' 'Feb-96' 'Mar-93' 'Apr-78' 'Jul-95'
'May-95' 'Apr-91' 'Mar-98' 'Aug-91' 'Jul-02' 'Oct-89' 'Apr-84' 'Dec-09'
'Sep-00' 'Jan-82' 'Jun-98' 'Jan-96' 'Nov-87' 'May-10' 'Jul-89' 'Jun-87'
'Oct-87' 'Aug-95' 'Feb-04' 'Oct-91' 'Dec-89' 'Oct-92' 'Feb-05' 'Apr-93'
'Dec-85' 'Sep-79' 'Feb-07' 'Nov-89' 'Apr-05' 'Mar-78' 'Sep-85' 'Nov-94'
'Jun-08' 'Apr-87' 'Dec-83' 'Dec-07' 'May-79' 'May-92' 'Jul-90' 'Mar-95'
'Feb-06' 'Feb-85' 'Sep-89' 'Aug-09' 'Nov-08' 'Nov-81' 'Jan-08' 'Aug-87'
'Nov-85' 'Dec-65' 'Sep-95' 'Jan-86' 'Oct-09' 'May-02' 'Aug-80' 'Sep-77'
'Sep-88' 'Oct-84' 'May-88' 'Aug-84' 'Nov-88' 'May-74' 'Nov-82' 'Oct-83'
'Sep-91' 'Feb-84' 'Feb-91' 'Jan-81' 'Jun-85' 'Dec-76' 'Dec-94' 'Dec-80'
'Sep-84' 'Jun-07' 'Aug-79' 'Sep-08' 'Apr-83' 'Mar-06' 'Jun-84' 'Jul-84'
'Jan-85' 'Dec-95' 'Apr-08' 'Mar-08' 'Jan-83' 'Dec-86' 'Jun-79' 'Dec-75'
'Jul-86' 'Nov-77' 'Dec-82' 'May-85' 'Feb-83' 'Aug-82' 'Oct-80' 'Mar-79'
'Jan-78' 'Mar-84' 'Nov-83' 'May-83' 'Jul-08' 'Apr-82' 'Jul-83' 'Feb-90'
'Dec-08' 'Jul-75' 'Dec-71' 'Feb-08' 'Mar-11' 'Feb-87' 'Feb-89' 'Aug-85'
'Jul-10' 'Apr-89' 'Feb-80' 'May-06' 'Nov-10' 'Apr-09' 'Feb-10' 'May-76'
'Feb-81' 'Jan-12' 'Oct-88' 'Nov-84' 'May-82' 'Oct-75' 'Jun-88' 'May-72'
'Apr-13' 'Sep-90' 'Oct-82' 'Feb-13' 'Mar-92' 'Aug-81' 'Feb-11' 'Nov-74'
'Feb-78' 'Sep-83' 'Jul-11' 'Nov-79' 'Aug-83' 'Apr-85' 'Jul-09' 'Jan-71'
'Jul-87' 'Aug-78' 'Aug-10' 'Oct-76' 'Aug-86' 'Jan-91' 'Dec-91' 'May-09'
'Aug-11' 'Jun-64' 'Jan-74' 'May-81' 'Jun-72' 'Jun-78' 'Sep-86' 'Jan-87'
'Jan-75' 'Feb-82' 'Jan-80' 'Feb-77' 'Sep-80' 'Nov-78' 'Jul-74' 'Jun-70'
'Jan-84' 'Nov-80' 'May-87' 'Sep-70' 'Jan-76' 'Feb-86' 'Oct-10' 'Apr-79'
'Oct-79' 'Jan-79' 'Sep-11' 'Jul-79' 'Sep-75' 'Mar-81' 'Aug-71' 'Apr-80'
'Apr-77' 'Jan-65' 'Nov-76' 'Nov-70' 'Nov-11' 'Nov-73' 'Sep-81' 'Jul-80'
'Mar-12' 'Dec-74' 'Mar-77' 'Dec-77' 'May-12' 'Dec-79' 'Jan-09' 'Jan-70'
'Dec-11' 'Feb-79' 'Mar-76' 'Jan-73' 'Oct-73' 'Mar-69' 'Oct-77' 'Mar-75'
'Aug-77' 'Jun-69' 'Oct-63' 'Nov-60' 'Aug-70' 'Feb-75' 'Sep-74' 'May-66'
'Apr-72' 'Apr-73' 'Apr-12' 'May-75' 'Sep-66' 'Feb-69' 'Feb-12' 'Jan-61'
'Aug-73' 'Feb-72' 'Apr-75' 'Jul-78' 'Oct-70' 'Mar-80' 'Sep-76' 'Apr-11'
'Nov-12' 'Aug-76' 'Apr-81' 'Mar-09' 'Jun-77' 'Apr-71' 'Sep-69' 'Jun-12'
'Apr-76' 'Feb-65' 'Jul-77' 'Jun-76' 'Mar-73' 'Oct-72' 'Dec-78' 'Jun-75'
'Nov-67' 'Sep-67' 'Nov-71' 'Jun-80' 'May-64' 'Feb-71' 'May-70' 'Apr-70'
'Mar-71' 'Apr-69' 'Jan-63' 'Jun-74' 'Oct-74' 'May-77' 'Dec-81' 'Jan-69'
'Feb-76' 'Mar-70' 'Aug-68' 'Feb-70' 'Jun-71' 'Jun-63' 'Jun-13' 'Mar-72'
'Aug-12' 'Jan-67' 'Feb-68' 'Dec-69' 'Jan-77' 'Jul-70' 'Feb-73' 'Mar-74'
'Feb-74' 'Dec-60' 'Jul-72' 'Jul-73' 'Sep-64' 'Jul-65' 'Oct-58' 'Jul-12'
```

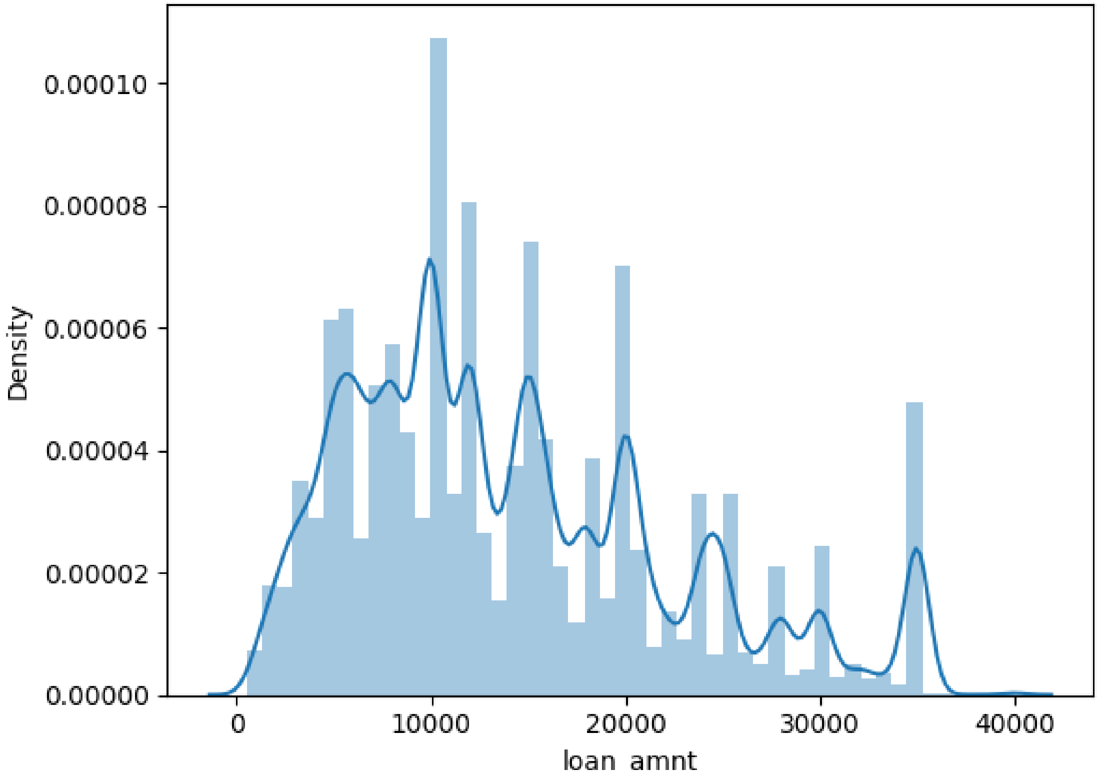
```
 'Jun-73' 'Sep-78' 'Nov-75' 'Jul-63' 'Jan-64' 'Dec-68' 'May-58' 'Sep-73'
 'May-71' 'Dec-72' 'Aug-65' 'Jul-76' 'Oct-12' 'May-73' 'Apr-55' 'Apr-66'
 'Jan-68' 'Nov-68' 'Oct-69' 'Mar-13' 'Jan-13' 'Jul-67' 'Oct-65' 'Jan-66'
 'Aug-72' 'Jul-69' 'May-65' 'Aug-74' 'May-68' 'Aug-69' 'May-13' 'Oct-67'
 'Aug-75' 'Apr-74' 'Sep-71' 'Apr-68' 'Jul-71' 'Jan-72' 'Nov-65' 'Dec-70'
 'Dec-73' 'Nov-72' 'Oct-59' 'Oct-62' 'Apr-67' 'Oct-71' 'Nov-63' 'Oct-68'
 'Dec-62' 'Jun-60' 'Jan-60' 'Sep-13' 'May-69' 'Dec-66' 'Feb-67' 'Dec-67'
 'Aug-61' 'Sep-68' 'Oct-64' 'Aug-66' 'Jul-66' 'Apr-64' 'Sep-62' 'Jul-13'
 'Jun-67' 'Apr-65' 'Jun-66' 'Jan-55' 'Jan-62' 'Feb-64' 'Aug-58' 'Jul-68'
 'May-67' 'Dec-59' 'Sep-63' 'Dec-12' 'Dec-63' 'Jan-44' 'Jun-65' 'May-62'
 'Mar-67' 'Mar-68' 'Jan-56' 'Sep-65' 'Dec-51' 'Aug-13' 'Jun-68' 'Mar-65'
 'Oct-57' 'Nov-66' 'Dec-58' 'Feb-57' 'Feb-63' 'Mar-63' 'Jan-59' 'May-55'
 'Feb-66' 'Nov-50' 'Mar-64' 'Jan-58' 'Sep-61' 'Apr-63' 'Jul-64' 'Nov-55'
 'Jun-57' 'Dec-64' 'Nov-53' 'Nov-64' 'Apr-61' 'Mar-66' 'Oct-60' 'Jul-59'
 'Jul-61' 'Jan-54' 'Dec-56' 'Mar-62' 'Jul-60' 'Sep-59' 'Dec-50' 'Oct-66'
 'Apr-60' 'Jul-58' 'Nov-54' 'Nov-57' 'Jun-62' 'May-63' 'Jul-55' 'Oct-50'
 'Dec-61' 'Aug-51' 'Oct-13' 'Aug-64' 'Apr-62' 'Jun-55' 'Jul-62' 'Jan-57'
 'Nov-58' 'Jul-51' 'Nov-59' 'Apr-58' 'Mar-60' 'Sep-57' 'Nov-61' 'Sep-60'
 'May-59' 'Jun-59' 'Feb-62' 'Sep-56' 'Aug-60' 'Feb-61' 'Jan-48' 'Aug-63'
 'Oct-61' 'Aug-62' 'Aug-59']
The unique values in initial_list_status are ['w' 'f']
The unique values in application_type are ['INDIVIDUAL' 'JOINT' 'DIRECT_PAY']
The unique values in state are ['OK' 'SD' 'WV' 'MA' 'VA' 'DE' 'TX' 'AE' 'AP'
'NM' 'MS' 'OR' 'NH' 'HI'
 'PA' 'CO' 'AL' 'FL' 'AZ' 'WI' 'NC' 'IN' 'MO' 'AA' 'TN' 'KS' 'ND' 'CT'
 'WY' 'NE' 'RI' 'AR' 'MI' 'IL' 'LA' 'NY' 'IA' 'AK' 'UT' 'MD' 'WA' 'MN'
 'OH' 'MT' 'NJ' 'DC' 'NV' 'VT' 'CA' 'ME' 'ID' 'GA' 'KY' 'SC']
```

[101]:
```python
df['sub_grade']=df['sub_grade'].str[1:]
df['sub_grade']=df['sub_grade'].astype(int)
```

[102]:
```python
sns.distplot(df['loan_amnt'])
```

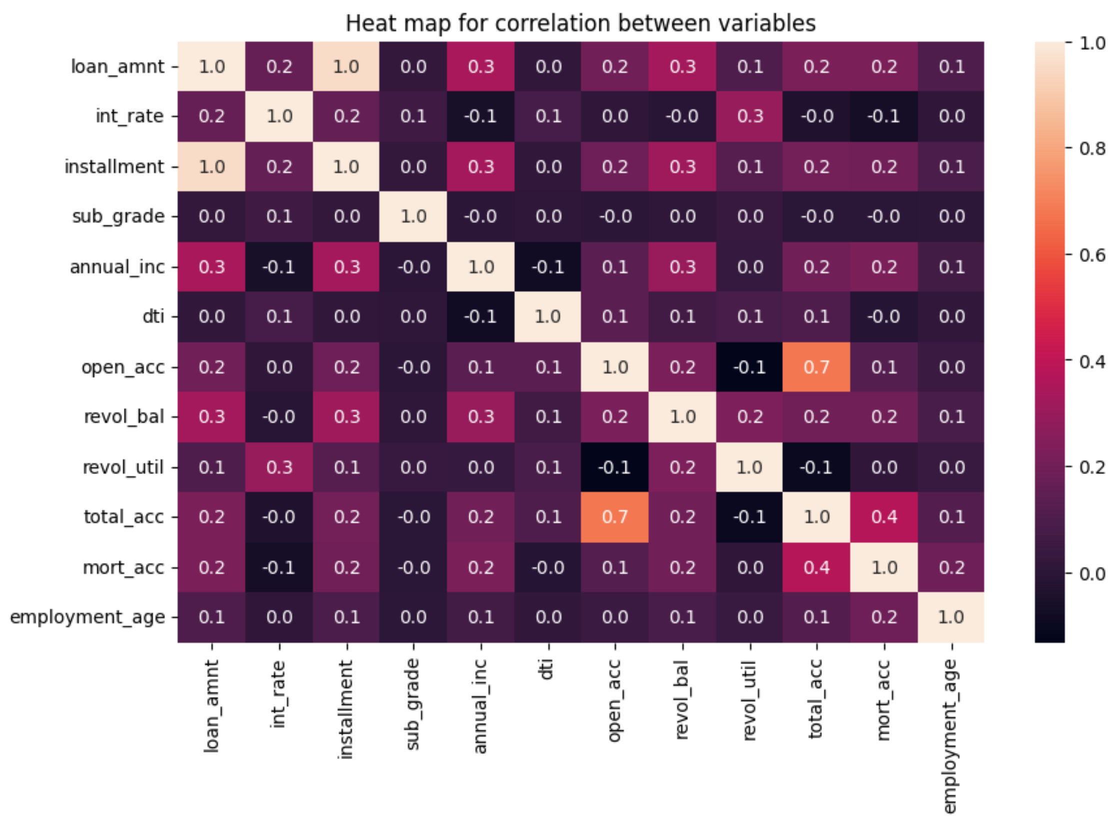
```
<ipython-input-102-6e47c8af991e>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['loan_amnt'])
```

[102]: <Axes: xlabel='loan_amnt', ylabel='Density'>



[103]:
```
##Now, let's see the correlation between the numerical variables
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, fmt=".1f")
plt.title('Heat map for correlation between variables')
plt.show()
```

