Business Case: LoanTap Logistic Regression

In [301]:
```
import pandas as pd
pd.set_option('display.max_columns', 500)
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import warnings
warnings.filterwarnings("ignore")

from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, KFold, cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report,precision_recall_curve
from sklearn.preprocessing import MinMaxScaler,StandardScaler

from imblearn.over_sampling import SMOTE
```

Problem Statement

LoanTap is an online platform committed to delivering customized loan products to millennials. They innovate in an otherwise dull loan segment, to deliver instant, flexible loans on consumer friendly terms to salaried professionals and businessmen.

The data science team at LoanTap is building an underwriting layer to determine the creditworthiness of MSMEs as well as individuals.

LoanTap different types of loans, This case study will focus on the underwriting process behind Personal Loan only and determine if a credit line should be extended to them. If so, what should the repayment terms be in business recommendations.

Since this is a Classification problem, here both Precision and Recall are important because comprimising on precision leads to loss in opportunity to give loans to good customers where we could earn more money and comprimising on recall leads to increase in NPA which will affect in profitability of the finance company, so lets build a model we get the best of both.

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

Out[2]:

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

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

Out[3]: (396030, 27)

In [4]:
```
df.dtypes
```

Out[4]:
```
loan_amnt                float64
term                      object
int_rate                 float64
installment              float64
grade                     object
sub_grade                 object
emp_title                 object
emp_length                object
home_ownership            object
annual_inc               float64
verification_status       object
issue_d                   object
loan_status               object
purpose                   object
title                     object
dti                      float64
earliest_cr_line          object
open_acc                 float64
pub_rec                  float64
revol_bal                float64
revol_util               float64
total_acc                float64
initial_list_status       object
application_type          object
mort_acc                 float64
pub_rec_bankruptcies     float64
address                   object
dtype: object
```

In [5]:
```
df['issue_d'] = pd.to_datetime(df['issue_d'])  ## converting object types to 
df['earliest_cr_line'] = pd.to_datetime(df['earliest_cr_line'])
```

In [ ]:
```

```

In [6]:
```
df.describe().T  # Statistical summary of numerical columns
```

Out[6]:

	count	mean	std	min	25%	50%	75
loan_amnt	396030.0	14113.888089	8357.441341	500.00	8000.00	12000.00	20000.
int_rate	396030.0	13.639400	4.472157	5.32	10.49	13.33	16.
installment	396030.0	431.849698	250.727790	16.08	250.33	375.43	567.
annual_inc	396030.0	74203.175798	61637.621158	0.00	45000.00	64000.00	90000.
dti	396030.0	17.379514	18.019092	0.00	11.28	16.91	22.
open_acc	396030.0	11.311153	5.137649	0.00	8.00	10.00	14.
pub_rec	396030.0	0.178191	0.530671	0.00	0.00	0.00	0.
revol_bal	396030.0	15844.539853	20591.836109	0.00	6025.00	11181.00	19620.
revol_util	395754.0	53.791749	24.452193	0.00	35.80	54.80	72.
total_acc	396030.0	25.414744	11.886991	2.00	17.00	24.00	32.
mort_acc	358235.0	1.813991	2.147930	0.00	0.00	1.00	3.
pub_rec_bankruptcies	395495.0	0.121648	0.356174	0.00	0.00	0.00	0.

In [7]:
```
df.describe(include='object').T # Statistical summary of categorical columns
```

Out[7]:

	count	unique	top	freq
term	396030	2	36 months	302005
grade	396030	7	B	116018
sub_grade	396030	35	B3	26655
emp_title	373103	173105	Teacher	4389
emp_length	377729	11	10+ years	126041
home_ownership	396030	6	MORTGAGE	198348
verification_status	396030	3	Verified	139563
loan_status	396030	2	Fully Paid	318357
purpose	396030	14	debt_consolidation	234507
title	394275	48817	Debt consolidation	152472
initial_list_status	396030	2	f	238066
application_type	396030	3	INDIVIDUAL	395319
address	396030	393700	USCGC Smith\r\nFPO AE 70466	8

In [ ]:

In [ ]:

In [8]:
```python
# defaults % increase with rise in interest rates
```

In [9]:
```python
num_cols = df.columns[df.dtypes=='float64']
cat_cols = df.columns[df.dtypes=='O']
date_cols = df.columns[df.dtypes=='datetime64[ns]']
```

In [ ]:

In [10]:
```python
print("Numeric columns : ", num_cols)
print()
print("Categorical columns : ", cat_cols)
print()
print("datetype columns : ", date_cols)
```

```
Numeric columns :  Index(['loan_amnt', 'int_rate', 'installment', 'annual_in
c', 'dti', 'open_acc',
       'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'mort_acc',
       'pub_rec_bankruptcies'],
      dtype='object')

Categorical columns :  Index(['term', 'grade', 'sub_grade', 'emp_title', 'em
p_length',
       'home_ownership', 'verification_status', 'loan_status', 'purpose',
       'title', 'initial_list_status', 'application_type', 'address'],
      dtype='object')

datetype columns :  Index(['issue_d', 'earliest_cr_line'], dtype='object')
```

In [ ]:

In [11]:
```
(100*df.isna().sum()/df.shape[0]).round(2)   # Percentage of nulls
```

Out[11]:
```
loan_amnt               0.00
term                    0.00
int_rate                0.00
installment             0.00
grade                   0.00
sub_grade               0.00
emp_title               5.79
emp_length              4.62
home_ownership          0.00
annual_inc              0.00
verification_status     0.00
issue_d                 0.00
loan_status             0.00
purpose                 0.00
title                   0.44
dti                     0.00
earliest_cr_line        0.00
open_acc                0.00
pub_rec                 0.00
revol_bal               0.00
revol_util              0.07
total_acc               0.00
initial_list_status     0.00
application_type        0.00
mort_acc                9.54
pub_rec_bankruptcies    0.14
address                 0.00
dtype: float64
```

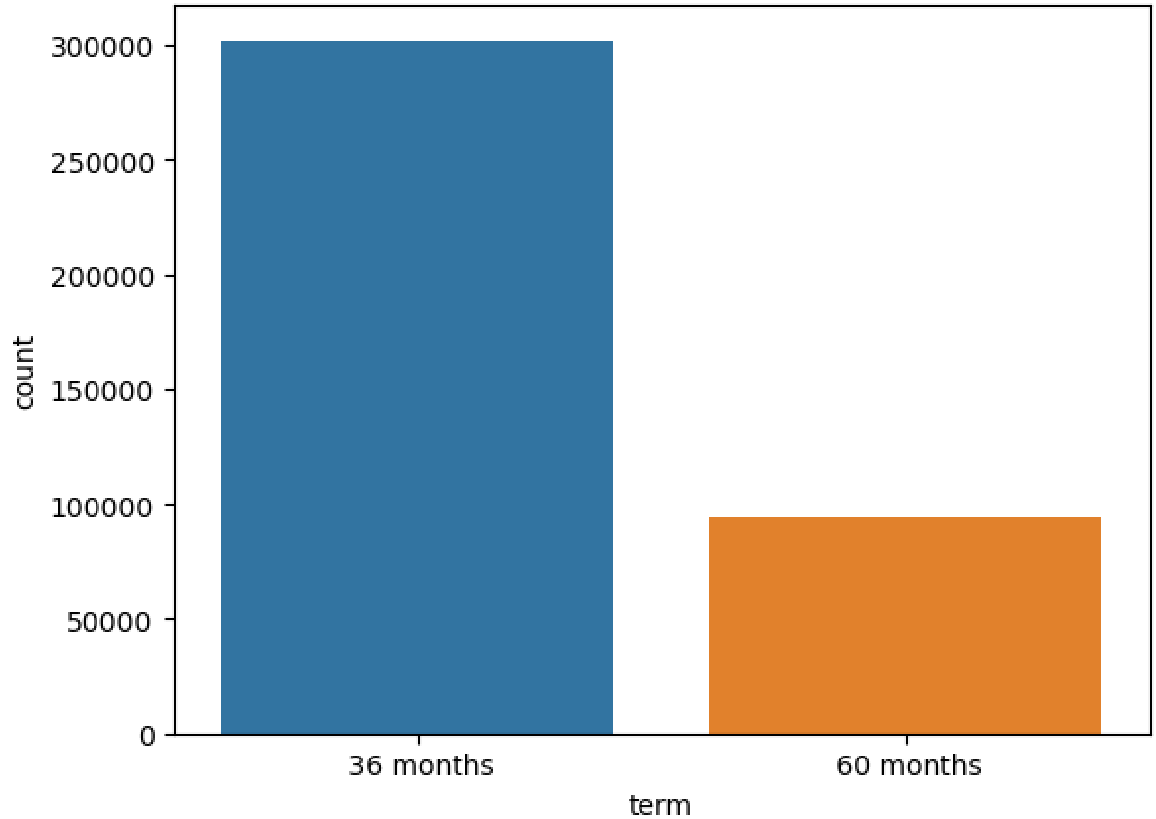
In [12]:
```
def parse_numeric(x):
    return '' if pd.isna(x) else re.findall(r'\d+',x)[0]
```

In [13]:
```
df['term'].unique()
```

Out[13]:
```
array([' 36 months', ' 60 months'], dtype=object)
```

In [14]:
```
sns.countplot(df['term'])
```

Out[14]: <AxesSubplot:xlabel='term', ylabel='count'>



In [15]:
```
df['term'] = df['term'].apply(lambda x : parse_numeric(x)).apply(int)   # con
```

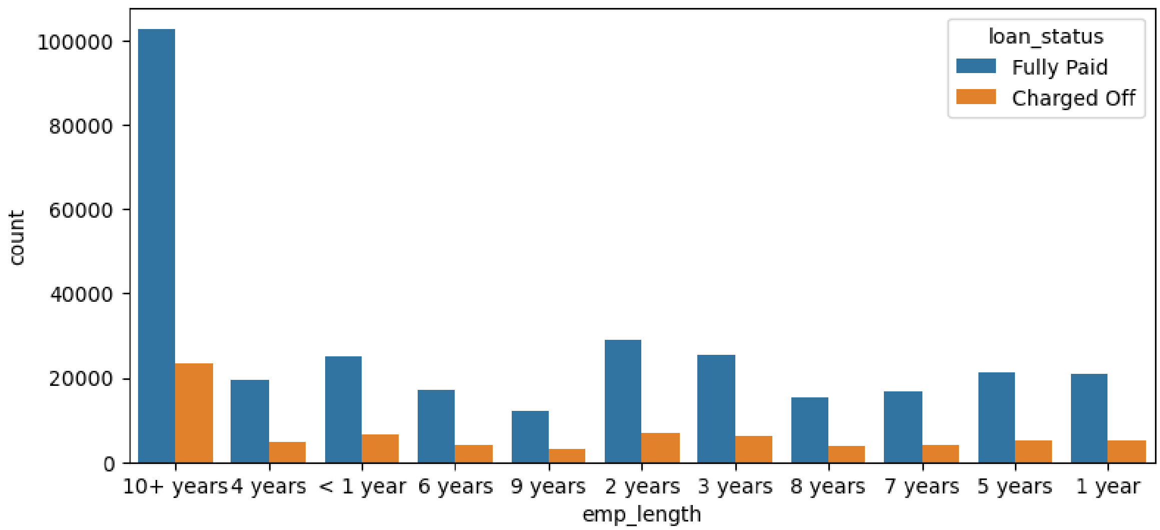
In [ ]:
```

```

In [16]:
```
df['emp_length'].unique()  # We can parse to remove text from the column and 
```

Out[16]:
```
array(['10+ years', '4 years', '< 1 year', '6 years', '9 years',
       '2 years', '3 years', '8 years', '7 years', '5 years', '1 year',
       nan], dtype=object)
```

In [17]:
```
plt.figure(figsize=(9,4))
sns.countplot(df['emp_length'],hue=df['loan_status'])
```

Out[17]: <AxesSubplot:xlabel='emp_length', ylabel='count'>



In [18]:
```
df['emp_length'] = pd.to_numeric(df['emp_length'].apply(lambda x : parse_nume
```

In [19]:
```
df['emp_length'].unique()
```

Out[19]: array([10., 4., 1., 6., 9., 2., 3., 8., 7., 5., nan])

In [ ]:

Filling missing values of emp_len with difference between initial credit line year to issue year as this could give us the approximate employment years

In [20]:
```
temp_emp_length_fill = df['issue_d'].dt.year-df['earliest_cr_line'].dt.year
```

In [21]:
```
temp_emp_length_fill[temp_emp_length_fill>10]=10
```

In [22]:
```
temp_emp_length_fill.value_counts()
```

Out[22]:
```
10    329781
9      16382
8      14369
7      12281
6       9507
5       6969
4       4997
3       1744
dtype: int64
```

In [23]:
```
df['emp_length'] = df['emp_length'].fillna(temp_emp_length_fill)
```

In [24]:
```
df['grade'].unique()  # As grade can be formulated to label encoding we label
```

Out[24]:
```
array(['B', 'A', 'C', 'E', 'D', 'F', 'G'], dtype=object)
```

In [25]:
```
grade_dict = dict(zip(list('ABCDEFG'),[i for i in range(7)]))
df['grade'] = df['grade'].replace(grade_dict)
```

