In [1]:

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

In [42]:

```
kisan=pd.read_csv('data_folder/kisan_net_log.csv')
kisan.head()
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\IPython\core\interactiveshell.p
y:3063: DtypeWarning: Columns (19,29,30) have mixed types.Specify dtype option
on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[42]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ow
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	
1	Mendozaberg	OK 22690"	NaN	NaN	NaN	NaN	NaN	NaN	
2	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MOR
3	Loganmouth	SD 05113"	NaN	NaN	NaN	NaN	NaN	NaN	
4	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	

5 rows × 31 columns

In [43]:
```
kisan1=kisan.iloc[0::2,0:27]
kisan1.head()
```

Out[43]:

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ow
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	
2	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MOR
4	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	
6	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	
8	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MOR

5 rows × 27 columns

EDA

Target Variable-Loan_Status

In [44]:
```
kisan1['loan_status'].value_counts()
```

Out[44]:
```
Fully Paid      78498
Charged Off     19294
Jun-13            125
Aug-13            107
May-13             99
                ...  
35000               1
40000               1
Sep-07              1
42000               1
81090               1
Name: loan_status, Length: 127, dtype: int64
```

In [45]:
```
kisan2=kisan1[(kisan1['loan_status']=='Fully Paid')|(kisan1['loan_status']=='Char
kisan2.shape
```

Out[45]:
```
(97792, 27)
```

In [46]:
```
kisan2['loan_status'].value_counts(normalize=True)
#Highly Imbalanced dataset
```

Out[46]:
```
Fully Paid     0.802704
Charged Off    0.197296
Name: loan_status, dtype: float64
```

In [47]:
```
kisan2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 97792 entries, 0 to 200488
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             97792 non-null  object 
 1   term                  97792 non-null  object 
 2   int_rate              97792 non-null  float64
 3   installment           97792 non-null  float64
 4   grade                 97792 non-null  object 
 5   sub_grade             97792 non-null  object 
 6   emp_title             92061 non-null  object 
 7   emp_length            93190 non-null  object 
 8   home_ownership        97792 non-null  object 
 9   annual_inc            97792 non-null  object 
 10  verification_status   97792 non-null  object 
 11  issue_d               97792 non-null  object 
 12  loan_status           97792 non-null  object 
 13  purpose               97792 non-null  object 
 14  title                 97351 non-null  object 
 15  dti                   97792 non-null  object 
 16  earliest_cr_line      97792 non-null  object 
 17  open_acc              97792 non-null  object 
 18  pub_rec               97792 non-null  object 
 19  revol_bal             97792 non-null  object 
 20  revol_util            97723 non-null  float64
 21  total_acc             97792 non-null  float64
 22  initial_list_status   97792 non-null  object 
 23  application_type      97792 non-null  object 
 24  mort_acc              88924 non-null  object 
 25  pub_rec_bankruptcies  97607 non-null  object 
 26  address               97782 non-null  object 
dtypes: float64(4), object(23)
memory usage: 20.9+ MB
```

In [48]:
```
kisan2.isnull().sum()
```

Out[48]:
```
loan_amnt                  0
term                       0
int_rate                   0
installment                0
grade                      0
sub_grade                  0
emp_title               5731
emp_length              4602
home_ownership             0
annual_inc                 0
verification_status        0
issue_d                    0
loan_status                0
purpose                    0
title                    441
dti                        0
earliest_cr_line           0
open_acc                   0
pub_rec                    0
revol_bal                  0
revol_util                69
total_acc                  0
initial_list_status        0
application_type           0
mort_acc                8868
pub_rec_bankruptcies     185
address                   10
dtype: int64
```

In [49]:
```
import re
pattern='[a-zA-Z]'
cols=['loan_amnt','annual_inc','dti','revol_bal','mort_acc']
for i in cols:
    kisan2[i]=list(map(lambda x:'nan'if re.search(pattern,str(x))else x,kisan2[i]
    kisan2=kisan2[kisan2[i]!='nan']
kisan2.shape
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:5: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """
```

Out[49]:
```
(88741, 27)
```

Changing the data types of some features to floats

In [50]:

```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
     'revol_bal','revol_util']
for i in cols:
    kisan2[i]=list(map(lambda x:float(x),kisan2[i]))
kisan2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88741 entries, 0 to 200488
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             88741 non-null  float64
 1   term                  88741 non-null  object 
 2   int_rate              88741 non-null  float64
 3   installment           88741 non-null  float64
 4   grade                 88741 non-null  object 
 5   sub_grade             88741 non-null  object 
 6   emp_title             83607 non-null  object 
 7   emp_length            84403 non-null  object 
 8   home_ownership        88741 non-null  object 
 9   annual_inc            88741 non-null  float64
 10  verification_status   88741 non-null  object 
 11  issue_d               88741 non-null  object 
 12  loan_status           88741 non-null  object 
 13  purpose               88741 non-null  object 
 14  title                 88302 non-null  object 
 15  dti                   88741 non-null  float64
 16  earliest_cr_line      88741 non-null  object 
 17  open_acc              88741 non-null  object 
 18  pub_rec               88741 non-null  object 
 19  revol_bal             88741 non-null  float64
 20  revol_util            88684 non-null  float64
 21  total_acc             88741 non-null  float64
 22  initial_list_status   88741 non-null  object 
 23  application_type      88741 non-null  object 
 24  mort_acc              88741 non-null  object 
 25  pub_rec_bankruptcies  88741 non-null  object 
 26  address               88741 non-null  object 
dtypes: float64(8), object(19)
memory usage: 19.0+ MB
```

In [51]:

```
cols=['open_acc', 'pub_rec','mort_acc',
      'pub_rec_bankruptcies']
for i in cols:
    kisan2[i]=list(map(lambda x:int(x),kisan2[i]))
kisan2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88741 entries, 0 to 200488
Data columns (total 27 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   loan_amnt             88741 non-null  float64
 1   term                  88741 non-null  object 
 2   int_rate              88741 non-null  float64
 3   installment           88741 non-null  float64
 4   grade                 88741 non-null  object 
 5   sub_grade             88741 non-null  object 
 6   emp_title             83607 non-null  object 
 7   emp_length            84403 non-null  object 
 8   home_ownership        88741 non-null  object 
 9   annual_inc            88741 non-null  float64
 10  verification_status   88741 non-null  object 
 11  issue_d               88741 non-null  object 
 12  loan_status           88741 non-null  object 
 13  purpose               88741 non-null  object 
 14  title                 88302 non-null  object 
 15  dti                   88741 non-null  float64
 16  earliest_cr_line      88741 non-null  object 
 17  open_acc              88741 non-null  int64  
 18  pub_rec               88741 non-null  int64  
 19  revol_bal             88741 non-null  float64
 20  revol_util            88684 non-null  float64
 21  total_acc             88741 non-null  float64
 22  initial_list_status   88741 non-null  object 
 23  application_type      88741 non-null  object 
 24  mort_acc              88741 non-null  int64  
 25  pub_rec_bankruptcies  88741 non-null  int64  
 26  address               88741 non-null  object 
dtypes: float64(8), int64(4), object(15)
memory usage: 19.0+ MB
```

In [52]:

```
kisan2['title'].value_counts()
```

Out[52]:

```
Debt consolidation                       37962
Credit card refinancing                  12970
Home improvement                          3865
Other                                     3179
Debt Consolidation                        2191
                                         ...  
Home Improvement/Consolidation               1
loan payback then purchase new shop          1
consolidation of credit cards                1
Debt Consolidation & CC Reduction            1
Persona                                      1
Name: title, Length: 10289, dtype: int64
```

In [53]:
```
kisan2['purpose'].value_counts()
```

Out[53]:
```
debt_consolidation    53511
credit_card           19352
home_improvement       5359
other                  4275
major_purchase         1723
small_business          939
medical                 889
car                     808
moving                  599
vacation                549
house                   468
wedding                 215
renewable_energy         53
educational               1
Name: purpose, dtype: int64
```

Categorising purpose into:debt_consolidation,credit_card and others

In [54]:
```
z=['debt_consolidation','credit_card']
kisan2['purpose']=list(map(lambda x:x if x in z else 'others',kisan2['purpose'])
kisan2['purpose'].value_counts()
```

Out[54]:
```
debt_consolidation    53511
credit_card           19352
others                15878
Name: purpose, dtype: int64
```

In [55]:
```
kisan2['home_ownership'].value_counts()
```

Out[55]:
```
MORTGAGE    45023
RENT        35083
OWN          8613
OTHER          13
NONE            7
ANY             2
Name: home_ownership, dtype: int64
```

Categorising home_ownership into:Mortgage,Rent and Others

In [56]:
```
z=['MORTGAGE','RENT']
kisan2['home_ownership']=list(map(lambda x:x if x in z else 'OTHERS',kisan2['home
kisan2['home_ownership'].value_counts()
```

Out[56]:
```
MORTGAGE    45023
RENT        35083
OTHERS       8635
Name: home_ownership, dtype: int64
```

In [57]:
```
kisan2['verification_status'].value_counts()
```

Out[57]:
```
Verified           31144
Source Verified    30265
Not Verified       27332
Name: verification_status, dtype: int64
```

Merging the Veified and Source Verified for verification_status

In [58]:
```
z=['Verified','Source Verified']
kisan2['verification_status']=list(map(lambda x:'Verified' if x in z else x,kisan
kisan2['verification_status'].value_counts()
```

Out[58]:
```
Verified        61409
Not Verified    27332
Name: verification_status, dtype: int64
```

pub_rec:No. of derogatory public records

In [59]:
```
kisan2['pub_rec'].value_counts()
```

Out[59]:
```
0     74769
1     11950
2      1393
3       360
4       147
5        63
6        27
8        15
7         8
10        3
11        2
9         2
19        1
13        1
Name: pub_rec, dtype: int64
```

In [60]:
```
z=[0,1]
kisan2['pub_rec']=list(map(lambda x:x if x in z else 1,kisan2['pub_rec']))
kisan2['pub_rec'].value_counts()
```

Out[60]:
```
0    74769
1    13972
Name: pub_rec, dtype: int64
```

In [61]:
```
kisan2['term'].value_counts()
```

Out[61]:
```
 36 months    67798
 60 months    20943
Name: term, dtype: int64
```

In [62]:
```
kisan2['is_36mnths']=list(map(lambda x:1 if x==' 36 months'else 0,kisan2['term'])
kisan2['is_36mnths'].value_counts()
```

Out[62]:
```
1    67798
0    20943
Name: is_36mnths, dtype: int64
```

In [63]:
```
kisan2['is_verified']=list(map(lambda x:1 if x=='Verified'else 0,kisan2['verifica
kisan2['is_verified'].value_counts()
```

Out[63]:
```
1    61409
0    27332
Name: is_verified, dtype: int64
```

In [64]:
```
kisan2['grade'].value_counts()
```

Out[64]:
```
B    25968
C    24431
D    14598
A    13215
E     7207
F     2642
G      680
Name: grade, dtype: int64
```

In [65]:
```
d={'A':0,'B':1,'C':2,'D':3,'E':4,'F':5,'G':6}
kisan2['mod_grade']=list(map(lambda x:d[x],kisan2['grade']))
kisan2[['grade','mod_grade']].head(20)
```

Out[65]:

	grade	mod_grade
0	B	1
2	B	1
4	B	1
6	A	0
8	C	2
10	C	2
12	A	0
14	B	1
16	B	1
18	C	2
20	B	1
22	C	2
24	B	1
26	C	2
28	A	0
30	A	0
32	E	4
34	C	2
36	A	0
38	A	0

In [66]:
```
kisan2=pd.get_dummies(kisan2,columns=['home_ownership','purpose'],drop_first=True
```

In [67]:
```
cols=['home_ownership_OTHERS','home_ownership_RENT','purpose_debt_consolidation',
     'purpose_others']
for i in cols:
    kisan2[i]=list(map(lambda x:int(x),kisan2[i]))
kisan2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 88741 entries, 0 to 200488
Data columns (total 32 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   loan_amnt                   88741 non-null  float64
 1   term                        88741 non-null  object 
 2   int_rate                    88741 non-null  float64
 3   installment                 88741 non-null  float64
 4   grade                       88741 non-null  object 
 5   sub_grade                   88741 non-null  object 
 6   emp_title                   83607 non-null  object 
 7   emp_length                  84403 non-null  object 
 8   annual_inc                  88741 non-null  float64
 9   verification_status         88741 non-null  object 
 10  issue_d                     88741 non-null  object 
 11  loan_status                 88741 non-null  object 
 12  title                       88302 non-null  object 
 13  dti                         88741 non-null  float64
 14  earliest_cr_line            88741 non-null  object 
 15  open_acc                    88741 non-null  int64  
 16  pub_rec                     88741 non-null  int64  
 17  revol_bal                   88741 non-null  float64
 18  revol_util                  88684 non-null  float64
 19  total_acc                   88741 non-null  float64
 20  initial_list_status         88741 non-null  object 
 21  application_type            88741 non-null  object 
 22  mort_acc                    88741 non-null  int64  
 23  pub_rec_bankruptcies        88741 non-null  int64  
 24  address                     88741 non-null  object 
 25  is_36mnths                  88741 non-null  int64  
 26  is_verified                 88741 non-null  int64  
 27  mod_grade                   88741 non-null  int64  
 28  home_ownership_OTHERS       88741 non-null  int64  
 29  home_ownership_RENT         88741 non-null  int64  
 30  purpose_debt_consolidation  88741 non-null  int64  
 31  purpose_others              88741 non-null  int64  
dtypes: float64(8), int64(11), object(13)
memory usage: 22.3+ MB
```

Type *Markdown* and LaTeX: α^2

In [68]:

```
kisan2['application_type'].value_counts(normalize=True)
```

Out[68]:

```
INDIVIDUAL    0.998073
JOINT         0.001127
DIRECT_PAY    0.000800
Name: application_type, dtype: float64
```

In [69]:

```
kisan2['initial_list_status'].value_counts(normalize=True)
```

Out[69]:

```
f    0.555775
w    0.444225
Name: initial_list_status, dtype: float64
```

In [70]:

```
d={'f':0,'w':1}
kisan2['initial_list_status']=list(map(lambda x:d[x] ,kisan2['initial_status
kisan2['initial_list_status'].value_counts()
```

Out[70]:

```
0    49320
1    39421
Name: initial_list_status, dtype: int64
```

In [72]:

```
kisan2['pub_rec_bankruptcies'].value_counts()
```

Out[72]:

```
0    77846
1    10303
2      475
3       79
4       25
5        8
6        4
7        1
Name: pub_rec_bankruptcies, dtype: int64
```

pub_rec_bankruptcies:Number of public record bankruptcies

In [73]:

```
z=[0,1]
kisan2['pub_rec_bankruptcies']=list(map(lambda x:x if x in z else 1,kisan2['pub_r
kisan2['pub_rec_bankruptcies'].value_counts()
```

Out[73]:

```
0    77846
1    10895
Name: pub_rec_bankruptcies, dtype: int64
```

In [86]:

```
kisan3=kisan2.dropna()
kisan3.shape
```

Out[86]:

```
(83135, 32)
```

In [87]:
```
kisan3.isnull().sum()
```

Out[87]:
```
loan_amnt                     0
term                          0
int_rate                      0
installment                   0
grade                         0
sub_grade                     0
emp_title                     0
emp_length                    0
annual_inc                    0
verification_status           0
issue_d                       0
loan_status                   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
is_36mnths                    0
is_verified                   0
mod_grade                     0
home_ownership_OTHERS         0
home_ownership_RENT           0
purpose_debt_consolidation    0
purpose_others                0
dtype: int64
```

Splitting the Kisan3 dataset into train,validation and test sets

In [89]:
```
XS=kisan3[['loan_amnt','int_rate', 'installment','mod_grade','sub_grade',
          'annual_inc','is_verified','dti','open_acc', 'pub_rec',
           'revol_bal', 'revol_util', 'total_acc','initial_list_status',
          'mort_acc','pub_rec_bankruptcies','is_36mnths',
          'home_ownership_OTHERS', 'home_ownership_RENT',
          'purpose_debt_consolidation', 'purpose_others']]
```

In [91]:
```
XS.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83135 entries, 0 to 200488
Data columns (total 21 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   loan_amnt                    83135 non-null  float64
 1   int_rate                     83135 non-null  float64
 2   installment                  83135 non-null  float64
 3   mod_grade                    83135 non-null  int64  
 4   sub_grade                    83135 non-null  object 
 5   annual_inc                   83135 non-null  float64
 6   is_verified                  83135 non-null  int64  
 7   dti                          83135 non-null  float64
 8   open_acc                     83135 non-null  int64  
 9   pub_rec                      83135 non-null  int64  
 10  revol_bal                    83135 non-null  float64
 11  revol_util                   83135 non-null  float64
 12  total_acc                    83135 non-null  float64
 13  initial_list_status          83135 non-null  int64  
 14  mort_acc                     83135 non-null  int64  
 15  pub_rec_bankruptcies         83135 non-null  int64  
 16  is_36mnths                   83135 non-null  int64  
 17  home_ownership_OTHERS        83135 non-null  int64  
 18  home_ownership_RENT          83135 non-null  int64  
 19  purpose_debt_consolidation   83135 non-null  int64  
 20  purpose_others               83135 non-null  int64  
dtypes: float64(8), int64(12), object(1)
memory usage: 14.0+ MB
```

In [92]:
```
y=kisan3['loan_status']
```

In [96]:

```
kisan3['y_obs']=list(map(lambda x:0 if x=='Fully Paid' else 1,kisan3['loan_status
kisan3[['loan_status','y_obs']].head(14)
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

Out[96]:

	loan_status	y_obs
0	Fully Paid	0
2	Fully Paid	0
4	Fully Paid	0
6	Fully Paid	0
8	Charged Off	1
10	Fully Paid	0
12	Fully Paid	0
14	Fully Paid	0
16	Fully Paid	0
18	Fully Paid	0
20	Fully Paid	0
22	Fully Paid	0
24	Fully Paid	0
26	Fully Paid	0

In [97]:

```
y=kisan3['y_obs']
```

In [98]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(XS,y,test_size=0.15,random_state=2
```

In [99]:

```
X_trainn,X_dev,y_trainn,y_dev=train_test_split(X_train,y_train,test_size=0.15,ran
```

In [100]:

```
print(X_trainn.shape)
print(X_dev.shape)
print(X_test.shape)
print(y_trainn.shape)
print(y_dev.shape)
print(y_test.shape)
```

```
(60064, 21)
(10600, 21)
(12471, 21)
(60064,)
(10600,)
(12471,)
```

In [101]:

```
X_trainn=pd.concat([X_trainn,y_trainn],axis=1)
```

In [103]:

```
z=X_trainn.groupby('sub_grade')['y_obs'].mean().to_dict()
X_trainn['sub_grade']=list(map(lambda x:z[x],X_trainn['sub_grade']))
X_trainn['sub_grade']
```

Out[103]:

```
14488     0.192450
94254     0.154802
24670     0.341498
20048     0.168598
55286     0.075792
            ...
71550     0.108300
86534     0.347036
117018    0.085448
190226    0.213446
184816    0.125066
Name: sub_grade, Length: 60064, dtype: float64
```

In [104]:

```
X_dev['sub_grade']=list(map(lambda x:z[x],X_dev['sub_grade']))
X_dev['sub_grade']
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
```

Out[104]:

```
39684     0.404795
16006     0.122754
165846    0.154802
151720    0.062593
15474     0.192450
            ...
192714    0.154802
38598     0.122754
132708    0.028833
90502     0.241294
179298    0.322265
Name: sub_grade, Length: 10600, dtype: float64
```

In [105]:

```
X_test['sub_grade']=list(map(lambda x:z[x],X_test['sub_grade']))
X_test['sub_grade']
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  """Entry point for launching an IPython kernel.
```

Out[105]:

```
56938     0.322265
63494     0.154802
120154    0.240930
26788     0.265314
77820     0.108300
            ...
143328    0.125066
177034    0.265314
48176     0.240930
49746     0.101064
110436    0.154802
Name: sub_grade, Length: 12471, dtype: float64
```

In [107]:
```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
d={}
for i in cols:
    d[i]=[X_trainn[i].mean(),X_trainn[i].std()]
print(d)
```

```
{'loan_amnt': [14491.336324587108, 8388.981997556977], 'int_rate': [13.78460392
2482301, 4.490693801881828], 'installment': [444.271407332167, 251.863757655670
16], 'annual_inc': [75394.66977274243, 48671.130821498744], 'dti': [17.68224127
597239, 8.108662503973125], 'open_acc': [11.59971030900373, 5.213968658647141],
'revol_bal': [16033.318560202451, 19607.84551585026], 'revol_util': [54.1856752
7970166, 23.89877657577768], 'total_acc': [25.88725359616409, 11.92963296750360
4], 'mort_acc': [1.8127164358018113, 2.148206675715537], 'mod_grade': [1.852124
400639318, 1.321668894368757]}
```

In [108]:

```python
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
for i in cols:
    mean=X_trainn[i].mean()
    std=X_trainn[i].std()
    X_trainn[i]=(X_trainn[i]-mean)/std
X_trainn.head(20)
```

Out[108]:

	loan_amnt	int_rate	installment	mod_grade	sub_grade	annual_inc	is_verified	d
14488	-0.869156	0.339679	-0.768596	0.111886	0.192450	-0.735439	0	0.55468
94254	0.418247	0.068006	0.681792	-0.644734	0.154802	0.053529	0	-0.05330
24670	0.060635	0.907520	-0.256136	0.868505	0.341498	-0.110839	1	0.43012
20048	-0.296977	-0.176945	-0.158822	0.111886	0.168598	-0.234116	0	0.85436
55286	1.610286	-1.310400	1.714652	-1.401353	0.075792	1.347931	1	-0.07057
138636	0.418247	0.442113	0.740554	0.868505	0.265314	-0.521760	1	0.63114
47806	-1.155246	-1.595434	-1.178778	-1.401353	0.040411	-0.850497	0	-1.59363
151938	-1.369813	-1.726816	-1.401398	-1.401353	0.028833	-0.727221	0	0.99372
153512	0.626854	-0.150223	0.882376	-0.644734	0.125066	-0.460122	1	-0.36901
151486	-1.012201	-1.582073	-1.031833	-1.401353	0.062593	0.505543	0	-0.58483
106528	2.444714	1.381835	1.917023	1.625124	0.391566	0.711003	1	0.50535
167006	-0.589027	0.713786	-0.412252	0.868505	0.298160	-0.932682	1	1.51415
129208	0.418247	0.994367	0.060622	0.868505	0.268987	-0.460122	1	0.56825
162968	-0.907898	1.604517	-0.735641	1.625124	0.391566	-1.035412	1	1.06401
163026	-0.726112	-0.319462	-0.650595	-0.644734	0.125066	0.074075	1	-0.53674
77624	1.610286	1.381835	1.180831	0.868505	0.322265	0.197352	1	0.20074
17694	-0.654589	0.402031	-0.514847	0.111886	0.241294	-0.809405	1	1.41302
20454	0.954665	-0.266018	1.228476	0.111886	0.192450	-0.521760	1	-1.61829
23510	-0.535385	-0.522103	-0.455768	-0.644734	0.125066	0.505543	1	-1.25695
161670	-0.535385	-0.176945	-0.426307	0.111886	0.168598	-0.829951	1	1.53635

20 rows × 22 columns

In [109]:
```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
for i in cols:
    X_dev[i]=(X_dev[i]-d[i][0])/d[i][1]
print(X_dev)
```

```
        loan_amnt  int_rate  installment  mod_grade  sub_grade  annual_inc  \
39684    2.444714  1.929634     2.109746   1.625124   0.404795    7.285743   
16006   -1.310211 -0.733651    -1.312302  -0.644734   0.122754   -0.398484   
165846   0.060635 -0.399627     0.213920  -0.644734   0.154802   -0.213570   
151720   1.610286 -1.372751     1.700318  -1.401353   0.062593    1.841037   
15474   -0.177773 -0.176945    -0.589769   0.111886   0.192450    0.197352   
...           ...       ...          ...        ...        ...         ...   
192714  -0.296977 -0.502061    -0.715353  -0.644734   0.154802   -0.727221   
38598   -0.714191 -0.399627    -0.643131  -0.644734   0.122754   -0.419030   
132708  -0.535385 -1.884921    -0.568249  -1.401353   0.028833   -0.012218   
90502    1.133471  0.667023     0.593013   0.111886   0.241294    0.300082   
179298   1.783132  0.842942     1.178012   0.868505   0.322265   -0.336846   

        is_verified       dti  open_acc  pub_rec  ...  revol_util  total_acc  
\
39684             1 -0.561405  0.268565        0  ...    1.172207   1.434474   
16006             1 -0.758725  0.652150        0  ...   -1.601993  -0.912623   
165846            0  0.622514  0.076773        0  ...   -0.288118   0.512400   
151720            1 -0.996742  1.035735        0  ...   -0.555914   1.182999   
```

In [110]:
```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
for i in cols:
    X_test[i]=(X_test[i]-d[i][0])/d[i][1]
print(X_test)
```

```
        loan_amnt  int_rate  installment  mod_grade  sub_grade  annual_inc  \
56938    1.875515  1.210369     2.648371   0.868505   0.322265   -0.069747   
63494    1.133471 -0.502061     1.379709  -0.644734   0.154802   -0.143230   
120154   0.179839  0.339679    -0.242279   0.111886   0.240930   -0.316300   
26788   -0.392340  0.406484    -0.691729   0.868505   0.265314   -0.710784   
77820    0.060635 -0.588908     0.189819  -0.644734   0.108300    0.094621   
...           ...       ...          ...        ...        ...         ...   
143328  -0.535385 -0.176945    -0.426307  -0.644734   0.125066   -0.521760   
177034  -0.296977  0.406484    -0.615140   0.868505   0.265314   -0.768313   
48176   -0.773793  0.045738    -0.678468   0.111886   0.240930   -0.747767   
49746   -0.535385 -0.844993    -0.482965  -0.644734   0.101064    1.121924   
110436  -0.010888  0.068006     0.192638  -0.644734   0.154802   -0.656337   

        is_verified       dti  open_acc  pub_rec  ...  revol_util  total_acc  
\
56938             1  0.951792 -0.882190        0  ...    1.327027  -0.912623   
63494             1  0.967824  1.227527        0  ...   -1.003636   0.847700   
120154            1 -1.043605 -0.690397        0  ...    0.732854  -0.158199   
26788             1  1.932225  0.268565        0  ...    0.448321  -0.912623   
```

In [111]:

```
X_trainn=X_trainn.drop('y_obs',axis=1)
X_trainn.columns
```

Out[111]:

```
Index(['loan_amnt', 'int_rate', 'installment', 'mod_grade', 'sub_grade',
       'annual_inc', 'is_verified', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status', 'mort_acc',
       'pub_rec_bankruptcies', 'is_36mnths', 'home_ownership_OTHERS',
       'home_ownership_RENT', 'purpose_debt_consolidation', 'purpose_others'],
      dtype='object')
```

In [113]:

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

In [118]:

```
y_trainn.value_counts(normalize=True)
```

Out[118]:

```
0    0.801345
1    0.198655
Name: y_obs, dtype: float64
```

In [120]:

```
from sklearn import metrics
```

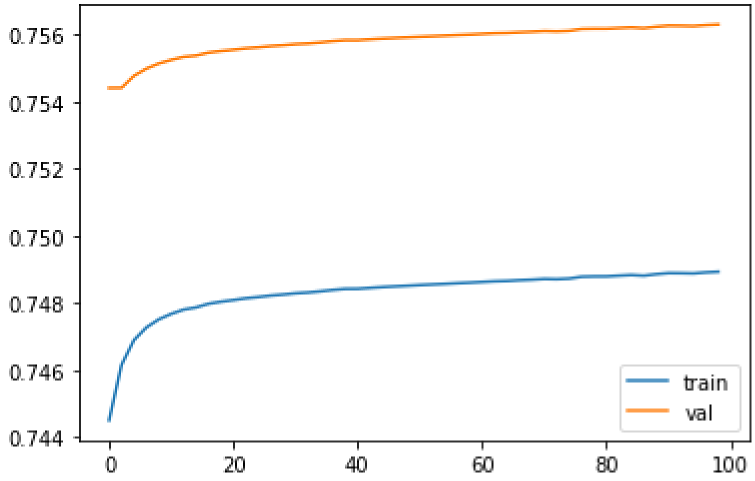
Making a Logistic regression model

In [121]:

```
train_cost=[]
val_cost=[]
lambdaa=[]
for i in np.arange(0.01,100,2):
    lambdaa.append(i)
    model=LogisticRegression(C=1/i,class_weight={0:0.1,1:0.6})
    model.fit(X_trainn,y_trainn)
    y_train_probs=model.predict_proba(X_trainn)
    y_val_probs=model.predict_proba(X_dev)
    train_loss=metrics.log_loss(y_trainn,y_train_probs)
    val_loss=metrics.log_loss(y_dev,y_val_probs)
    train_cost.append(train_loss)
    val_cost.append(val_loss)
plt.plot(lambdaa,train_cost,label='train')
plt.plot(lambdaa,val_cost,label='val')
plt.legend()
plt.show()
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```



In [122]:

```
z=np.argmin(val_cost)
z
```

Out[122]:

```
0
```

In [123]:

```
model=LogisticRegression(C=1,class_weight={0:0.1,1:0.6})
model.fit(X_trainn,y_trainn)
print(model.score(X_trainn,y_trainn))
print(model.score(X_dev,y_dev))
print(model.score(X_test,y_test))
```

```
0.5457844965370272
0.5323584905660378
0.5452650148344158
```

In [124]:

```
from sklearn.metrics import confusion_matrix, precision_score, recall_score
ypred=model.predict(X_test)
confusion_matrix(y_test,ypred)
```

Out[124]:

```
array([[4788, 5176],
       [ 495, 2012]], dtype=int64)
```

In [125]:

```
print(precision_score(y_test,ypred))
print(recall_score(y_test,ypred))
```

```
0.27991096271563715
0.8025528520143598
```

Removing the outliers which are on the wrong side of the plane theta_T*x+theta_not

In [126]:

```
z=np.array(list(model.intercept_)+list(model.coef_[0]))
z=z.T
z
```

Out[126]:

```
array([-0.37986903, -0.09505128, -0.24157436,  0.16026868,  0.29393671,
        3.91784125, -0.07684596,  0.10735067,  0.21085049,  0.10616443,
        0.08805733, -0.07201472,  0.08561362, -0.1274847 , -0.0507649 ,
       -0.07308559, -0.08033209, -0.49011554,  0.10939622,  0.26325411,
        0.02387176,  0.12101527])
```

In [127]:

```
X_trainn.insert(0,'x_not',1)
```

In [128]:
```
X_trainn
```

Out[128]:

	x_not	loan_amnt	int_rate	installment	mod_grade	sub_grade	annual_inc	is_verified
14488	1	-0.869156	0.339679	-0.768596	0.111886	0.192450	-0.735439	0
94254	1	0.418247	0.068006	0.681792	-0.644734	0.154802	0.053529	0
24670	1	0.060635	0.907520	-0.256136	0.868505	0.341498	-0.110839	1
20048	1	-0.296977	-0.176945	-0.158822	0.111886	0.168598	-0.234116	0
55286	1	1.610286	-1.310400	1.714652	-1.401353	0.075792	1.347931	1
...	...	...	...	...	...	...	...	...
71550	1	0.656655	-0.956334	0.779384	-0.644734	0.108300	-0.117003	1
86534	1	0.299043	1.159152	-0.013386	1.625124	0.347036	-0.291645	1
117018	1	0.662615	-1.087717	0.763860	-1.401353	0.085448	-0.624491	1
190226	1	-1.369813	0.266194	-1.351133	0.111886	0.213446	0.197352	0
184816	1	-1.143325	-0.622310	-1.127083	-0.644734	0.125066	-0.172477	1

60064 rows × 22 columns

In [129]:
```
z2=X_trainn.dot(z)
z3=1/(1+np.exp(-z2))
z3
```

Out[129]:
```
14488     0.555917
94254     0.504452
24670     0.769651
20048     0.539214
55286     0.339882
            ...
71550     0.491943
86534     0.854377
117018    0.431970
190226    0.471579
184816    0.391628
Length: 60064, dtype: float64
```

In [130]:
```
X_trainn['log_loss']=-1*(y_trainn*np.log(z3)+(1-y_trainn)*(1-np.log(z3)))
```

In [131]:

```
X_trainn['log_loss'].describe()
```

Out[131]:

```
count    60064.000000
mean        -1.268352
std          0.921445
min         -5.582546
25%         -1.851313
50%         -1.563946
75%         -1.232072
max          2.381357
Name: log_loss, dtype: float64
```

In [132]:

```
X_trainn=pd.concat([X_trainn,y_trainn],axis=1)
```

In [133]:

```
q05=X_trainn['log_loss'].quantile(0.05)
q95=X_trainn['log_loss'].quantile(0.95)
X_trainn=X_trainn[(X_trainn['log_loss']>q05)&(X_trainn['log_loss']<q95)]
```

In [134]:

```
y_trainn=X_trainn['y_obs']
y_trainn
```

Out[134]:

```
14488     0
94254     0
24670     0
20048     0
55286     0
         ..
71550     0
86534     1
117018    0
190226    0
184816    0
Name: y_obs, Length: 54056, dtype: int64
```

In [136]:

```
X_trainn=X_trainn.drop(['x_not','y_obs','log_loss'],axis=1)
```

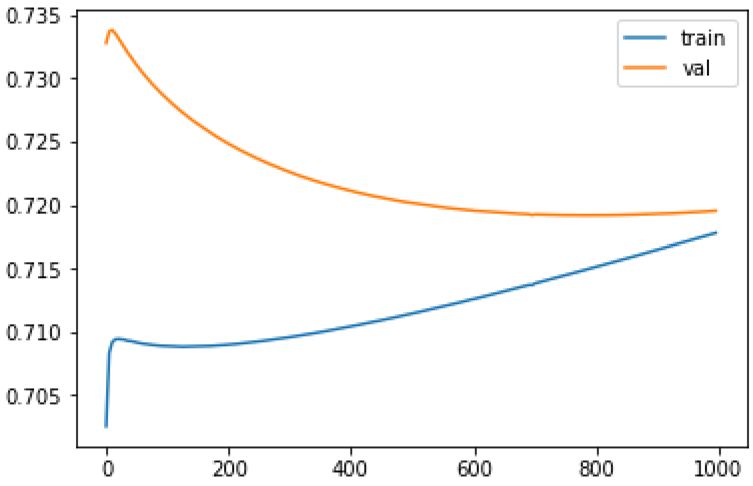
Performing LogisticRegression after removing outliers

In [173]:

```
train_cost=[]
val_cost=[]
lambdaa=[]
for i in np.arange(0.1,1000,5):
    lambdaa.append(i)
    model=LogisticRegression(C=1/i,class_weight={0:0.1,1:0.9})
    model.fit(X_trainn,y_trainn)
    y_train_probs=model.predict_proba(X_trainn)
    y_val_probs=model.predict_proba(X_dev)
    train_loss=metrics.log_loss(y_trainn,y_train_probs)
    val_loss=metrics.log_loss(y_dev,y_val_probs)
    train_cost.append(train_loss)
    val_cost.append(val_loss)
plt.plot(lambdaa,train_cost,label='train')
plt.plot(lambdaa,val_cost,label='val')
plt.legend()
plt.show()
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.
py:940: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-
learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regressi
on)
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```



In [ ]:

```

```

In [174]:

```
z=np.argmin(val_cost)
lbest=0.1+z*5
lbest
```

Out[174]:

```
785.1
```

In [175]:

```
model=LogisticRegression(C=1/lbest,class_weight={0:0.1,1:0.9})
model.fit(X_trainn,y_trainn)
print(model.score(X_trainn,y_trainn))
print(model.score(X_dev,y_dev))
print(model.score(X_test,y_test))
```

```
0.6240565339647772
0.6033962264150944
0.6118194210568519
```

In [178]:

```
ypred=model.predict(X_test)
confusion_matrix(y_test,ypred)
```

Out[178]:

```
array([[5850, 4114],
       [ 727, 1780]], dtype=int64)
```

In [177]:

```
print(precision_score(y_test,ypred))
print(recall_score(y_test,ypred))
```

```
0.30200203596878183
0.7100119664938173
```

Introducing polynomial terms in the logistic regression model

In [180]:

```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
t=0
for i in cols:
    z='f'+str(t)
    XS[z]=XS[i]**2
    t+=1
print(XS)
```

```
        loan_amnt  int_rate  installment  mod_grade sub_grade  annual_inc  \
0         10000.0     11.44       329.48          1        B4    117000.0   
2          8000.0     11.99       265.68          1        B5     65000.0   
4         15600.0     10.49       506.97          1        B3     43057.0   
6          7200.0      6.49       220.65          0        A2     54000.0   
8         24375.0     17.27       609.33          2        C5     55000.0   
...           ...       ...          ...        ...       ...         ...   
200476    30225.0     17.27       755.57          2        C5     68000.0   
200478     6000.0      5.32       180.69          0        A1     48000.0   
200480    15450.0      7.90       483.44          0        A4     42000.0   
200486     5000.0     13.53       169.75          1        B5     43000.0   
200488    15000.0     14.30       514.86          2        C1     44000.0   

        is_verified    dti  open_acc  pub_rec  ...        f1            f2  \
0                 0  26.24        16        0  ...  130.8736  108557.0704   
2                 0  22.05        17        0  ...  143.7601   70585.8624   
4                 1  12.79        13        0  ...  110.0401  257018.5809   
6                 0   2.60         6        0  ...   42.1201   48686.4225   
8                 1  33.95        13        0  ...  298.2529  371283.0489   
...             ...    ...       ...      ...  ...       ...          ...   
200476            1  24.55         7        0  ...  298.2529  570886.0249   
200478            1   3.78         7        0  ...   28.3024   32648.8761   
200480            0  28.09        13        0  ...   62.4100  233714.2336   
200486            1  18.39         6        0  ...  183.0609   28815.0625   
200488            1  26.02        10        1  ...  204.4900  265080.8196   

                  f3         f4   f5            f6       f7      f8  f9  f10  
0       1.368900e+10   688.5376  256  1.322704e+09  1747.24   625.0   0    1  
2       4.225000e+09   486.2025  289  4.052572e+08  2840.89   729.0   9    1  
4       1.853905e+09   163.5841  169  1.436882e+08  8500.84   676.0   0    1  
6       2.916000e+09     6.7600   36  2.994278e+07   462.25   169.0   0    0  
8       3.025000e+09  1152.6025  169  6.043731e+08  4872.04  1849.0   1    4  
...              ...        ...  ...           ...      ...     ...  ..  ...  
200476  4.624000e+09   602.7025   49  1.494905e+09  7072.81   289.0  16    4  
200478  2.304000e+09    14.2884   49  5.346534e+07  1705.69   841.0   0    0  
200480  1.764000e+09   789.0481  169  2.196028e+08  3203.56   729.0   0    0  
200486  1.849000e+09   338.1921   36  5.290000e+06  1747.24   289.0  16    1  
200488  1.936000e+09   677.0404  100  1.937107e+08  3757.69   529.0   0    4  

[83135 rows x 32 columns]
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:7: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  import sys
```

In [183]:

```
XS.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83135 entries, 0 to 200488
Data columns (total 32 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   loan_amnt                    83135 non-null  float64
 1   int_rate                     83135 non-null  float64
 2   installment                  83135 non-null  float64
 3   mod_grade                    83135 non-null  int64  
 4   sub_grade                    83135 non-null  object 
 5   annual_inc                   83135 non-null  float64
 6   is_verified                  83135 non-null  int64  
 7   dti                          83135 non-null  float64
 8   open_acc                     83135 non-null  int64  
 9   pub_rec                      83135 non-null  int64  
 10  revol_bal                    83135 non-null  float64
 11  revol_util                   83135 non-null  float64
 12  total_acc                    83135 non-null  float64
 13  initial_list_status          83135 non-null  int64  
 14  mort_acc                     83135 non-null  int64  
 15  pub_rec_bankruptcies         83135 non-null  int64  
 16  is_36mnths                   83135 non-null  int64  
 17  home_ownership_OTHERS        83135 non-null  int64  
 18  home_ownership_RENT          83135 non-null  int64  
 19  purpose_debt_consolidation   83135 non-null  int64  
 20  purpose_others               83135 non-null  int64  
 21  f0                           83135 non-null  float64
 22  f1                           83135 non-null  float64
 23  f2                           83135 non-null  float64
 24  f3                           83135 non-null  float64
 25  f4                           83135 non-null  float64
 26  f5                           83135 non-null  int64  
 27  f6                           83135 non-null  float64
 28  f7                           83135 non-null  float64
 29  f8                           83135 non-null  float64
 30  f9                           83135 non-null  int64  
 31  f10                          83135 non-null  int64  
dtypes: float64(16), int64(15), object(1)
memory usage: 20.9+ MB
```

In [184]:

```
X_train,X_test,y_train,y_test=train_test_split(XS,y,test_size=0.15,random_state=
```

In [185]:

```
X_trainn,X_dev,y_trainn,y_dev=train_test_split(X_train,y_train,test_size=0.15,ran
```

In [186]:

```
print(X_trainn.shape)
print(X_dev.shape)
print(X_test.shape)
print(y_trainn.shape)
print(y_dev.shape)
print(y_test.shape)
```

```
(60064, 32)
(10600, 32)
(12471, 32)
(60064,)
(10600,)
(12471,)
```

In [187]:

```
X_trainn=pd.concat([X_trainn,y_trainn],axis=1)
```

In [188]:

```
z=X_trainn.groupby('sub_grade')['y_obs'].mean().to_dict()
X_trainn['sub_grade']=list(map(lambda x:z[x],X_trainn['sub_grade']))
X_trainn['sub_grade']
```

Out[188]:

```
14488     0.192450
94254     0.154802
24670     0.341498
20048     0.168598
55286     0.075792
            ...
71550     0.108300
86534     0.347036
117018    0.085448
190226    0.213446
184816    0.125066
Name: sub_grade, Length: 60064, dtype: float64
```

In [189]:

```
X_dev['sub_grade']=list(map(lambda x:z[x],X_dev['sub_grade']))
X_dev['sub_grade']
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

Out[189]:

```
39684     0.404795
16006     0.122754
165846    0.154802
151720    0.062593
15474     0.192450
            ...
192714    0.154802
38598     0.122754
132708    0.028833
90502     0.241294
179298    0.322265
Name: sub_grade, Length: 10600, dtype: float64
```

In [190]:

```
X_test['sub_grade']=list(map(lambda x:z[x],X_test['sub_grade']))
X_test['sub_grade']
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  """Entry point for launching an IPython kernel.
```

Out[190]:

```
56938     0.322265
63494     0.154802
120154    0.240930
26788     0.265314
77820     0.108300
            ...
143328    0.125066
177034    0.265314
48176     0.240930
49746     0.101064
110436    0.154802
Name: sub_grade, Length: 12471, dtype: float64
```

In [191]:
```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
d={}
for i in cols:
    d[i]=[X_trainn[i].mean(),X_trainn[i].std()]
print(d)
```

```
{'loan_amnt': [14491.336324587108, 8388.981997556977], 'int_rate': [13.78460392
2482301, 4.490693801881828], 'installment': [444.271407332167, 251.863757655670
16], 'annual_inc': [75394.66977274243, 48671.130821498744], 'dti': [17.68224127
597239, 8.108662503973125], 'open_acc': [11.59971030900373, 5.213968658647141],
'revol_bal': [16033.318560202451, 19607.84551585026], 'revol_util': [54.1856752
7970166, 23.89877657577768], 'total_acc': [25.88725359616409, 11.92963296750360
4], 'mort_acc': [1.8127164358018113, 2.148206675715537], 'mod_grade': [1.852124
400639318, 1.321668894368757]}
```

In [192]:

```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
for i in cols:
    mean=X_trainn[i].mean()
    std=X_trainn[i].std()
    X_trainn[i]=(X_trainn[i]-mean)/std
X_trainn.head(20)
```

Out[192]:

	loan_amnt	int_rate	installment	mod_grade	sub_grade	annual_inc	is_verified	d
14488	-0.869156	0.339679	-0.768596	0.111886	0.192450	-0.735439	0	0.55468
94254	0.418247	0.068006	0.681792	-0.644734	0.154802	0.053529	0	-0.05330
24670	0.060635	0.907520	-0.256136	0.868505	0.341498	-0.110839	1	0.43012
20048	-0.296977	-0.176945	-0.158822	0.111886	0.168598	-0.234116	0	0.85436
55286	1.610286	-1.310400	1.714652	-1.401353	0.075792	1.347931	1	-0.07057
138636	0.418247	0.442113	0.740554	0.868505	0.265314	-0.521760	1	0.63114
47806	-1.155246	-1.595434	-1.178778	-1.401353	0.040411	-0.850497	0	-1.59363
151938	-1.369813	-1.726816	-1.401398	-1.401353	0.028833	-0.727221	0	0.99372
153512	0.626854	-0.150223	0.882376	-0.644734	0.125066	-0.460122	1	-0.36901
151486	-1.012201	-1.582073	-1.031833	-1.401353	0.062593	0.505543	0	-0.58483
106528	2.444714	1.381835	1.917023	1.625124	0.391566	0.711003	1	0.50535
167006	-0.589027	0.713786	-0.412252	0.868505	0.298160	-0.932682	1	1.51415
129208	0.418247	0.994367	0.060622	0.868505	0.268987	-0.460122	1	0.56825
162968	-0.907898	1.604517	-0.735641	1.625124	0.391566	-1.035412	1	1.06401
163026	-0.726112	-0.319462	-0.650595	-0.644734	0.125066	0.074075	1	-0.53674
77624	1.610286	1.381835	1.180831	0.868505	0.322265	0.197352	1	0.20074
17694	-0.654589	0.402031	-0.514847	0.111886	0.241294	-0.809405	1	1.41302
20454	0.954665	-0.266018	1.228476	0.111886	0.192450	-0.521760	1	-1.61829
23510	-0.535385	-0.522103	-0.455768	-0.644734	0.125066	0.505543	1	-1.25695
161670	-0.535385	-0.176945	-0.426307	0.111886	0.168598	-0.829951	1	1.53635

20 rows × 33 columns

In [193]:

```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
for i in cols:
    X_dev[i]=(X_dev[i]-d[i][0])/d[i][1]
print(X_dev)
```

```
        loan_amnt  int_rate  installment  mod_grade  sub_grade  annual_inc  \
39684    2.444714  1.929634     2.109746   1.625124   0.404795    7.285743   
16006   -1.310211 -0.733651    -1.312302  -0.644734   0.122754   -0.398484   
165846   0.060635 -0.399627     0.213920  -0.644734   0.154802   -0.213570   
151720   1.610286 -1.372751     1.700318  -1.401353   0.062593    1.841037   
15474   -0.177773 -0.176945    -0.589769   0.111886   0.192450    0.197352   
...           ...       ...          ...        ...        ...         ...   
192714  -0.296977 -0.502061    -0.715353  -0.644734   0.154802   -0.727221   
38598   -0.714191 -0.399627    -0.643131  -0.644734   0.122754   -0.419030   
132708  -0.535385 -1.884921    -0.568249  -1.401353   0.028833   -0.012218   
90502    1.133471  0.667023     0.593013   0.111886   0.241294    0.300082   
179298   1.783132  0.842942     1.178012   0.868505   0.322265   -0.336846   

        is_verified       dti  open_acc  pub_rec  ...        f1           f2  \
39684             1 -0.561405  0.268565        0  ...  504.0025  951873.4096   
16006             1 -0.758725  0.652150        0  ...  110.0401   12939.0625   
165846            0  0.622514  0.076773        0  ...  143.7601  248153.4225   
151720            1 -0.996742  1.035735        0  ...   58.0644  761291.1504   
15474             0 -0.737759 -1.073982        0  ...  168.7401   87456.2329   
...             ...       ...       ...      ...  ...       ...          ...   
192714            0  0.390664 -0.498605        0  ...  132.9409   69748.8100   
38598             0 -0.192663  1.035735        0  ...  143.7601   79687.6441   
132708            1 -1.187895 -0.690397        1  ...   28.3024   90691.3225   
90502             1  1.224340  0.076773        0  ...  281.5684  352396.5769   
179298            1  0.521388 -0.498605        0  ...  308.7049  549036.5409   

                  f3        f4   f5            f6       f7      f8  f9  f10  
39684   1.849000e+11  172.3969  169  2.655959e+09  6756.84  1849.0   9   16  
16006   3.136000e+09  132.9409  225  1.227101e+07   252.81   225.0   0    1  
165846  4.225000e+09  516.6529  144  2.757260e+08  2237.29  1024.0   0    1  
151720  2.722500e+10   92.1600  289  4.789970e+08  1672.81  1600.0   1    0  
15474   7.225000e+09  136.8900   36  1.100191e+08  5227.29   324.0   1    4  
...              ...       ...  ...           ...      ...     ...  ..  ...  
192714  1.600000e+09  434.7225   81  1.029961e+09  4747.21  1296.0   9    1  
38598   3.025000e+09  259.8544  289  9.329628e+07  1962.49  1024.0   1    1  
132708  5.595040e+09   64.8025   64  1.651610e+07  2520.04   144.0   0    0  
90502   8.100000e+09  762.3121  144  7.464917e+08  4692.25   900.0   4    4  
179298  3.481000e+09  480.0481   81  3.013002e+08  3660.25   289.0   0    9  

[10600 rows x 32 columns]


C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:5: Settin
gWithCopyWarning: 
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
    """
```

In [194]:

```
cols=['loan_amnt','int_rate','installment','annual_inc','dti',
      'open_acc','revol_bal','revol_util','total_acc','mort_acc',
     'mod_grade']
for i in cols:
    X_test[i]=(X_test[i]-d[i][0])/d[i][1]
print(X_test)
```

```
        loan_amnt  int_rate  installment  mod_grade  sub_grade  annual_inc  \
56938    1.875515  1.210369     2.648371   0.868505   0.322265   -0.069747   
63494    1.133471 -0.502061     1.379709  -0.644734   0.154802   -0.143230   
120154   0.179839  0.339679    -0.242279   0.111886   0.240930   -0.316300   
26788   -0.392340  0.406484    -0.691729   0.868505   0.265314   -0.710784   
77820    0.060635 -0.588908     0.189819  -0.644734   0.108300    0.094621   
...           ...       ...          ...        ...        ...         ...   
143328  -0.535385 -0.176945    -0.426307  -0.644734   0.125066   -0.521760   
177034  -0.296977  0.406484    -0.615140   0.868505   0.265314   -0.768313   
48176   -0.773793  0.045738    -0.678468   0.111886   0.240930   -0.747767   
49746   -0.535385 -0.844993    -0.482965  -0.644734   0.101064    1.121924   
110436  -0.010888  0.068006     0.192638  -0.644734   0.154802   -0.656337   

        is_verified       dti  open_acc  pub_rec  ...         f1           f2  
\
56938             1  0.951792 -0.882190        0  ...  369.4084  1.234988e+06   
63494             1  0.967824  1.227527        0  ...  132.9409  6.268997e+05   
120154            1 -1.043605 -0.690397        0  ...  234.3961  1.468806e+05   
26788             1  1.932225  0.268565        0  ...  243.6721  7.292700e+04   
77820             1  1.336566 -0.306812        0  ...  124.0996  2.421427e+05   
...             ...       ...       ...      ...  ...       ...           ...   
143328            1  0.267339  0.268565        0  ...  168.7401  1.135016e+05   
177034            1 -0.069338 -0.690397        1  ...  243.6721  8.371764e+04   
48176             1  1.917426  0.843942        0  ...  195.7201  7.474209e+04   
49746             0 -1.324786 -1.073982        0  ...   99.8001  1.040901e+05   
110436            1  2.103646  0.268565        1  ...  198.5281  2.428420e+05   

                  f3         f4   f5            f6       f7      f8  f9  f10  
56938   5.184000e+09   645.1600   49  3.330329e+09  7378.81   225.0   1    9  
63494   4.681778e+09   651.7809  324  1.038565e+08   912.04  1296.0   1    1  
120154  3.600000e+09    85.0084   64  2.815364e+07  5140.89   576.0   4    4  
26788   1.664640e+09  1112.2225  169  2.984256e+08  4212.01   225.0   0    9  
77820   6.400000e+09   813.3904  100  1.514377e+09  7921.00   676.0  36    1  
...              ...        ...  ...           ...      ...     ...  ..  ...  
143328  2.500000e+09   394.0225  169  3.053668e+07  6225.21   841.0   0    1  
177034  1.444000e+09   293.0944   64  9.672100e+06  2034.01   841.0   4    9  
48176   1.521000e+09  1104.2329  256  5.406661e+07  2016.01   400.0   0    4  
49746   1.690000e+10    48.1636   36  1.314233e+08  3080.25   529.0   0    1  
110436  1.887902e+09  1206.8676  169  3.661460e+07  7259.04  2116.0  81    1  

[12471 rows x 32 columns]
```

```
C:\Users\Atul kumar\anaconda3\lib\site-packages\ipykernel_launcher.py:5: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
```

```
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  """
```

In [195]:

```
X_trainn=X_trainn.drop('y_obs',axis=1)
X_trainn.columns
```

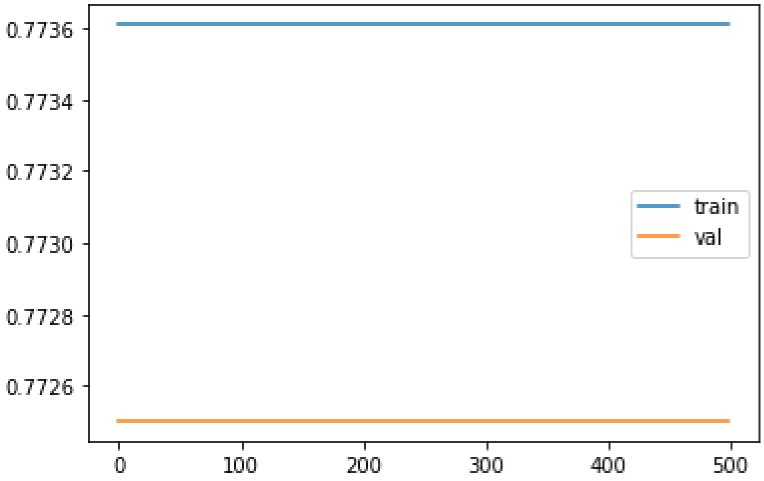
Out[195]:

```
Index(['loan_amnt', 'int_rate', 'installment', 'mod_grade', 'sub_grade',
       'annual_inc', 'is_verified', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status', 'mort_acc',
       'pub_rec_bankruptcies', 'is_36mnths', 'home_ownership_OTHERS',
       'home_ownership_RENT', 'purpose_debt_consolidation', 'purpose_others',
       'f0', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10'],
      dtype='object')
```

Making a LogisticRegression model

In [196]:

```
train_cost=[]
val_cost=[]
lambdaa=[]
for i in np.arange(0.01,500,2):
    lambdaa.append(i)
    model=LogisticRegression(C=1/i,class_weight={0:0.1,1:0.6})
    model.fit(X_trainn,y_trainn)
    y_train_probs=model.predict_proba(X_trainn)
    y_val_probs=model.predict_proba(X_dev)
    train_loss=metrics.log_loss(y_trainn,y_train_probs)
    val_loss=metrics.log_loss(y_dev,y_val_probs)
    train_cost.append(train_loss)
    val_cost.append(val_loss)
plt.plot(lambdaa,train_cost,label='train')
plt.plot(lambdaa,val_cost,label='val')
plt.legend()
plt.show()
```



```
In [197]: z=np.argmin(val_cost)
          z

Out[197]: 0
```

```
In [198]: model=LogisticRegression(C=1,class_weight={0:0.1,1:0.6})
          model.fit(X_trainn,y_trainn)
          print(model.score(X_trainn,y_trainn))
          print(model.score(X_dev,y_dev))
          print(model.score(X_test,y_test))

          0.2590570058604156
          0.25669811320754715
          0.26124609093095985
```

```
In [199]: ypred=model.predict(X_test)
          confusion_matrix(y_test,ypred)

Out[199]: array([[ 889, 9075],
                 [ 138, 2369]], dtype=int64)
```

```
In [200]: print(precision_score(y_test,ypred))
          print(recall_score(y_test,ypred))

          0.20700803914715135
          0.944954128440367
```

Removing the outliers which are on the wrong side of the plane theta_T*x+theta_not

```
In [201]: z=np.array(list(model.intercept_)+list(model.coef_[0]))
          z=z.T
          z

Out[201]: array([ 2.73464847e-18,  9.53859513e-19,  5.42256037e-18,  5.66917095e-19,
                  5.71174031e-18,  1.17718627e-18, -1.77511311e-18,  2.65818229e-18,
                  2.89722012e-18,  3.63015482e-19,  4.71306960e-19, -4.59448671e-19,
                  1.59365616e-18, -7.35330550e-19,  1.11152473e-18, -1.78406958e-18,
                  3.26092317e-19,  4.97488431e-19,  2.77981799e-19,  1.88506989e-18,
                  1.91517594e-18,  5.73489833e-19,  9.90760756e-10,  1.29122727e-15,
                  7.95629367e-13, -4.02482995e-12,  1.95358871e-15,  4.97865851e-16,
                 -1.74604483e-11,  1.33647727e-14,  1.74795612e-15,  5.97045872e-19,
                  4.80592142e-17])
```

```
In [202]: X_trainn.insert(0,'x_not',1)
```

In [203]:

```
z2=X_trainn.dot(z)
z3=1/(1+np.exp(-z2))
z3
```

Out[203]:

```
14488     0.510121
94254     0.570642
24670     0.549522
20048     0.529288
55286     0.660947
            ...
71550     0.592478
86534     0.566896
117018    0.596300
190226    0.492246
184816    0.497507
Length: 60064, dtype: float64
```

In [204]:

```
X_trainn['log_loss']=-1*(y_trainn*np.log(z3)+(1-y_trainn)*(1-np.log(z3)))
```

In [205]:

```
X_trainn['log_loss'].describe()
```

Out[205]:

```
count    60064.000000
mean        -1.165980
std          0.884166
min        -25.048947
25%         -1.665072
50%         -1.598013
75%         -1.367070
max          4.028935
Name: log_loss, dtype: float64
```

In [206]:

```
X_trainn=pd.concat([X_trainn,y_trainn],axis=1)
```

In [207]:

```
q10=X_trainn['log_loss'].quantile(0.10)
q90=X_trainn['log_loss'].quantile(0.90)
X_trainn=X_trainn[(X_trainn['log_loss']>q10)&(X_trainn['log_loss']<q90)]
```

In [208]:

```
X_trainn['log_loss'].describe()
```

Out[208]:

```
count    48050.000000
mean        -1.324091
std          0.687728
min         -1.688692
25%         -1.655102
50%         -1.598013
75%         -1.457196
max          0.607496
Name: log_loss, dtype: float64
```

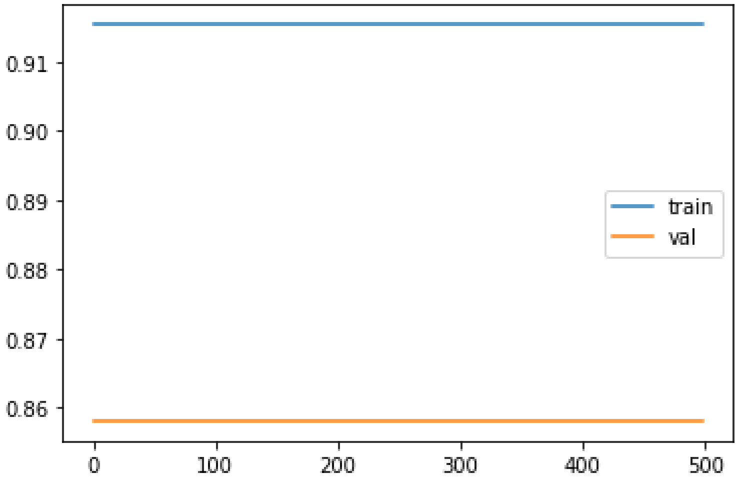
In [209]:
```
y_trainn=X_trainn['y_obs']
y_trainn
```

Out[209]:
```
14488     0
94254     0
24670     0
20048     0
55286     0
         ..
162398    0
2676      0
71550     0
86534     1
117018    0
Name: y_obs, Length: 48050, dtype: int64
```

In [210]:
```
X_trainn=X_trainn.drop(['x_not','y_obs','log_loss'],axis=1)
```

Performing LogisticRegression after removing outliers

In [221]:
```
train_cost=[]
val_cost=[]
lambdaa=[]
for i in np.arange(0.1,500,2):
    lambdaa.append(i)
    model=LogisticRegression(C=1/i,class_weight={0:0.1,1:1.2})
    model.fit(X_trainn,y_trainn)
    y_train_probs=model.predict_proba(X_trainn)
    y_val_probs=model.predict_proba(X_dev)
    train_loss=metrics.log_loss(y_trainn,y_train_probs)
    val_loss=metrics.log_loss(y_dev,y_val_probs)
    train_cost.append(train_loss)
    val_cost.append(val_loss)
plt.plot(lambdaa,train_cost,label='train')
plt.plot(lambdaa,val_cost,label='val')
plt.legend()
plt.show()
```



In [222]:
```
z=np.argmin(val_cost)
lbest=0.1+z*5
lbest
```

Out[222]:
```
240.1
```

In [223]:
```
model=LogisticRegression(C=1/lbest,class_weight={0:0.1,1:1.2})
model.fit(X_trainn,y_trainn)
print(model.score(X_trainn,y_trainn))
print(model.score(X_dev,y_dev))
print(model.score(X_test,y_test))
```

```
0.24276795005202914
0.36933962264150944
0.36612942025499157
```

In [224]:
```
ypred=model.predict(X_test)
confusion_matrix(y_test,ypred)
```

Out[224]:
```
array([[2471, 7493],
       [ 412, 2095]], dtype=int64)
```

In [225]:
```
print(precision_score(y_test,ypred))
print(recall_score(y_test,ypred))
```

```
0.21850229453483522
0.8356601515755884
```

EDA