```
from google.colab import drive
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

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

```
df=pd.read_csv('/content/gdrive/My Drive/logistic_regression.txt')
```

```
df.head()
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pu
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	
		months					ent Advocate	6 years	RENT	54000.0	...	6.0	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	

5 rows × 27 columns


```
df['loan_status'].value_counts()
```

```
Fully Paid     318357
Charged Off     77673
Name: loan_status, dtype: int64
```

```
df.shape
```

```
(396030, 27)
```

```
df.columns
```

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

```
df['mort_acc'].value_counts()
```

```
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
19.0        15
20.0        13
24.0        10
22.0         7
21.0         4
25.0         4
27.0         3
32.0         2
31.0         2
23.0         2
26.0         2
28.0         1
30.0         1
34.0         1
Name: mort_acc, dtype: int64
```

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

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

```
df['emp_title'].value_counts()
```

```
Teacher                    4389
Manager                    4250
Registered Nurse           1856
RN                         1846
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
McCarthy & Holthus, LLC       1
jp flooring                   1
Histology Technologist        1
Gracon Services, Inc          1
Name: emp_title, Length: 173105, dtype: int64
```

```
df['loan_status'].value_counts()
```

```
Fully Paid     318357
Charged Off     77673
Name: loan_status, dtype: int64
```

```
df['purpose'].value_counts()
```

```
debt_consolidation    234507
credit_card            83019
home_improvement       24030
other                  21185
major_purchase          8790
small_business          5701
car                     4697
medical                 4196
moving                  2854
vacation                2452
house                   2201
wedding                 1812
renewable_energy         329
educational              257
Name: purpose, dtype: int64
```

```
## Start
```

Define Problem Statement and perform Exploratory Data Analysis (10 points)

Definition of problem (as per given problem statement with additional views)

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary.

Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables)

Bivariate Analysis (Relationships between important variable)

Illustrate the insights based on EDA

Comments on range of attributes, outliers of various attributes

Comments on the distribution of the variables and relationship between them

Comments for each univariate and bivariate plots

Observations on shape of data, data types of all the attributes, conversion of categorical attributes to 'category' (If required), missing value detection, statistical summary.

```
df.head()
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pu
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	6.0	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
df.shape
```

```
(396030, 27)
```

```
df.dtypes
```

```
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
```

```
df.describe()
```

	loan_amnt	int_rate	installment	annual_inc	dti	open_acc	pub_rec	revol_bal	revol_u
count	396030.000000	396030.000000	396030.000000	3.960300e+05	396030.000000	396030.000000	396030.000000	3.960300e+05	395754.000
mean	14113.888089	13.639400	431.849698	7.420318e+04	17.379514	11.311153	0.178191	1.584454e+04	53.791
std	8357.441341	4.472157	250.727790	6.163762e+04	18.019092	5.137649	0.530671	2.059184e+04	24.452
min	500.000000	5.320000	16.080000	0.000000e+00	0.000000	0.000000	0.000000	0.000000e+00	0.000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 396030 entries, 0 to 396029
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   loan_amnt             396030 non-null  float64
 1   term                  396030 non-null  object
 2   int_rate              396030 non-null  float64
 3   installment           396030 non-null  float64
 4   grade                 396030 non-null  object
 5   sub_grade             396030 non-null  object
 6   emp_title             373103 non-null  object
 7   emp_length            377729 non-null  object
 8   home_ownership        396030 non-null  object
 9   annual_inc            396030 non-null  float64
 10  verification_status   396030 non-null  object
 11  issue_d               396030 non-null  object
 12  loan_status           396030 non-null  object
 13  purpose               396030 non-null  object
 14  title                 394275 non-null  object
 15  dti                   396030 non-null  float64
 18  pub_rec               396030 non-null  float64
 19  revol_bal             396030 non-null  float64
 20  revol_util            395754 non-null  float64
 21  total_acc             396030 non-null  float64
 22  initial_list_status   396030 non-null  object
 23  application_type      396030 non-null  object
 24  mort_acc              358235 non-null  float64
 25  pub_rec_bankruptcies  395495 non-null  float64
 26  address               396030 non-null  object
dtypes: float64(12), object(15)
memory usage: 81.6+ MB
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
## null values check
```

```
df.isna().sum()
```

```
loan_amnt                   0
term                        0
int_rate                    0
installment                 0
grade                       0
sub_grade                   0
emp_title               22927
emp_length              18301
home_ownership              0
annual_inc                  0
verification_status         0
issue_d                     0
loan_status                 0
purpose                     0
title                    1755
dti                         0
earliest_cr_line            0
open_acc                    0
pub_rec                     0
revol_bal                   0
revol_util                276
total_acc                   0
initial_list_status         0
application_type            0
mort_acc                37795
pub_rec_bankruptcies      535
address                     0
dtype: int64
```

```
df.dropna(subset=['emp_title','emp_length','title','revol_util','mort_acc','pub_rec_bankruptcies']).isna().sum()
```

```
loan_amnt               0
term                    0
int_rate                0
installment             0
grade                   0
```

```
sub_grade               0
emp_title               0
emp_length              0
home_ownership          0
annual_inc              0
verification_status     0
issue_d                 0
loan_status             0
purpose                 0
title                   0
dti                     0
earliest_cr_line        0
open_acc                0
pub_rec                 0
revol_bal               0
revol_util              0
total_acc               0
initial_list_status     0
application_type        0
mort_acc                0
pub_rec_bankruptcies    0
address                 0
dtype: int64
```

```
## dropping null values
```

```
df.dropna(subset=['emp_title','emp_length','title','revol_util','mort_acc','pub_rec_bankruptcies'],inplace=True)
```

```
df.isna().sum()
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

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

```
# duplicates check
```

```
df[df.duplicated()]
```

loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_re

0 rows × 27 columns

```
## no duplicated records found
```

```
## Univariate Analysis (distribution plots of all the continuous variable(s) barplots/countplots of all the categorical variables)
```

```
df.dtypes
```

```
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
```

```
df.shape
```

```
(335868, 27)
```

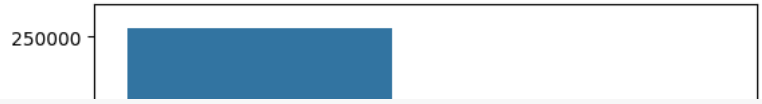
Automatic saving failed. This file was updated remotely or in another tab. Show diff