In [ ]:

In [26]:
```
df['sub_grade'].unique()
```

Out[26]:
```
array(['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'], dtype=object)
```

In [308]:
```
df['sub_grade'] = df['sub_grade'].str[1].apply(int) ## subgrade is extension 
```

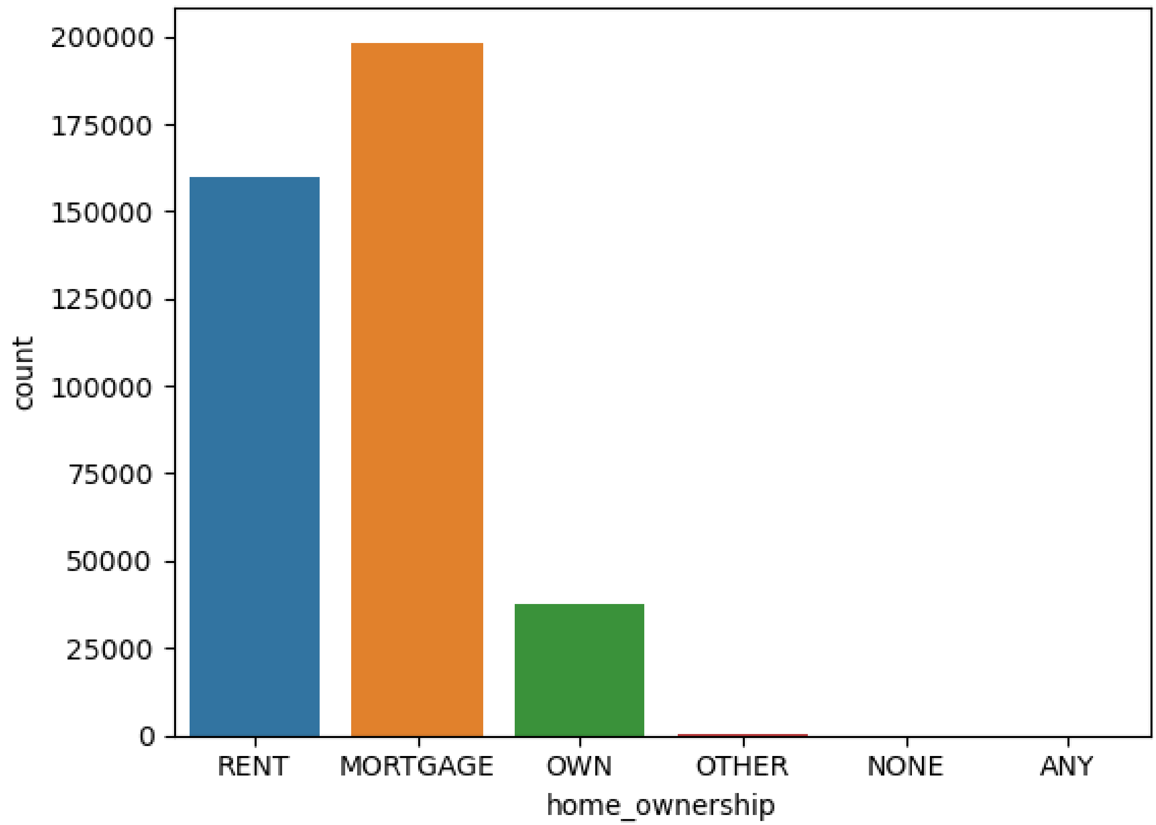
In [ ]:

In [28]:
```
df['home_ownership'].unique()
```

Out[28]:
```
array(['RENT', 'MORTGAGE', 'OWN', 'OTHER', 'NONE', 'ANY'], dtype=object)
```

In [29]:
```
sns.countplot(df['home_ownership'])
```

Out[29]: <AxesSubplot:xlabel='home_ownership', ylabel='count'>



In [30]:
```
home_ownership_dict = {'NONE':'OTHER','ANY':'OTHER'} # we will replace None an
df['home_ownership'] = df['home_ownership'].replace(home_ownership_dict)
```

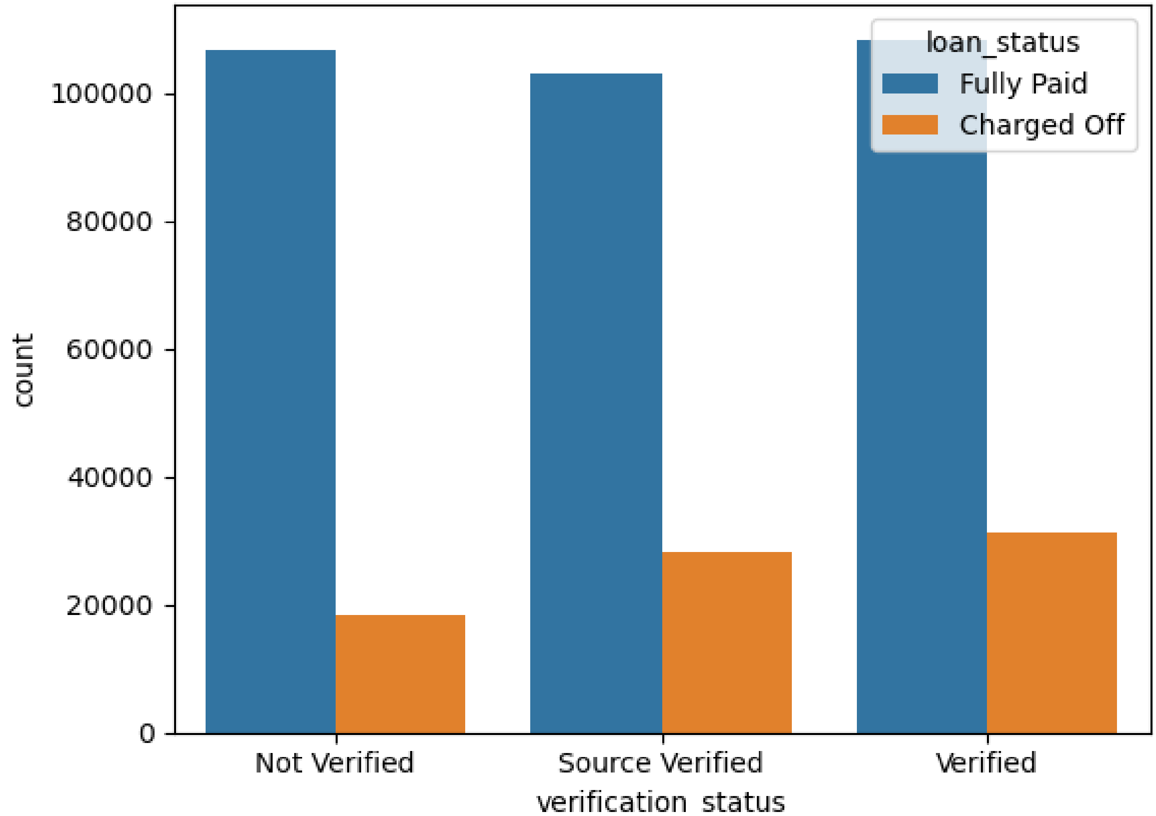
In [ ]:
```

```

In [31]:
```
df['verification_status'].unique()
```

Out[31]: array(['Not Verified', 'Source Verified', 'Verified'], dtype=object)

In [32]:
```
sns.countplot(df['verification_status'],hue=df['loan_status'])
```

Out[32]:
```
<AxesSubplot:xlabel='verification_status', ylabel='count'>
```



In [33]:
```
verification_status_dict = dict(zip(['Not Verified', 'Source Verified', 'Veri
df['verification_status'] = df['verification_status'].replace(verification_st
```

In [ ]:
```
```

In [34]:
```
df['loan_status'].unique()
```

Out[34]:
```
array(['Fully Paid', 'Charged Off'], dtype=object)
```

In [35]:
```
sns.countplot(df['loan_status'])
```

Out[35]:
```
<AxesSubplot:xlabel='loan_status', ylabel='count'>
```

In [36]:
```
loan_status_dict = dict(zip(['Fully Paid', 'Charged Off'],[0,1]))
df['loan_status'] = df['loan_status'].replace(loan_status_dict)
```

In [ ]:

In [37]:
```
df['initial_list_status'].value_counts()
```

Out[37]:
```
f    238066
w    157964
Name: initial_list_status, dtype: int64
```

In [38]:
```
initial_list_status_dict = dict(zip(['w', 'f'],[1,0]))
df['initial_list_status'] = df['initial_list_status'].replace(initial_list_st
```

In [ ]:

In [39]:
```
df['address'].unique()
```

Out[39]:
```
array(['0174 Michelle Gateway\r\nMendozaberg, OK 22690',
       '1076 Carney Fort Apt. 347\r\nLoganmouth, SD 05113',
       '87025 Mark Dale Apt. 269\r\nNew Sabrina, WV 05113', ...,
       '953 Matthew Points Suite 414\r\nReedfort, NY 70466',
       '7843 Blake Freeway Apt. 229\r\nNew Michael, FL 29597',
       '787 Michelle Causeway\r\nBriannaton, AR 48052'], dtype=object)
```

In [40]:
```
df['address'].value_counts()
```

Out[40]:
```
USCGC Smith\r\nFPO AE 70466                           8
USS Johnson\r\nFPO AE 48052                           8
USNS Johnson\r\nFPO AE 05113                          8
USS Smith\r\nFPO AP 70466                             8
USNS Johnson\r\nFPO AP 48052                          7
                                                     ..
455 Tricia Cove\r\nAustinbury, FL 00813               1
7776 Flores Fall\r\nFernandezshire, UT 05113          1
6577 Mia Harbors Apt. 171\r\nRobertshire, OK 22690    1
8141 Cox Greens Suite 186\r\nMadisonstad, VT 05113    1
787 Michelle Causeway\r\nBriannaton, AR 48052         1
Name: address, Length: 393700, dtype: int64
```

In [41]:
```
df['address'] = df['address'].str[-5:]
```

In [42]:
```
df['address'].value_counts()
```

Out[42]:
```
70466    56985
30723    56546
22690    56527
48052    55917
00813    45824
29597    45471
05113    45402
11650    11226
93700    11151
86630    10981
Name: address, dtype: int64
```

In [ ]:
```

```

In [167]:
```
df['purpose'].unique()
```

Out[167]:
```
array(['vacation', 'debt_consolidation', 'credit_card',
       'home_improvement', 'small_business', 'major_purchase', 'other',
       'medical', 'wedding', 'car', 'moving', 'house', 'educational',
       'renewable_energy'], dtype=object)
```

In [159]:
```
(100*df.isna().sum()/df.shape[0]).round(2)
```

Out[159]:
```
loan_amnt               0.00
term                    0.00
int_rate                0.00
installment             0.00
grade                   0.00
sub_grade               0.00
emp_title               0.00
emp_length              0.00
home_ownership          0.00
annual_inc              0.00
verification_status     0.00
issue_d                 0.00
loan_status             0.00
purpose                 0.00
title                   0.44
dti                     0.00
earliest_cr_line        0.00
open_acc                0.00
pub_rec                 0.00
revol_bal               0.00
revol_util              0.00
total_acc               0.00
initial_list_status     0.00
application_type        0.00
mort_acc                0.00
pub_rec_bankruptcies    0.00
address                 0.00
revol_util_na           0.00
mort_acc_na             0.00
dtype: float64
```

In [44]:
```
df['revol_util_na']=df['revol_util'].isna().apply(int)
```

In [45]:
```
df['revol_util'] = df['revol_util'].fillna(0)
```

In [ ]:
```

```

In [46]:
```
df['emp_title'] = df['emp_title'].str.lower().str.strip().fillna('other')
```

