In [3]:

```
cd drive/My\ Drive/Artivatic
```

```
[Errno 2] No such file or directory: 'drive/My Drive/Artivatic'
/content/drive/My Drive
```

In [11]:

```
import nltk
nltk.download('stopwords')
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Using GridSearchCV to find the best algorithm for this problem
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
# Importing essential libraries for visualization
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import pandas as pd
import seaborn as sns
import numpy as np
import xgboost as xgb
from tqdm import tqdm
from sklearn.svm import SVC
from keras.models import Sequential
from keras.layers.recurrent import LSTM, GRU
from keras.layers.core import Dense, Activation, Dropout
from keras.layers.embeddings import Embedding
from keras.layers.normalization import BatchNormalization
from keras.utils import np_utils
from sklearn import preprocessing, decomposition, model_selection, metrics, pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from keras.layers import GlobalMaxPooling1D, Conv1D, MaxPooling1D, Flatten, Bidirectional, SpatialD
ropout1D
from keras.preprocessing import sequence, text
from keras.callbacks import EarlyStopping
from nltk import word_tokenize
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
from sklearn.metrics import roc_curve, auc


# Configure visualisations
%matplotlib inline
sns.set_style( 'white' )
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [6]:

```
train = pd.read_csv('train_indessa.csv')
test= pd.read_csv('test_indessa.csv')
submission = pd.read_csv('sample_submission.csv')
```

In [7]:

In [7]:

```
train.shape , test.shape
```

Out[7]:

```
((532428, 45), (354951, 44))
```

In [8]:

```
train.head()
```

Out[8]:

	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	batch_enrolled	int_rate	grade	sub_grade	emp_title	emp_leng
0	58189336	14350	14350	14350.0	36 months		19.19	E	E3	clerk	9 ye
1	70011223	4800	4800	4800.0	36 months	BAT1586599	10.99	B	B4	Human Resources Specialist	< 1 y
2	70255675	10000	10000	10000.0	36 months	BAT1586599	7.26	A	A4	Driver	2 ye
3	1893936	15000	15000	15000.0	36 months	BAT4808022	19.72	D	D5	Us office of Personnel Management	10+ ye
4	7652106	16000	16000	16000.0	36 months	BAT2833642	10.64	B	B2	LAUSD-HOLLYWOOD HIGH SCHOOL	10+ ye

In [12]:

```
# Returns different datatypes for each columns (float, int, string, bool, etc.)
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 532428 entries, 0 to 532427
Data columns (total 45 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   member_id                    532428 non-null  int64
 1   loan_amnt                    532428 non-null  int64
 2   funded_amnt                  532428 non-null  int64
 3   funded_amnt_inv              532428 non-null  float64
 4   term                         532428 non-null  object
 5   batch_enrolled               447279 non-null  object
 6   int_rate                     532428 non-null  float64
 7   grade                        532428 non-null  object
 8   sub_grade                    532428 non-null  object
 9   emp_title                    501595 non-null  object
 10  emp_length                   505537 non-null  object
 11  home_ownership               532428 non-null  object
 12  annual_inc                   532425 non-null  float64
 13  verification_status          532428 non-null  object
 14  pymnt_plan                   532428 non-null  object
 15  desc                         75599 non-null   object
 16  purpose                      532428 non-null  object
 17  title                        532338 non-null  object
 18  zip_code                     532428 non-null  object
 19  addr_state                   532428 non-null  object
 20  dti                          532428 non-null  float64
 21  delinq_2yrs                  532412 non-null  float64
 22  inq_last_6mths               532412 non-null  float64
 23  mths_since_last_delinq       259874 non-null  float64
 24  mths_since_last_record       82123 non-null   float64
 25  open_acc                     532412 non-null  float64
 26  pub_rec                      532412 non-null  float64
 27  revol_bal                    532428 non-null  float64
```

```
 28  revol_util                  532141 non-null  float64
 29  total_acc                   532412 non-null  float64
 30  initial_list_status         532428 non-null  object
 31  total_rec_int               532428 non-null  float64
 32  total_rec_late_fee          532428 non-null  float64
 33  recoveries                  532428 non-null  float64
 34  collection_recovery_fee     532428 non-null  float64
 35  collections_12_mths_ex_med  532333 non-null  float64
 36  mths_since_last_major_derog 132980 non-null  float64
 37  application_type            532428 non-null  object
 38  verification_status_joint   305 non-null     object
 39  last_week_pay               532428 non-null  object
 40  acc_now_delinq              532412 non-null  float64
 41  tot_coll_amt                490424 non-null  float64
 42  tot_cur_bal                 490424 non-null  float64
 43  total_rev_hi_lim            490424 non-null  float64
 44  loan_status                 532428 non-null  int64
dtypes: float64(23), int64(4), object(18)
memory usage: 182.8+ MB
```

In [13]:

```
# Returns different datatypes for each columns (float, int, string, bool, etc.)
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 354951 entries, 0 to 354950
Data columns (total 44 columns):
 #   Column                      Non-Null Count   Dtype
---  ------                      --------------   -----
 0   member_id                   354951 non-null  int64
 1   loan_amnt                   354951 non-null  int64
 2   funded_amnt                 354951 non-null  int64
 3   funded_amnt_inv             354951 non-null  float64
 4   term                        354951 non-null  object
 5   batch_enrolled              309352 non-null  object
 6   int_rate                    354951 non-null  float64
 7   grade                       354951 non-null  object
 8   sub_grade                   354951 non-null  object
 9   emp_title                   334322 non-null  object
 10  emp_length                  337017 non-null  object
 11  home_ownership              354951 non-null  object
 12  annual_inc                  354950 non-null  float64
 13  verification_status         354951 non-null  object
 14  pymnt_plan                  354951 non-null  object
 15  desc                        50181 non-null   object
 16  purpose                     354951 non-null  object
 17  title                       354889 non-null  object
 18  zip_code                    354951 non-null  object
 19  addr_state                  354951 non-null  object
 20  dti                         354951 non-null  float64
 21  delinq_2yrs                 354938 non-null  float64
 22  inq_last_6mths              354938 non-null  float64
 23  mths_since_last_delinq      173193 non-null  float64
 24  mths_since_last_record      54930 non-null   float64
 25  open_acc                    354938 non-null  float64
 26  pub_rec                     354938 non-null  float64
 27  revol_bal                   354951 non-null  int64
 28  revol_util                  354736 non-null  float64
 29  total_acc                   354938 non-null  float64
 30  initial_list_status         354951 non-null  object
 31  total_rec_int               354951 non-null  float64
 32  total_rec_late_fee          354951 non-null  float64
 33  recoveries                  354951 non-null  float64
 34  collection_recovery_fee     354951 non-null  float64
 35  collections_12_mths_ex_med  354901 non-null  float64
 36  mths_since_last_major_derog 88723 non-null   float64
 37  application_type            354951 non-null  object
 38  verification_status_joint   206 non-null     object
 39  last_week_pay               354951 non-null  object
 40  acc_now_delinq              354938 non-null  float64
 41  tot_coll_amt                326679 non-null  float64
 42  tot_cur_bal                 326679 non-null  float64
 43  total_rev_hi_lim            326679 non-null  float64
dtypes: float64(22), int64(4), object(18)
memory usage: 119.2+ MB
```