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pu
0	10000.0	36 months	11.44	329.48	B	B4	Marketing	10+ years	RENT	117000.0	...	16.0	
1	8000.0	36 months	11.99	265.68	B	B5	Credit analyst	4 years	MORTGAGE	65000.0	...	17.0	
2	15600.0	36 months	10.49	506.97	B	B3	Statistician	< 1 year	RENT	43057.0	...	13.0	
3	7200.0	36 months	6.49	220.65	A	A2	Client Advocate	6 years	RENT	54000.0	...	6.0	
4	24375.0	60 months	17.27	609.33	C	C5	Destiny Management Inc.	9 years	MORTGAGE	55000.0	...	13.0	

5 rows × 27 columns

```
# term count
```

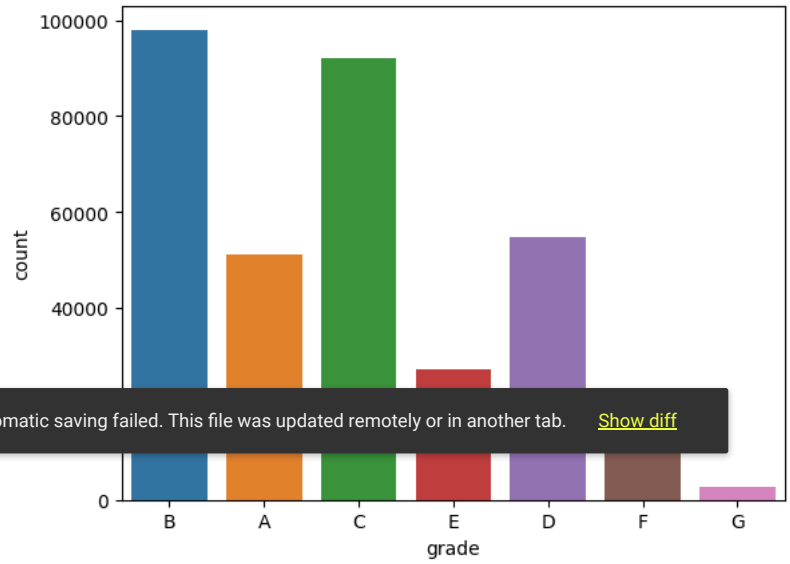
```
sns.countplot(x='term',data=df)
plt.show()
```




```
# Observations: Short term loans more than long term
```

```
## grade count
```

```
sns.countplot(x='grade',data=df)
plt.show()
```

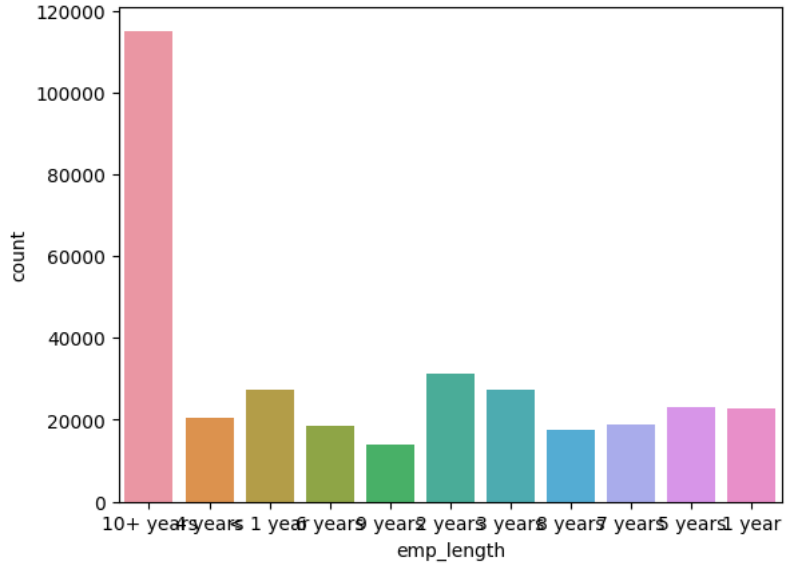


Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
# Observation: A,B,C Grade more
```

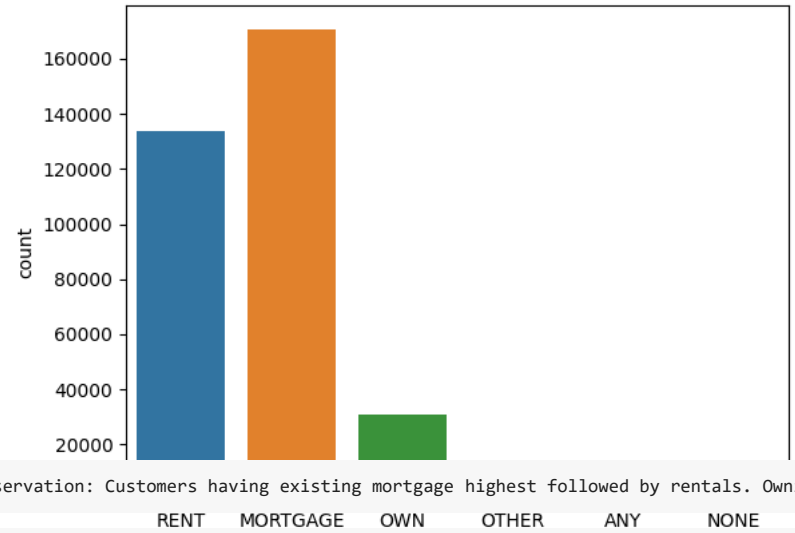
```
# employee title count
```