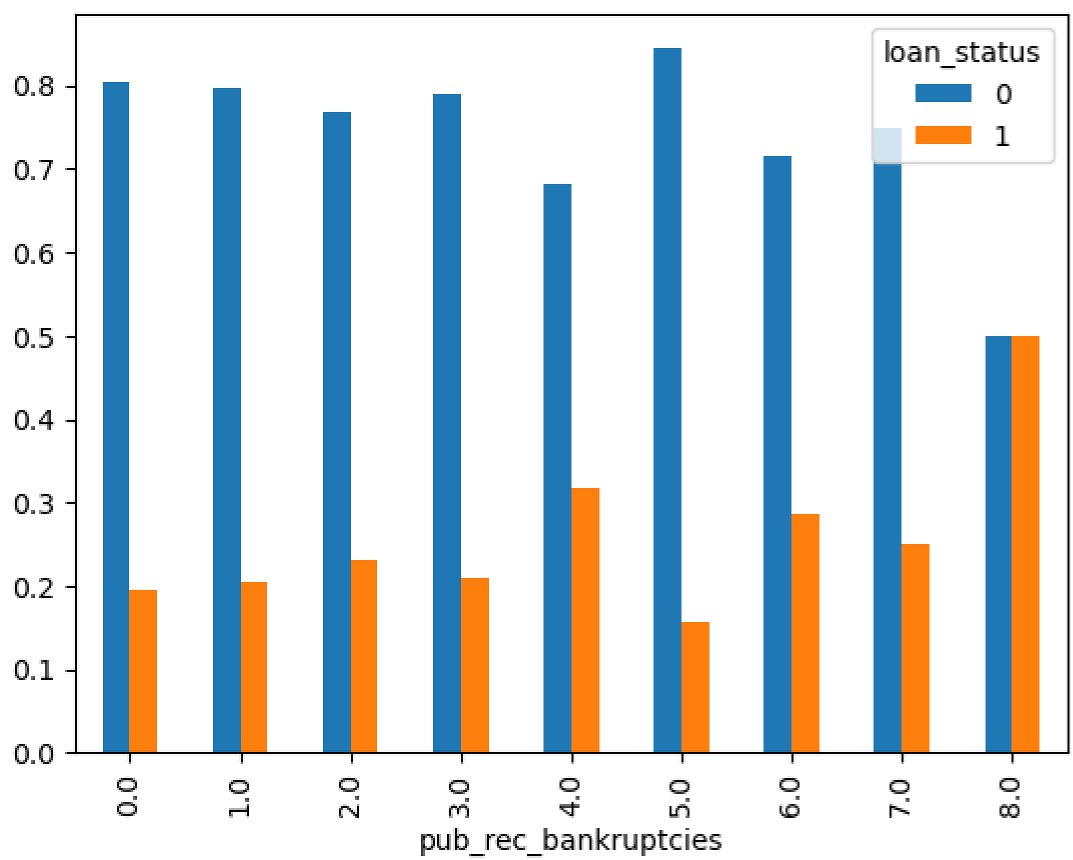
In [ ]:
```

```

In [54]:
```
df['pub_rec_bankruptcies'].value_counts()
```

Out[54]:
```
0.0    350380
1.0     42790
2.0      1847
3.0       351
4.0        82
5.0        32
6.0         7
7.0         4
8.0         2
Name: pub_rec_bankruptcies, dtype: int64
```

In [49]:
```
pd.crosstab(columns = df["loan_status"],
   index=df['pub_rec_bankruptcies'],
   normalize="index").plot(kind="bar")
```

Out[49]: <AxesSubplot:xlabel='pub_rec_bankruptcies'>

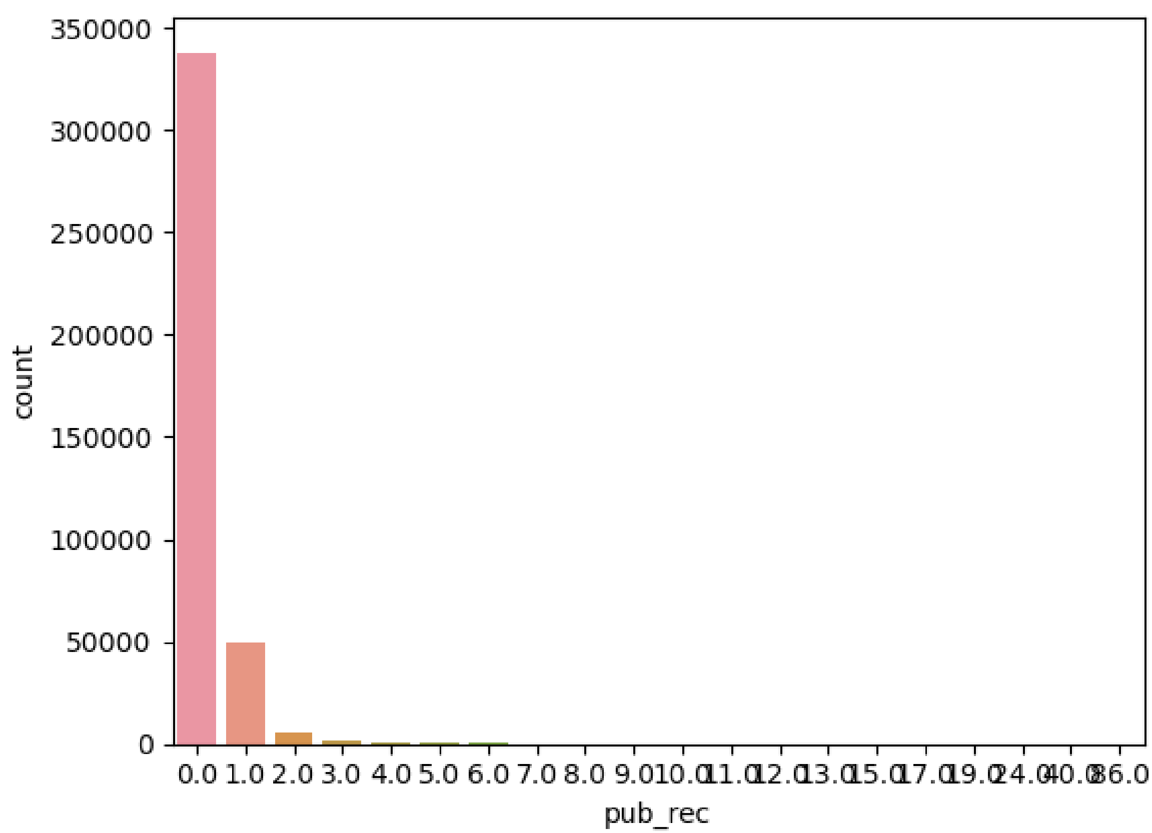


In [62]:
```
df['pub_rec'].value_counts()
```

Out[62]:
```
0.0     338272
1.0      49739
2.0       5476
3.0       1521
4.0        527
5.0        237
6.0        122
7.0         56
8.0         34
9.0         12
10.0        11
11.0         8
13.0         4
12.0         4
19.0         2
40.0         1
17.0         1
86.0         1
24.0         1
15.0         1
Name: pub_rec, dtype: int64
```

In [53]:
```
sns.countplot(df['pub_rec'])
```

Out[53]:
```
<AxesSubplot:xlabel='pub_rec', ylabel='count'>
```



In [67]:
```
df['pub_rec_bankruptcies'] = df['pub_rec_bankruptcies'].fillna(df['pub_rec'])
```

In [ ]:

In [ ]:

In [70]:
```
df['mort_acc'].value_counts()
```

Out[70]:
```
0.0     139777
1.0      60416
2.0      49948
3.0      38049
4.0      27887
5.0      18194
6.0      11069
7.0       6052
8.0       3121
9.0       1656
10.0       865
11.0       479
12.0       264
13.0       146
14.0       107
15.0        61
16.0        37
17.0        22
18.0        18
```

In [120]:
```
df['mort_acc_na'] = df['mort_acc'].isna().apply(int)
```

In [122]:
```
df['home_ownership'].unique()
```

Out[122]:
```
array(['RENT', 'MORTGAGE', 'OWN', 'OTHER'], dtype=object)
```

In [124]:
```
df['purpose'].unique()
```

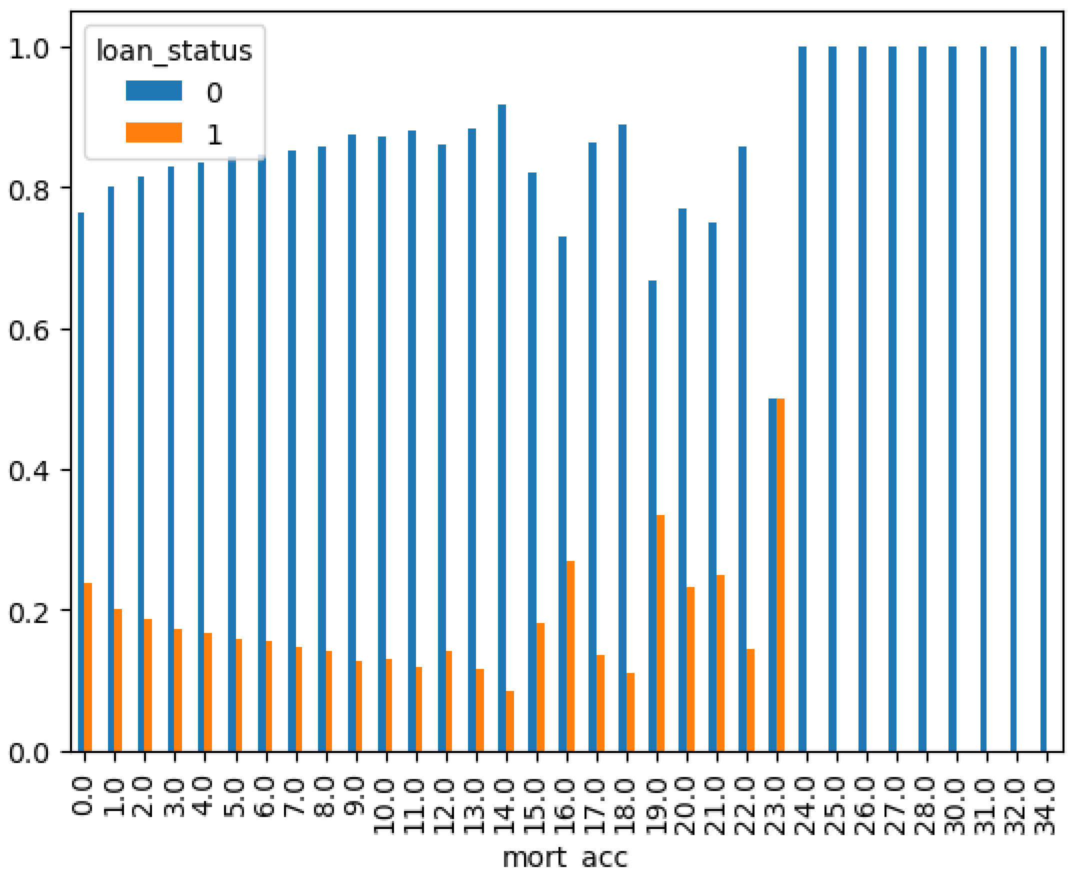
Out[124]:
```
array(['vacation', 'debt_consolidation', 'credit_card',
       'home_improvement', 'small_business', 'major_purchase', 'other',
       'medical', 'wedding', 'car', 'moving', 'house', 'educational',
       'renewable_energy'], dtype=object)
```

In [155]:
```
def mort_acc_fillna(row):   # filling the mortgage with the best possible val
    if pd.isnull(row['mort_acc']):
        if row['home_ownership'] in ['MORTGAGE','OTHER','OWN'] or row['purpos
            return 1
        else:
            return 0
    else:
        return row['mort_acc']
```

In [158]:
```
df['mort_acc'] = df.apply(mort_acc_fillna, axis=1)
```

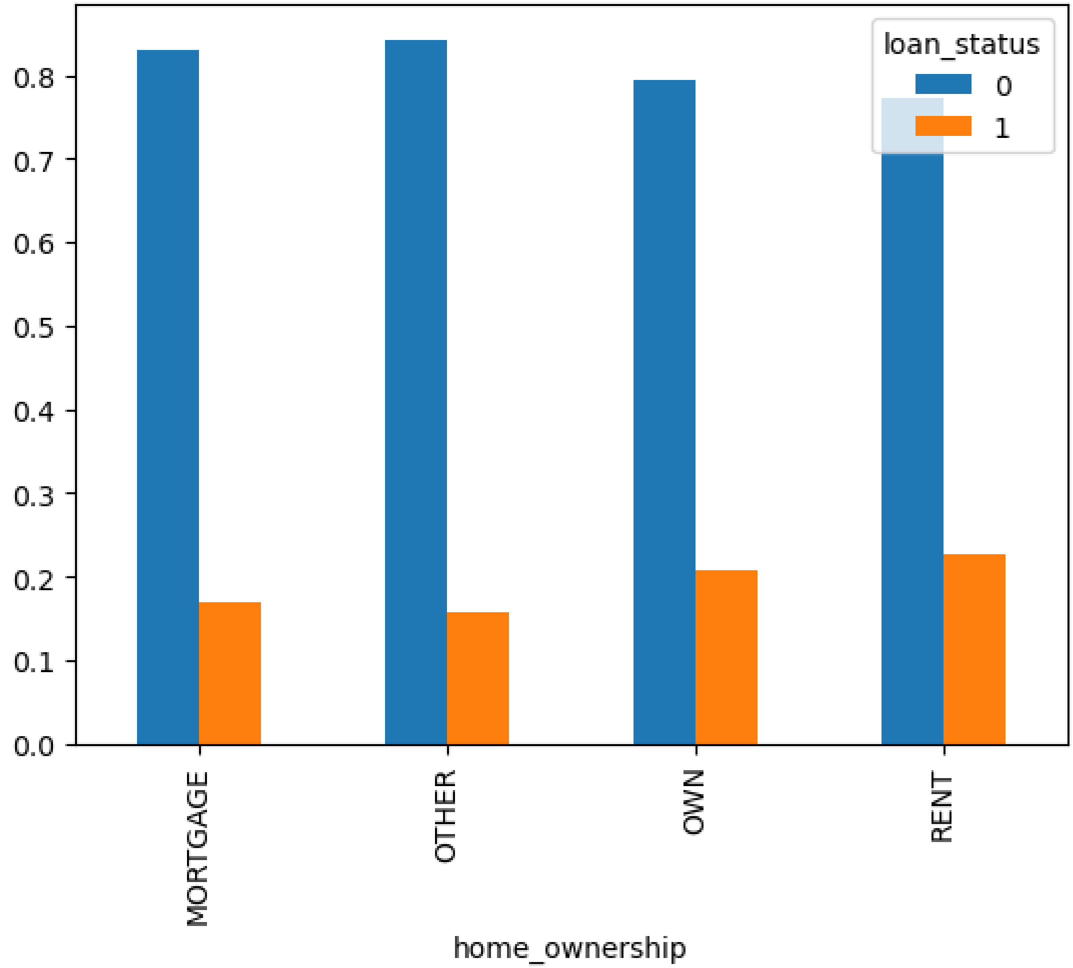
In [71]:
```
pd.crosstab(columns = df["loan_status"],
            index=df['mort_acc'],
            normalize="index").plot(kind="bar")
```