```
memory usage: 119.2+ MB
```

Checking Null Values

In [9]:

```
train.isnull().sum()
```

Out[9]:

```
member_id                        0
loan_amnt                        0
funded_amnt                      0
funded_amnt_inv                  0
term                             0
batch_enrolled               85149
int_rate                         0
grade                            0
sub_grade                        0
emp_title                    30833
emp_length                   26891
home_ownership                   0
annual_inc                       3
verification_status              0
pymnt_plan                       0
desc                        456829
purpose                          0
title                           90
zip_code                         0
addr_state                       0
dti                              0
delinq_2yrs                     16
inq_last_6mths                  16
mths_since_last_delinq      272554
mths_since_last_record      450305
open_acc                        16
pub_rec                         16
revol_bal                        0
revol_util                     287
total_acc                       16
initial_list_status              0
total_rec_int                    0
total_rec_late_fee               0
recoveries                       0
collection_recovery_fee          0
collections_12_mths_ex_med      95
mths_since_last_major_derog 399448
application_type                 0
verification_status_joint   532123
last_week_pay                    0
acc_now_delinq                  16
tot_coll_amt                 42004
tot_cur_bal                  42004
total_rev_hi_lim             42004
loan_status                      0
dtype: int64
```

In [10]:

```
test.isnull().sum()
```

Out[10]:

```
member_id                        0
loan_amnt                        0
funded_amnt                      0
funded_amnt_inv                  0
term                             0
batch_enrolled               45599
int_rate                         0
grade                            0
sub_grade                        0
emp_title                    20629
emp_length                   17934
```

```
emp_length                     17934
home_ownership                     0
annual_inc                         1
verification_status                0
pymnt_plan                         0
desc                          304770
purpose                            0
title                             62
zip_code                           0
addr_state                         0
dti                                0
delinq_2yrs                       13
inq_last_6mths                    13
mths_since_last_delinq        181758
mths_since_last_record        300021
open_acc                          13
pub_rec                           13
revol_bal                          0
revol_util                       215
total_acc                         13
initial_list_status                0
total_rec_int                      0
total_rec_late_fee                 0
recoveries                         0
collection_recovery_fee            0
collections_12_mths_ex_med        50
mths_since_last_major_derog   266228
application_type                   0
verification_status_joint     354745
last_week_pay                      0
acc_now_delinq                    13
tot_coll_amt                   28272
tot_cur_bal                    28272
total_rev_hi_lim               28272
dtype: int64
```

- Lots of features has Null Values

Exploratory Data Analysis :

- Let's dive deep into each feature and gain some insights about the data.

CATEGORICAL FEATURES

In [30]:

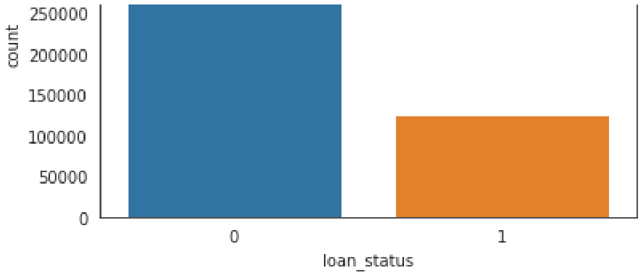
```
### pivot_table for two categorical features one being loan_status
def pivot(col):
  return pd.pivot_table(train,'member_id',index=[col],columns=['loan_status'],aggfunc='count')
```

Lets check the Data Imbalance

In [60]:

```
#Imbalanced dataset
ax = sns.countplot(x=train['loan_status'], data=train)
print(train['loan_status'].value_counts())
```

```
0    406601
1    125827
Name: loan_status, dtype: int64
```



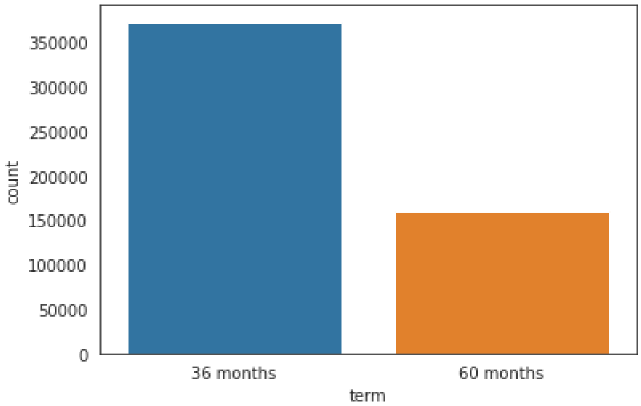
- Imbalanced Dataset, so ROC-AUC score will be the best metric for this problem

Let's check term of loan (in months)

In [15]:

```
ax = sns.countplot(x=train['term'], data=train)
print(train['term'].value_counts())
```

```
36 months    372793
60 months    159635
Name: term, dtype: int64
```



In [37]:

```
pivot('term')
```

Out[37]:

loan_status	0	1
term		
36 months	271120	101673
60 months	135481	24154

- Most people opt for 36 months or 3 years term loan

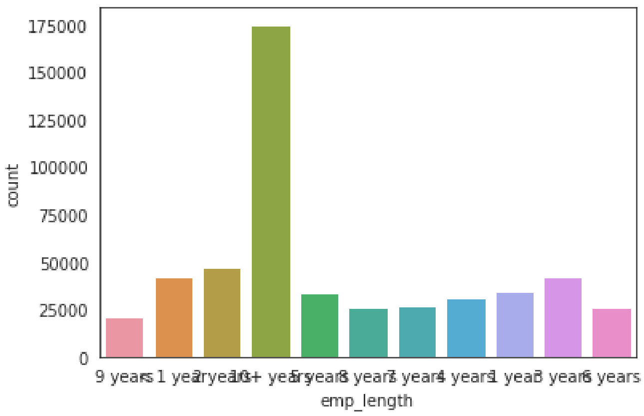
Let's check employment length, where 0 means less than one year and 10 means ten or more years

In [27]:

```
ax = sns.countplot(x=train['emp_length'], data=train)
print(train['emp_length'].value_counts())
```