```
<ipython-input-103-616ec37b365e>:3: FutureWarning: The default value of
numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only
to silence this warning.
  sns.heatmap(df.corr(), annot=True, fmt=".1f")
```

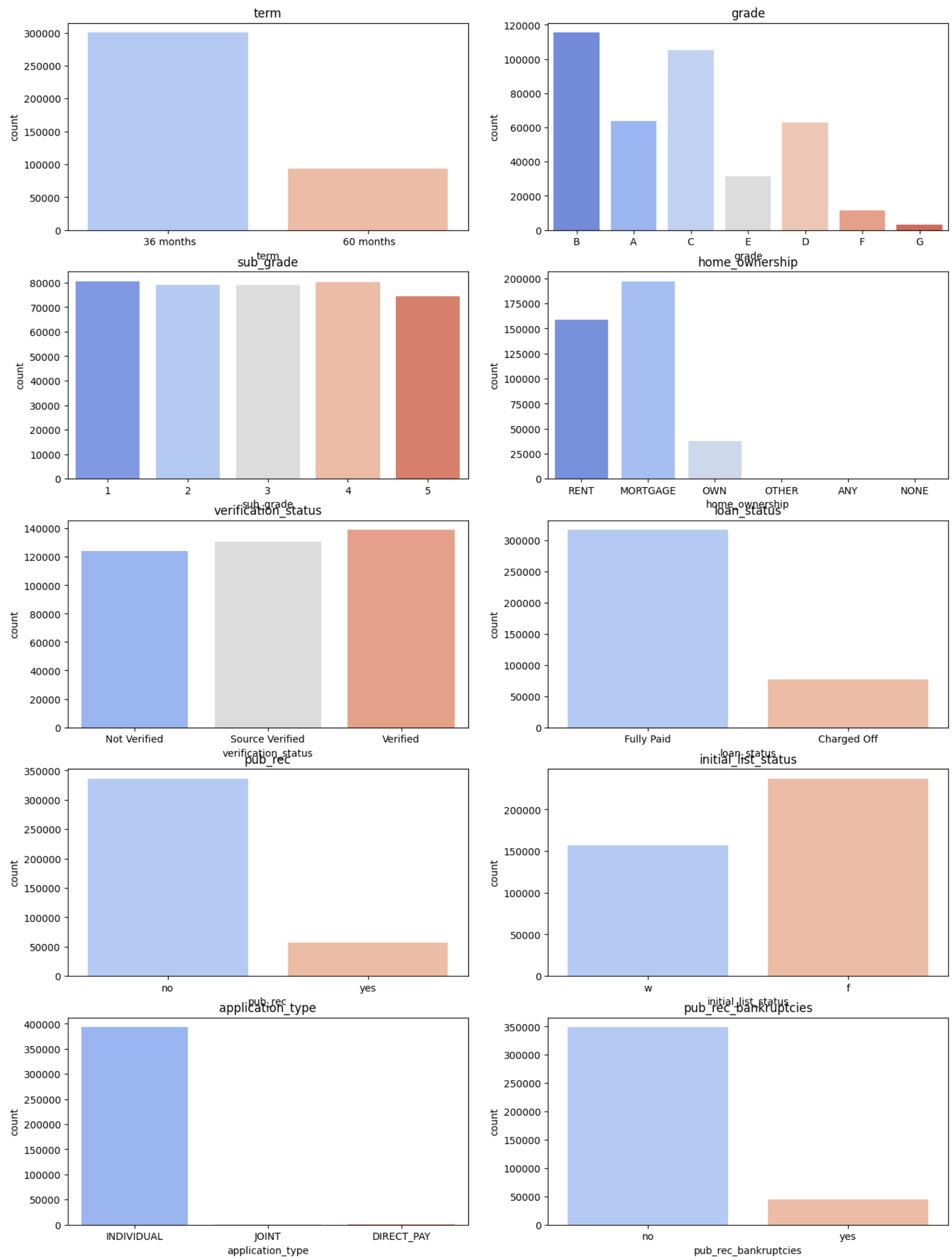


- total_acc is highly correlated with open_acc
- loan_amnt and installment are perfectly correlated
- total_acc is moderately correlated with mort_acc

We can avoid multi-collinearity by removing some of tehse correlated features.

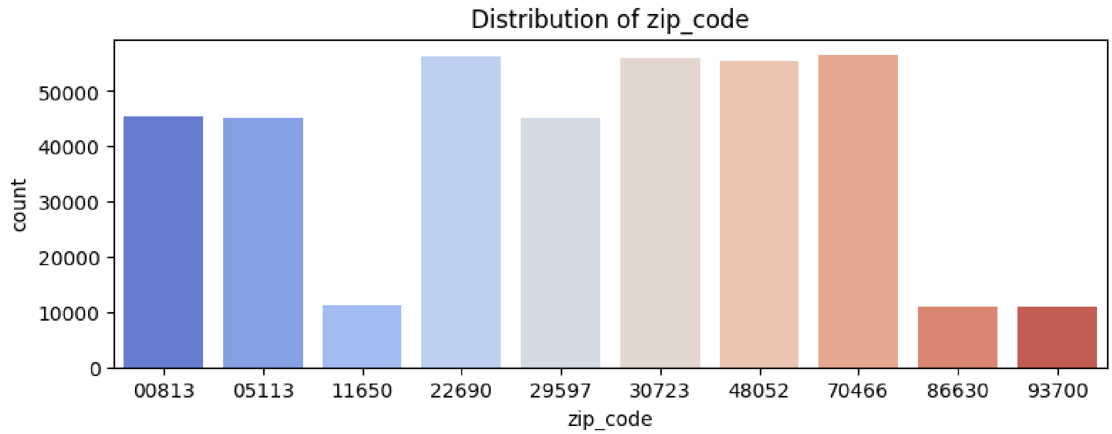
[104]:
```
df.drop(columns=['installment'],inplace=True)
```

[105]:
```
plot=['term','grade','sub_grade','home_ownership','verification_status','loan_status','pub_rec
plt.figure(figsize=(16,22))
j=1
for col in plot:
  ax=plt.subplot(5,2,j)
  sns.countplot(x=df[col],palette='coolwarm')
  plt.title(f'{col}')
  j=j+1
plt.show()
```

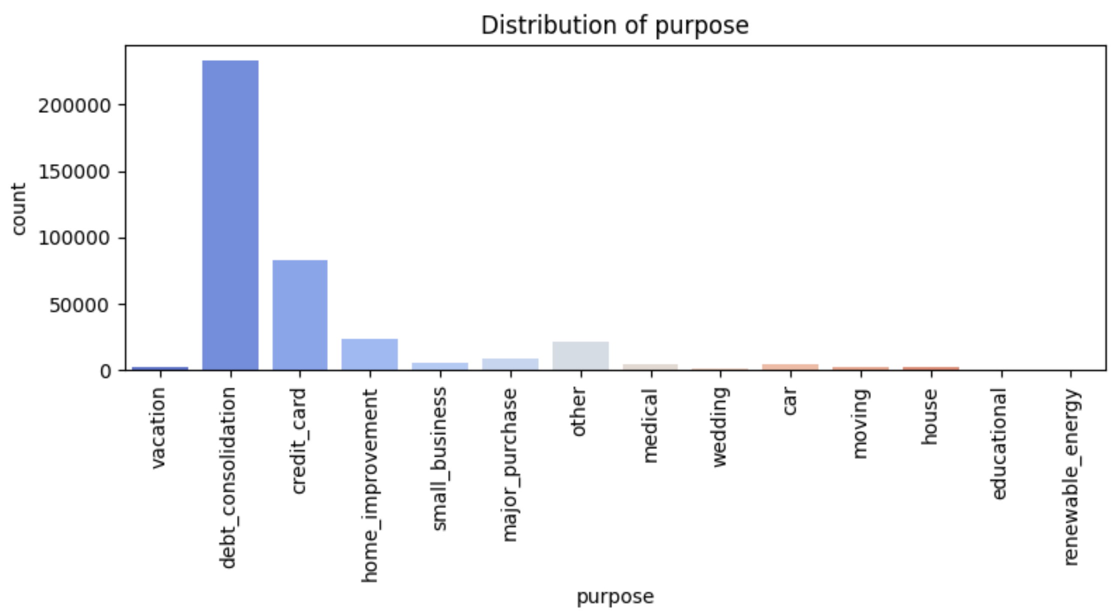


[106]:
```
plt.figure(figsize=(9,3))
sns.countplot(x=df['zip_code'],palette='coolwarm')
plt.title('Distribution of zip_code')
```