Out[71]: <AxesSubplot:xlabel='mort_acc'>



In [72]:
```
pd.crosstab(columns = df["loan_status"],
           index=df['home_ownership'],
           normalize="index").plot(kind="bar")
```

Out[72]:
```
<AxesSubplot:xlabel='home_ownership'>
```



In [ ]:

In [84]:
```
df[df['home_ownership']=='OWN'][['mort_acc','total_acc','open_acc']].corr()
```

Out[84]:

	mort_acc	total_acc	open_acc
mort_acc	1.000000	0.352143	0.104398
total_acc	0.352143	1.000000	0.696238
open_acc	0.104398	0.696238	1.000000

In [82]:
```
(df['open_acc']>=df['mort_acc']).value_counts()
```

Out[82]:
```
True     355208
False     40822
dtype: int64
```

In [ ]:

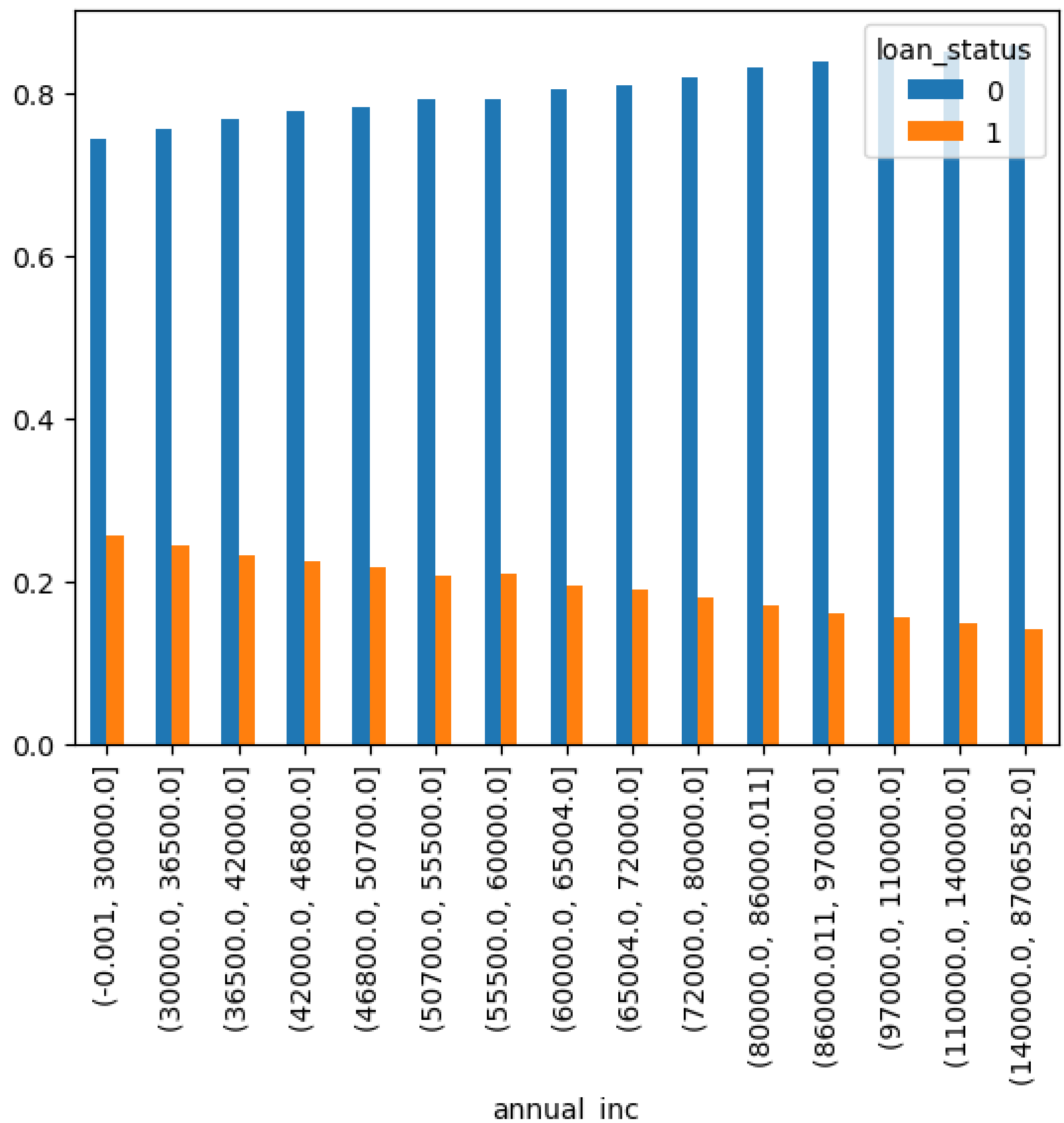
In [ ]:

In [ ]:

In [173]:

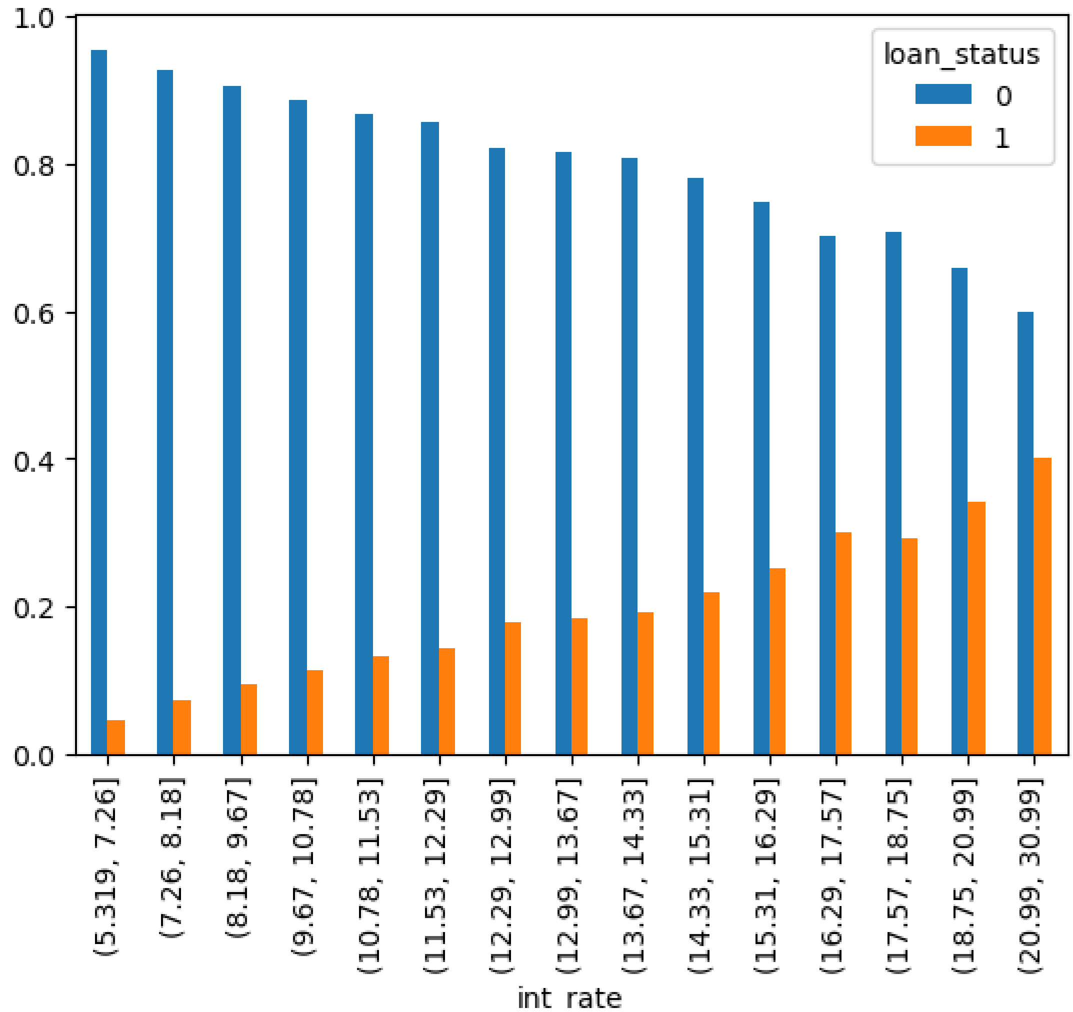
```python
pd.crosstab(columns = df["loan_status"],
            index=pd.qcut(df["annual_inc"],15),
            normalize="index").plot(kind="bar")
```

Out[173]: <AxesSubplot:xlabel='annual_inc'>



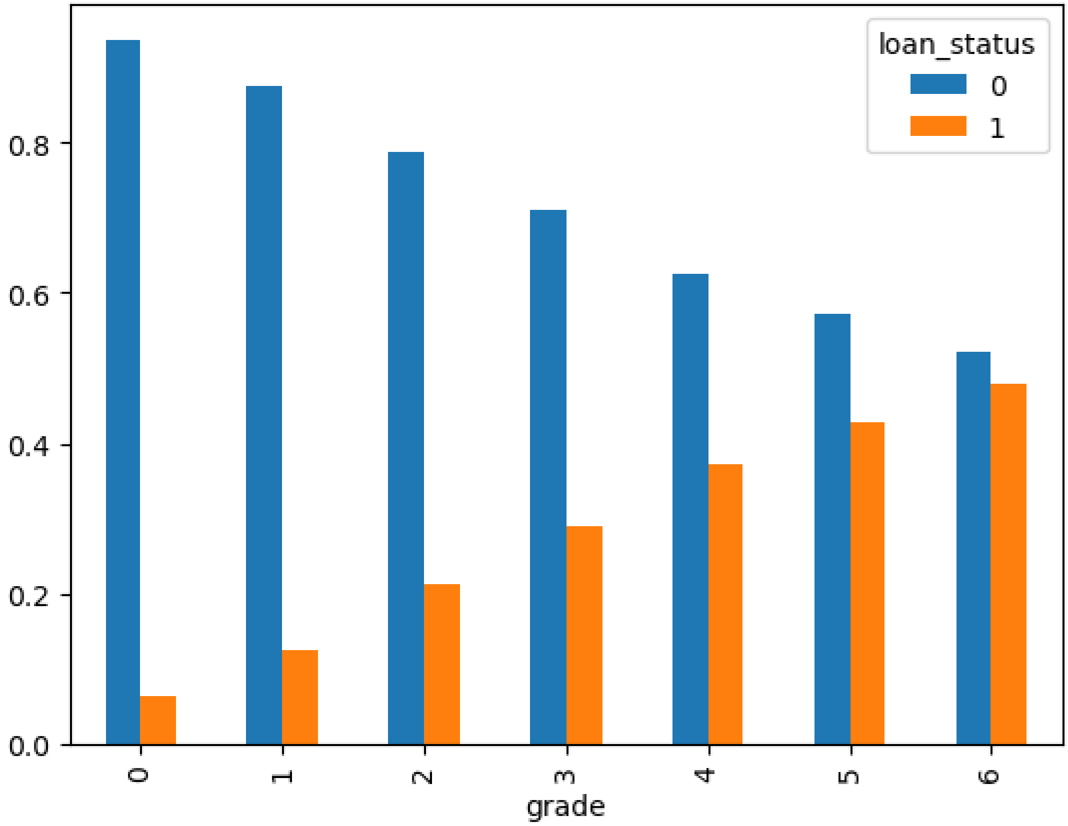
In [164]:
```
pd.crosstab(columns = df["loan_status"],
            index=pd.qcut(df["int_rate"],15),
            normalize="index").plot(kind="bar")
```