```
sns.countplot(x='emp_length',data=df)
plt.show()
```




```
# Observation: 10+ years more of employment more
```

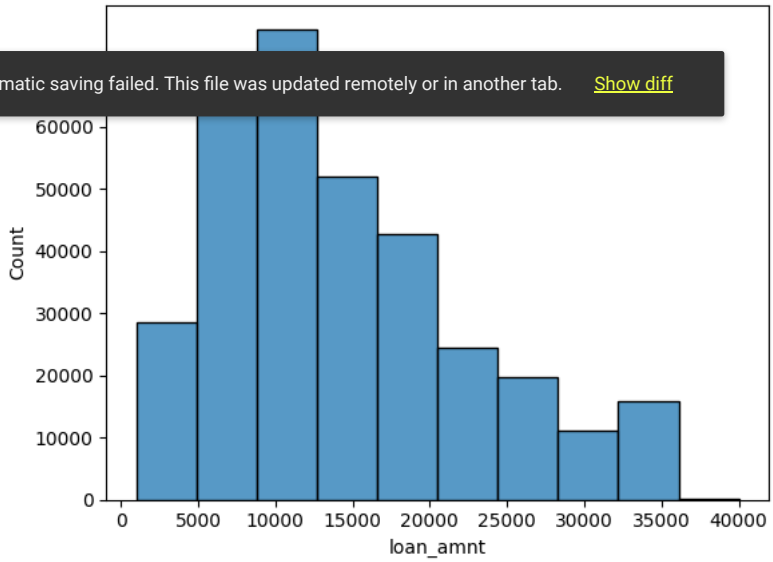
```
sns.countplot(x='home_ownership',data=df)
plt.show()
```



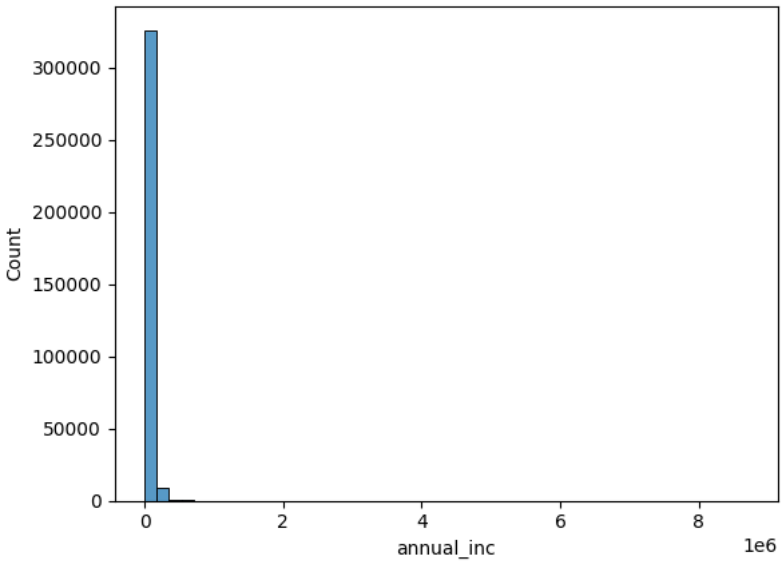

```
# Observation: Customers having existing mortgage highest followed by rentals. Owning home is very less
```

```
# Distribution plots
```

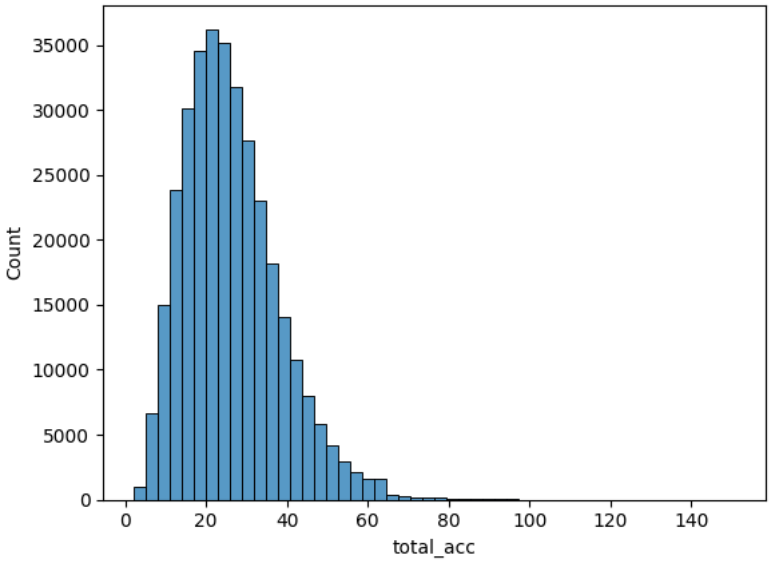
```
sns.histplot(df['loan_amnt'],bins=10)
plt.show()
```




```
sns.histplot(df['annual_inc'],bins=50)
plt.show()
```

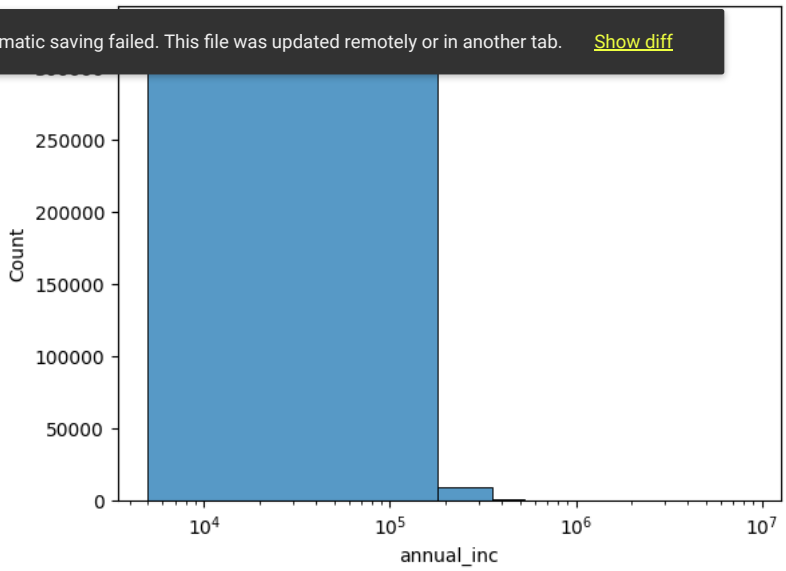



```
sns.histplot(df['total_acc'],bins=50)
plt.show()
```




```
sns.histplot(df['annual_inc'],bins=50)

plt.xscale('log')
plt.show()
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