```
plt.show()
```



[107]:
```
plt.figure(figsize=(9,3))
sns.countplot(x=df['purpose'],palette='coolwarm')
plt.xticks(rotation=90)
plt.title('Distribution of purpose')
plt.show()
```



Almost 85% of applicants don't have a public record/haven't filed for bankruptcy.

99% applicants have applied for individual application type.

55% of loans are taken for the purpose of debt consolidation.

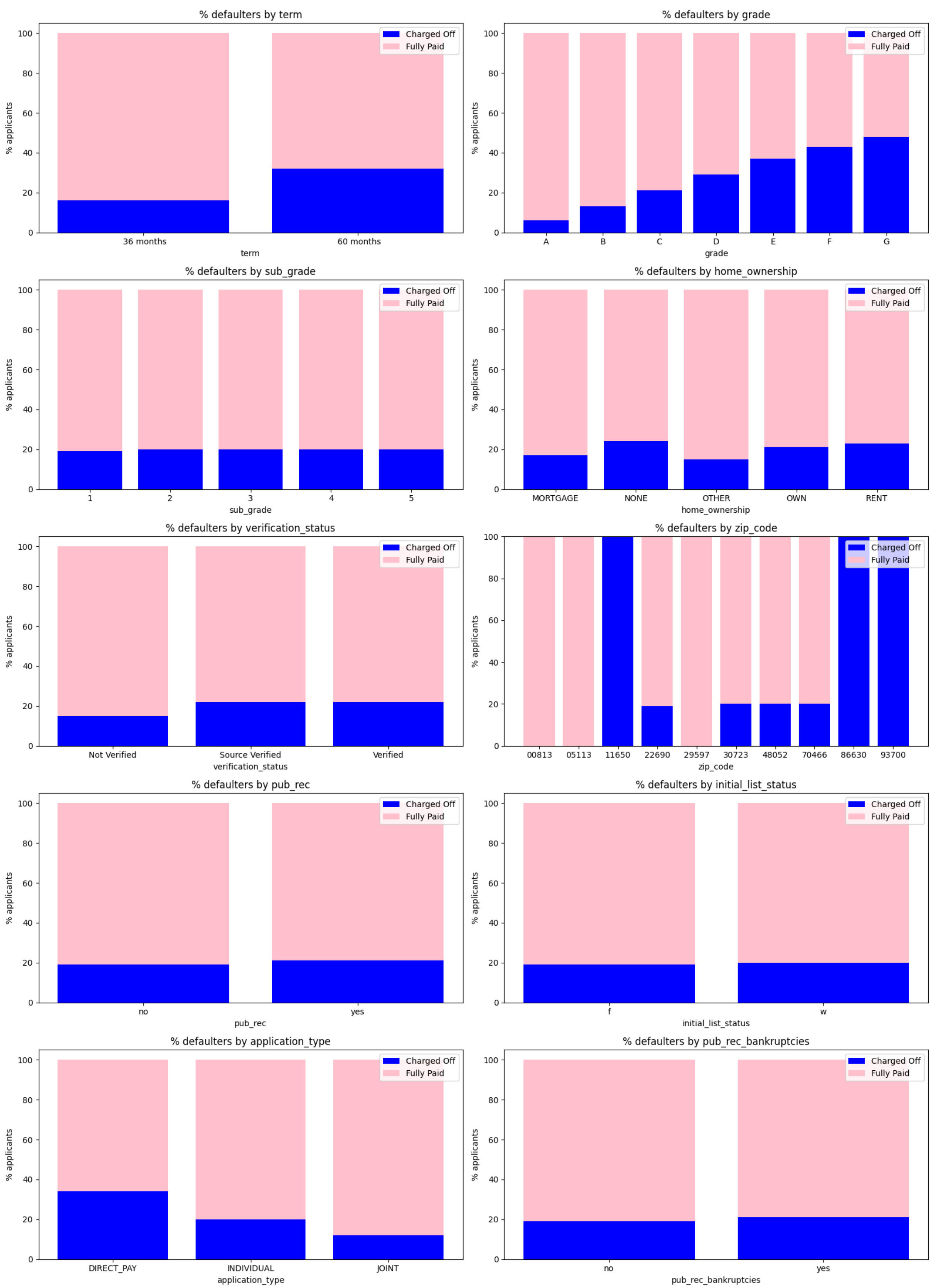
20% of loans are taken on credit card.

Almost 80% of the loans are for 36 months.

Maximum loans fall in B grade.

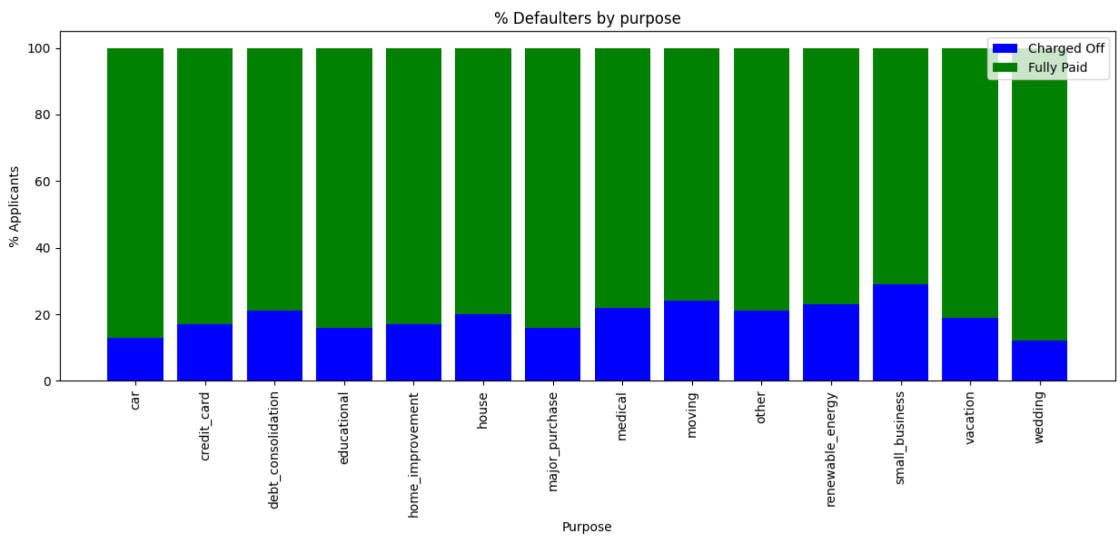
The type of home ownership for 50% cases is morgage.

[108]:
```
##finding the impact of categorical factors on loan status
plot_dig=['term','grade','sub_grade','home_ownership','verification_status','zip_code','pub_re
plt.figure(figsize=(16,22))
j=1
for x in plot_dig:
  ax=plt.subplot(5,2,j)
  data_1=df.
 ↪pivot_table(index=x,columns='loan_status',aggfunc='count',values='purpose')
  data_1=data_1.div(data_1.sum(axis=1),axis=0).multiply(100).round()
  data_1.reset_index(inplace=True)
  plt.bar(data_1[x],data_1['Charged Off'],color='blue')
  plt.bar(data_1[x],data_1['Fully Paid'],color='pink',bottom=data_1['Charged␣
 ↪Off'])
  plt.ylabel('% applicants')
  plt.xlabel(f'{x}')
  plt.title(f'% defaulters by {x}')
  plt.legend(['Charged Off','Fully Paid'])
  j=j+1