Out[164]: <AxesSubplot:xlabel='int_rate'>



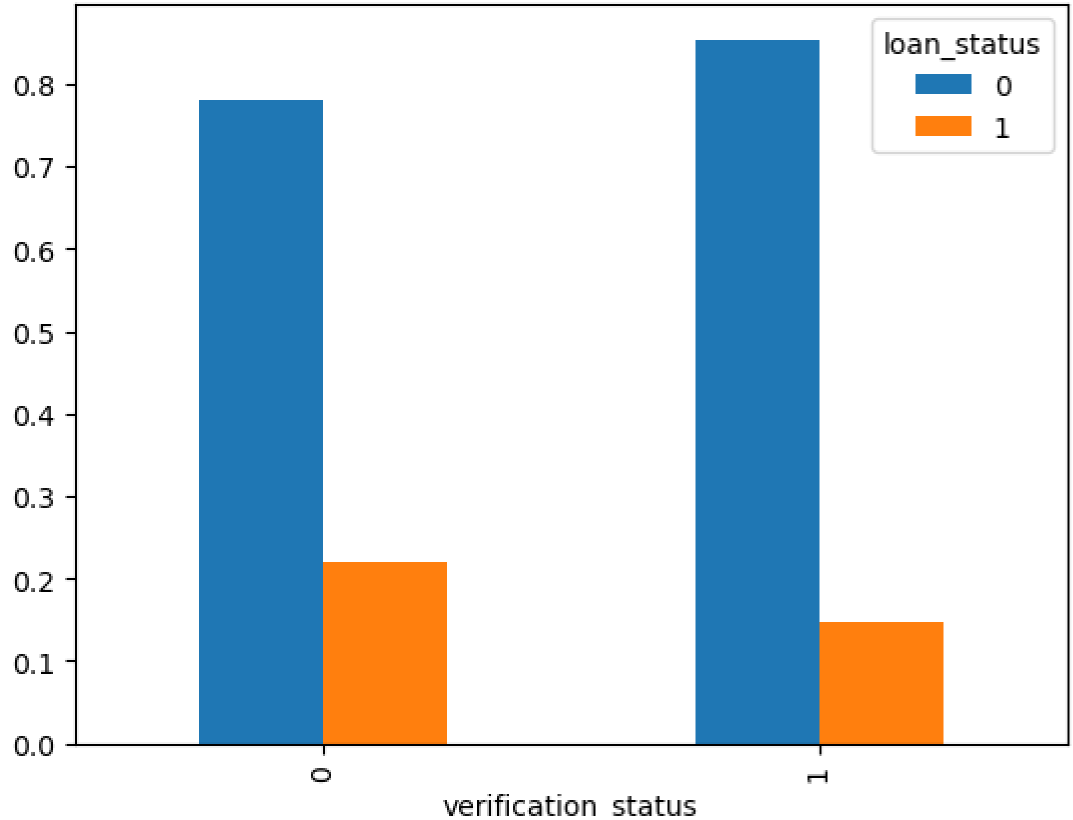
In [165]:
```
pd.crosstab(columns = df["loan_status"],
            index=df["grade"],
            normalize="index").plot(kind="bar")
```

Out[165]: <AxesSubplot:xlabel='grade'>



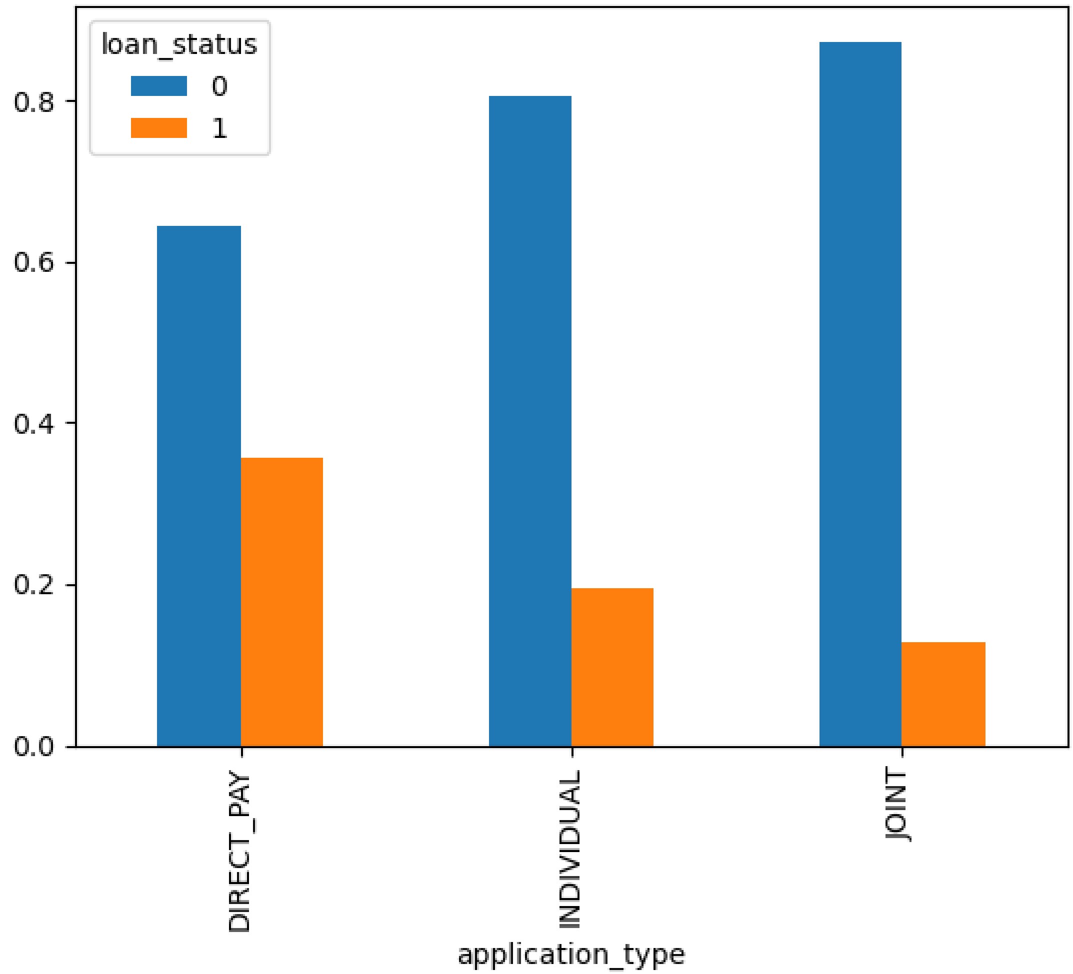
In [166]:
```
pd.crosstab(columns = df["loan_status"],
            index=df["verification_status"],
            normalize="index").plot(kind="bar")
```

Out[166]: <AxesSubplot:xlabel='verification_status'>



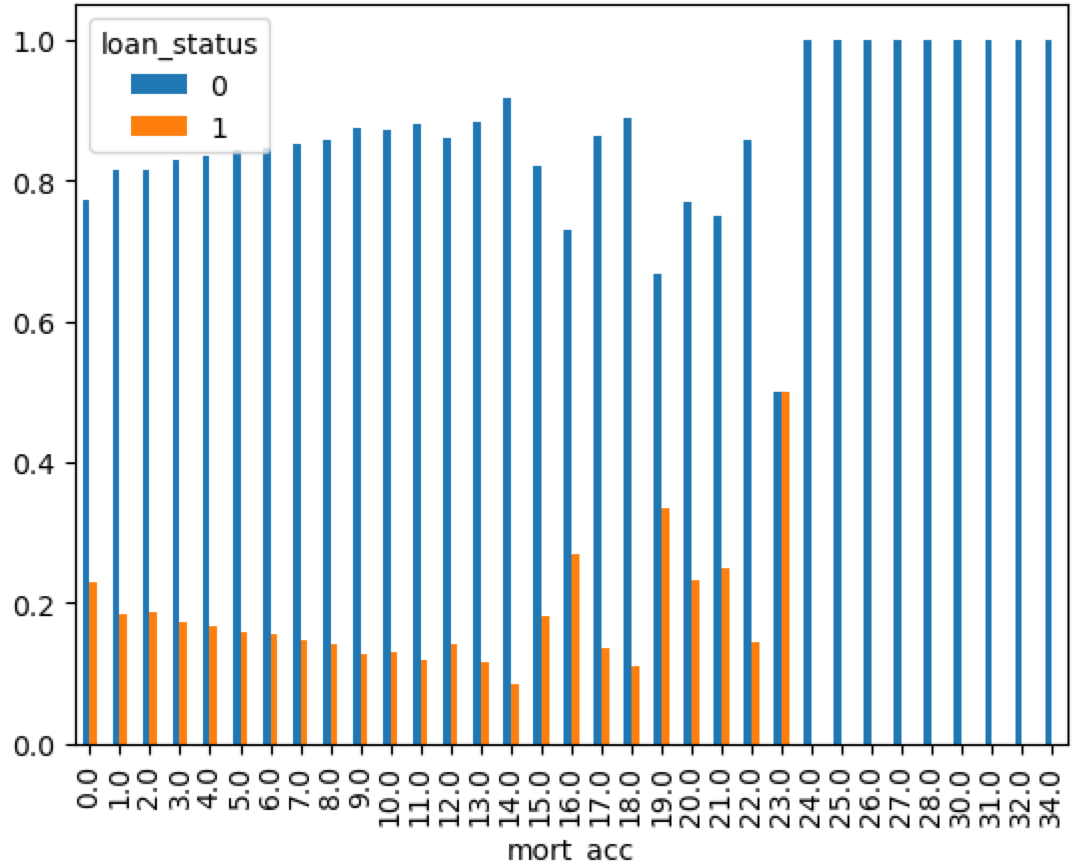
In [167]:
```
pd.crosstab(columns = df["loan_status"],
            index=df["application_type"],
            normalize="index").plot(kind="bar")
```

Out[167]: <AxesSubplot:xlabel='application_type'>



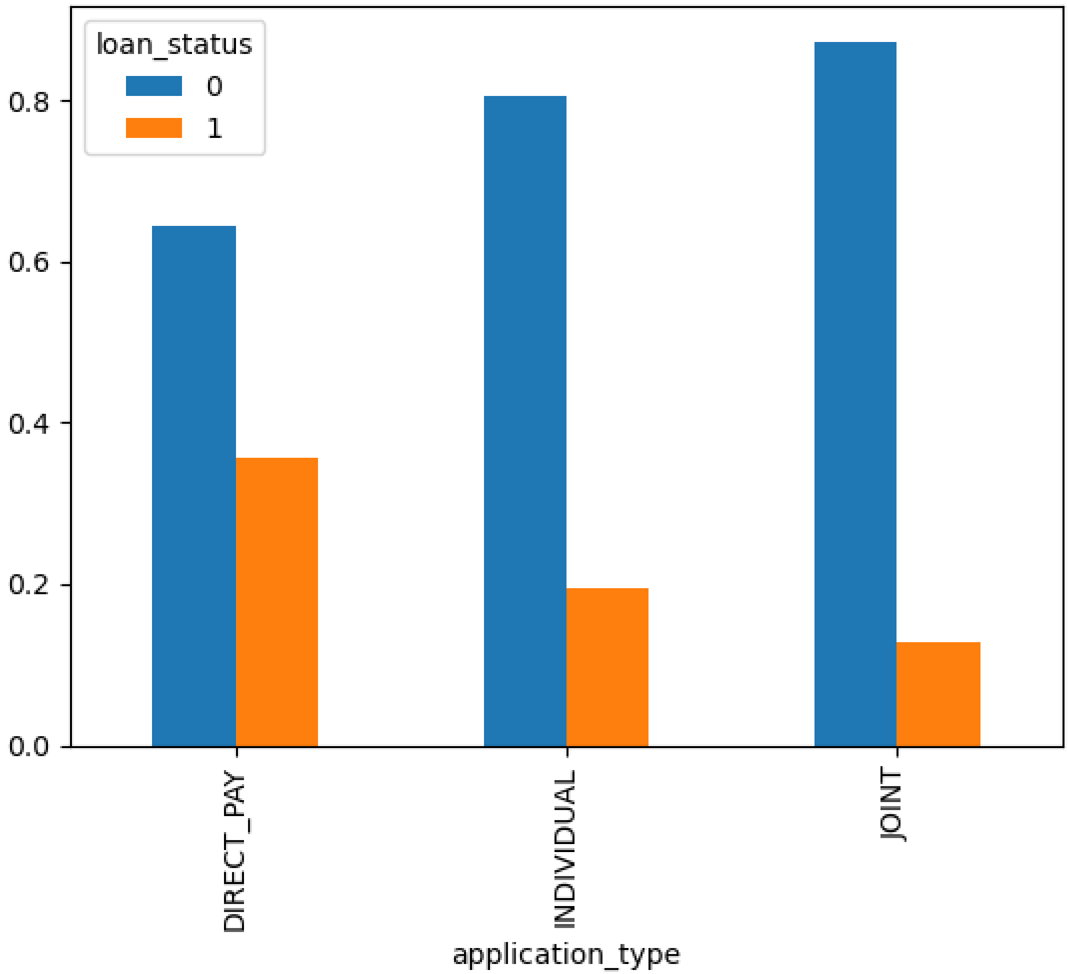
In [172]:
```
pd.crosstab(columns = df["loan_status"],
            index=df['mort_acc'],
            normalize="index").plot(kind="bar")
```

Out[172]: <AxesSubplot:xlabel='mort_acc'>



In [175]:
```
pd.crosstab(columns = df["loan_status"],
            index=df["application_type"],
            normalize="index").plot(kind="bar")
```

Out[175]: <AxesSubplot:xlabel='application_type'>



In [ ]:

In [ ]:

In [92]:
```
mort_acc_na = df['mort_acc'].apply(mort)
```

In [89]:
```
pd.crosstab(columns = df[df['loan_status']=='Charged Off']["home_ownership"],
            index=df[df['loan_status']=='Charged Off']['mort'],
            normalize="index")
```

Out[89]:

home_ownership mort	MORTGAGE	NONE	OTHER	OWN	RENT
-1	0.406126	0.000000	0.001622	0.082703	0.509550
0	0.095063	0.000060	0.000121	0.125554	0.779202
1	0.724375	0.000128	0.000077	0.081712	0.193707

In [94]:
```
pd.crosstab(columns = df["home_ownership"],
            index=df['mort'],
            normalize="index")
```

Out[94]:

home_ownership mort	ANY	MORTGAGE	NONE	OTHER	OWN	RENT
-1	0.000000	0.440323	0.000053	0.002064	0.078502	0.479058
0	0.000000	0.104209	0.000064	0.000107	0.122896	0.772724
1	0.000014	0.765090	0.000092	0.000087	0.080569	0.154149

In [93]:
```
pd.crosstab(columns = df["home_ownership"],
            index=mort_acc_na)
```

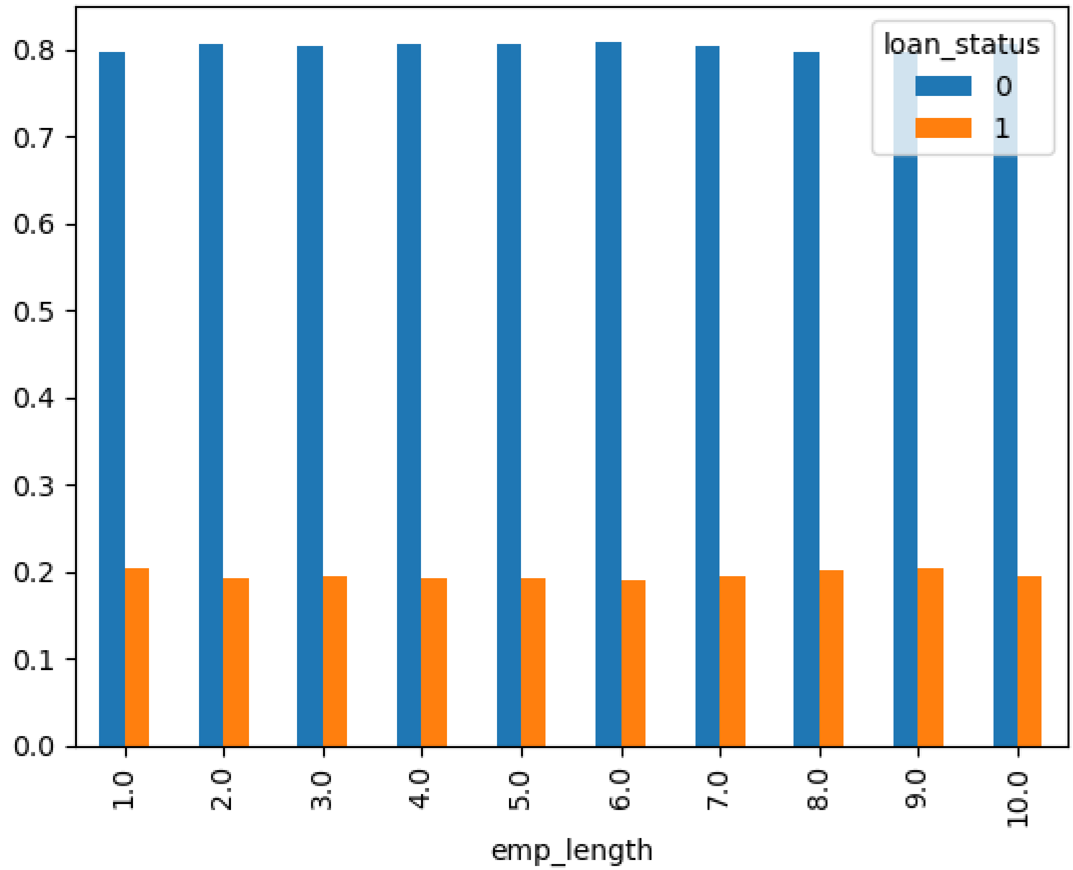
Out[93]:

home_ownership mort_acc	MORTGAGE	OTHER	OWN	RENT
-1	16642	80	2967	18106
0	14566	24	17178	108009
1	167140	42	17601	33675

In [104]:
```
df_ = df[df["home_ownership"]=='OWN']
```

In [108]:
```
pd.crosstab(columns = df["loan_status"],
            index=(df["emp_length"]),
            normalize="index").plot(kind="bar")
```

Out[108]: <AxesSubplot:xlabel='emp_length'>



In [94]:
```
pd.crosstab(columns = df["home_ownership"],
            index=mort_acc_na,
            normalize="columns")
```

Out[94]:

home_ownership mort_acc	MORTGAGE	OTHER	OWN	RENT
-1	0.083903	0.547945	0.078604	0.113311
0	0.073437	0.164384	0.455095	0.675943
1	0.842660	0.287671	0.466301	0.210745

In [93]:
```
df[df['mort_acc'].isna()].groupby(['home_ownership'])['loan_status'].count()
```

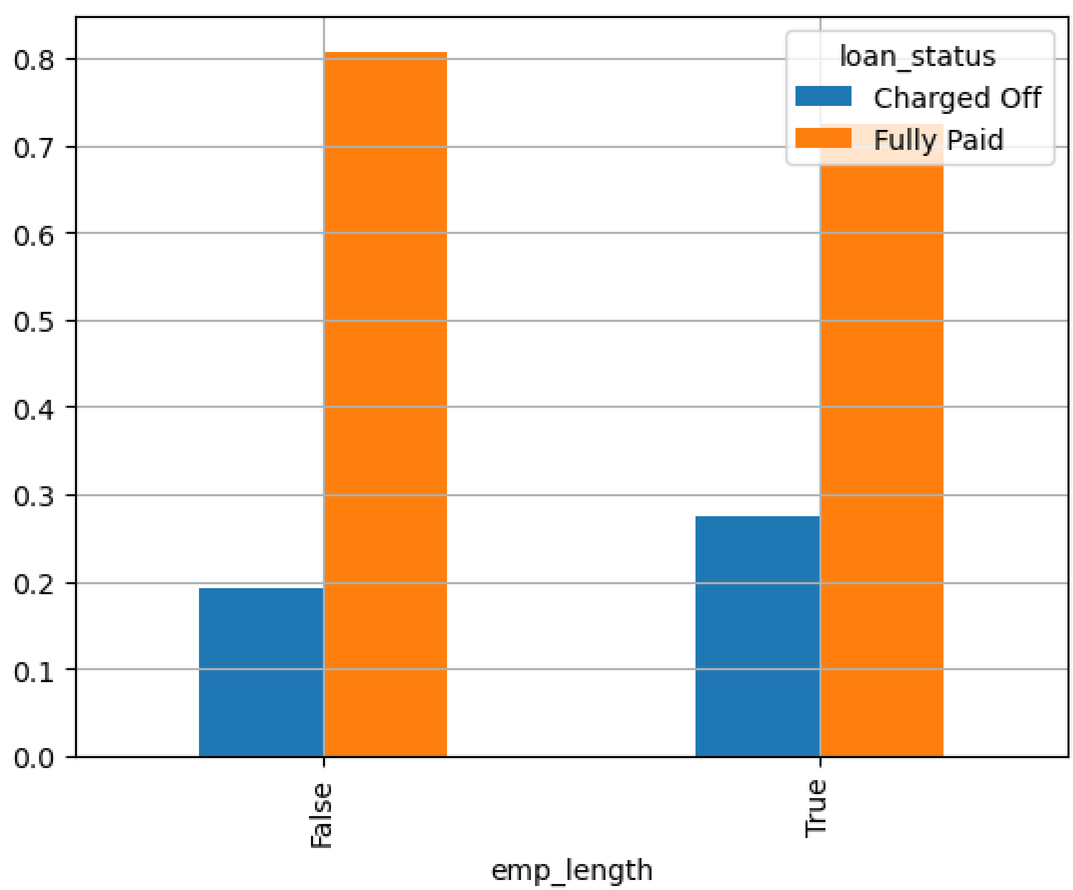
Out[93]:
```
home_ownership
MORTGAGE    16642
NONE            2
OTHER          78
OWN          2967
RENT        18106
Name: loan_status, dtype: int64
```

In [ ]:
```
pd.crosstab(columns = df["loan_status"],
           index=df['emp_length_na'].isna(),
           normalize="index").plot(kind="bar",grid=True)
```

In [102]:
```
df['emp_length_na'] = df['emp_length'].isna()
```

In [101]:
```
pd.crosstab(columns = df["loan_status"],
           index=df['emp_length'].isna(),
           normalize="index").plot(kind="bar",grid=True)
```

Out[101]:
```
<AxesSubplot:xlabel='emp_length'>
```



```
In [110]: df['purpose']=='house'
```

```
Out[110]: array(['vacation', 'debt_consolidation', 'credit_card',
                 'home_improvement', 'small_business', 'major_purchase', 'other',
                 'medical', 'wedding', 'car', 'moving', 'house', 'educational',
                 'renewable_energy'], dtype=object)
```

```
In [116]: df['purpose'].value_counts(normalize=True)*100
```

```
Out[116]: debt_consolidation    59.214453
          credit_card           20.962806
          home_improvement       6.067722
          other                  5.349342
          major_purchase         2.219529
          small_business         1.439537
          car                    1.186021
          medical                1.059516
          moving                 0.720652
          vacation               0.619145
          house                  0.555766
          wedding                0.457541
          renewable_energy       0.083075
          educational            0.064894
          Name: purpose, dtype: float64
```

```
In [117]: df[df['mort_acc'].isna()]['purpose'].value_counts(normalize=True)*100
```

```
Out[117]: debt_consolidation    47.712660
          credit_card           14.075936
          other                  9.564757
          home_improvement       7.082947
          major_purchase         5.125017
          small_business         4.651409
          car                    3.728006
          wedding                2.262204
          medical                1.656304
          moving                 1.349385
          house                  1.010716
          vacation               0.870486
          educational            0.677338
          renewable_energy       0.232835
          Name: purpose, dtype: float64
```

```
In [113]: (df['mort_acc'].isna()).value_counts()
```

```
Out[113]: False    358235
          True      37795
          Name: mort_acc, dtype: int64
```

In [ ]:

In [ ]:

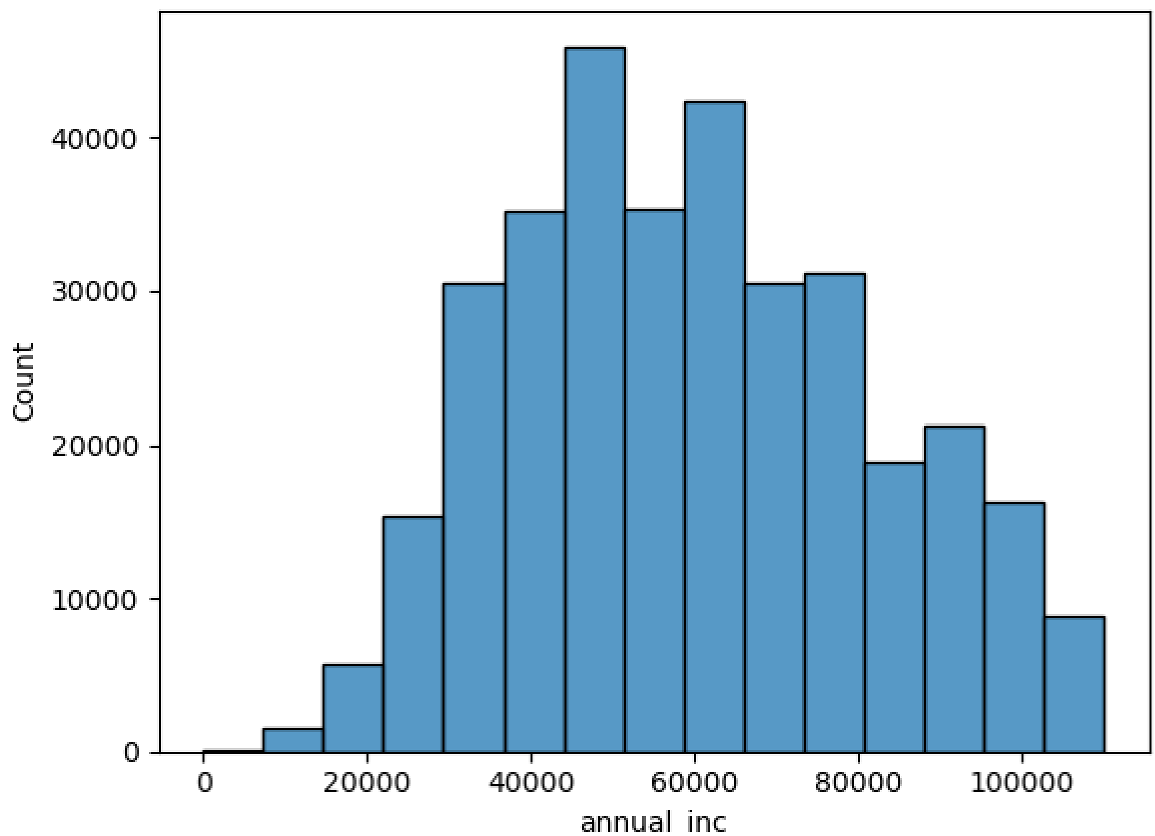
In [176]:
```
df.head()
```

Out[176]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ov
0	10000.0	36	11.44	329.48	1	4	marketing	10.0	
1	8000.0	36	11.99	265.68	1	5	credit analyst	4.0	MOI
2	15600.0	36	10.49	506.97	1	3	statistician	1.0	
3	7200.0	36	6.49	220.65	0	2	client advocate	6.0	
4	24375.0	60	17.27	609.33	2	5	destiny management inc.	9.0	MOI