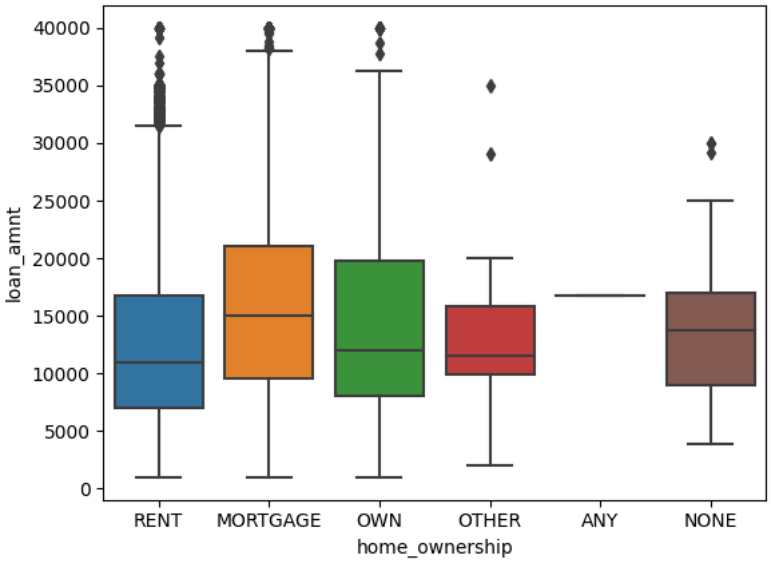



```
## Bivariate Analysis (Relationships between important variable)
```

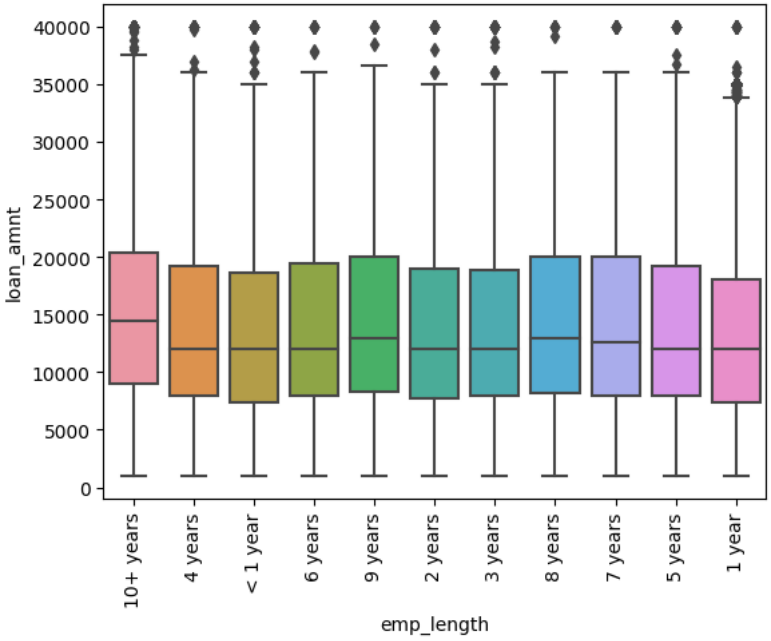
```

```

```
sns.boxplot(x='grade',y='loan_amnt',data=df);
```

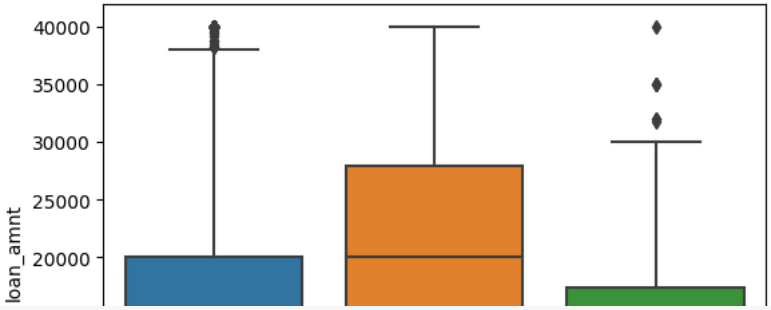

```
sns.boxplot(x='home_ownership',y='loan_amnt',data=df);
```



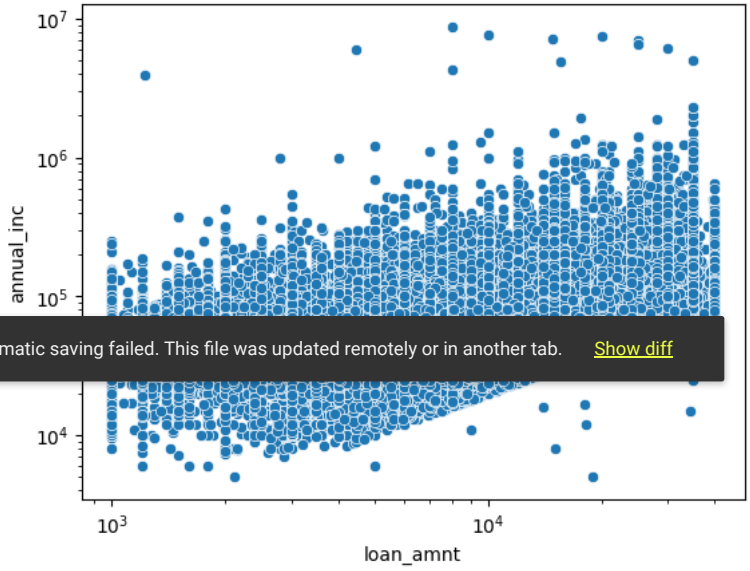
Automatic saving failed. This file was updated remotely or in another tab. Show diff




```
sns.boxplot(x='application_type',y='loan_amnt',data=df)
plt.xticks(rotation = 90)
plt.show()
```

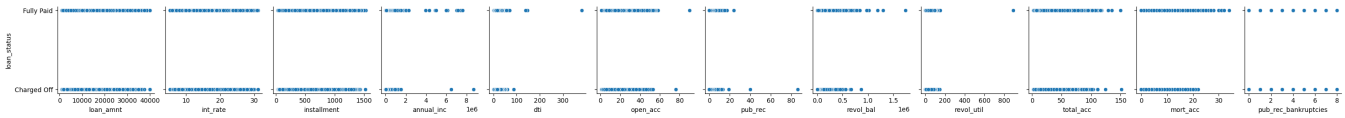



```
sns.scatterplot(x='loan_amnt',y='annual_inc',data=df)
plt.xscale('log')
plt.yscale('log');
```




```
# pairplot
```

```
sns.pairplot(df, y_vars=["loan_status"]);
```



2. Data Preprocessing

Duplicate value check(done above)

Missing value treatment(done above)

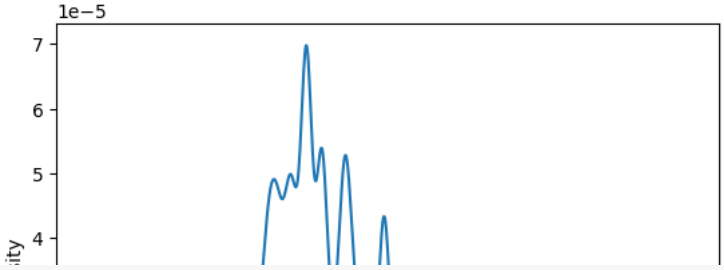
Outlier treatment

Feature engineering

Data preparation for modeling

```
## Outlier treatment
```

```
df["loan_amnt"].plot.density();
```

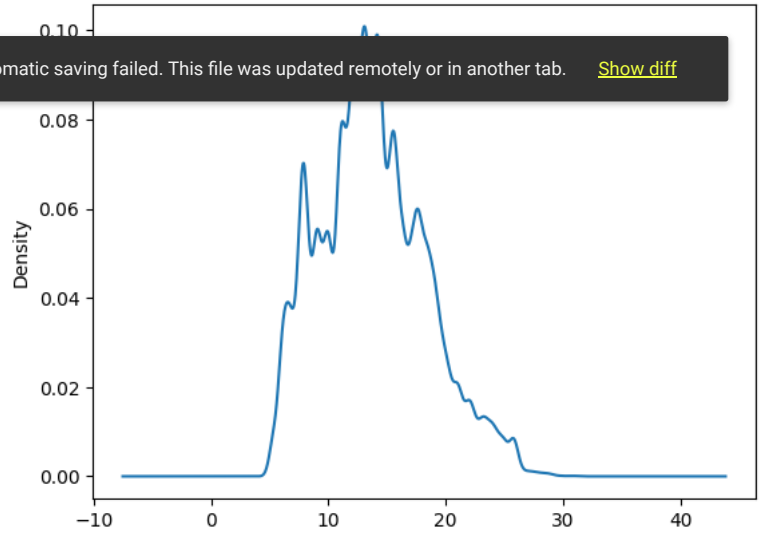



```
df[df["loan_amnt"]>40000].loan_amnt.describe()
```

```
count    0.0
mean     NaN
std      NaN
min      NaN
25%      NaN
50%      NaN
75%      NaN
max      NaN
Name: loan_amnt, dtype: float64
```

```
## nothing significant as outlier here
```

```
df["int_rate"].plot.density();
```



Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
df[df["int_rate"]>33]
```

loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_re

0 rows × 27 columns

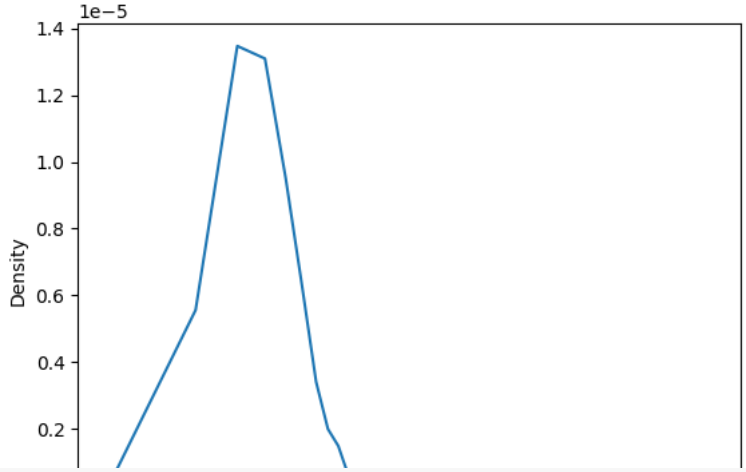
```
df[df["int_rate"]<=0]
```

loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_acc	pub_re

0 rows × 27 columns

```
## I am not touching the very high interest rate(>=30). Wilfull defaulters!!!!!
```

```
df["annual_inc"].plot.density()
plt.xscale('log');
```




```
df[df["annual_inc"]>=1000000]
```

	loan_amnt	term	int_rate	installment	grade	sub_grade	emp_title	emp_length	home_ownership	annual_inc	...	open_ac
15303	4000.0	36 months	10.49	130.00	B	B3	Owner	< 1 year	RENT	1000000.0	...	12.(
16169	30000.0	36 months	12.12	998.15	B	B3	BOONE AND SONS JEWELERS	10+ years	MORTGAGE	6100000.0	...	8.(
							owner	6 years	OWN	1900000.0	...	7.(
22740	28000.0	36 months	12.29	933.89	C	C1	COO-VP Operations	6 years	RENT	1250000.0	...	14.(
28108	35000.0	36 months	12.79	1175.76	C	C1	Managing Director	10+ years	MORTGAGE	1250000.0	...	5.(
...	...	...	...	...	...	...	...	...	...	...	...	..
377460	20000.0	36 months	7.89	625.72	A	A5	Senior Vice President Trading	10+ years	RENT	1100000.0	...	7.(
380456	35000.0	36 months	11.22	1149.51	B	B5	Managing Director	7 years	MORTGAGE	2300000.0	...	8.(
384128	35000.0	36 months	8.49	1104.71	B	B1	Owner	10+ years	OWN	1250000.0	...	4.(
387397	35000.0	36 months	18.99	1282.79	D	D3	Senior Vice President	< 1 year	MORTGAGE	1100000.0	...	15.(
391587	4475.0	36 months	7.89	140.01	A	A5	consultant	10+ years	MORTGAGE	6000000.0	...	4.(

61 rows × 27 columns

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
df[df["annual_inc"]>=1000000]['emp_title'].value_counts()
```

```
Managing Director                       4
Owner                                   2
Manager                                 2
Group Supervisor                        2
Child Nutrition Assistant I             1
Senior Vice President/Partner           1
quality lead                            1
Financial Advisor                       1
sr manager                              1
Sales                                   1
President, Alternative and Late night   1
Director of Engg                        1
CFO                                     1
Fiam Pack corp                          1
Harris C, Siskind, P.A.                 1
Correctional Sgt.                       1
equity analyst                          1
R Markey &Sons Inc                      1
```

```
VISIUM asset management                 1
Interim Director of Case Management     1
UPS                                     1
Murray's Cheese                         1
Business unit controller                1
Case Manager                            1
president                               1
Us postal service                       1
Senior Vice President Trading           1
Senior Vice President                   1
Field Account Manager                   1
Strategist                              1
Vice President                          1
Argus Health Stystems, Inc              1
owner                                   1
COO-VP Operations                       1
Texas A&M university-Kingsville         1
Portfolio Manager                       1
sr. national sales director             1
Highbridge Capital                      1
Chief Culinary Officer                  1
RBC Capital Markets                     1
Registered Nurse                        1
Legal admin asst                        1
operator                                1
BOONE AND SONS JEWELERS                 1
Dispatcher                              1
sales                                   1
Commercial Finance Manager              1
Executive Director                      1
PACAF PMO                               1
CEO                                     1
3rd cook                                1
Sr. Network Engineer                    1
consultant                              1
Name: emp_title, dtype: int64
```

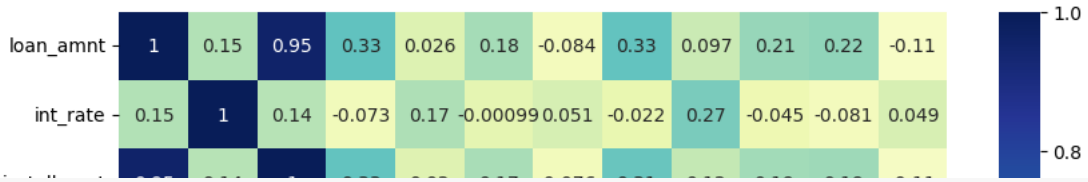
Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
# High profile guys asking for big loans. Cannot be a problem to be considered as outlier
```

```
# correlations
```

```
plt.figure(figsize=(10,8))
ax = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
plt.show()
```

```
<ipython-input-76-368fb77b0ee3>:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  ax = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True)
```




```
## Feature engineering
```

```
def merge(x):
    if x in ['1 years', '2 years', '3 years', '4 years', '5 years']:
        return '1-5 years'
    elif x in ['6 years', '7 years', '8 years', '9 years', '10 years']:
        return '6-10 years'
    elif x=='10+ years':
        return '10+ years'
    else:
        return '< 1 year'
```

```
df['emp_length']=df['emp_length'].apply(merge)
df['loan_status']=df['loan_status'].apply(lambda x:1 if x=='Fully Paid' else 0)
```

```
df['loan_status'].value_counts()
```

```
1    269556
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
df = pd.concat([df, pd.get_dummies(df['term']).iloc[: , 1:]], axis=1)
df = pd.concat([df, pd.get_dummies(df['home_ownership']).iloc[: , [0,2,3]]], axis=1)
df = pd.concat([df, pd.get_dummies(df['emp_length']).iloc[: , 1:]], axis=1)
df = pd.concat([df, pd.get_dummies(df['grade']).iloc[: , 1:]], axis=1)
df['emp_title'] = df.groupby('emp_title')['loan_amnt'].transform('median')
df.drop(columns=['term','grade','sub_grade','emp_length','home_ownership'],inplace=True)
```

```
df.head()
```

	loan_amnt	int_rate	installment	emp_title	annual_inc	verification_status	issue_d	loan_status	purpose	title
0	10000.0	11.44	329.48	12000.0	117000.0	Not Verified	Jan-2015	1	vacation	Vacation
1	8000.0	11.99	265.68	12000.0	65000.0	Not Verified	Jan-2015	1	debt_consolidation	Debt consolidation
2	15600.0	10.49	506.97	12000.0	43057.0	Source Verified	Jan-2015	1	credit_card	Credit card refinancing
3	7200.0	6.49	220.65	7200.0	54000.0	Not Verified	Nov-2014	1	credit_card	Credit card refinancing
4	24375.0	17.27	609.33	24375.0	55000.0	Verified	Apr-2013	0	credit_card	Credit Card Refinance

5 rows × 35 columns

```
df.columns
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'emp_title', 'annual_inc',
       'verification_status', 'issue_d', 'loan_status', 'purpose', 'title',
       'dti', 'earliest_cr_line', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'initial_list_status', 'application_type',
       'mort_acc', 'pub_rec_bankruptcies', 'address', ' 60 months', 'ANY',
       'NONE', 'OTHER', '10+ years', '6-10 years', '< 1 year', 'B', 'C', 'D',
       'E', 'F', 'G'],
      dtype='object')
```

```
df = pd.concat([df, pd.get_dummies(df['verification_status']).iloc[: , 1:]], axis=1)

df['purpose'] = df.groupby('purpose')['loan_amnt'].transform('median')
df = pd.concat([df, pd.get_dummies(df['application_type']).iloc[: , 1:]], axis=1)
df = pd.concat([df, pd.get_dummies(df['initial_list_status']).iloc[: , 1:]], axis=1)
```

```
df.drop(columns=['verification_status', 'issue_d', 'title', 'earliest_cr_line', 'initial_list_status', 'application_type', 'address'],inp
```

```
df.columns
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'emp_title', 'annual_inc',
       'loan_status', 'purpose', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'mort_acc', 'pub_rec_bankruptcies',
       ' 60 months', 'ANY', 'NONE', 'OTHER', '10+ years', '6-10 years',
       '< 1 year', 'B', 'C', 'D', 'E', 'F', 'G', 'Source Verified', 'Verified',
       'INDIVIDUAL', 'JOINT', 'w'],
      dtype='object')
```

```
df['pub_rec']=df['pub_rec'].apply(lambda x:1 if x>1.0 else 0)
df['mort_acc']=df['mort_acc'].apply(lambda x:1 if x>1.0 else 0)
df['pub_rec_bankruptcies']=df['pub_rec_bankruptcies'].apply(lambda x:1 if x>1.0 else 0)
```

```
df['mort_acc'].unique()
```

```
array([0, 1])
```

```
df.head()
```

	loan_amnt	int_rate	installment	emp_title	annual_inc	loan_status	purpose	dti	open_acc	pub_rec	...	C	D	E	F	G	Sou Verif
0	10000.0	11.44	329.48	12000.0	117000.0	1	5000.0	26.24	16.0	0	...	0	0	0	0	0	
1	8000.0	11.99	265.68	12000.0	65000.0	1	14000.0	22.05	17.0	0	...	0	0	0	0	0	
							13000.0	12.79	13.0	0	...	0	0	0	0	0	
							13000.0	2.60	6.0	0	...	0	0	0	0	0	
4	24375.0	17.27	609.33	24375.0	55000.0	0	13000.0	33.95	13.0	0	...	1	0	0	0	0	

Automatic saving failed. This file was updated remotely or in another tab. Show diff

5 rows × 33 columns

```
df.columns
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'emp_title', 'annual_inc',
       'loan_status', 'purpose', 'dti', 'open_acc', 'pub_rec', 'revol_bal',
       'revol_util', 'total_acc', 'mort_acc', 'pub_rec_bankruptcies',
       ' 60 months', 'ANY', 'NONE', 'OTHER', '10+ years', '6-10 years',
       '< 1 year', 'B', 'C', 'D', 'E', 'F', 'G', 'Source Verified', 'Verified',
       'INDIVIDUAL', 'JOINT', 'w'],
      dtype='object')
```

```
df.head()
```

	loan_amnt	int_rate	installment	emp_title	annual_inc	loan_status	purpose	dti	open_acc	pub_rec	...	C	D	E	F	G	Sou Verif
0	10000.0	11.44	329.48	12000.0	117000.0	1	5000.0	26.24	16.0	0	...	0	0	0	0	0	
1	8000.0	11.99	265.68	12000.0	65000.0	1	14000.0	22.05	17.0	0	...	0	0	0	0	0	
2	15600.0	10.49	506.97	12000.0	43057.0	1	13000.0	12.79	13.0	0	...	0	0	0	0	0	
3	7200.0	6.49	220.65	7200.0	54000.0	1	13000.0	2.60	6.0	0	...	0	0	0	0	0	
4	24375.0	17.27	609.33	24375.0	55000.0	0	13000.0	33.95	13.0	0	...	1	0	0	0	0	

5 rows × 33 columns

```
# scaling(Scaling - Using MinMaxScaler or StandardScaler)
```

```
df.shape
```

```
(335868, 33)
```

```
df['loan_status'].shape
```

```
(396030,)
```

```
df1=df.copy()
```

```
df['loan_status'].shape
```

```
(335868,)
```

```
# Lets split the dataset with training and testing set and prepare the inputs and outputs
```

```
df['loan_status']
```

```
0         1
1         1
2         1
3         1
4         0
         ..
396024    1
396025    1
396026    1
396027    1
396028    1
Name: loan_status, Length: 335868, dtype: int64
```

```
from sklearn.model_selection import train_test_split

X = df.drop(columns='loan_status',axis=1)
y = df['loan_status']
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
X.columns
```

```
Index(['loan_amnt', 'int_rate', 'installment', 'emp_title', 'annual_inc',
       'purpose', 'dti', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util',
       'total_acc', 'mort_acc', 'pub_rec_bankruptcies', ' 60 months', 'ANY',
       'NONE', 'OTHER', '10+ years', '6-10 years', '< 1 year', 'B', 'C', 'D',
       'E', 'F', 'G', 'Source Verified', 'Verified', 'INDIVIDUAL', 'JOINT',
       'w'],
      dtype='object')
```

```
y.value_counts()
```

```
1    269556
0     66312
Name: loan_status, dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.20, shuffle=True)
```

```
y_test.value_counts()
```

```
1    53841
0    13333
Name: loan_status, dtype: int64
```

```
# scaling(Scaling - Using MinMaxScaler or StandardScaler)
```

```
#Standardization
from sklearn.preprocessing import StandardScaler
X_train_columns=X_train.columns
X_test_columns=X_test.columns
std=StandardScaler()
X_train_std=std.fit_transform(X_train)
X_test_std=std.fit_transform(X_test)
```

```
X_train_std
```

```
array([[ 0.05763423, -1.35245413,  0.09087636, ...,  0.04026869,
        -0.02989997,  1.12442259],
       [-1.49262731, -1.24378233, -1.51883374, ...,  0.04026869,
        -0.02989997,  1.12442259],
       [-1.01562376, -1.02422094, -1.00842526, ...,  0.04026869,
        -0.02989997, -0.88934535],
       ...,
       [-0.89935415, -0.69820554, -0.86580177, ...,  0.04026869,
        -0.02989997, -0.88934535],
```

```
 [ 1.68242757,  0.29092962,  2.1803164 , ...,  0.04026869,
  -0.02989997, -0.88934535],
 [-0.53862021, -1.36797868, -0.53020541, ...,  0.04026869,
  -0.02989997, -0.88934535]])
```

```
X_train=pd.DataFrame(X_train_std, columns=X_train_columns)
X_test=pd.DataFrame(X_test_std, columns=X_test_columns)
```

```
X_train.head()
```

	loan_amnt	int_rate	installment	emp_title	annual_inc	purpose	dti	open_acc	pub_rec	revol_bal	...	C	
0	0.581639	-0.027405	0.853897	-0.581943	-0.062515	0.506508	-0.502103	0.465397	-0.14852	0.073907	...	-0.615727	-0.4405
1	-0.419800	0.187960	-0.261113	-0.483184	-0.195986	0.506508	-0.140282	-0.308344	-0.14852	-0.067376	...	1.624095	-0.4405
2	2.107642	0.598709	1.376508	0.175209	-0.095883	0.506508	0.904706	0.658832	-0.14852	0.984601	...	-0.615727	2.2700
3	-0.449605	0.263449	-0.288297	-0.524333	-0.763238	-0.026074	-0.437098	-0.114909	-0.14852	-0.120842	...	1.624095	-0.4405
4	0.602503	0.445510	0.114690	0.963224	-0.396193	0.506508	1.757131	2.593186	-0.14852	0.540105	...	1.624095	-0.4405

5 rows × 32 columns

```
X_test.head()
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

	loan_amnt	int_rate	installment	emp_title	annual_inc	purpose	dti	open_acc	pub_rec	revol_bal	...	C	
0	0.652803	1.196337	0.300618	-0.344297	-0.015789	0.508981	-1.880414	0.073545	-0.144612	-0.625284	...	-0.613035	2.263
1	-1.136920	1.200772	-1.040296	-1.473584	-0.663136	-2.657440	0.853934	-0.882691	-0.144612	-0.668632	...	-0.613035	2.263
2	-1.023571	1.814953	-0.867820	0.636617	-0.432968	-2.657440	0.469993	-0.691444	-0.144612	-0.580024	...	-0.613035	-0.441
3	-1.107091	0.187483	-1.051193	-1.432369	-0.231571	0.508981	-0.007472	0.456039	-0.144612	0.147155	...	1.631229	-0.441
4	-0.301716	-1.138439	-0.261152	-1.588985	0.487703	0.508981	1.028677	0.073545	-0.144612	12.447727	...	-0.613035	-0.441

5 rows × 32 columns

```
# 7.Use Logistic Regression Model from Sklearn/Statsmodel library and explain the results
```

```
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
y_test
```

```
233730    1
173017    0
211515    1
71205     1
353872    1
         ..
279636    1
124178    1
3506      1
23509     1
228284    1
Name: loan_status, Length: 67174, dtype: int64
```

```
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
y_pred = log_reg.predict(X_test)
```

Results Evaluation:

ROC AUC Curve & comments

Precision Recall Curve & comments

Classification Report (Confusion Matrix etc)

Tradeoff Questions: How can we make sure that our model can detect real defaulters and there are less false positives? This is important as we can lose out on an opportunity to finance more individuals and earn interest on it. (10 Points) Since NPA (non-performing asset) is a real problem in this industry, it's important we play safe and shouldn't disb

```
# classification reports
```

```
# confusion matrix
```

```
from sklearn import metrics
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[  814, 12380],
       [  657, 53323]])
```

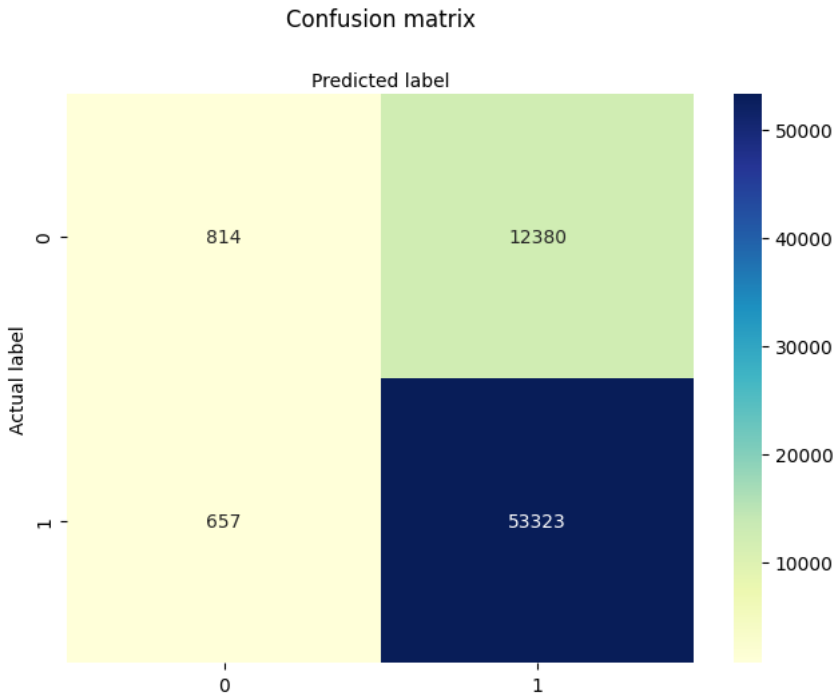
```
#
```

```
# import required modules
```



```
class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')

#Text(0.5,257.44,'Predicted label');
```

```
Text(0.5, 427.9555555555555, 'Predicted label')
```




```
# Accuracy: 80%
```

```
from sklearn.metrics import classification_report
```

```
from sklearn.metrics import classification_report
target_names = ['Fully Paid', 'Charged off']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

   Fully Paid       0.55      0.06      0.11     13194
  Charged off       0.81      0.99      0.89     53980

     accuracy                           0.81     67174
    macro avg       0.68      0.52      0.50     67174
 weighted avg       0.76      0.81      0.74     67174
```

Well, you got a classification rate of 80%, considered as good accuracy.

Precision: Precision is about being precise, i.e., how accurate your model is. In other words, you can say, when a model makes a prediction, how often it is correct. In your prediction case, when your Logistic Regression model predicted who fully padi is 80% of the time.

Recall: If there are ones uso fully paid in the test set and your Logistic Regression model can identify it 99% of the time.

In a credit card fraud detection(or in our case of loantab) system, you would want to have a higher recall score of the predictive models predicting loan payments. A lower recall score would mean a higher false-negative which would mean people not paying back theire loans and hence loss to business.

```
pd.Series(y_pred).value_counts()
```

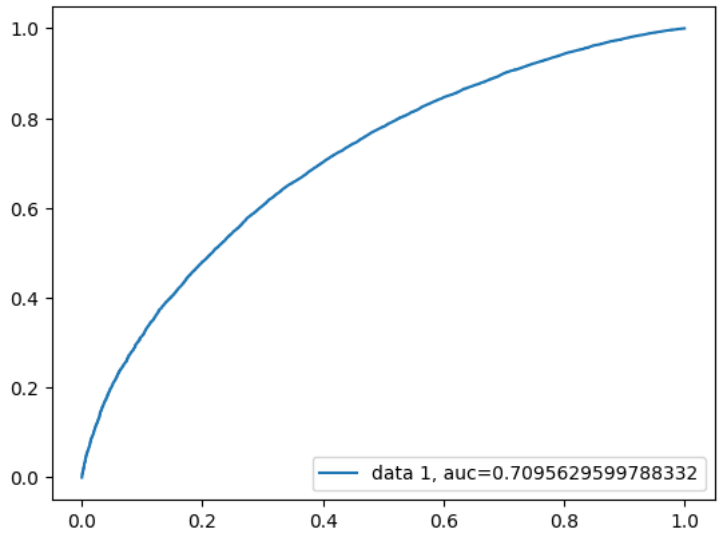
```
1    65785
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

Double-click (or enter) to edit

```
# roc curve
```

```
y_pred_proba = log_reg.predict_proba(X_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```




```
# AUC score for the case is 0.71. AUC score 1 represents a perfect classifier, and 0.5 represents a worthless classifier.
```

✓ 0s completed at 6:30 PM

Automatic saving failed. This file was updated remotely or in another tab. Show diff