plt.tight_layout()
plt.show()
```


% defaulters by term
Charged Off
Fully Paid
% applicants
36 months
60 months
term
% defaulters by grade
A
B
C
D
E
F
G
grade
% defaulters by sub_grade
1
2
3
4
5
sub_grade
% defaulters by home_ownership
MORTGAGE
NONE
OTHER
OWN
RENT
home_ownership
% defaulters by verification_status
Not Verified
Source Verified
Verified
verification_status
% defaulters by zip_code
00813
05113
11650
22690
29597
30723
48052
70466
86630
93700
zip_code
% defaulters by pub_rec
no
yes
pub_rec
% defaulters by initial_list_status
f
w
initial_list_status
% defaulters by application_type
DIRECT_PAY
INDIVIDUAL
JOINT
application_type
% defaulters by pub_rec_bankruptcies
no
yes
pub_rec_bankruptcies

[109]:

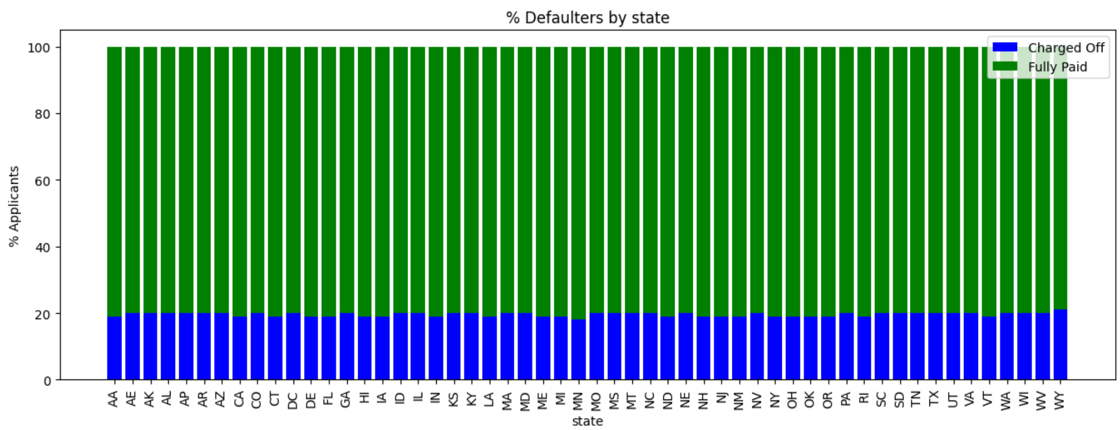
```
##finding impact of purpose on loan status
purpose=df.
 ↪pivot_table(index='purpose',columns='loan_status',aggfunc='count',values='sub_grade')
purpose=purpose.div(purpose.sum(axis=1),axis=0).multiply(100).round()
purpose.reset_index(inplace=True)