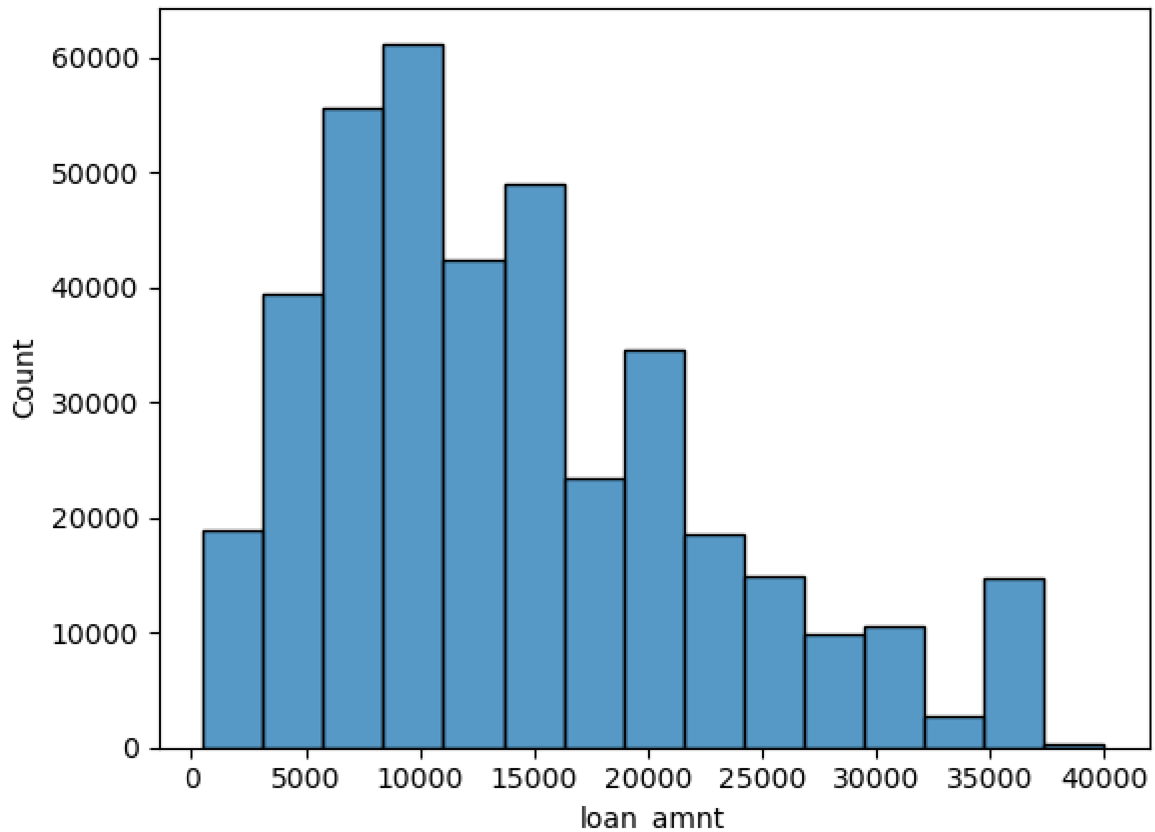
In [248]:
```
for col in ['annual_inc','dti','open_acc','revol_bal','revol_util']:
    sns.boxplot(df[col])
    plt.show()
```

In [ ]:

In [192]:
```
df['annual_inc'].shape[0]-df.loc[df['annual_inc']<260000,'annual_inc'].shape[
```
Out[192]:
```
3194
```
In [210]:
```
df.loc[df['annual_inc']<1100000,'loan_status'].value_counts()
```
Out[210]:
```
0    50
1     7
Name: loan_status, dtype: int64
```
In [209]:
```
sns.histplot((df.loc[df['annual_inc']<110000,'annual_inc']),bins=15)
```
Out[209]:
```
<AxesSubplot:xlabel='annual_inc', ylabel='Count'>
```


In [177]:
```
sns.histplot(df['loan_amnt'],bins=15)
```

Out[177]: <AxesSubplot:xlabel='loan_amnt', ylabel='Count'>



In [262]:
```
df_1=df.copy(deep=True)
```

In [263]:
```
df_1.to_csv("preprocessed-data.csv",index=False)
```

In [250]:
```
df.shape
```

Out[250]: (396030, 29)

In [251]:
```
data = df[(df['annual_inc']<1100000) & (df['dti']<60) & (df['open_acc']<60) &
```

In [252]:
```
data.shape
```

Out[252]: (395945, 29)

In [254]:
```
data['revol_bal'] = np.log1p(data['revol_bal'])
```

In [ ]:

```python

```

In [261]:

```python
data.head()
```

Out[261]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_length	home_ownership	ann
0	10000.0	36	11.44	329.48	1	4	10.0	RENT	1
1	8000.0	36	11.99	265.68	1	5	4.0	MORTGAGE	6
2	15600.0	36	10.49	506.97	1	3	1.0	RENT	4
3	7200.0	36	6.49	220.65	0	2	6.0	RENT	5
4	24375.0	60	17.27	609.33	2	5	9.0	MORTGAGE	5

In [257]:

```python
data.drop(columns=['emp_title','title'],inplace=True)
```

In [260]:

```python
data['issue_d'] = data['issue_d'].dt.year
data['earliest_cr_line'] = data['earliest_cr_line'].dt.year
```

In [264]:

```python
dummies = ['home_ownership','purpose', 'application_type','address']
data = pd.get_dummies(data, columns=dummies, drop_first=True)
```

In [265]:

```python
data.shape
```

Out[265]:

```
(395945, 50)
```

In [266]:

```python
data.head()
```

Out[266]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_length	annual_inc	verificatio
0	10000.0	36	11.44	329.48	1	4	10.0	117000.0	
1	8000.0	36	11.99	265.68	1	5	4.0	65000.0	
2	15600.0	36	10.49	506.97	1	3	1.0	43057.0	
3	7200.0	36	6.49	220.65	0	2	6.0	54000.0	
4	24375.0	60	17.27	609.33	2	5	9.0	55000.0	

In [267]:

```python
data['pub_rec'].apply(lambda x :0 if x==0 else 1).value_counts()
```

Out[267]:

```
0    338194
1     57751
Name: pub_rec, dtype: int64
```

In [270]:
```python
for col in ['pub_rec','mort_acc','pub_rec_bankruptcies']:
    data[col+'_flag'] = data[col].apply(lambda x :0 if x==0 else 1)
```

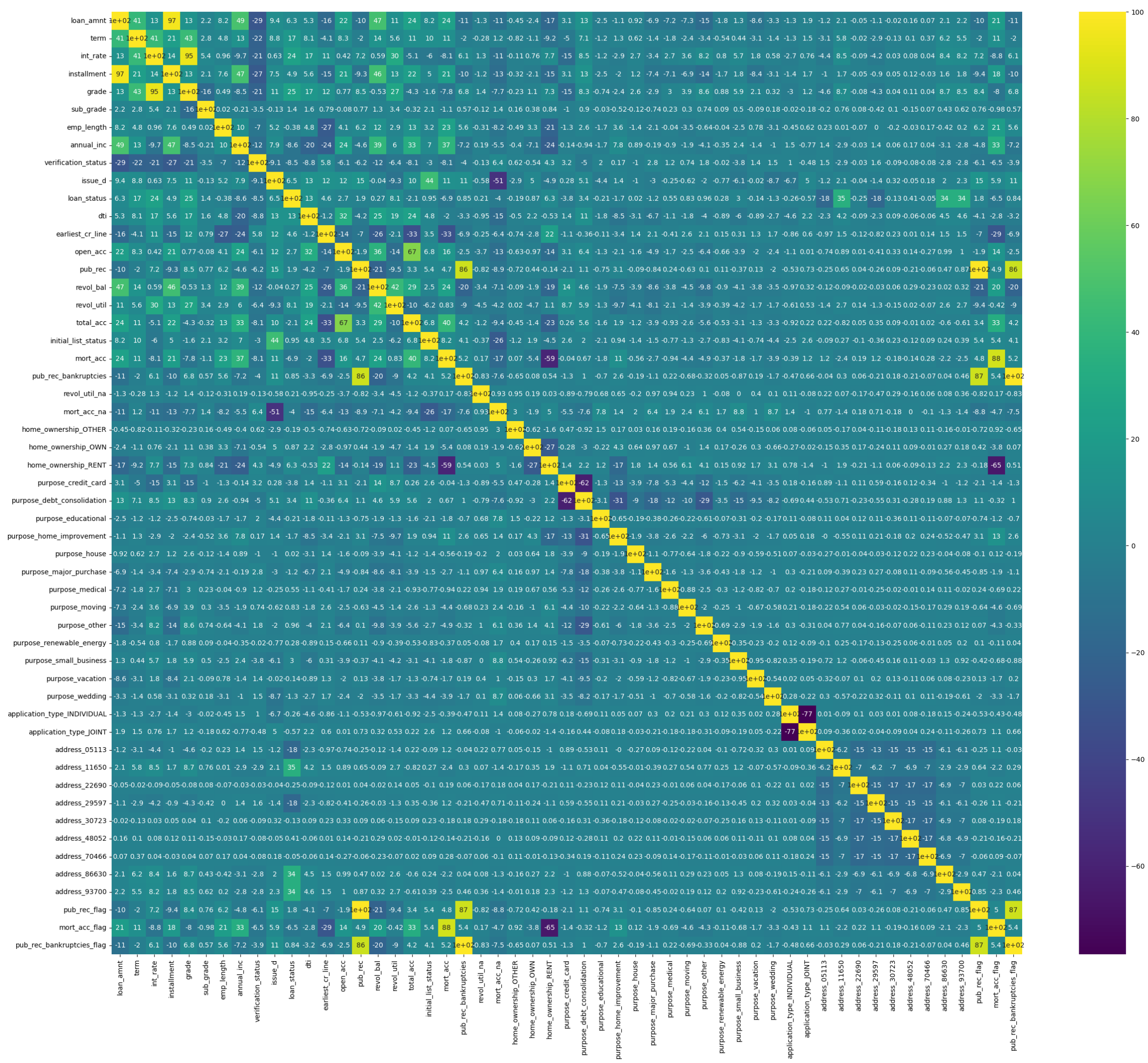
In [271]:
```python
data.shape
```

Out[271]: (395945, 53)