```
10+ years    175105
2 years       47276
```

```
< 1 year      42253
3 years       42175
1 year        34202
5 years       33393
4 years       31581
7 years       26680
8 years       26443
6 years       25741
9 years       20688
Name: emp_length, dtype: int64
```



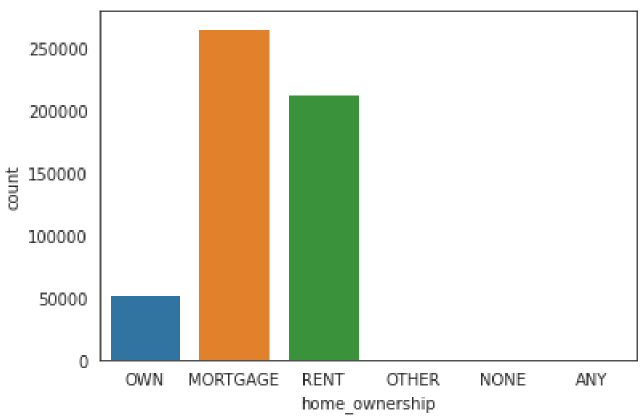
- People who are employed for 10+ years tend to take loans more since they might have stable income source now . So, it will be beneficial to target those audience with 'emp_length = 10+years' .

Let's check status of home ownership

In [28]:

```
ax = sns.countplot(x=train['home_ownership'], data=train)
print(train['home_ownership'].value_counts())
```

```
MORTGAGE    265940
RENT        213668
OWN          52664
OTHER          117
NONE            36
ANY              3
Name: home_ownership, dtype: int64
```



- We can see that most people keep their home as mortgage to take huge amounts of loan .
- Bigger the house , more is the income of the person, more is the employment lenth, more amount of loan can be taken

In [32]:

```
## Defaulter =1 , Non-Defaulter=0
pivot('home_ownership')
```

```
pivot('home_ownership')
```

Out[32]:

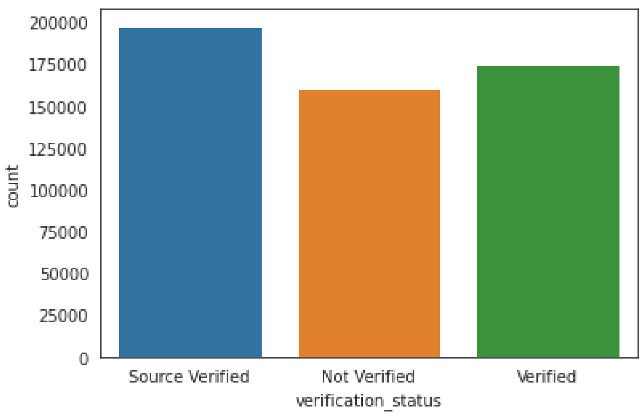
loan_status	0	1
home_ownership		
ANY	2	1
MORTGAGE	202344	63596
NONE	8	28
OTHER	27	90
OWN	41737	10927
RENT	162483	51185

Let's check status of income verified by the bank

In [33]:

```
ax = sns.countplot(x=train['verification_status'], data=train)
print(train['verification_status'].value_counts())
```

```
Source Verified    197750
Verified           174702
Not Verified       159976
Name: verification_status, dtype: int64
```



In [34]:

```
## Defaulter =1 , Non-Defaulter=0
pivot('verification_status')
```

Out[34]:

loan_status	0	1
verification_status		
Not Verified	115028	44948
Source Verified	161329	36421
Verified	130244	44458

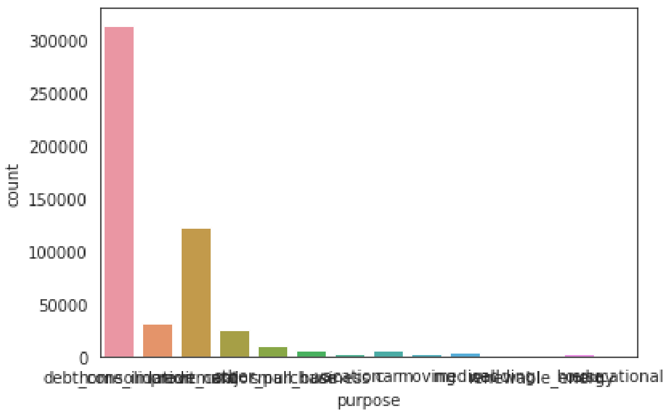
- We can see that : **For Defaulters - status of verified income by the bank is also less. So, not trustworthy**

Let's view the purpose of loans

In [35]:

```
ax = sns.countplot(x=train['purpose'], data=train)
print(train['purpose'].value_counts())
```

```
debt_consolidation    314989
credit_card           123670
home_improvement       31087
other                  25652
major_purchase         10284
small_business          6146
car                     5266
medical                 5117
moving                  3243
vacation                2812
house                   2170
wedding                 1401
renewable_energy         331
educational              260
Name: purpose, dtype: int64
```

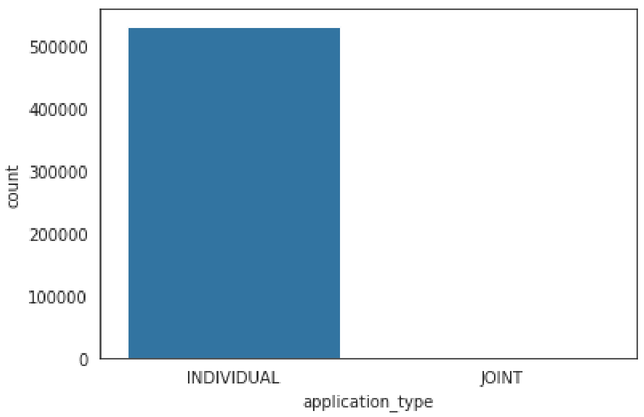


Let's check for loan application types

In [38]:

```
ax = sns.countplot(x=train['application_type'], data=train)
print(train['application_type'].value_counts())
```

```
INDIVIDUAL    532123
JOINT            305
Name: application_type, dtype: int64
```



- Most people opt for individual loans

In [39]:

```
pivot('application_type')
```

Out[39]:

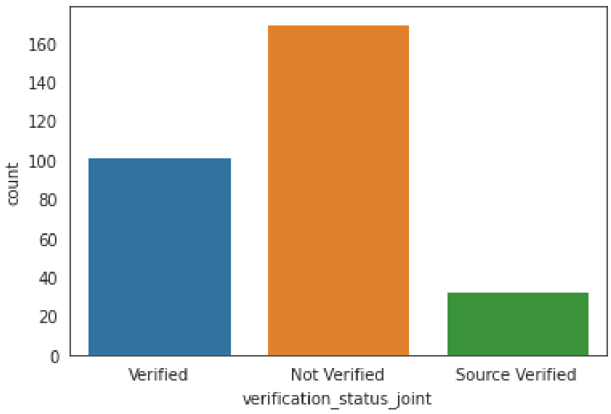
loan_status	0	1
application_type		
INDIVIDUAL	406297	125826
JOINT	304	1

Let's check 'verification_status_joint'. This should be ignorable quantities for "Defaulters"

In [43]:

```
ax = sns.countplot(x=train['verification_status_joint'], data=train)
print(train['verification_status_joint'].value_counts())
```

```
Not Verified       170
Verified           102
Source Verified     33
Name: verification_status_joint, dtype: int64
```



In [44]:

```
## Defaulter =1 , Non-Defaulter=0
pivot('verification_status_joint')
```

Out[44]:

loan_status	0	1
verification_status_joint		
Not Verified	170.0	NaN
Source Verified	32.0	1.0
Verified	102.0	NaN

- Previously we saw in 'application_type' , there was ignorable quantity of Defaulters in Joint loan category.
- Here also we can see that , if the Defaulters dont take the joint application option then obviously they wont have any income to be verified by the bank.
- Makes Sense !

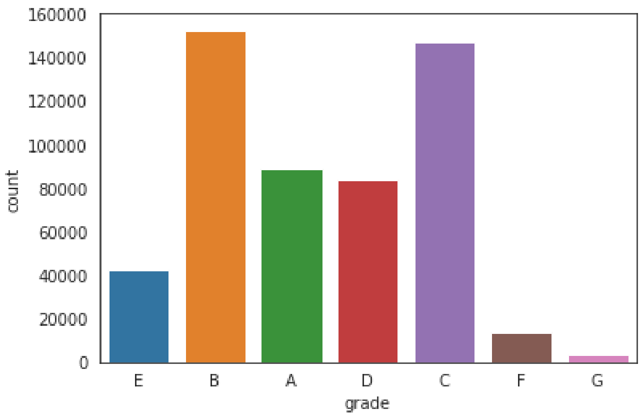
Let's check the Grade assigned by the bank

- Loan grading is a classification system that involves assigning a quality score to a loan based on a borrower's credit history, quality of the collateral, and the likelihood of repayment of the principal and interest.

In [45]:

```
ax = sns.countplot(x=train['grade'], data=train)
print(train['grade'].value_counts())
```

```
B    152713
C    147499
A     89107
D     83567
E     42495
F     13826
G      3221
Name: grade, dtype: int64
```



In [46]:

```
## Defaulter =1 , Non-Defaulter=0
pivot('grade')
```

Out[46]:

loan_status	0	1
grade		
A	65148	23959
B	112507	40206
C	115579	31920
D	65419	18148
E	34553	7942
F	10934	2892
G	2461	760

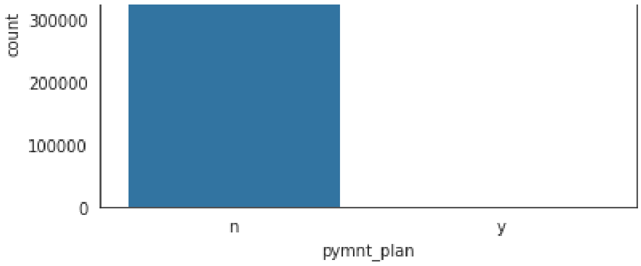
Let's see the 'payment plan'

- indicates if any payment plan has started against loan

In [47]:

```
ax = sns.countplot(x=train['pymnt_plan'], data=train)
print(train['pymnt_plan'].value_counts())
```

```
n    532420
y         8
Name: pymnt_plan, dtype: int64
```



In [48]:

```
## Defaulter =1 , Non-Defaulter=0
pivot('pymnt_plan')
```

Out[48]:

loan_status	**0**	**1**
pymnt_plan		
n	406595	125825
y	6	2

- The status = 'n' is more , so payment plan for the loans granted have not started for majority.

Let's check the 'State' where loan is granted maximum.

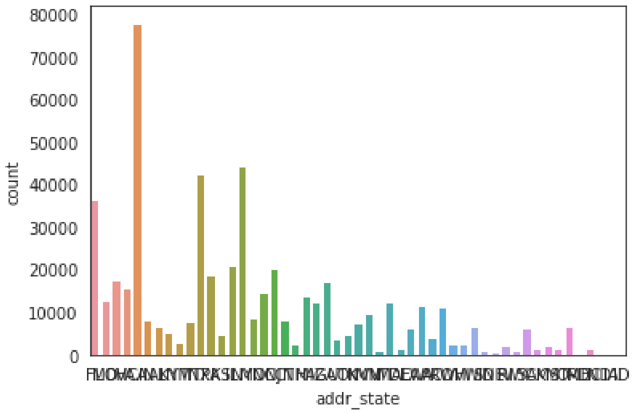
- Should be places where income is comparatively larger than usual

In [49]:

```
ax = sns.countplot(x=train['addr_state'], data=train)
print(train['addr_state'].value_counts())
```

```
CA    77911
NY    44406
TX    42527
FL    36575
IL    21205
NJ    20103
PA    18882
OH    17778
GA    17292
VA    15826
NC    14812
MI    13869
MD    12667
MA    12385
AZ    12320
WA    11664
CO    11233
MN     9577
MO     8538
IN     8197
CT     8075
TN     7817
NV     7408
WI     6880
AL     6699
OR     6549
SC     6331
LA     6304
KY     5140
KS     4818
OK     4797
AR     3988
UT     3829
NM     2958
```

```
HI      2705
WV      2615
NH      2568
RI      2322
MS      2296
MT      1545
DE      1512
DC      1477
AK      1346
WY      1198
SD      1091
VT      1062
NE       709
ME       324
ND       284
IA         7
ID         7
Name: addr_state, dtype: int64
```



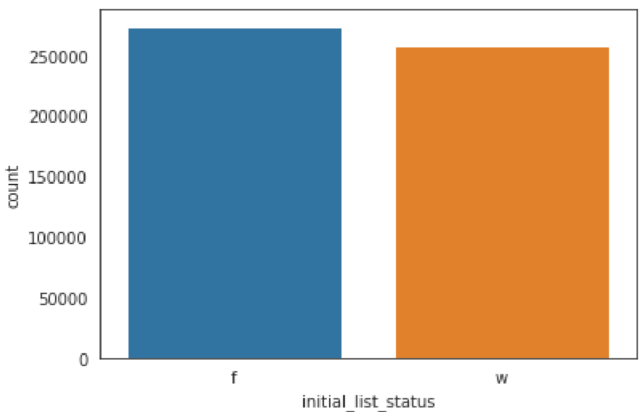
- We can see that for CA i.e. California , it is highest. Since the incomes are also very high there.

Let's check the unique listing status of the loan - W(Waiting), F(Forwarded)

In [51]:

```
ax = sns.countplot(x=train['initial_list_status'], data=train)
print(train['initial_list_status'].value_counts())
```

```
f    274018
w    258410
Name: initial_list_status, dtype: int64
```



In [52]:

```
pivot('initial_list_status')
```

Out[52]:

loan_status	0	1
initial_list_status		
f	183320	90698
w	223281	35129

- Unique pattern :
- For Non Defaulters : loans are in waiting state more than in forwarded state.
- For Defaulters : loans are in forwarded state more than in waiting state.

NUMERICAL FEATURES

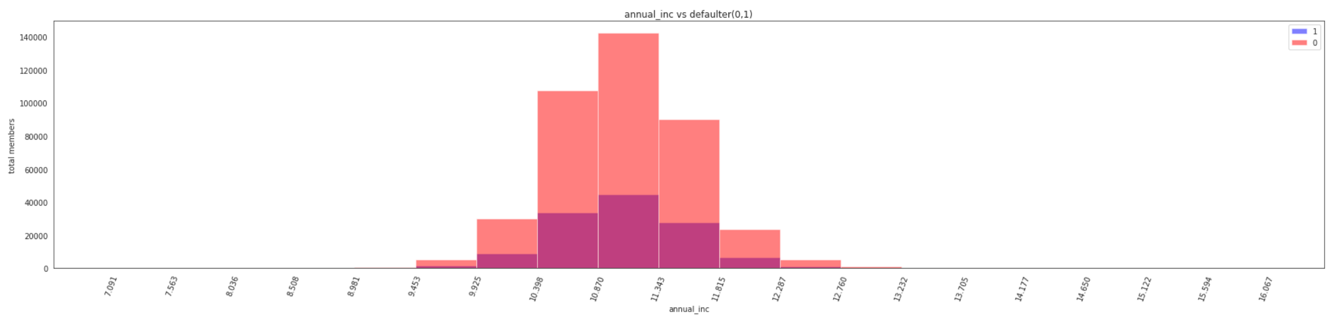
In [64]:

```
def plot_hist(df,col,bin_size,log=None):
    fig = plt.figure(figsize=(30, 6))
    ax = fig.add_subplot(111)
    if log==True:
        x0 = np.log(df[df.loan_status==0][col].dropna().values+1)
        x1 = np.log(df[df.loan_status==1][col].dropna().values+1)
        min_ = min(np.log(df[col].dropna().values+1))
        max_ = max(np.log(df[col].dropna().values+1))
        bins = np.linspace(min_,max_,bin_size)
    else:
        x0 = df[df.loan_status==0][col].dropna().values
        x1 = df[df.loan_status==1][col].dropna().values
        bins = np.linspace(df[col].min(),df[col].max(),bin_size)
    ax.hist(x1,bins=bins,label='1',color='b',alpha=0.5)
    ax.hist(x0,bins=bins,label='0',color='r',alpha=0.5)
    ax.set_ylabel("total members")
    ax.set_xlabel(col)
    ax.set_title("{} vs defaulter(0,1)".format(col))
    ax.set_xticks(bins)
    plt.xticks(rotation=70)
    plt.legend(loc='upper right')
    plt.show()
    return
```

Firstly let's chek the annual incomes

In [80]:

```
plot_hist(train,'annual_inc',20,True)
```



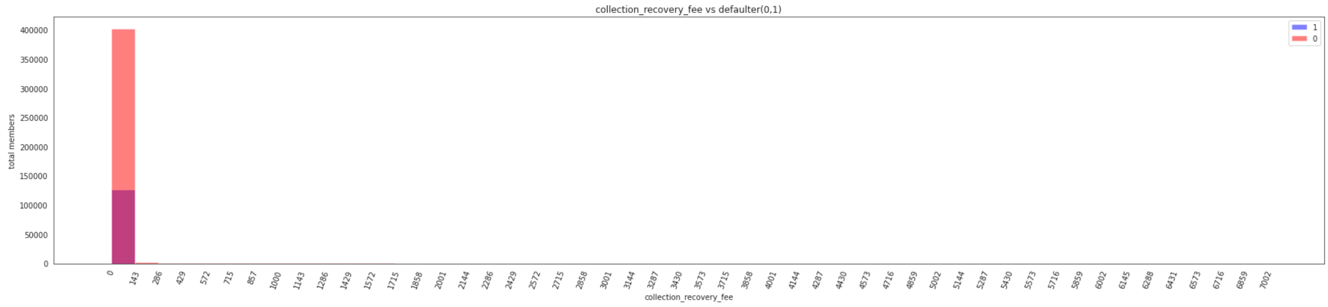
- Annual Income of Defaulter is very less, no doubt that they are referred to as Defaulters by bank

Let's check post charge off collection fee :

- A charge-off refers to debt that a company believes it will no longer collect as the borrower has become delinquent on payments.

In [65]:

```
plot_hist(train,'collection_recovery_fee',50)
```

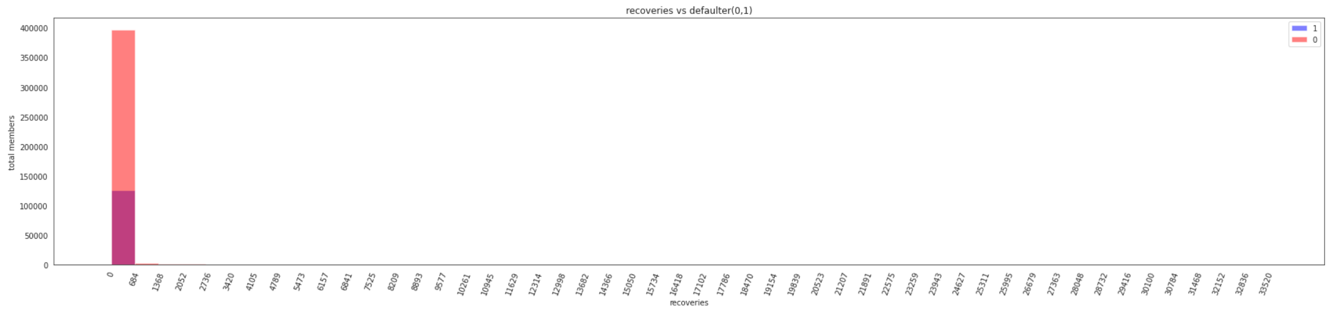


- post charge off collection fee is 0 means, most loan bearers have no charge-off debt to the bank,
- Defaulters are again less in this.

Let;s check post charge off gross recovery fee:

In [66]:

```
plot_hist(train,'recoveries',50)
```

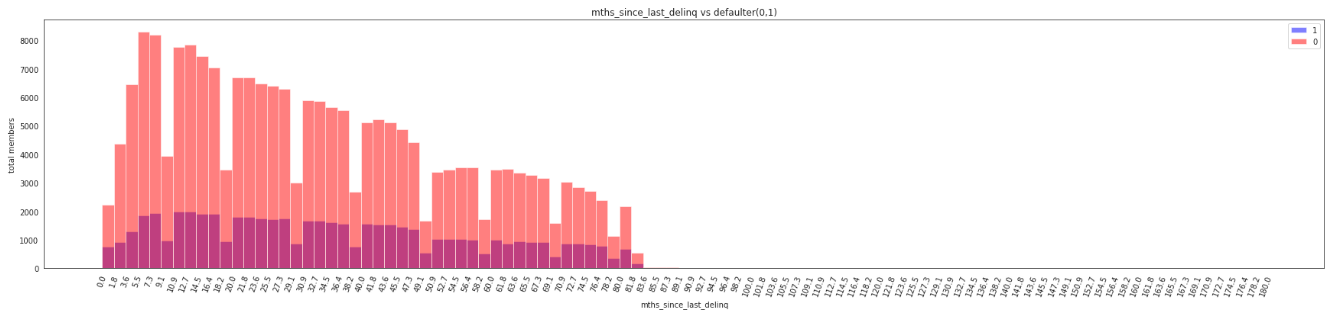


Let's check 'mths_since_last_delinq'.

- the term "delinquent" commonly refers to a situation where a borrower is late or overdue on a payment, such as income taxes, a mortgage, an automobile loan, or a credit card account. There are consequences for being delinquent, depending on the type, duration and cause of the delinquency.

In [67]:

```
plot_hist(train,'mths_since_last_delinq',100)
```

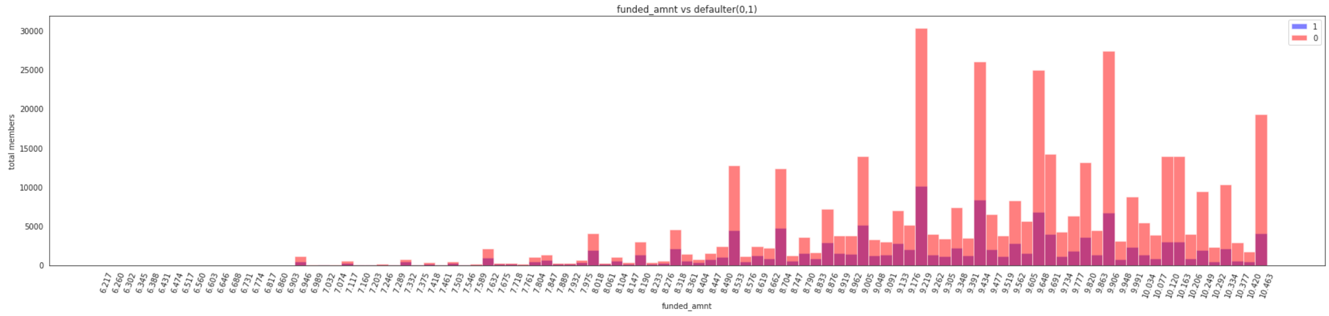


Let's check the funded amount

- loan amount ($) sanctioned by the bank

In [72]:

```
plot_hist(train,'funded_amnt',100,True)
```

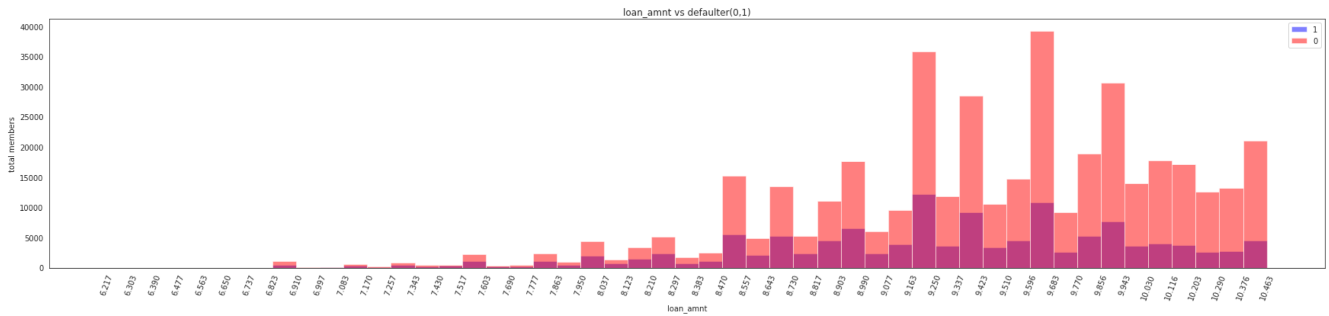


- max loan amount santioned in dollar : between 9.176 to 9.219

Let'check the loan amount ($) applied by the member

In [74]:

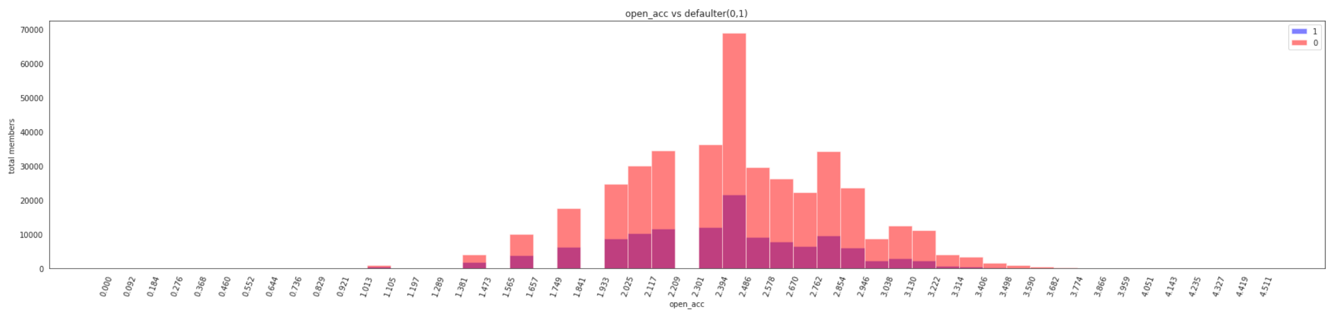
```
plot_hist(train,'loan_amnt',50,True)
```



Let's check number of open credit line in member's credit line

In [75]:

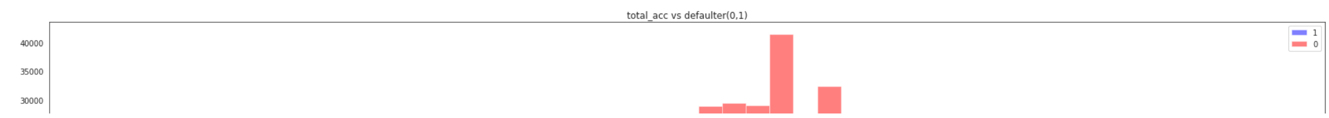
```
plot_hist(train,'open_acc',50,True)
```

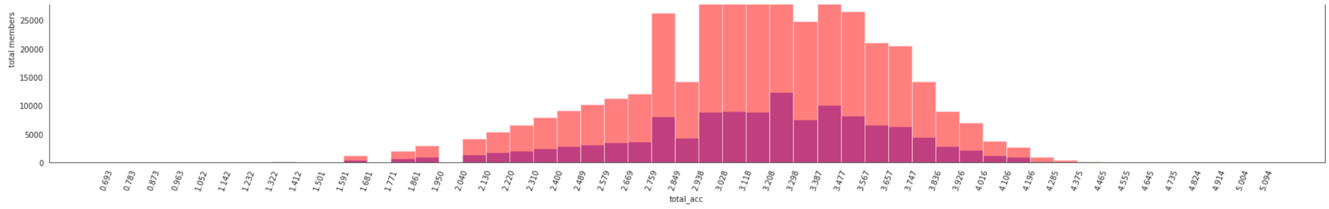


Let's check total number of credit lines available in members credit line

In [76]:

```
plot_hist(train,'total_acc',50,True)
```

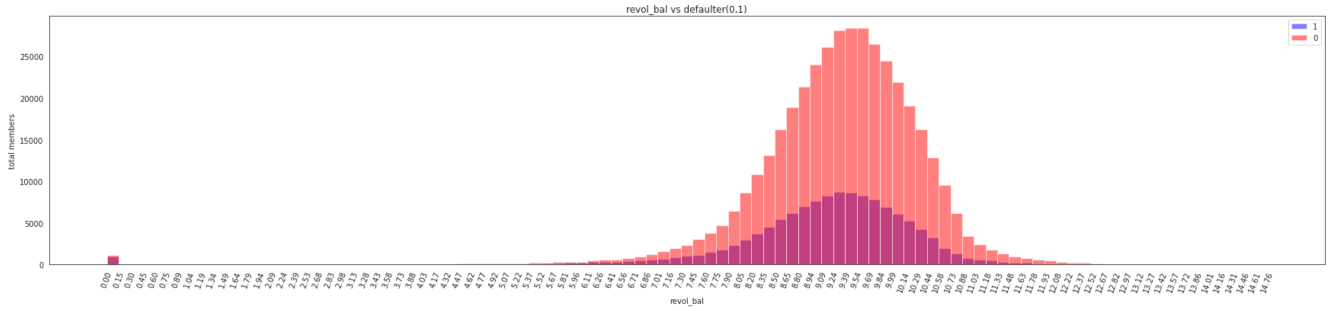




Let's check total credit revolving balance

In [77]:

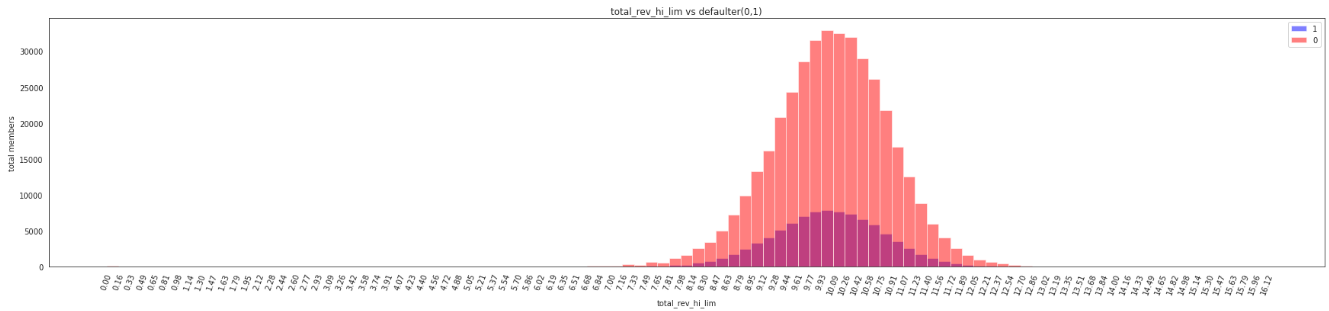
```
plot_hist(train,'revol_bal',100,True)
```



Let's check total revolving credit limit

In [78]:

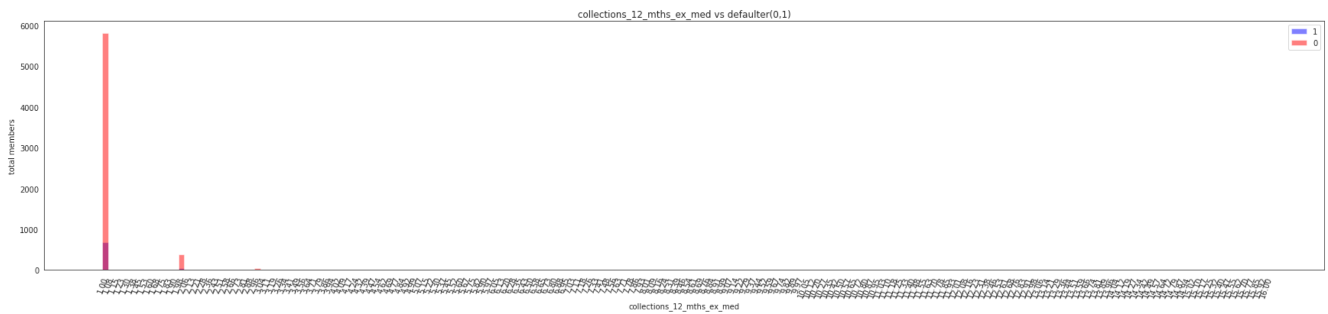
```
plot_hist(train,'total_rev_hi_lim',100,True)
```



Let's check number of collections in last 12 months excluding medical collections

In [82]:

```
plot_hist(train[train.collections_12_mths_ex_med!=0],'collections_12_mths_ex_med',200)
```

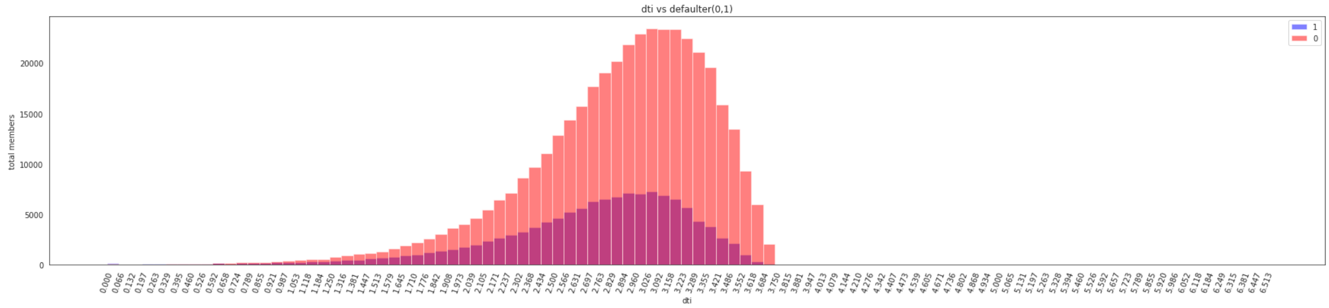


Let's check ratio of member's total monthly debt repayment excluding

Let's check ratio of member's total monthly debt repayment excluding mortgage divided by self reported monthly income

In [83]:

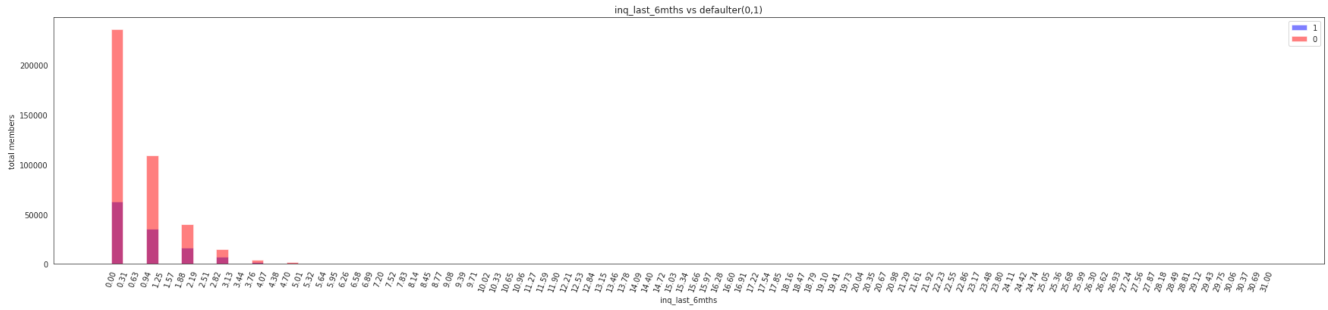
```
plot_hist(train,'dti',100,True)
```



Let's check number of inquiries in last 6 months

In [84]:

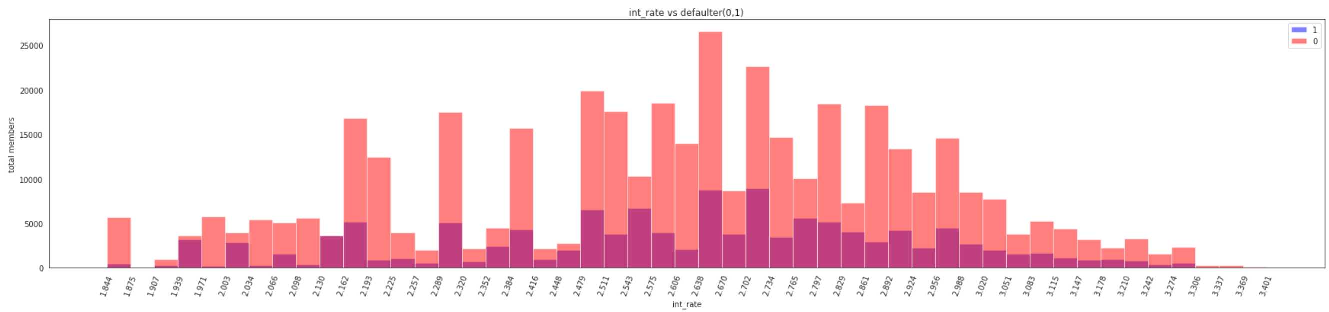
```
plot_hist(train,'inq_last_6mths',100)
```



Let's check interest rate (%) on loan

In [85]:

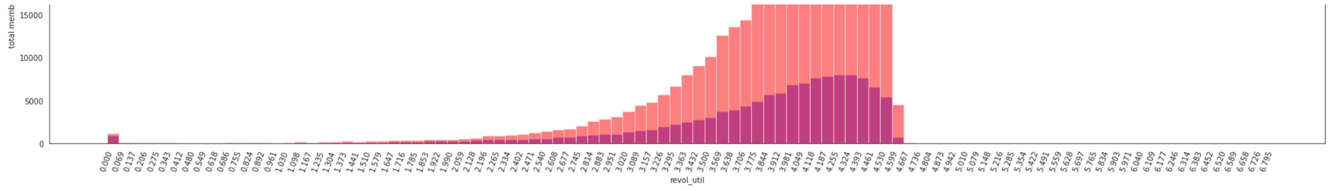
```
plot_hist(train,'int_rate',50,True)
```



Let's check amount of credit a member is using relative to revol_bal

In [86]:

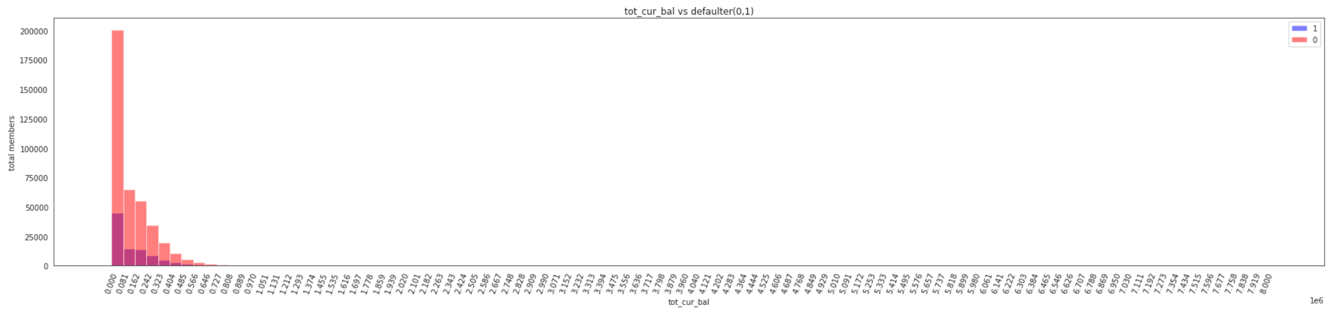
```
plot_hist(train,'revol_util',100,True)
```



Let's check total current balance of all accounts

In [87]:

```
plot_hist(train,'tot_cur_bal',100)
```



- More people have less total current balance of all accounts

In [ ]:

```
```