plt.figure(figsize=(15,5))
plt.bar(purpose['purpose'], purpose['Charged Off'], color='blue')
plt.bar(purpose['purpose'], purpose['Fully Paid'],␣
 ↪color='green',bottom=purpose['Charged Off'])
plt.xlabel('Purpose')
plt.ylabel('% Applicants')
plt.title('% Defaulters by purpose')
plt.legend(['Charged Off', 'Fully Paid'])
plt.xticks(rotation=90)
plt.show()
```



[110]:

```
##finding impact of state on loan status
state=df.
 ↪pivot_table(index='state',columns='loan_status',aggfunc='count',values='sub_grade')
state=state.div(state.sum(axis=1),axis=0).multiply(100).round()
state.reset_index(inplace=True)

plt.figure(figsize=(15,5))
plt.bar(state['state'], state['Charged Off'], color='blue')
plt.bar(state['state'], state['Fully Paid'],␣
 ↪color='green',bottom=state['Charged Off'])
plt.xlabel('state')
plt.ylabel('% Applicants')
```

```python
plt.title('% Defaulters by state')
plt.legend(['Charged Off', 'Fully Paid'])
plt.xticks(rotation=90)
plt.show()
```

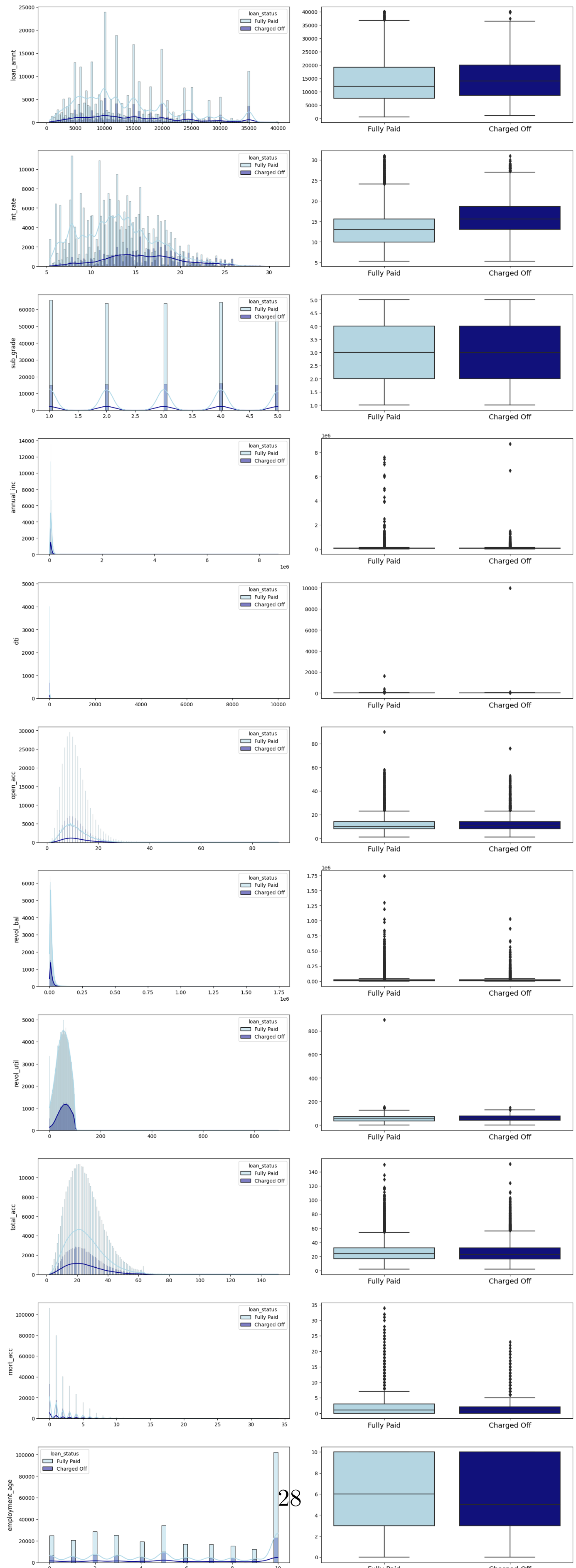


##Observations:-

- The percent of defaulters is much higher for long term (60 months) loans.
- As expected, grade has the maximum impact on loan status. The loan having the highest grade are having the maximum defaulters.
- Loan taken for small business has highest rate of default.
- We can remove state and initial_list_status as they don't have major impact on loan status.

[ ]:
```python
##finding impact of numerical features on loan status
num_cols = df.select_dtypes(include='number').columns
fig, ax = plt.subplots(len(num_cols), 2, figsize=(16, 4 * len(num_cols)))

color_dict = {'Fully Paid': '#add8e8', 'Charged Off': '#00008c'}

for i, col in enumerate(num_cols):

  sns.histplot(data=df, x=col, hue='loan_status', ax=ax[i, 0], legend=True,
               palette=color_dict, kde=True, fill=True)
  sns.boxplot(data=df, y=col, x='loan_status', ax=ax[i, 1],
              palette=list(color_dict.values()))

  ax[i, 0].set_ylabel(col, fontsize=12)
  ax[i, 0].set_xlabel(' ')
  ax[i, 1].set_xlabel(' ')
  ax[i, 1].set_ylabel(' ')
  ax[i, 1].xaxis.set_tick_params(labelsize=14)
```

```
plt.tight_layout()
plt.show()
```


loan_amnt
int_rate
sub_grade
annual_inc
dti
open_acc
revol_bal
revol_util
total_acc
mort_acc
employment_age
loan_status
Fully Paid
Charged Off
1e6

Observing the boxplots, it's evident that defaulters tend to exhibit slightly higher mean values for loan_amnt, int_rate, dti, open_acc, and revol_util, while the annual income is comparatively lower.

```
[ ]: ##now let's remove columns that do not have much impact on loan status
df.
  ↪drop(columns=['initial_list_status','state','emp_title','issue_d','sub_grade','earliest_cr_
```

##One Hot Encoding

```
[ ]: dummies = ['purpose', 'zip_code', 'grade', 'verification_status',␣
  ↪'application_type', 'home_ownership']
df = pd.get_dummies(df, columns=dummies, drop_first=True)
```

```
[121]: pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

df.head()
```

```
[121]:    loan_amnt        term  int_rate  annual_inc  loan_status    dti  open_acc  \
0      10000   36 months     11.44    117000.0   Fully Paid  26.24        16
1       8000   36 months     11.99     65000.0   Fully Paid  22.05        17
2      15600   36 months     10.49     43057.0   Fully Paid  12.79        13
3       7200   36 months      6.49     54000.0   Fully Paid   2.60         6
4      24375   60 months     17.27     55000.0  Charged Off  33.95        13

  pub_rec  revol_bal  revol_util  total_acc  mort_acc pub_rec_bankruptcies  \
0      no      36369        41.8         25       0.0                   no
1      no      20131        53.3         27       3.0                   no
2      no      11987        92.2         26       0.0                   no
3      no       5472        21.5         13       0.0                   no
4      no      24584        69.8         43       1.0                   no

   employment_age  purpose_credit_card  purpose_debt_consolidation  \
0              10                    0                           0
1               4                    0                           1
2               0                    1                           0
3               6                    1                           0
4               9                    1                           0

   purpose_educational  purpose_home_improvement  purpose_house  \
0                    0                         0              0
1                    0                         0              0
2                    0                         0              0
3                    0                         0              0
```

```
4                    0                          0              0

   purpose_major_purchase  purpose_medical  purpose_moving  purpose_other  \
0                       0                0               0              0
1                       0                0               0              0
2                       0                0               0              0
3                       0                0               0              0
4                       0                0               0              0

   purpose_renewable_energy  purpose_small_business  purpose_vacation  \
0                         0                       0                 1
1                         0                       0                 0
2                         0                       0                 0
3                         0                       0                 0
4                         0                       0                 0

   purpose_wedding  zip_code_05113  zip_code_11650  zip_code_22690  \
0                0               0               0               1
1                0               1               0               0
2                0               1               0               0
3                0               0               0               0
4                0               0               1               0

   zip_code_29597  zip_code_30723  zip_code_48052  zip_code_70466  \
0               0               0               0               0
1               0               0               0               0
2               0               0               0               0
3               0               0               0               0
4               0               0               0               0

   zip_code_86630  zip_code_93700  grade_B  grade_C  grade_D  grade_E  \
0               0               0        1        0        0        0
1               0               0        1        0        0        0
2               0               0        1        0        0        0
3               0               0        0        0        0        0
4               0               0        0        1        0        0

   grade_F  grade_G  verification_status_Source Verified  \
0        0        0                                    0
1        0        0                                    0
2        0        0                                    1
3        0        0                                    0
4        0        0                                    0

   verification_status_Verified  application_type_INDIVIDUAL  \
0                             0                            1
1                             0                            1
```

```
2                            0                            1
3                            0                            1
4                            1                            1

   application_type_JOINT  home_ownership_MORTGAGE  home_ownership_NONE  \
0                       0                        0                    0
1                       0                        1                    0
2                       0                        0                    0
3                       0                        0                    0
4                       0                        1                    0

   home_ownership_OTHER  home_ownership_OWN  home_ownership_RENT
0                     0                   0                    1
1                     0                   0                    0
2                     0                   0                    1
3                     0                   0                    1
4                     0                   0                    0
```

[122]:
```
df.shape
```

[122]:
```
(393465, 51)
```

[140]:
```
df['loan_status']=df['loan_status'].map({'Fully Paid': 0, 'Charged Off': 1}).
↪astype(int)
```

[123]:
```
df_x=df.drop(columns=['loan_status'])
df_x.reset_index(inplace= True, drop= True)
df_y=df['loan_status']
df_y.reset_index(drop= True, inplace= True)
```

[124]:
```
term_values = {' 36 months': 36, ' 60 months': 60}
df_x['term'] = df_x.term.map(term_values)
```

[125]:
```
# Encoding Binary features into numerical dtype
df_x['pub_rec']=df_x['pub_rec'].map({'no': 0, 'yes':1}).astype(int)
df_x['pub_rec_bankruptcies']=df_x['pub_rec_bankruptcies'].map({'no': 0, 'yes':
↪1}).astype(int)
```

[126]:
```
df_x.head(4)
```

[126]:
```
   loan_amnt  term  int_rate  annual_inc    dti  open_acc  pub_rec  revol_bal  \
0      10000    36     11.44    117000.0  26.24        16        0      36369
1       8000    36     11.99     65000.0  22.05        17        0      20131
2      15600    36     10.49     43057.0  12.79        13        0      11987
3       7200    36      6.49     54000.0   2.60         6        0       5472

   revol_util  total_acc  mort_acc  pub_rec_bankruptcies  employment_age  \
```

```
0        41.8         25       0.0                        0              10
1        53.3         27       3.0                        0               4
2        92.2         26       0.0                        0               0
3        21.5         13       0.0                        0               6

   purpose_credit_card  purpose_debt_consolidation  purpose_educational  \
0                    0                           0                    0
1                    0                           1                    0
2                    1                           0                    0
3                    1                           0                    0

   purpose_home_improvement  purpose_house  purpose_major_purchase  \
0                         0              0                       0
1                         0              0                       0
2                         0              0                       0
3                         0              0                       0

   purpose_medical  purpose_moving  purpose_other  purpose_renewable_energy  \
0                0               0              0                         0
1                0               0              0                         0
2                0               0              0                         0
3                0               0              0                         0

   purpose_small_business  purpose_vacation  purpose_wedding  zip_code_05113  \
0                       0                 1                0               0
1                       0                 0                0               1
2                       0                 0                0               1
3                       0                 0                0               0

   zip_code_11650  zip_code_22690  zip_code_29597  zip_code_30723  \
0               0               1               0               0
1               0               0               0               0
2               0               0               0               0
3               0               0               0               0

   zip_code_48052  zip_code_70466  zip_code_86630  zip_code_93700  grade_B  \
0               0               0               0               0        1
1               0               0               0               0        1
2               0               0               0               0        1
3               0               0               0               0        0

   grade_C  grade_D  grade_E  grade_F  grade_G  \
0        0        0        0        0        0
1        0        0        0        0        0
2        0        0        0        0        0
3        0        0        0        0        0
```

```
   verification_status_Source Verified  verification_status_Verified  \
0                                    0                             0
1                                    0                             0
2                                    1                             0
3                                    0                             0

   application_type_INDIVIDUAL  application_type_JOINT  \
0                            1                       0
1                            1                       0
2                            1                       0
3                            1                       0

   home_ownership_MORTGAGE  home_ownership_NONE  home_ownership_OTHER  \
0                        0                    0                     0
1                        1                    0                     0
2                        0                    0                     0
3                        0                    0                     0

   home_ownership_OWN  home_ownership_RENT
0                   0                    1
1                   0                    0
2                   0                    1
3                   0                    1
```

##Data Preparation For Model

[127]:
```
x_train, x_test, y_train, y_test = train_test_split(df_x,df_y,test_size=0.
↪25,stratify=df_y,random_state=42)
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

[127]:
```
((295098, 50), (295098,), (98367, 50), (98367,))
```

##**MinMaxScaler**: In python, minmaxscaler scales each feature by subtracting its minimum value and tyhen dividing it by the range. The range is equals to Original maximum- Original minimum. This process preserves the original distribution and retains the embedded information.

[128]:
```
numeric_cols = x_train.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = list(set(x_train.columns) - set(numeric_cols))

scaler = MinMaxScaler()
x_train[numeric_cols] = scaler.fit_transform(x_train[numeric_cols])
x_test[numeric_cols] = scaler.transform(x_test[numeric_cols])
x_train.tail()
```

[128]:
```
        loan_amnt  term  int_rate  annual_inc       dti  open_acc  pub_rec  \
251729   0.291139   0.0  0.279314    0.010526  0.002535  0.101124      0.0
19297    0.367089   0.0  0.027659    0.014474  0.002389  0.202247      0.0
160631   0.756329   0.0  0.591352    0.012184  0.001408  0.123596      0.0
```

```
59254    0.620253   1.0  0.688352    0.018421  0.000203  0.101124      1.0
171714   0.240506   0.0  0.427347    0.005000  0.002217  0.044944      0.0

        revol_bal  revol_util  total_acc  mort_acc  pub_rec_bankruptcies  \
251729   0.016544    0.086854   0.147651   0.03125                   0.0
19297    0.021024    0.036087   0.268456   0.09375                   0.0
160631   0.029945    0.062199   0.167785   0.00000                   0.0
59254    0.007665    0.036423   0.201342   0.12500                   0.0
171714   0.006217    0.049647   0.154362   0.03125                   0.0

        employment_age  purpose_credit_card  purpose_debt_consolidation  \
251729             1.0                    0                           1
19297              0.3                    0                           1
160631             1.0                    0                           1
59254              1.0                    0                           1
171714             0.5                    0                           1

        purpose_educational  purpose_home_improvement  purpose_house  \
251729                    0                         0              0
19297                     0                         0              0
160631                    0                         0              0
59254                     0                         0              0
171714                    0                         0              0

        purpose_major_purchase  purpose_medical  purpose_moving  \
251729                       0                0               0
19297                        0                0               0
160631                       0                0               0
59254                        0                0               0
171714                       0                0               0

        purpose_other  purpose_renewable_energy  purpose_small_business  \
251729              0                         0                       0
19297               0                         0                       0
160631              0                         0                       0
59254               0                         0                       0
171714              0                         0                       0

        purpose_vacation  purpose_wedding  zip_code_05113  zip_code_11650  \
251729                 0                0               0               0
19297                  0                0               0               0
160631                 0                0               0               1
59254                  0                0               0               0
171714                 0                0               0               1

        zip_code_22690  zip_code_29597  zip_code_30723  zip_code_48052  \
251729               0               1               0               0
```

```
19297                0                0                0                0
160631               0                0                0                0
59254                0                0                0                0
171714               0                0                0                0

        zip_code_70466  zip_code_86630  zip_code_93700  grade_B  grade_C  \
251729               0               0               0        1        0
19297                1               0               0        0        0
160631               0               0               0        0        0
59254                1               0               0        0        0
171714               0               0               0        0        1

        grade_D  grade_E  grade_F  grade_G  \
251729        0        0        0        0
19297         0        0        0        0
160631        0        1        0        0
59254         0        0        1        0
171714        0        0        0        0

        verification_status_Source Verified  verification_status_Verified  \
251729                                    0                             1
19297                                     0                             1
160631                                    1                             0
59254                                     1                             0
171714                                    0                             1

        application_type_INDIVIDUAL  application_type_JOINT  \
251729                            1                       0
19297                             1                       0
160631                            1                       0
59254                             1                       0
171714                            1                       0

        home_ownership_MORTGAGE  home_ownership_NONE  home_ownership_OTHER  \
251729                        0                    0                     0
19297                         1                    0                     0
160631                        0                    0                     0
59254                         1                    0                     0
171714                        1                    0                     0

        home_ownership_OWN  home_ownership_RENT
251729                   0                    1
19297                    0                    0
160631                   0                    1
59254                    0                    0
171714                   0                    0
```

##Oversampling with SMOTE

[134]:
```
sm_ote=SMOTE(random_state=42)
x_train_res, y_train_res = sm_ote.fit_resample(x_train,y_train.ravel())

print(f"Before OverSampling, count of label 1: {sum(y_train == 1)}")
print(f"Before OverSampling, count of label 0: {sum(y_train == 0)}")
print(f"After OverSampling, count of label 1: {sum(y_train_res == 1)}")
print(f"After OverSampling, count of label 0: {sum(y_train_res == 0)}")
```

```
Before OverSampling, count of label 1: 0
Before OverSampling, count of label 0: 0
After OverSampling, count of label 1: 0
After OverSampling, count of label 0: 0
```

##Logistic Regression

[132]:
```
model = LogisticRegression(max_iter=1000)
model.fit(x_train, y_train)
train_preds = model.predict(x_train)
test_preds = model.predict(x_test)
```

[131]:
```
print(f"The accuracy of logistic regression is {model.score(x_test,y_test)}")
```

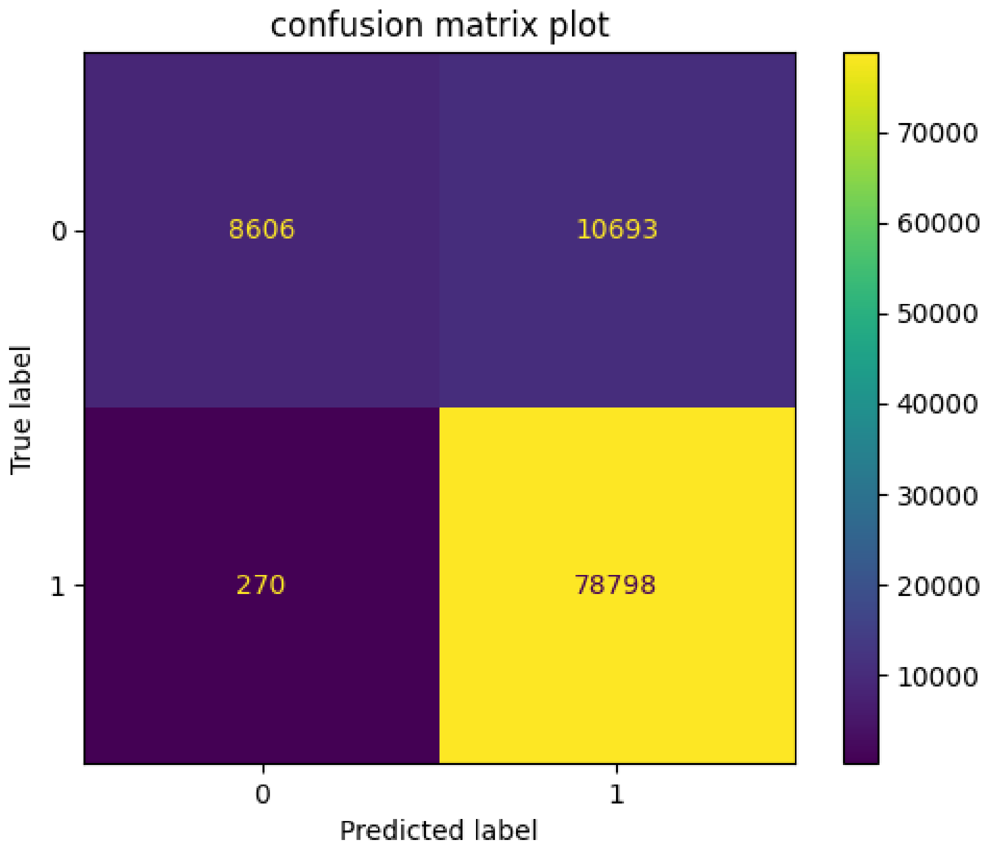
```
The accuracy of logistic regression is 0.8885500218569236
```

##Model Evaluation

[ ]:
```
print('Train Accuracy :', model.score(x_train, y_train).round(2))
print('Train F1 Score:',f1_score(y_train,train_preds).round(2))
print('Train Recall Score:',recall_score(y_train,train_preds).round(2))
print('Train Precision Score:',precision_score(y_train,train_preds).round(2))

print('\nTest Accuracy :',model.score(x_test,y_test).round(2))
print('Test F1 Score:',f1_score(y_test,test_preds).round(2))
print('Test Recall Score:',recall_score(y_test,test_preds).round(2))
print('Test Precision Score:',precision_score(y_test,test_preds).round(2))
```

##Confusion Matrix

[144]:
```
CM=confusion_matrix(y_test, test_preds)
display=ConfusionMatrixDisplay(CM)
display.plot()
plt.title('confusion matrix plot')
plt.show()
```



##Classification Report

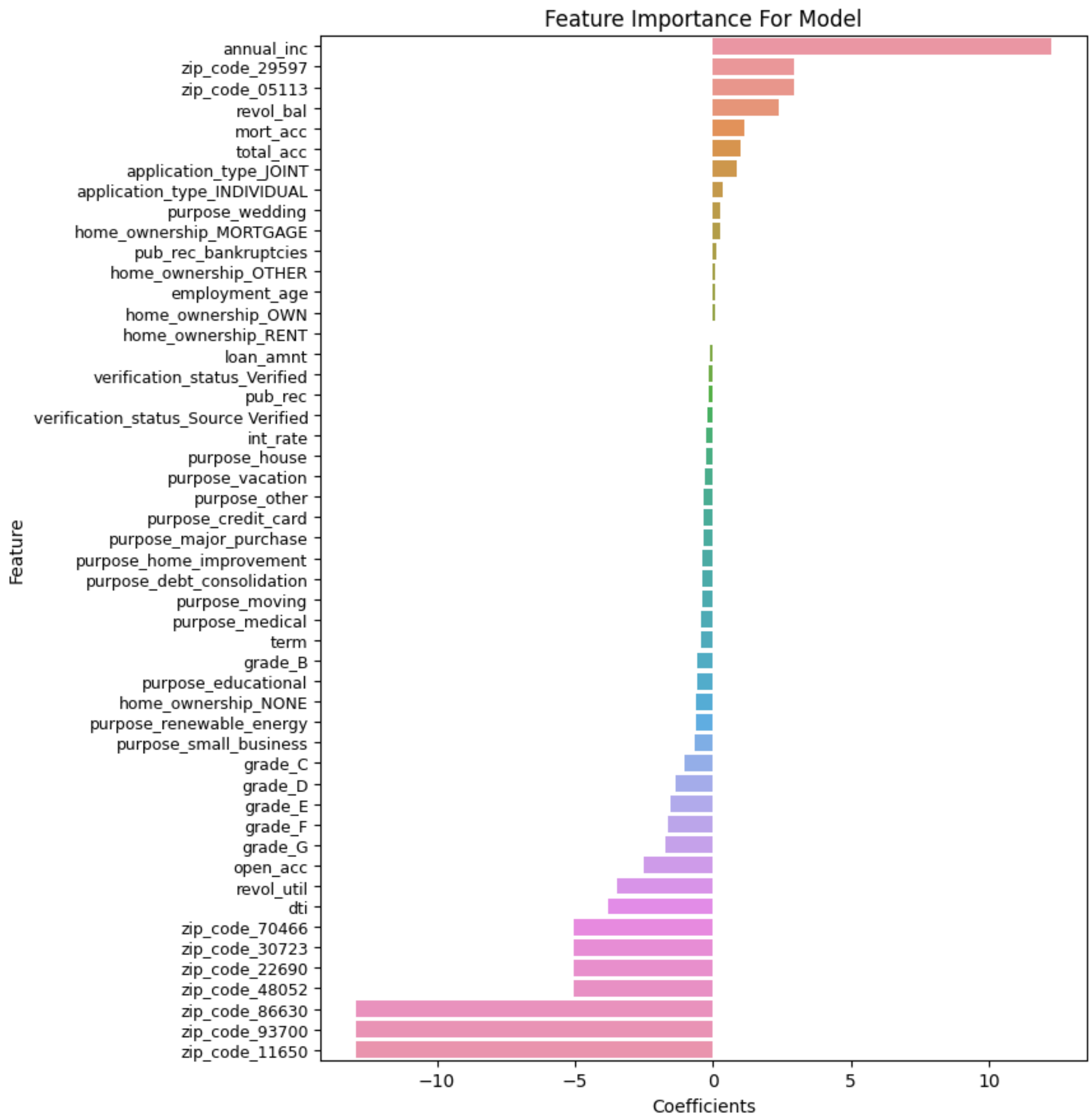
[146]:
```
print(classification_report(y_test,test_preds))
```

```
              precision    recall  f1-score   support

 Charged Off       0.97      0.45      0.61     19299
  Fully Paid       0.88      1.00      0.93     79068

    accuracy                           0.89     98367
   macro avg       0.93      0.72      0.77     98367
weighted avg       0.90      0.89      0.87     98367
```

The model exhibits high recall (80%), low precisoon(50%) for identifying defaulters, leading to a significant number of false positives, while effective at catching actual defaulters, the low precision raises concerns about denying loans to deserving customers. F1 score is impacted, dropping to 60%, despite of 80% accuracy rate. Balancing precision and recall is crucial to avoid unneccessary loan denials and maintain the effectiveness of model.

##Feature Importance

[147]:
```python
feature_imp=pd.DataFrame({'Columns': x_train.columns, 'Coefficients': model.
    ↪coef_[0]}).round(2).sort_values('Coefficients', ascending=False)
plt.figure(figsize=(9,9))
sns.barplot(y=feature_imp['Columns'], x=feature_imp['Coefficients'])
plt.title('Feature Importance For Model')
plt.tight_layout()
plt.yticks(fontsize=9)
plt.ylabel('Feature')
plt.show()
```



From the feature importance plot of model, we can see that the largest weightage is given to annual_income followed by zip_code, revol_bal,.....

##ROC Curve and AUC

An ROC curve depicts a classification models across various threshholds by plotting the true positive rate (recall) against the false positive rate. The true positive rate is the ratio of true positive to the sum of true positives and false negatives, while the false positive rate is the ratio of false positives to the sum of false positives and true negatives.

TPR= (TP)/(TP+FN)

FPR=(FP)/(FP+TN)

Lowering the classification threshold increases both true and false positives.

The curve helps assess the trade off between sensitivity and specificity at different decision thresholds.

The area under the ROC curve (AUC) is a widely used metric for evaluating classifier performance. A perfect classifier scores 1, while a random one scores 0.5. Higher AUC value indicates better discrimation between positive and negative instances reflecting superior classifier performance.

[ ]:
```python
# Predict probabilities for the test set
probs = model.predict_proba(x_test)[:, 1]

# Compute the false positive rate, true positive rate, and thresholds
fpr, tpr, thresholds = roc_curve(y_test, probs)

# Compute the area under the ROC curve
roc_auc = auc(fpr, tpr)

# Plot the ROC curve with modified colors
plt.figure()
plt.plot(fpr, tpr, color='mediumvioletred', lw=2, label=f'ROC curve (AUC =␣
↪{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='darkslategray', lw=2, linestyle='--')
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.show()
```

##Precision Recall Curve

The Precision Recall (PR) curve assesses the performance of a binary classification model by depicting the trade off between precision and recall at different classification thresholds. Precision gauges the accuracy of positive predictions, while recall focuses on capturing all positive instances. The curve is constructed by varying the threshold and plotting recall against precision. The area under the PR curve (AUPRC) serves as a performance metric with a perfect classifier scoring 1 and higher values indicating superior performance when compared to random classifier.

```
[ ]: # Compute the false precision and recall at all thresholds
precision, recall, thresholds = precision_recall_curve(y_test, probs)

# Area under Precision Recall Curve
auprc = average_precision_score(y_test, probs)

# Plot the precision-recall curve
plt.plot(recall, precision, marker='.', label='PR curve (area = %0.2f)' % auprc)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend(loc="lower left")
plt.show()
```

##Tradeoff Questions

Q1.How can we make sure that our model can detect real defaulters and there are less false positives? This is important as we can lose out on an opportunity to finance more individuals and earn interest on it.

Ans. Enhancing the precision score in a model is crucial to reduce the false positives. This ensures that the company makes accurate decisions in identifying and financing (giving loans) to deserving individuals, avoiding missed opportunities.

Q2.Since NPA (non-performing asset) is a real problem in this industry, it's important we play safe and shouldn't disburse loans to anyone.

Ans. Given the prevalent issues of non-performing assets (NPA) in the industry, adopting the cautious approach is imperitive. We should prioritize prudence and refrain from approving loans without careful consideration.

##**Insights** ##1.)Categorical attributes impact on loan_status

- Higher default rates for longer loan terms like 60 months has much higher higher percentage of defaulters.
- Grade/Sub-grade strongly influences loan status.
- Insignificant impact of initial_list_status and state on loan_status, suggesting their removal.
- Loans for small businesses show the highest default rate.
- Direct pay application type has higher default rate than individual/joint applications.

##2.)Impact of numerical attributes on loan_status

- Defaulters tend to have higher mean values for loan_amnt, int_rate, dti, open_acc, and revol_util.
- Mean annual income is lower for defaulters.

##3.) Logistic Regression model performance

- Balanced data training achieves 80% accuracy.

- Negative class: Precision 95%, Recall 80%, F1 score 87%.
- Positive class: Precision 49%, Recall 81%, F1 score 61%.

##4.) ROC and Precision-Recall Curve analysis

- ROC curve AUC of 0.91 indicates strong class differentiation.
- Precision-Recall curve AUC of 0.78 suggests room for improvement through hyperparameter tuning or increased model complexity.

##**Recommendations**:

- The most effective approach to striking a balance between the risk of rising NPAs due to loans to defaulters and the opportunity to extend loans to deserving customers, aiming to optimize the F1 score while considering the precision-recall trade-off. This strategy seeks to find a sweet spot that maximizes the overall performance, considering both the precision-recall curve's area and the F1 score.
- Sophisticated classifiers such as random forest tend to outperform logistic regression since they aren't confined by the linear nature of decision boundaries. Unlike logistic regression, these advanced models can capture intricate relationships and patterns in the data, leading to more accurate and nuanced predictions.

##**Questionnaire**

1.) What percentage of customers have fully paid their Loan Amount?

Ans. 80.39%

2.) Comment about the correlation between Loan Amount and Installment features.

Ans. A stronmg correlation of 0.97 exists between loan amount and installment indicating substancial multicollinearity. To mitigate this issue, we opted to eliminate one of the feature leading to the removal of installment variable.

3.) The majority of people have home ownership as "Mortgage".

4.) People with grades 'A' are more likely to fully pay their loan.

Ans. TRUE

5.) Name the top 2 afforded job titles.

Ans. Manager and Teacher

6.) Thinking from a bank's perspective, which metric should our primary focus be on..

Ans. F1 Score

7.) How does the gap in precision and recall affect the bank?

Ans. With the recall score of 0.79 and precision score of 0.5, the imbalance suggests a higher number of false positives compared to false negatives. Low recall implies the potential approval of loans for some actual defaulters while a low precision indicates the likelihood of identifying some credit worthy customers as defaulters.

8.)Which were the features that heavily affected the outcome?

Ans. Annual_income, zip_code- these features have positively affected the outcome. dti, revol_util, open_acc-these features have negatively affected the outcome.

9.)Will the results be affected by geographical location?

Ans. Yes, because of Zip Code.