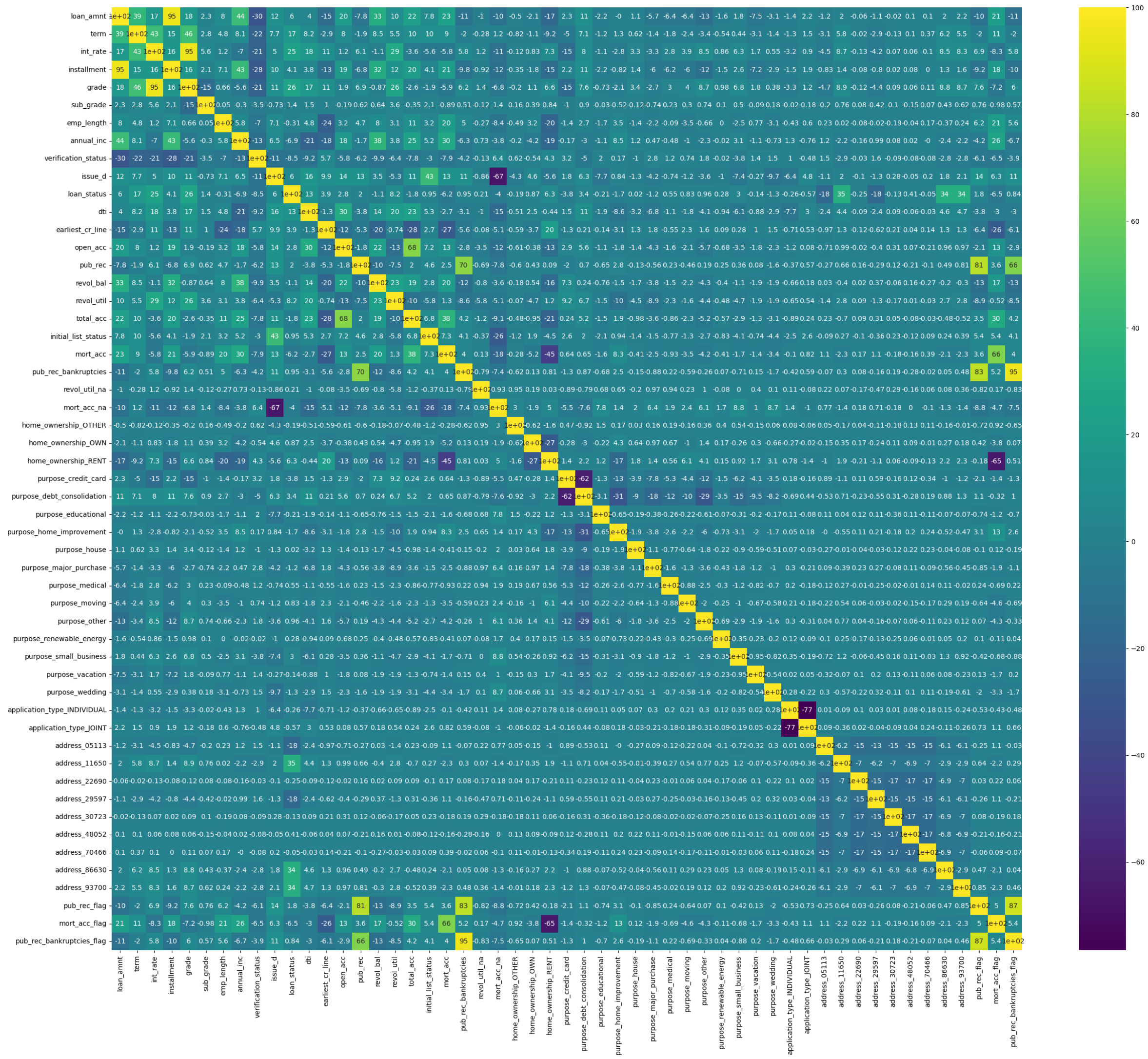
In [272]:
```python
spearman_corr = data.corr(method='spearman')
pearson_corr = data.corr()
```

In [273]:
```python
spearman_corr = (100*spearman_corr).round(2)
pearson_corr = (100*pearson_corr).round(2)
```

In [274]:
```python
plt.figure(figsize=(30, 25))
sns.heatmap(spearman_corr, annot=True, cmap='viridis')
plt.show()
```



In [275]:
```python
plt.figure(figsize=(30, 25))
sns.heatmap(pearson_corr, annot=True, cmap='viridis')
plt.show()
```



In [276]:
```python
data.describe()
```

Out[276]:

	loan_amnt	term	int_rate	installment	grade	sub_gra
count	395945.000000	395945.000000	395945.000000	395945.000000	395945.000000	395945.000(
mean	14112.684969	41.698185	13.639617	431.811657	1.822389	2.9717
std	8356.309080	10.212120	4.472038	250.683221	1.333799	1.4067
min	500.000000	36.000000	5.320000	16.080000	0.000000	1.000(
25%	8000.000000	36.000000	10.490000	250.330000	1.000000	2.000(
50%	12000.000000	36.000000	13.330000	375.430000	2.000000	3.000(
75%	20000.000000	36.000000	16.490000	567.300000	3.000000	4.000(
max	40000.000000	60.000000	30.990000	1533.810000	6.000000	5.000(

In [277]:
```
data.isna().sum()
```

Out[277]:
```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
sub_grade               0
emp_length              0
annual_inc              0
verification_status     0
issue_d                 0
loan_status             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
```

In [ ]:

In [278]:
```
X = data.drop(columns=['loan_status'])
y = data['loan_status']
```

In [279]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25, stratify
```

In [280]:
```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[280]:
```
((296958, 52), (98987, 52), (296958,), (98987,))
```

In [281]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [283]:
```
model = LogisticRegression(max_iter=1000)
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print("Train score : " , model.score(X_train,y_train))
print("Test score : " , model.score(X_test,y_test))
```

```
Train score :  0.8883680520477644
Test score :  0.8910968106923132
```

In [288]:
```
model = LogisticRegression(max_iter=1000,class_weight='balanced')
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
print("Train score : " , model.score(X_train,y_train))
print("Test score : " , model.score(X_test,y_test))
```

```
Train score :  0.8100707844206925
Test score :  0.809783102831685
```

In [ ]:

In [ ]:

In [285]:
```
print(confusion_matrix(y_test, y_test))
```

```
[[79572     0]
 [    0 19415]]
```

In [286]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[78779   793]
 [ 9987  9428]]
```

In [289]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[64751 14821]
 [ 4008 15407]]
```

In [ ]:

In [ ]:

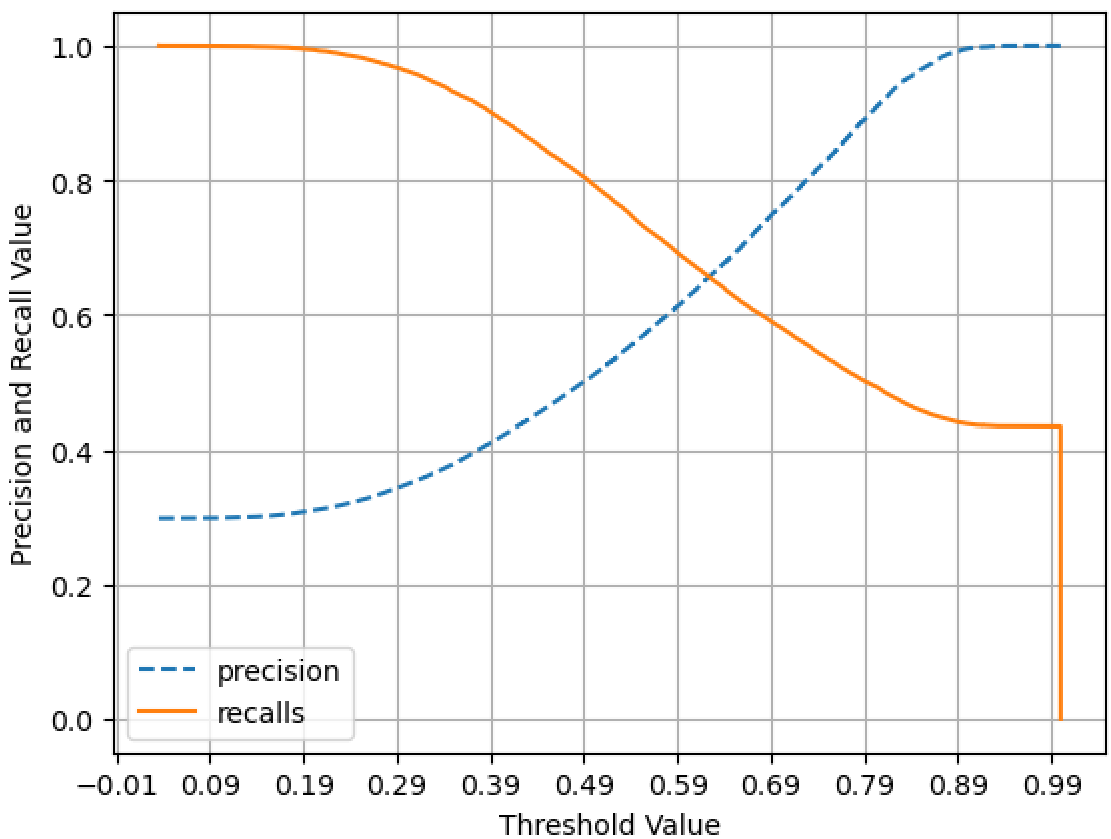
In [287]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.99      0.94     79572
           1       0.92      0.49      0.64     19415

    accuracy                           0.89     98987
   macro avg       0.90      0.74      0.79     98987
weighted avg       0.89      0.89      0.88     98987
```

In [290]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.81      0.87     79572
           1       0.51      0.79      0.62     19415

    accuracy                           0.81     98987
   macro avg       0.73      0.80      0.75     98987
weighted avg       0.86      0.81      0.82     98987
```

In [291]:
```
def precision_recall_curve_plot(y_test, pred_proba_c1):
    precisions, recalls, thresholds = precision_recall_curve(y_test, pred_prol

    threshold_boundary = thresholds.shape[0]
    # plot precision
    plt.plot(thresholds, precisions[0:threshold_boundary], linestyle='--', la
    # 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, model.predict_proba(X_test)[:,1])
```



In [ ]:

In [292]:
```
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.sha
    vif['VIF'] = round(vif['VIF'], 2)
    vif = vif.sort_values(by='VIF', ascending = False)
    return vif
```

In [293]:
```
data = data.drop(columns=['sub_grade_2','issue_d','earliest_cr_line','applica
                          'int_rate','loan_amnt','annual_inc','dti','term','tot
```

In [294]:
```
vif_result = calc_vif(df1)
```

In [295]:
```
vif_result
```

In [ ]:
```

```

In [296]:
```
sm = SMOTE(random_state=42)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())
```

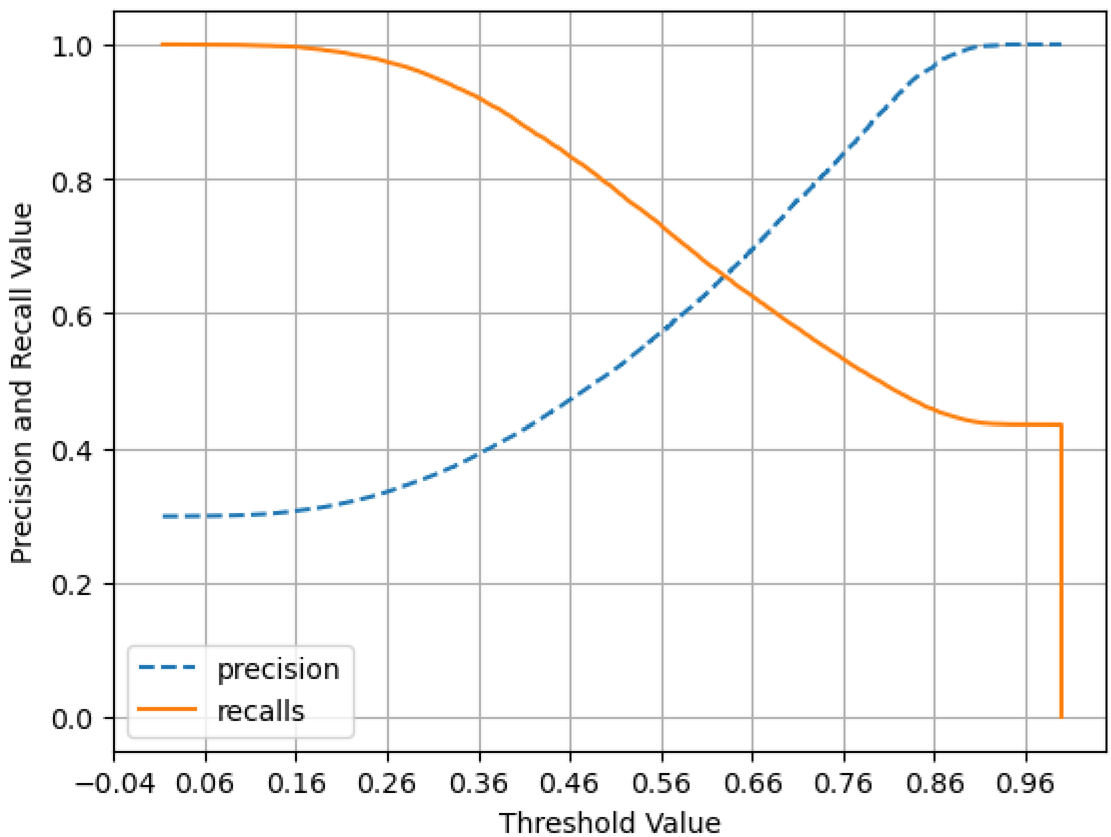
In [297]:
```
print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.sh

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

```
After OverSampling, the shape of train_X: (477424, 52)
After OverSampling, the shape of train_y: (477424,)

After OverSampling, counts of label '1': 238712
After OverSampling, counts of label '0': 238712
```

In [298]:
```
lr1 = LogisticRegression(max_iter=5000)
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       0.94      0.81      0.87     79572
           1       0.51      0.79      0.62     19415

    accuracy                           0.81     98987
   macro avg       0.72      0.80      0.75     98987
weighted avg       0.86      0.81      0.82     98987
```

In [303]:
```
precision_recall_curve_plot(y_test, lr1.predict_proba(X_test)[:,1])
```



In [ ]:

In [ ]:

In [304]:

```
logreg = LogisticRegression(max_iter=2000,class_weight='balanced')
```

In [305]:

```
X = scaler.fit_transform(X_train)

kfold = KFold(n_splits=5)
accuracy = np.mean(cross_val_score(logreg, X_train, y_train, cv=kfold, scorin
print("Cross Validation accuracy: {:.3f}".format(accuracy))
```

```
Cross Validation accuracy: 0.810
```

In [ ]:

In [ ]:

In [ ]: