In [1]:
```
# importing important libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [2]:
```
## Loading data into the sheet
df=pd.read_csv(r"D:\Projects\ML\loan_approval_dataset.csv")
```

In [3]:
```
# checking first 2 rows of the data
df.head(2)
```

Out[3]:

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_v
0	1	2	Graduate	No	9600000	29900000	12	778	240
1	2	0	Not Graduate	Yes	4100000	12200000	8	417	270

In [4]:
```
# checking how many rows and columns are there in the dataset
print(f"No. of rows: {df.shape[0]}")
print(f"No. of columns: {df.shape[1]}")
# We can see that there are 4269 rows and 13 columns in the dataset
```

```
No. of rows: 4269
No. of columns: 13
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   loan_id                   4269 non-null   int64 
 1    no_of_dependents         4269 non-null   int64 
 2    education                4269 non-null   object
 3    self_employed            4269 non-null   object
 4    income_annum             4269 non-null   int64 
 5    loan_amount              4269 non-null   int64 
 6    loan_term                4269 non-null   int64 
 7    cibil_score              4269 non-null   int64 
 8    residential_assets_value 4269 non-null   int64 
 9    commercial_assets_value  4269 non-null   int64 
 10   luxury_assets_value      4269 non-null   int64 
 11   bank_asset_value         4269 non-null   int64 
 12   loan_status              4269 non-null   object
dtypes: int64(10), object(3)
memory usage: 433.7+ KB
```

It can be observed that thre are total 13 columns and there are 3 columns with object type of datatype, and rest 10 columns are with integer datatype

In [6]:
```
1 df.isnull().sum()
```

Out[6]:
```
loan_id                    0
 no_of_dependents          0
 education                 0
 self_employed             0
 income_annum              0
 loan_amount               0
 loan_term                 0
 cibil_score               0
 residential_assets_value  0
 commercial_assets_value   0
 luxury_assets_value       0
 bank_asset_value          0
 loan_status               0
dtype: int64
```

There are no null values in the dataset

In [7]:
```
1 df.duplicated().sum()
```

Out[7]:
```
0
```

There are no duplicates either.

In [8]:
```
1 # All column names
2 df.columns
```

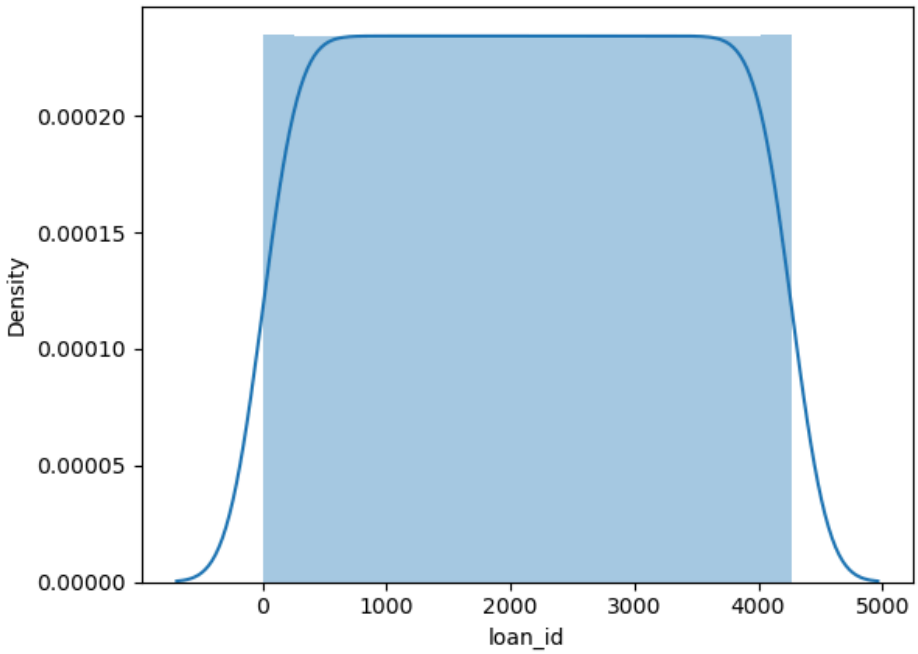
Out[8]:
```
Index(['loan_id', ' no_of_dependents', ' education', ' self_employed',
       ' income_annum', ' loan_amount', ' loan_term', ' cibil_score',
       ' residential_assets_value', ' commercial_assets_value',
       ' luxury_assets_value', ' bank_asset_value', ' loan_status'],
      dtype='object')
```

EDA

Univariate Analysis

In [9]:
```
1 sns.distplot(df["loan_id"])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\3704533142.py:1: UserWarning:

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

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

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

  sns.distplot(df["loan_id"])
```



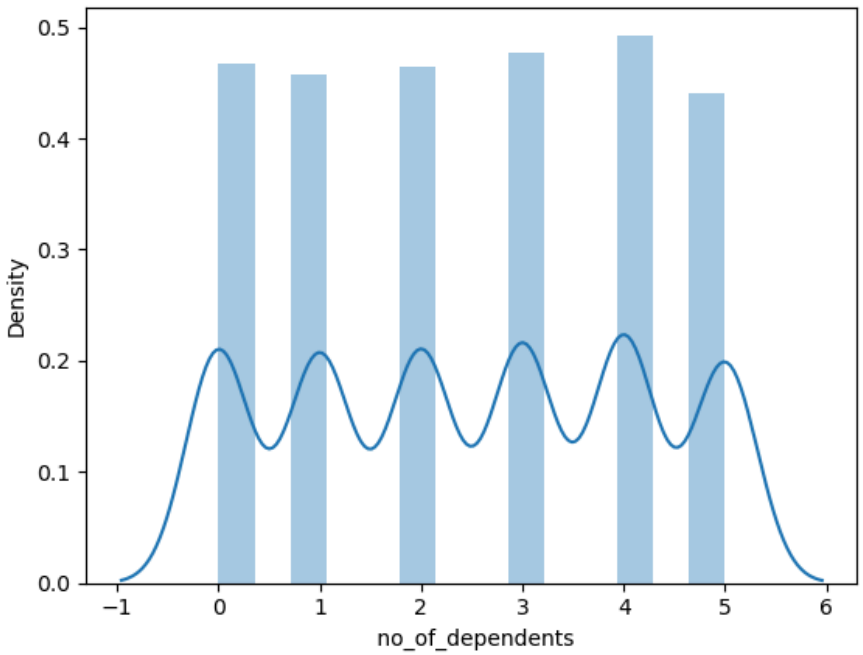
In [10]:
```
1 df["loan_id"].nunique()
```

Out[10]: 4269

In [11]:
```
1 df.drop(columns="loan_id", inplace=True)
```

In [12]:
```
1 sns.distplot(df[" no_of_dependents"])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\3524164572.py:1: UserWarning:

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

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

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

  sns.distplot(df[" no_of_dependents"])
```



In [13]:
```
1 df[' education'].nunique()
```

Out[13]: 2

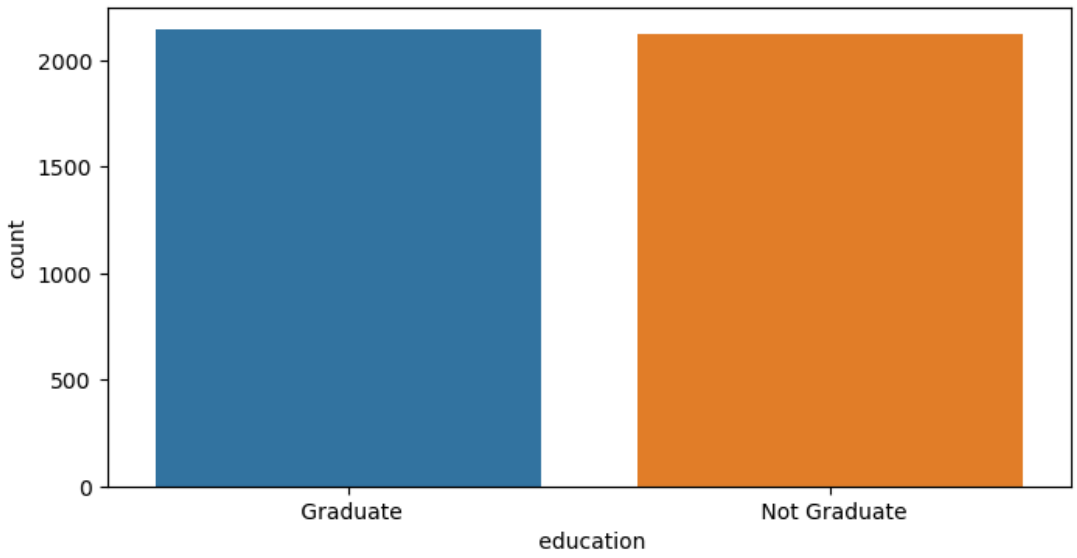
In [14]:
```
1 df[" education"].unique()
```

Out[14]: array([' Graduate', ' Not Graduate'], dtype=object)

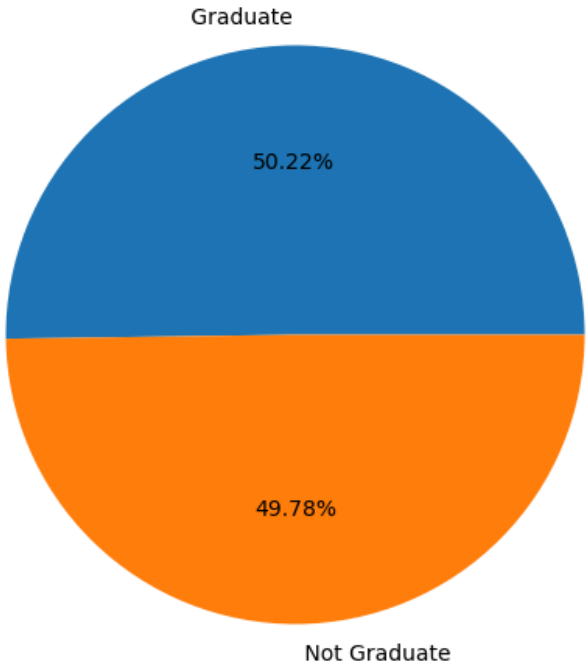
In [15]:
```
1 df[" education"].value_counts(dropna=False)
```

Out[15]:
```
 Graduate        2144
 Not Graduate    2125
Name:  education, dtype: int64
```

In [16]:
```
plt.figure(figsize=(8,4))
sns.countplot(x=" education",data=df)
plt.show()
```



In [17]:
```
plt.figure(figsize=(6,8))
plt.pie(df[" education"].value_counts().values,labels=df[" education"].value_counts().index,autopct=
plt.show()
```



- There are 50.22% who all are graduate.
- There are 49.78% who all are not graduate.

In [18]:
```
df[' self_employed'].nunique()
```

Out[18]: 2

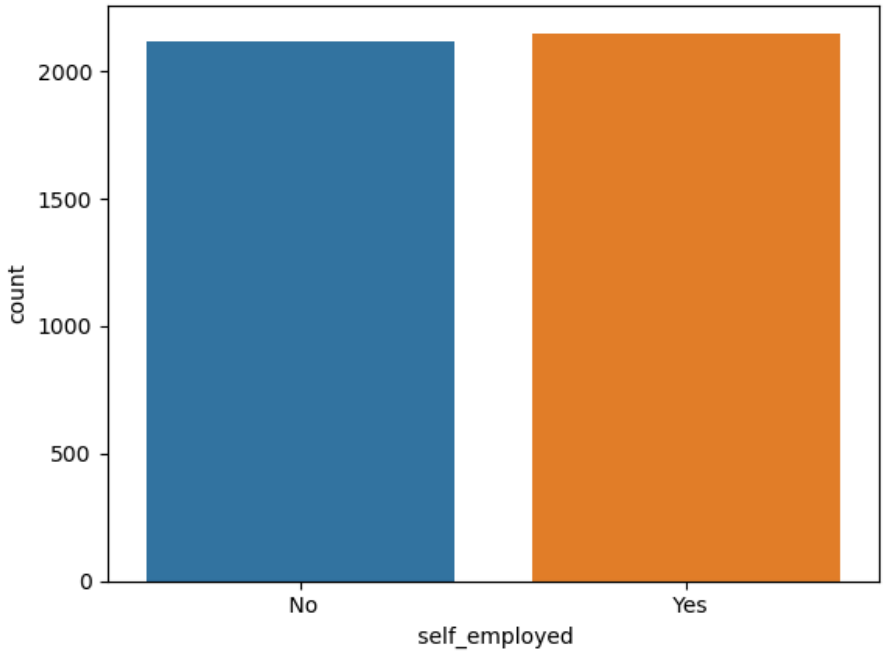
In [19]:
```
df[' self_employed'].unique()
```

Out[19]: array([' No', ' Yes'], dtype=object)

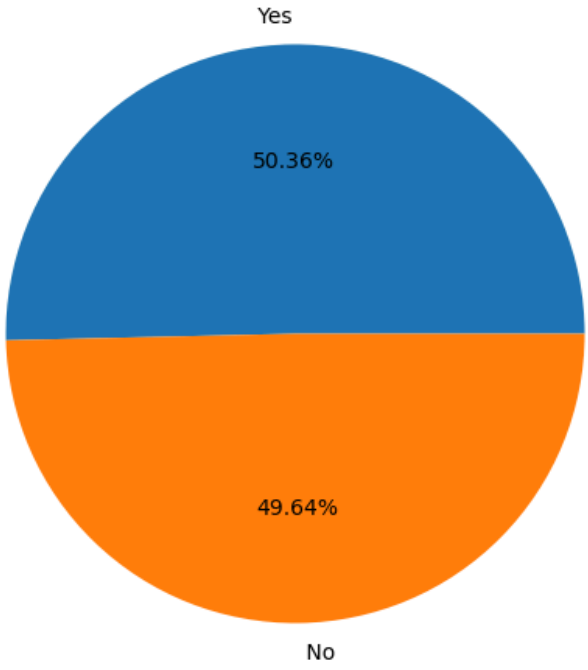
In [20]:
```
df[' self_employed'].value_counts(dropna=False)
```

Out[20]:
```
Yes     2150
No      2119
Name:  self_employed, dtype: int64
```

In [21]:
```
sns.countplot(x=' self_employed',data=df)
plt.show()
```



In [22]:
```
plt.figure(figsize=(6,8))
plt.pie(x=df[' self_employed'].value_counts().values,labels=df[' self_employed'].value_counts().inde
plt.show()
```



- There are 50.36% of people who are self employed
- There are 49.64% of people who are not self employed

In [23]:
```
df[' income_annum'].nunique()
```

Out[23]:
```
98
```

In [24]:
```
df[' income_annum'].unique()
```

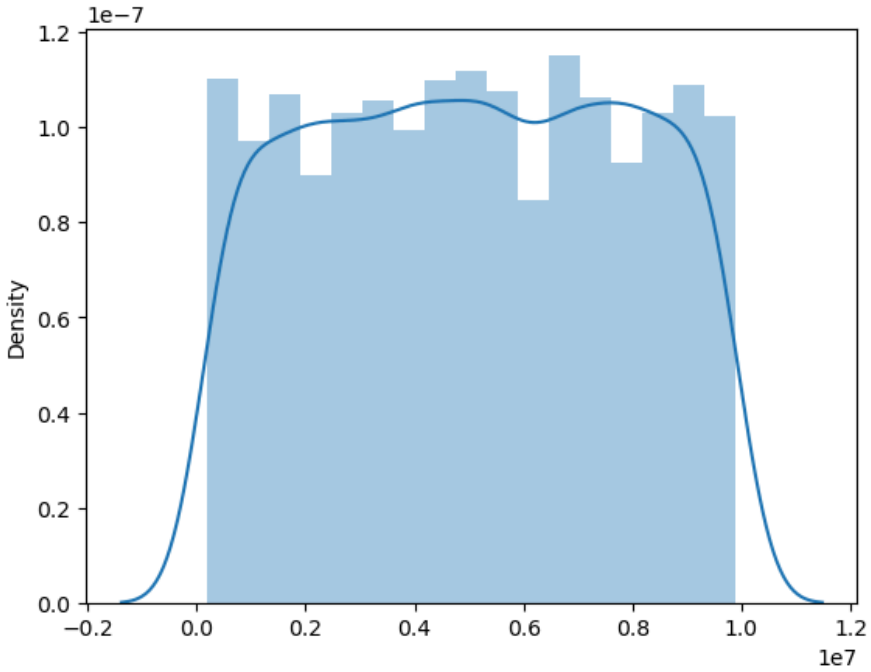
Out[24]:
```
array([9600000, 4100000, 9100000, 8200000, 9800000, 4800000, 8700000,
       5700000,  800000, 1100000, 2900000, 6700000, 5000000, 1900000,
       4700000,  500000, 2700000, 6300000, 5800000, 6500000, 4900000,
       3100000, 2400000, 7000000, 9000000, 8400000, 1700000, 1600000,
       8000000, 3600000, 1500000, 7800000, 1400000, 4200000, 5500000,
       9500000, 7300000, 3800000, 5100000, 4300000, 9300000, 7400000,
       8500000, 8800000, 3300000, 3900000, 8300000, 5600000, 5300000,
       2600000,  700000, 3500000, 9900000, 3000000, 6800000, 2000000,
       1000000,  300000, 6600000, 9400000, 4400000,  400000, 6200000,
       9700000, 7100000,  600000, 7200000,  900000,  200000, 1800000,
       4600000, 2200000, 2500000, 8600000, 4000000, 5200000, 8900000,
       1300000, 4500000, 8100000, 9200000, 2800000, 7500000, 6400000,
       6900000, 7700000, 3200000, 7900000, 5900000, 3400000, 2100000,
       3700000, 5400000, 2300000, 7600000, 6000000, 6100000, 1200000],
      dtype=int64)
```

In [25]:
```
df[' income_annum'].value_counts(dropna=False)
```

Out[25]:
```
7000000    62
4100000    59
7600000    57
4700000    56
6900000    55
           ..
3600000    33
3400000    33
9300000    33
8500000    32
6700000    30
Name:  income_annum, Length: 98, dtype: int64
```

In [27]:
```
1 sns.distplot(x=df[' income_annum'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\2273562137.py:1: UserWarning:

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

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

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

  sns.distplot(x=df[' income_annum'])
```



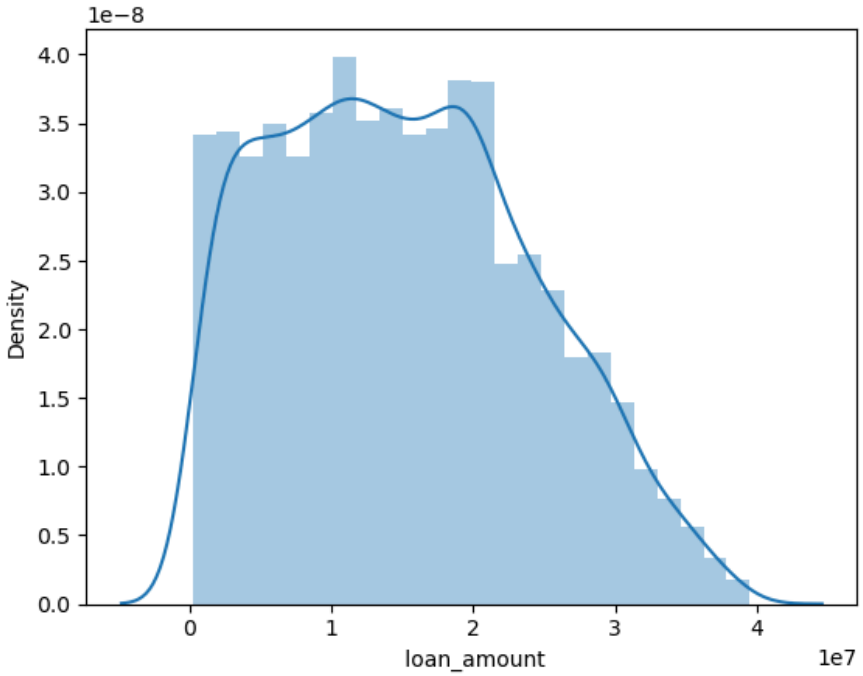
In [28]:
```
1 df[' loan_amount'].nunique()
```

Out[28]: 378

In [30]:
```
1 # df[' loan_amount'].unique()
```

In [31]:
```
1 sns.distplot(df[' loan_amount'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\162217760.py:1: UserWarning:

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

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

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

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



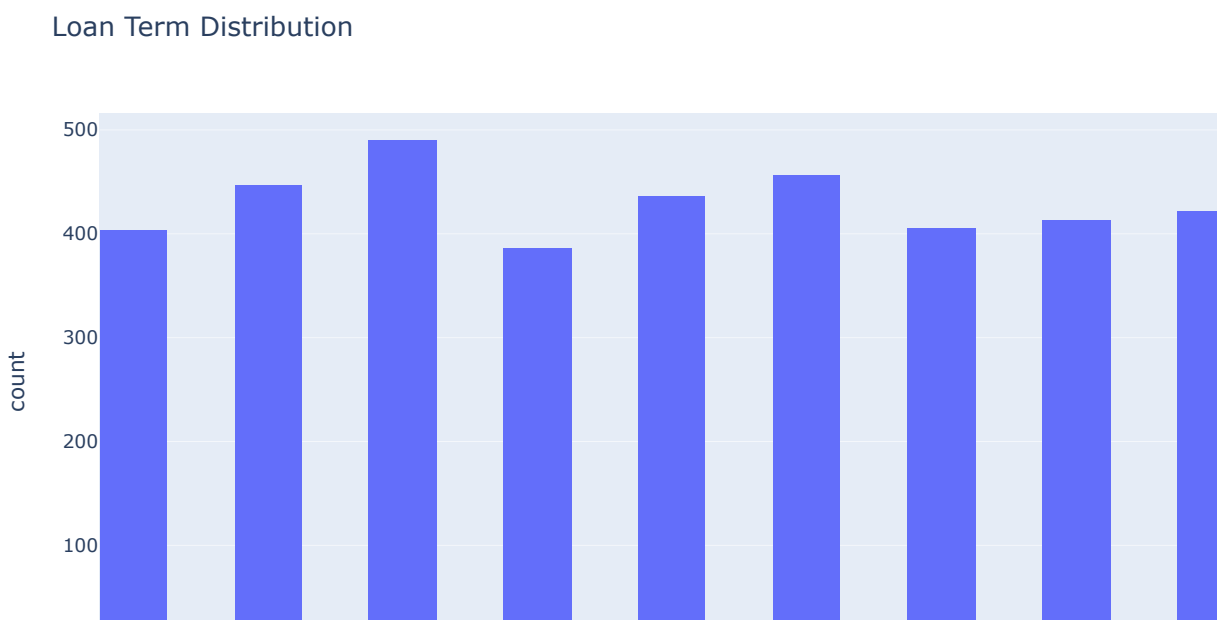
In [32]:
```
1 df[' loan_term'].nunique()
```

Out[32]: 10

In [33]:
```
1 df[' loan_term'].unique()
```

Out[33]: array([12, 8, 20, 10, 4, 2, 18, 16, 14, 6], dtype=int64)

In [39]:
```
import plotly.express as px
fig=px.histogram(df,x=df[' loan_term'],nbins=30,title="Loan Term Distribution")
fig.show()
```



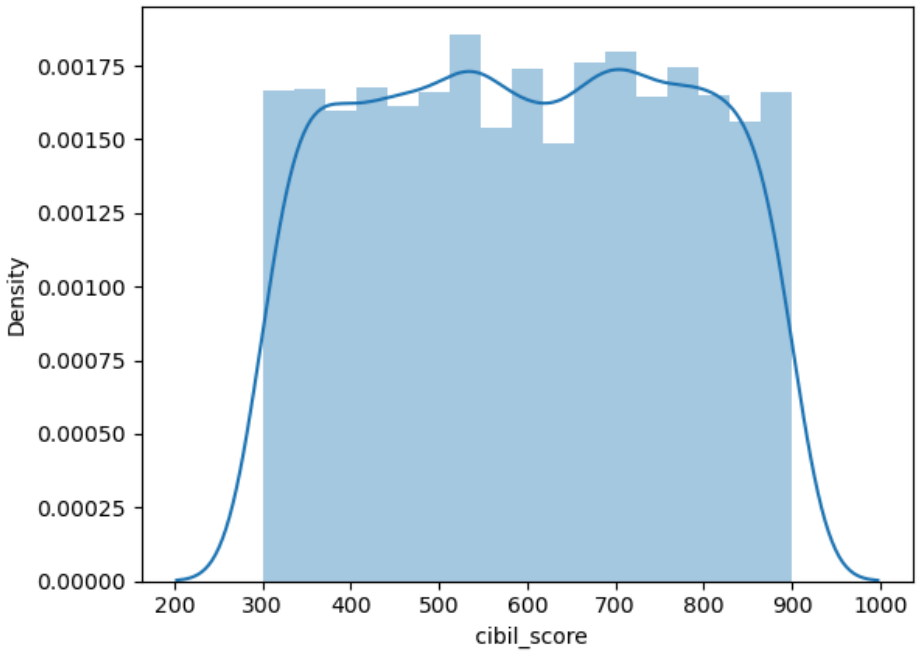
In [40]:
```
df[' cibil_score'].nunique()
```

Out[40]: 601

In [43]:
```
# df[' cibil_score'].unique()
```

In [45]:
```
1 sns.distplot(df[' cibil_score'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\2655950409.py:1: UserWarning:


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

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

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



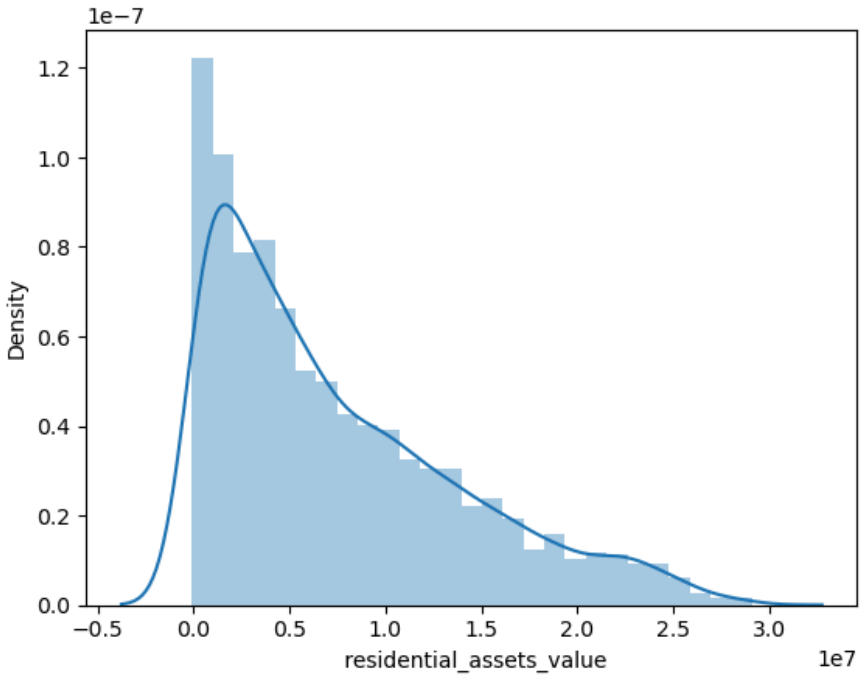
In [46]:
```
1 df[' residential_assets_value'].nunique()
```

Out[46]: 278

In [48]:
```
1 # df[' residential_assets_value'].unique()
```

In [50]:
```
1 sns.distplot(df[' residential_assets_value'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\1197037550.py:1: UserWarning:


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

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

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



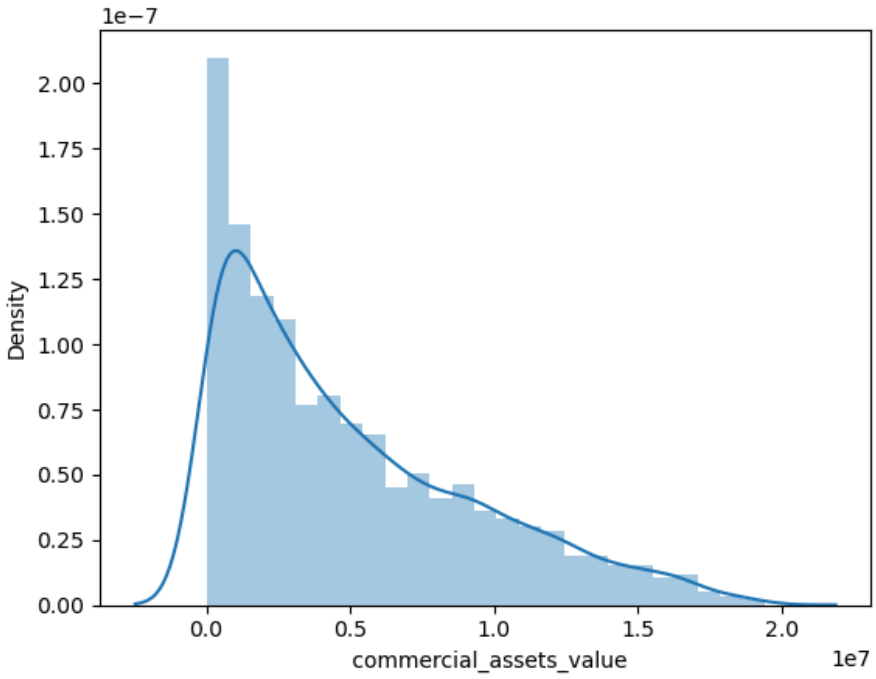
In [51]:
```
1 df[' commercial_assets_value'].nunique()
```

Out[51]: 188

In [53]:
```
1 # df[' commercial_assets_value'].unique()
```

In [54]:
```
1 sns.distplot(df[' commercial_assets_value'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\268431795.py:1: UserWarning:


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

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

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

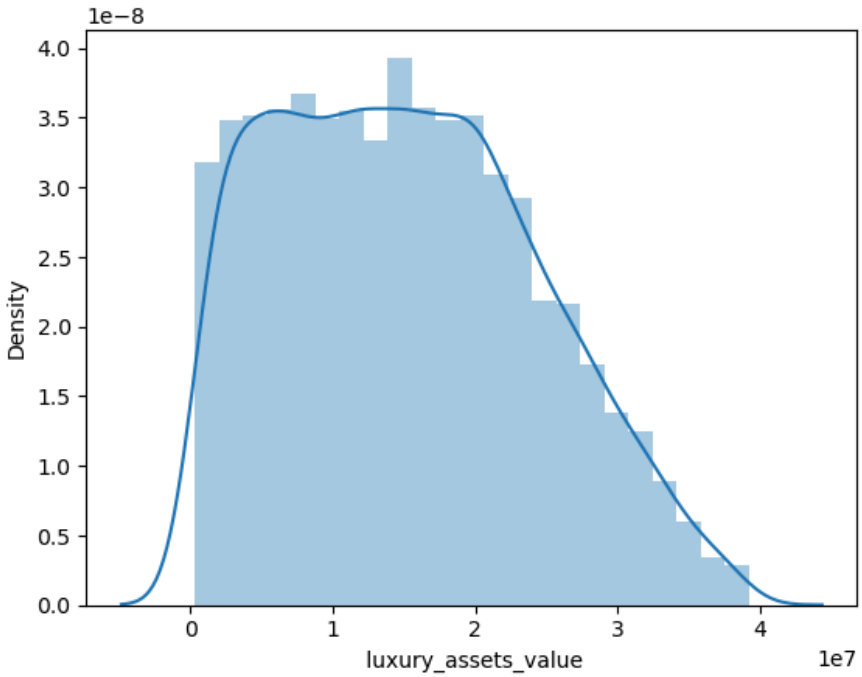


In [55]:
```
1 df[' luxury_assets_value'].nunique()
```

Out[55]: 379

In [56]:
```
1 sns.distplot(df[' luxury_assets_value'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\38931760.py:1: UserWarning:


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

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

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



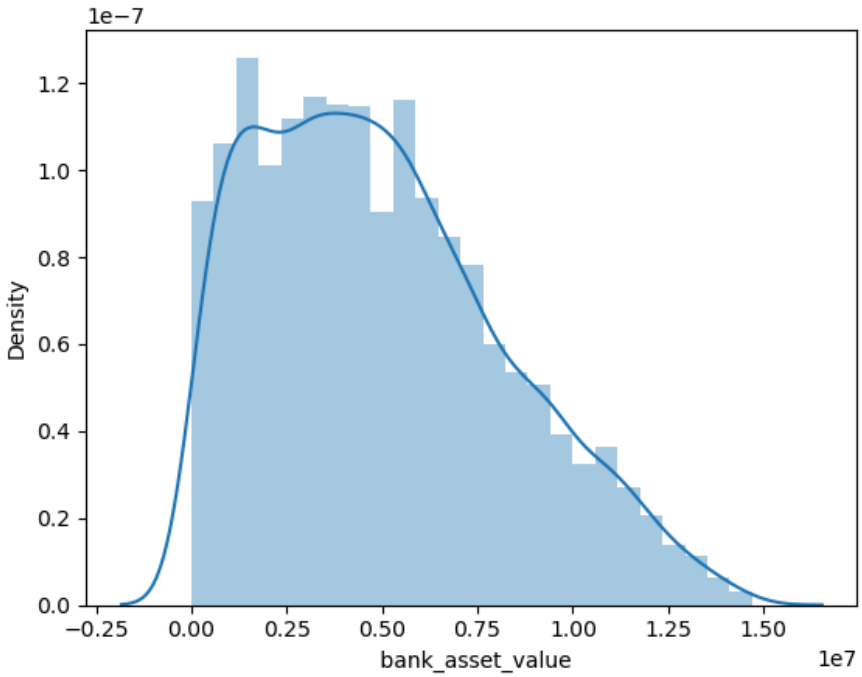
In [57]:
```
1 df[' bank_asset_value'].nunique()
```

Out[57]: 146

In [59]:
```
1 # df[' bank_asset_value'].unique()
```

In [61]:
```
1 sns.distplot(df[' bank_asset_value'])
2 plt.show()
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\2747197036.py:1: UserWarning:


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

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

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwaskom/de441
47ed2974457ad6372750bbe5751)
```



In [62]:
```
1 df[' loan_status'].nunique()
```

Out[62]: 2

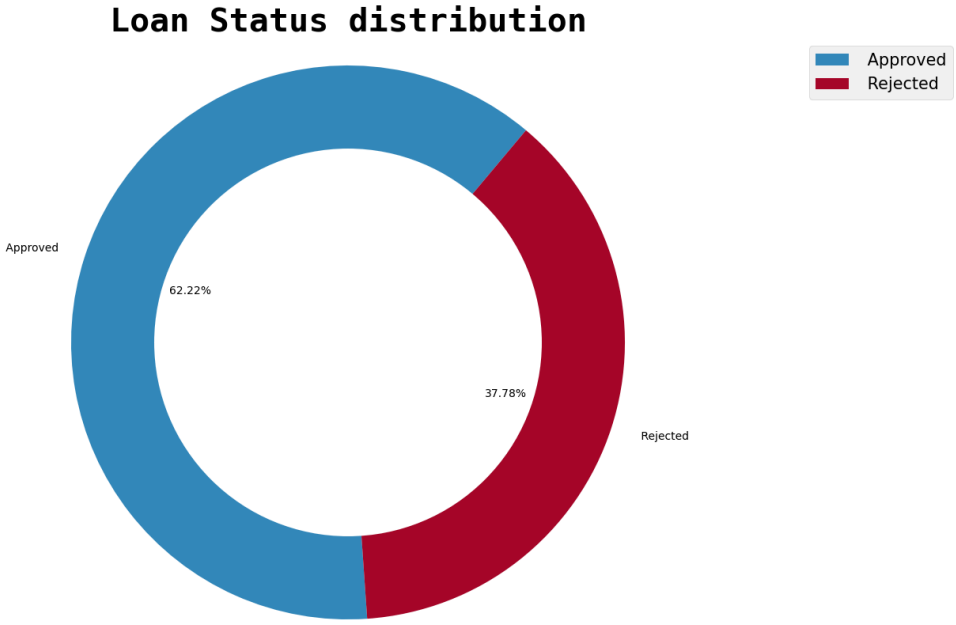
In [63]:
```
1 df[' loan_status'].unique()
```

Out[63]: array([' Approved', ' Rejected'], dtype=object)

In [64]:
```
1 df[' loan_status'].value_counts(dropna=False)
```

Out[64]:
```
 Approved    2656
 Rejected    1613
Name:  loan_status, dtype: int64
```

In [74]:
```
da=df[' loan_status'].value_counts(dropna=False)

plt.style.use("bmh")
plt.figure(figsize=(20,10))
plt.pie(x=da.values,labels=da.index,startangle=50,autopct="%1.2f%%")

centre_circle = plt.Circle((0,0),.7,fc="white")
fig=plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('Loan Status distribution', fontdict={
          'fontname': 'Monospace', 'fontsize': 30, 'fontweight': 'bold'})
plt.axis('equal')
plt.legend(prop={'size': 15})

plt.show()
```



- There are only 62.22% of the people who get approved for the loan.
- And 37.78% who got rejected for the loan.

Numerical

In [76]:
```
df.describe(percentiles=(.01,.02,.03,.04,.05,.1,.25,.5,.75,.9,.95,.96,.97,.98,.99)).T
```

Out[76]:

	count	mean	std	min	1%	2%	3%	4%	5%
no_of_dependents	4269.0	2.498712e+00	1.695910e+00	0.0	0.0	0.0	0.0	0.00	0.0
income_annum	4269.0	5.059124e+06	2.806840e+06	200000.0	300000.0	300000.0	500000.0	500000.00	600000.0
loan_amount	4269.0	1.513345e+07	9.043363e+06	300000.0	700000.0	1000000.0	1204000.0	1500000.00	1800000.0
loan_term	4269.0	1.090045e+01	5.709187e+00	2.0	2.0	2.0	2.0	2.00	2.0
cibil_score	4269.0	5.999361e+02	1.724304e+02	300.0	304.0	311.0	317.0	322.72	330.0
residential_assets_value	4269.0	7.472617e+06	6.503637e+06	-100000.0	0.0	100000.0	100000.0	200000.00	300000.0
commercial_assets_value	4269.0	4.973155e+06	4.388966e+06	0.0	0.0	0.0	100000.0	100000.00	200000.0
luxury_assets_value	4269.0	1.512631e+07	9.103754e+06	300000.0	700000.0	1000000.0	1300000.0	1600000.00	1900000.0
bank_asset_value	4269.0	4.976692e+06	3.250185e+06	0.0	200000.0	200000.0	300000.0	400000.00	500000.0

BiVariate Analysis

CAT-NUM

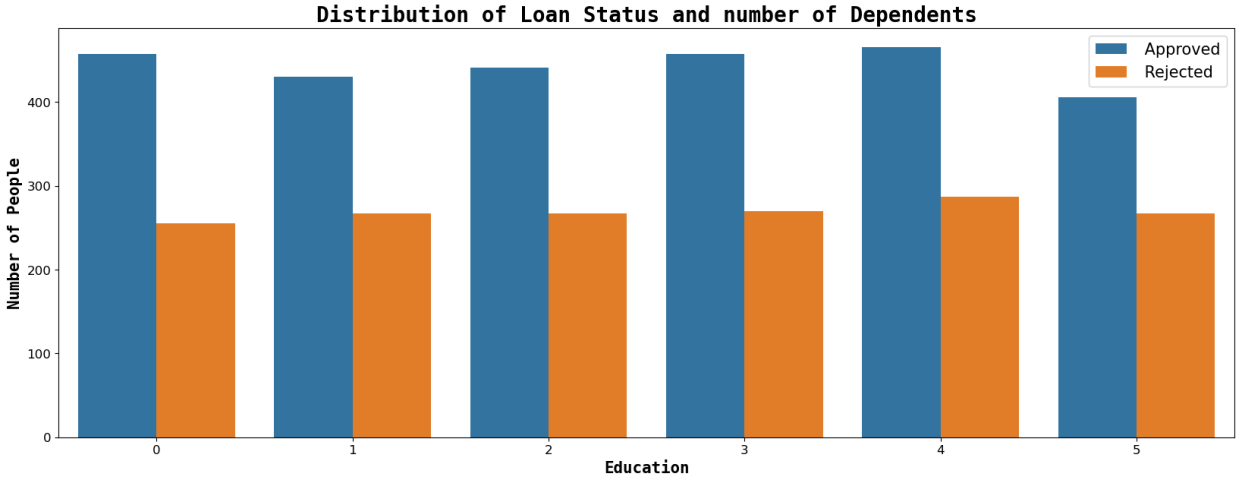
In [77]:
```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4269 entries, 0 to 4268
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0    no_of_dependents         4269 non-null   int64
 1    education                4269 non-null   object
 2    self_employed            4269 non-null   object
 3    income_annum             4269 non-null   int64
 4    loan_amount              4269 non-null   int64
 5    loan_term                4269 non-null   int64
 6    cibil_score              4269 non-null   int64
 7    residential_assets_value 4269 non-null   int64
 8    commercial_assets_value  4269 non-null   int64
 9    luxury_assets_value      4269 non-null   int64
 10   bank_asset_value         4269 non-null   int64
 11   loan_status              4269 non-null   object
dtypes: int64(9), object(3)
memory usage: 400.3+ KB
```

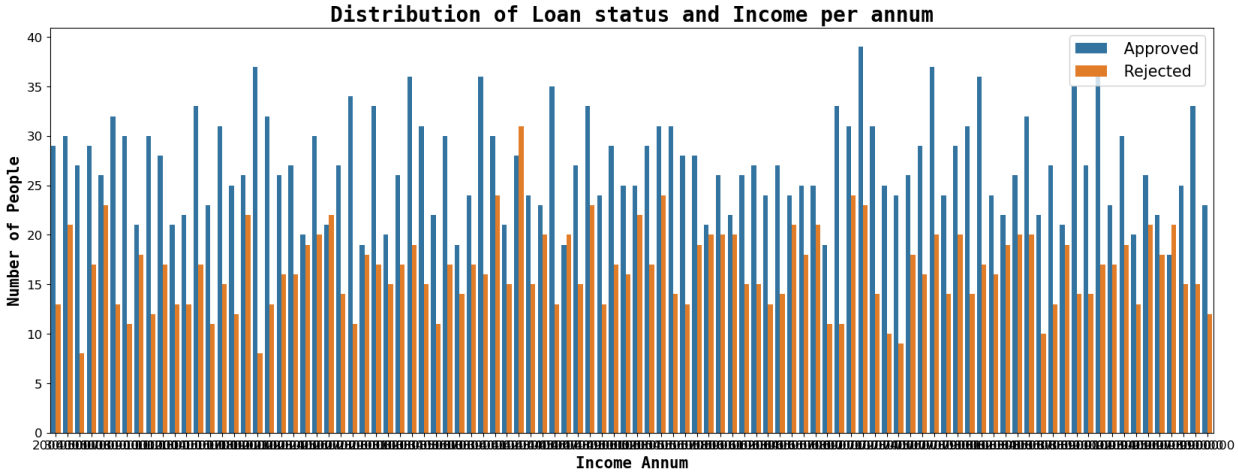
In [78]:
```
1 df.columns
```

Out[78]:
```
Index([' no_of_dependents', ' education', ' self_employed', ' income_annum',
       ' loan_amount', ' loan_term', ' cibil_score',
       ' residential_assets_value', ' commercial_assets_value',
       ' luxury_assets_value', ' bank_asset_value', ' loan_status'],
      dtype='object')
```

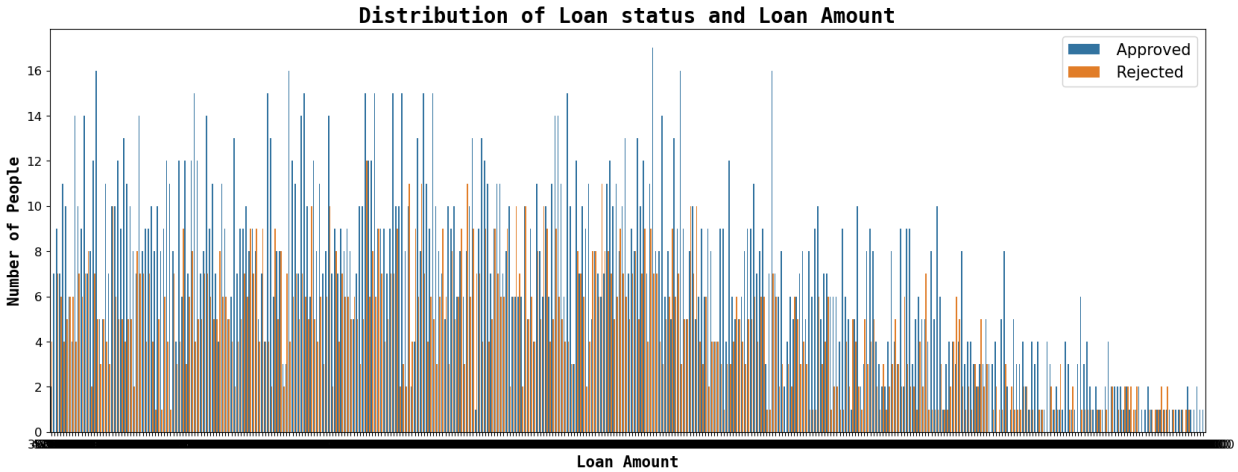
In [86]:
```
1  plt.style.use("default")
2  plt.figure(figsize=(20,7))
3  sns.countplot(x=' no_of_dependents',data=df,hue=' loan_status')
4  plt.title("Distribution of Loan Status and number of Dependents",fontdict={
5      "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
6  plt.xlabel("Education",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
7  plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
8
9  plt.tick_params(labelsize=12)
10 plt.legend(loc=1,prop={"size":15})
11 plt.show()
```



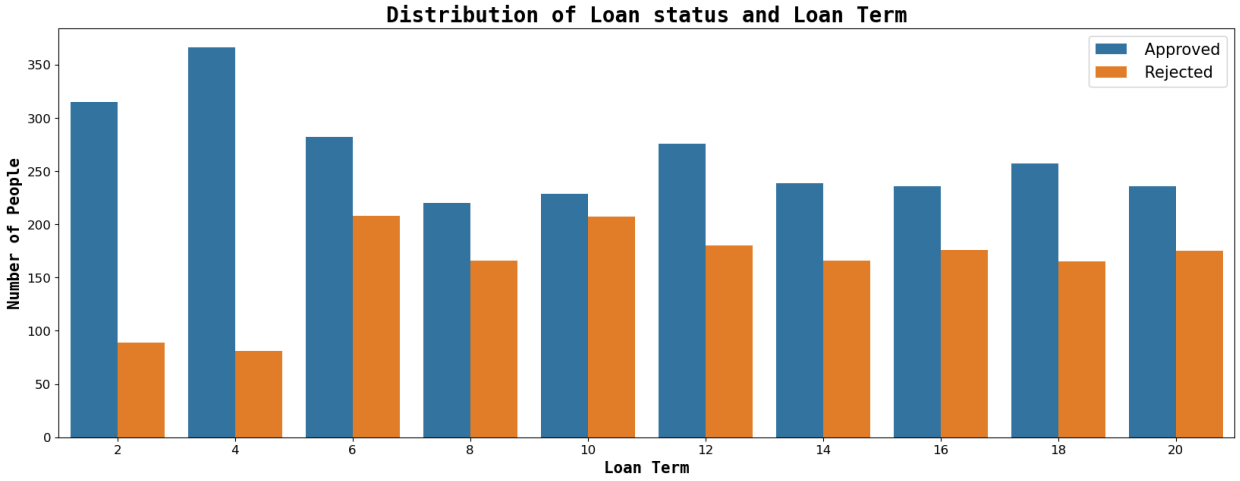
In [88]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' income_annum',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Income per annum",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Income Annum",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```



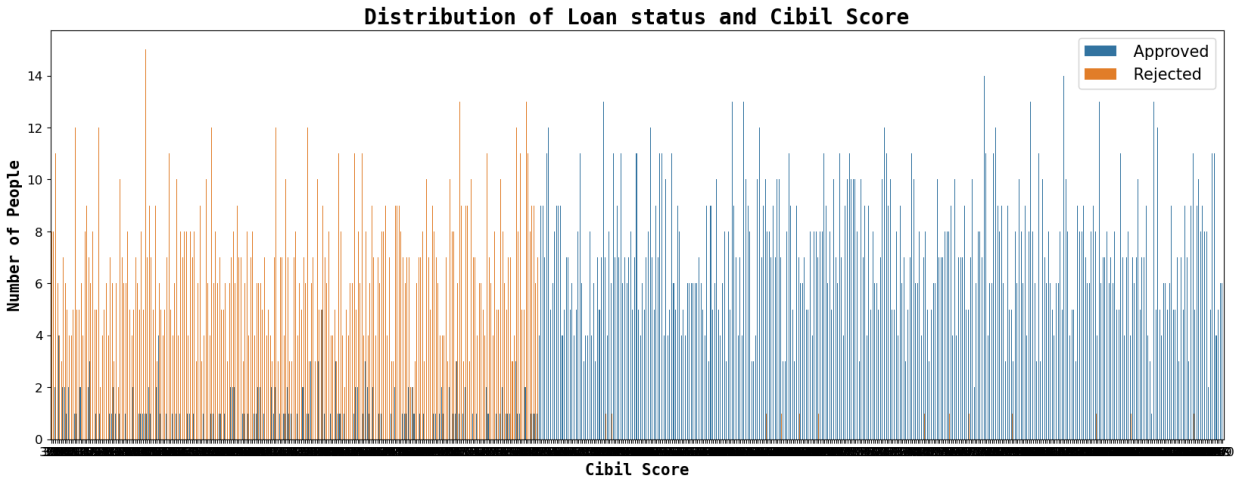
In [89]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' loan_amount',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Loan Amount",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Loan Amount",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```



In [90]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' loan_term',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Loan Term",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Loan Term",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```

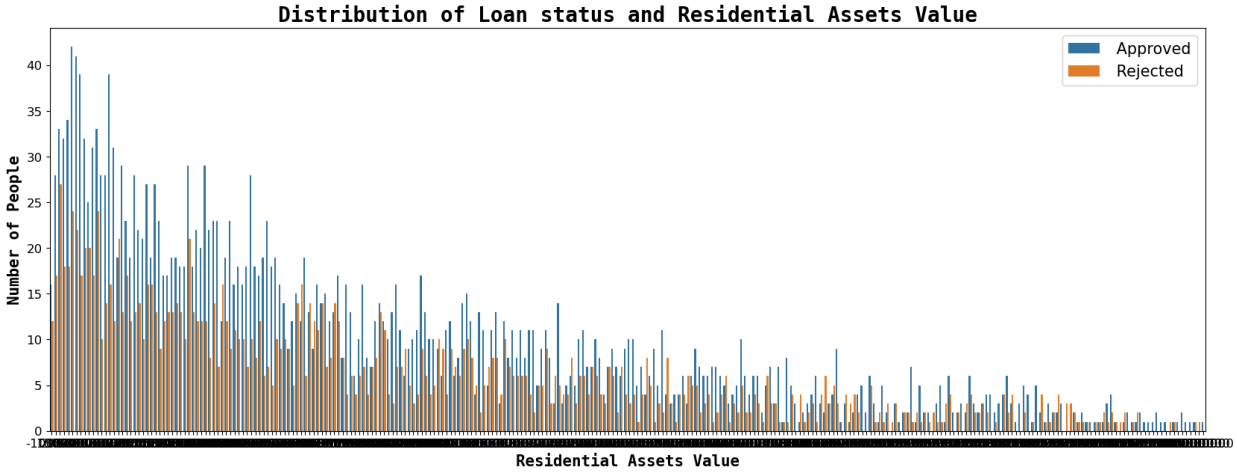


In [94]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' cibil_score',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Cibil Score",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Cibil Score",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```

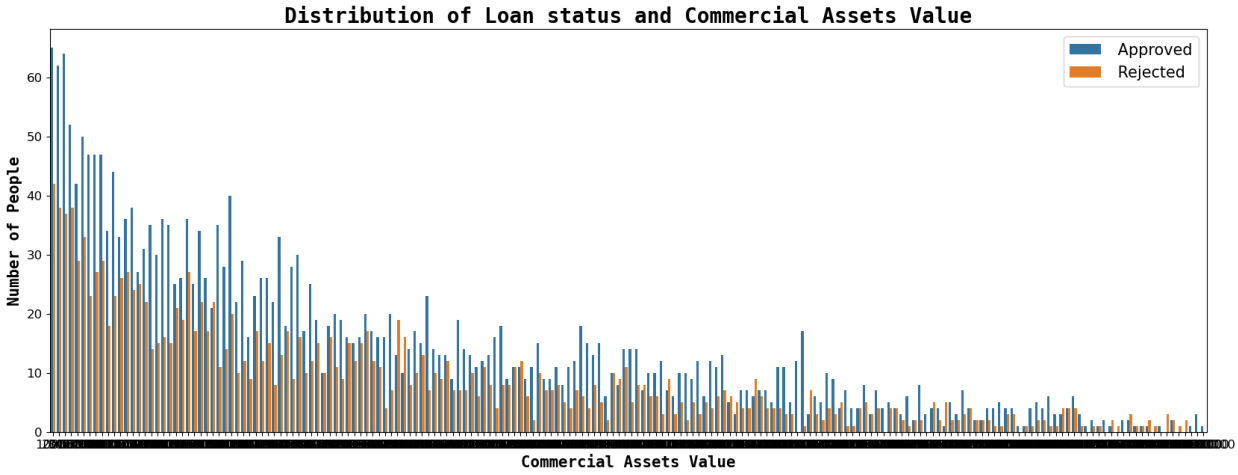


- It can be seen that persons with lower Cibil Score are higher chance of rejection.

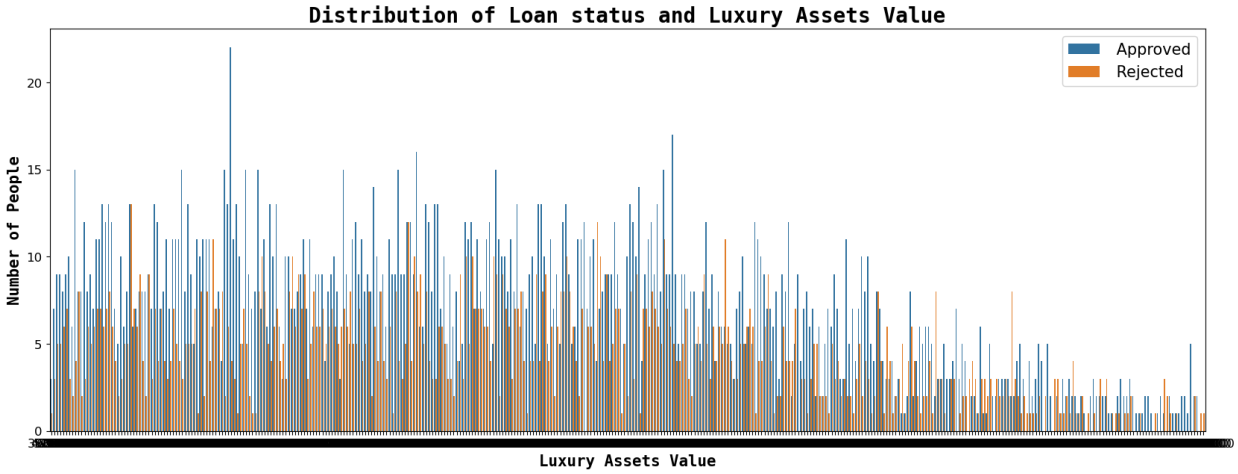
In [95]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' residential_assets_value',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Residential Assets Value",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Residential Assets Value",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"b
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```



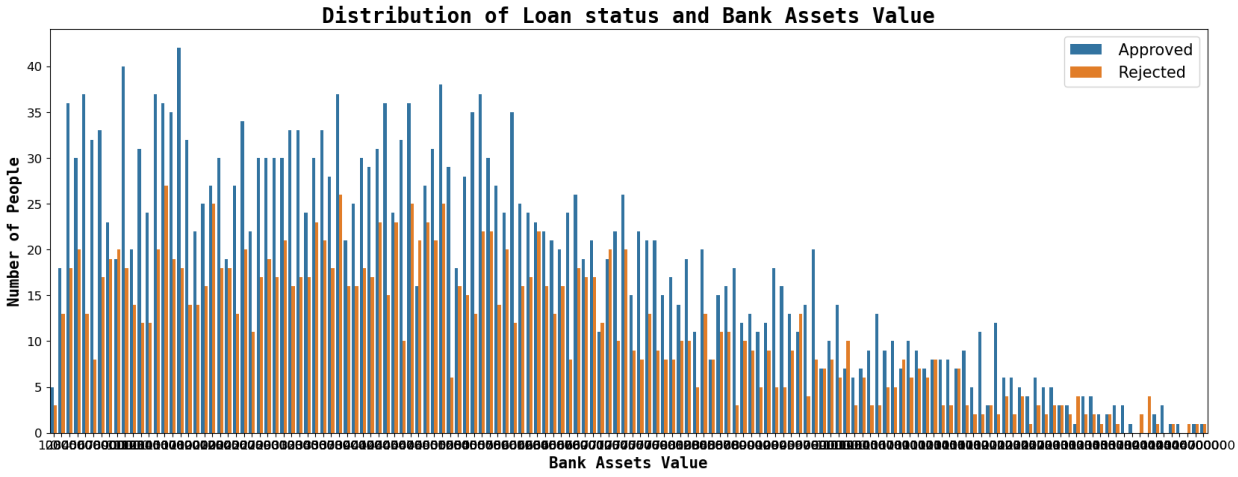
In [96]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' commercial_assets_value',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Commercial Assets Value",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Commercial Assets Value",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bo
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```



In [97]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' luxury_assets_value',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Luxury Assets Value",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Luxury Assets Value",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```



In [100]:
```
plt.style.use("default")
plt.figure(figsize=(20,7))
sns.countplot(x=' bank_asset_value',data=df,hue=" loan_status")
plt.title("Distribution of Loan status and Bank Assets Value",fontdict={
    "fontname":"Monospace","fontsize":20,"fontweight":"bold"})
plt.xlabel("Bank Assets Value",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})
plt.ylabel("Number of People",fontdict={"fontname":"Monospace","fontsize":15,"fontweight":"bold"})

plt.tick_params(labelsize=12)
plt.legend(loc=1,prop={"size":15})
plt.show()
```



MultiVariate Analysis

In [102]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [110]:
```
df1=df.copy()
```

In [112]:
```
cols=[' education',' self_employed',' loan_status']
for i in cols:
    df1[i]=le.fit_transform(df1[i])
```

In [114]:
```
df1[[' education',' self_employed',' loan_status']].value_counts()
```

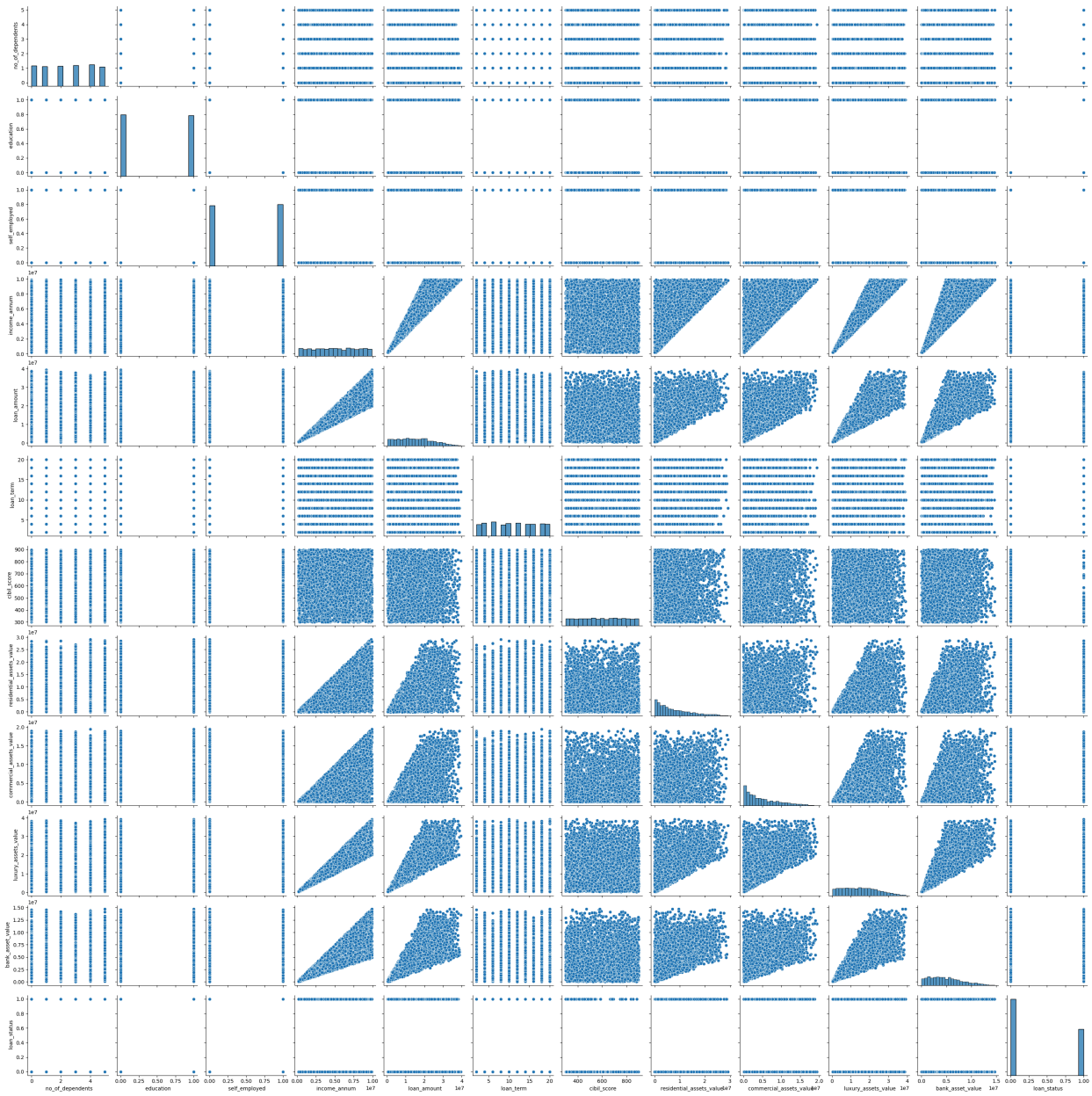
Out[114]:
```
 education   self_employed   loan_status
0            0               0              681
1            1               0              680
0            1               0              658
1            0               0              637
                             1              415
0            0               1              408
                             1              397
1            0               1              393
dtype: int64
```

In [115]:
```
df[[' education',' self_employed',' loan_status']].value_counts()
```

Out[115]:
```
 education      self_employed   loan_status
 Graduate       No              Approved       681
 Not Graduate   Yes             Approved       680
 Graduate       Yes             Approved       658
 Not Graduate   No              Approved       637
                Yes             Rejected       415
 Graduate       No              Rejected       408
                Yes             Rejected       397
 Not Graduate   No              Rejected       393
dtype: int64
```

In [119]:
```
sns.pairplot(df1)
plt.show()
```

Out[119]: <seaborn.axisgrid.PairGrid at 0x1d0663b79d0>



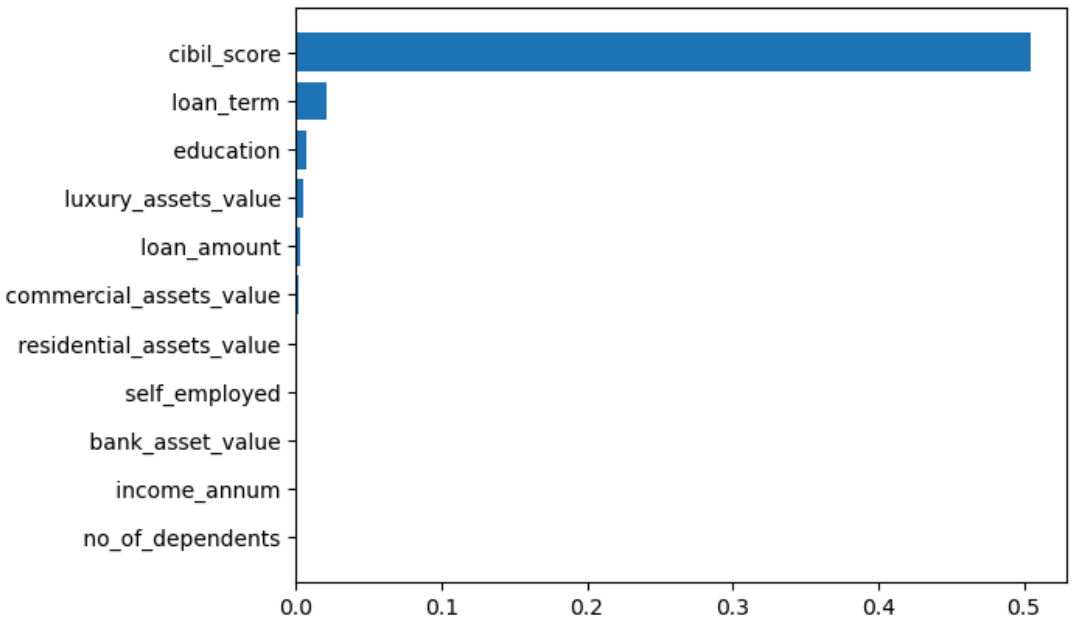
Outlier Treatment

In [122]:
```
var=[' no_of_dependents', ' education', ' self_employed', ' income_annum',
       ' loan_amount', ' loan_term', ' cibil_score',
       ' residential_assets_value', ' commercial_assets_value',
       ' luxury_assets_value', ' bank_asset_value', ' loan_status']

for i in var:
    q1=df1[i].quantile(.25)
    q3=df1[i].quantile(.75)
    iqr=q3-q1

    lower_bound=q1-1.5*iqr
    upper_bound=q3+1.5*iqr

    df1[i]=np.where(df1[i]>upper_bound,upper_bound,df1[i])
    df1[i]=np.where(df1[i]<=lower_bound,lower_bound,df1[i])
```

Feature Engineering

In [126]:
```
x=df1.drop(columns=" loan_status")
y=df1[" loan_status"]
```

In [127]:
```
from sklearn.feature_selection import mutual_info_classif
imp=mutual_info_classif(x,y)
m1=pd.DataFrame({"features":x.columns,"imp":imp}).sort_values(by=["imp"],ascending=True)
plt.barh(y=m1["features"],width=m1["imp"])
plt.show()
```



In [128]:
```
m1[m1["imp"]>0]["features"].values
```

Out[128]:
```
array([' self_employed', ' residential_assets_value',
       ' commercial_assets_value', ' loan_amount', ' luxury_assets_value',
       ' education', ' loan_term', ' cibil_score'], dtype=object)
```

In [148]:
```
x=df1[[' luxury_assets_value',
       ' education', ' loan_term', ' cibil_score']]
y=df1[" loan_status"]
```

In [149]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=0)
```

In [150]:
```
logR=LogisticRegression()
```

In [151]:
```
logR.fit(x_train,y_train)
```

Out[151]:
```
▾ LogisticRegression
LogisticRegression()
```

In [152]:
```
logR.score(x_train,y_train)
```

Out[152]:
```
0.7054886211512718
```

In [156]:
```
print("Accuracy Train:", str(round(logR.score(x_train,y_train)*100,2))+str("%"))
```

```
Accuracy Train: 70.55%
```

In [157]:
```
print("Accuracy Test:", str(round(logR.score(x_test,y_test)*100,2))+str("%"))
```

```
Accuracy Test: 70.49%
```

In [158]:
```
pred_train=logR.predict(x_train)
pred_test=logR.predict(x_test)
```

In [159]:
```
pred_test
```

Out[159]:
```
array([0., 0., 0., ..., 0., 0., 0.])
```

In [160]:
```
pd.DataFrame({"Actual":y_train,"Predicted":pred_train}).head(15)
```

Out[160]:

	Actual	Predicted
1023	0.0	0.0
728	1.0	0.0
133	1.0	1.0
2255	1.0	1.0
1044	0.0	0.0
366	0.0	0.0
1927	0.0	0.0
3594	0.0	0.0
1140	0.0	0.0
3773	0.0	1.0
259	0.0	0.0
2180	0.0	0.0
2487	0.0	0.0
3379	1.0	0.0
2678	0.0	0.0

In [161]:
```
pd.DataFrame({"Actual":y_test,"Predicted":pred_test}).head(15)
```

Out[161]:

	Actual	Predicted
1972	0.0	0.0
528	0.0	0.0
3540	0.0	0.0
87	1.0	1.0
1621	1.0	1.0
1949	1.0	1.0
520	0.0	0.0
1715	0.0	0.0
3994	0.0	0.0
2369	1.0	1.0
125	0.0	0.0
2714	0.0	0.0
3804	1.0	1.0
4228	1.0	0.0
3176	1.0	0.0

In [163]:
```
pd.DataFrame(metrics.confusion_matrix(y_train,pred_train), index=["Act0","Act1"],columns=["Pred0","P
```

Out[163]:

	Pred0	Pred1
Act0	1775	106
Act1	774	333

In [164]:
```
pd.DataFrame(metrics.confusion_matrix(y_test,pred_test),index=["Act0","Act1"],columns=["Pred0","Pred
```

Out[164]:

	Pred0	Pred1
Act0	734	41
Act1	337	169

In [165]:
```
print(metrics.classification_report(y_train,pred_train))
```

```
              precision    recall  f1-score   support

         0.0       0.70      0.94      0.80      1881
         1.0       0.76      0.30      0.43      1107

    accuracy                           0.71      2988
   macro avg       0.73      0.62      0.62      2988
weighted avg       0.72      0.71      0.66      2988
```

In [166]:
```
print(metrics.classification_report(y_test,pred_test))
```

```
              precision    recall  f1-score   support

         0.0       0.69      0.95      0.80       775
         1.0       0.80      0.33      0.47       506

    accuracy                           0.70      1281
   macro avg       0.75      0.64      0.63      1281
weighted avg       0.73      0.70      0.67      1281
```

In [167]:
```
prob_train=pd.DataFrame(logR.predict_proba(x_train),columns=["Prob0","Prob1"])
prob_train
```

Out[167]:

	Prob0	Prob1
0	0.840677	0.159323
1	0.618646	0.381354
2	0.447827	0.552173
3	0.329620	0.670380
4	0.933191	0.066809
...	...	...
2983	0.924180	0.075820
2984	0.742521	0.257479
2985	0.633809	0.366191
2986	0.773950	0.226050
2987	0.791846	0.208154

2988 rows × 2 columns

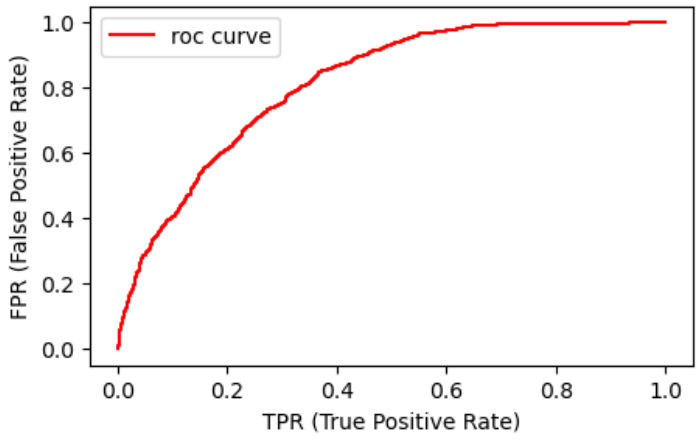
In [169]:
```
prob_test=pd.DataFrame(logR.predict_proba(x_test),columns=["Prob0","Prob1"])
prob_test
```

Out[169]:

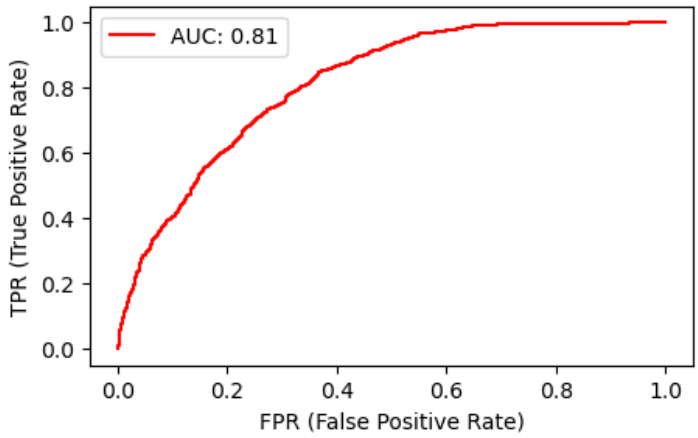
	Prob0	Prob1
0	0.563816	0.436184
1	0.659761	0.340239
2	0.638045	0.361955
3	0.343754	0.656246
4	0.435181	0.564819
...	...	...
1276	0.919445	0.080555
1277	0.864331	0.135669
1278	0.936008	0.063992
1279	0.835353	0.164647
1280	0.724170	0.275830

1281 rows × 2 columns

In [170]:
```
tpr,fpr,threshold= metrics.roc_curve(y_train,prob_train["Prob1"])
plt.figure(figsize=(5,3))
plt.plot(tpr,fpr,color="r",label="roc curve")
plt.xlabel("TPR (True Positive Rate)")
plt.ylabel("FPR (False Positive Rate)")
plt.legend()
plt.show()
```



In [172]:
```
fpr,tpr,threshold= metrics.roc_curve(y_train,prob_train["Prob1"])
auc=metrics.roc_auc_score(y_train,prob_train["Prob1"])
plt.figure(figsize=(5,3))
plt.plot(fpr,tpr,color="r",label="AUC: "+str(round(auc,2)))
plt.ylabel("TPR (True Positive Rate)")
plt.xlabel("FPR (False Positive Rate)")
plt.legend()
plt.show()
```



In [173]:
```
new_pred_train=np.where(prob_train["Prob1"]>.25,1,0)
new_pred_test=np.where(prob_test["Prob1"]>.25,1,0)
```

In [175]:
```
print(metrics.classification_report(y_train,new_pred_train))
print(metrics.classification_report(y_test,new_pred_test))
```

```
              precision    recall  f1-score   support

         0.0       0.86      0.64      0.74      1881
         1.0       0.58      0.83      0.68      1107

    accuracy                           0.71      2988
   macro avg       0.72      0.74      0.71      2988
weighted avg       0.76      0.71      0.72      2988

              precision    recall  f1-score   support

         0.0       0.86      0.66      0.75       775
         1.0       0.62      0.84      0.71       506

    accuracy                           0.73      1281
   macro avg       0.74      0.75      0.73      1281
weighted avg       0.77      0.73      0.73      1281
```

In [176]:
```
x_train.columns
```

Out[176]:
```
Index([' luxury_assets_value', ' education', ' loan_term', ' cibil_score'], dtype='object')
```

In [177]:
```
new=pd.read_excel(r"D:\Projects\ML\loan_approval_dataset\loandatatest.xlsx")
```

In [178]:
```
new
```

Out[178]:

	loan_id	no_of_dependents	education	self_employed	income_annum	loan_amount	loan_term	cibil_score	residential_assets_
0	1	2	Graduate	No	9600000	29900000	12	778	24
1	2	0	Not Graduate	Yes	4100000	12200000	8	417	27
2	3	3	Graduate	No	9100000	29700000	20	506	71
3	4	3	Graduate	No	8200000	30700000	8	467	182
4	5	5	Not Graduate	Yes	9800000	24200000	20	382	124
5	6	0	Graduate	Yes	4800000	13500000	10	319	68
6	7	5	Graduate	No	8700000	33000000	4	678	225
7	8	2	Graduate	Yes	5700000	15000000	20	382	132
8	9	0	Graduate	Yes	800000	2200000	20	782	13
9	10	5	Not Graduate	No	1100000	4300000	10	388	32
10	4265	5	Graduate	Yes	1000000	2300000	12	317	28
11	4266	0	Not Graduate	Yes	3300000	11300000	20	559	42
12	4267	2	Not Graduate	No	6500000	23900000	18	457	12
13	4268	1	Not Graduate	No	4100000	12800000	8	780	82
14	4269	1	Graduate	No	9200000	29700000	10	607	178

In [179]:
```
new_df=new[[' luxury_assets_value', ' education', ' loan_term', ' cibil_score']]
```

In [180]:
```
new_df
```

Out[180]:

	luxury_assets_value	education	loan_term	cibil_score
0	22700000	Graduate	12	778
1	8800000	Not Graduate	8	417
2	33300000	Graduate	20	506
3	23300000	Graduate	8	467
4	29400000	Not Graduate	20	382
5	13700000	Graduate	10	319
6	29200000	Graduate	4	678
7	11800000	Graduate	20	382
8	2800000	Graduate	20	782
9	3300000	Not Graduate	10	388
10	3300000	Graduate	12	317
11	11000000	Not Graduate	20	559
12	18100000	Not Graduate	18	457
13	14100000	Not Graduate	8	780
14	35700000	Graduate	10	607

In [181]:
```
for i in new_df.columns:
    if new_df[i].dtypes=="object":
        encoder=LabelEncoder()
        new_df[i]=encoder.fit_transform(new_df[i])
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\4001675252.py:4: 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)
```

In [183]:
```
pd.DataFrame(logR.predict_proba(new_df),columns=["Prob0","Prob1"])
```

Out[183]:

	Prob0	Prob1
0	0.745715	0.254285
1	0.695732	0.304268
2	0.336558	0.663442
3	0.481606	0.518394
4	0.303820	0.696180
5	0.528598	0.471402
6	0.559030	0.440970
7	0.617667	0.382333
8	0.928905	0.071095
9	0.756401	0.243599
10	0.706991	0.293009
11	0.762777	0.237223
12	0.568957	0.431043
13	0.848454	0.151546
14	0.377924	0.622076

In [184]:
```
new_prob = pd.DataFrame(logR.predict_proba(new_df),columns=["Prob0","Prob1"])
```

In [196]:
```
pred_new_status=np.where(new_prob["Prob1"]>=.35,1,0)
```

In [197]:
```
new_df.loc[:,"STATUS"]=pred_new_status
```

```
C:\Users\Pulkit\AppData\Local\Temp\ipykernel_33200\4205649271.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)
```

In [198]:
```
new_df
```

Out[198]:

	luxury_assets_value	education	loan_term	cibil_score	STATUS
0	22700000	0	12	778	0
1	8800000	1	8	417	0
2	33300000	0	20	506	1
3	23300000	0	8	467	1
4	29400000	1	20	382	1
5	13700000	0	10	319	1
6	29200000	0	4	678	1
7	11800000	0	20	382	1
8	2800000	0	20	782	0
9	3300000	1	10	388	0
10	3300000	0	12	317	0
11	11000000	1	20	559	0
12	18100000	1	18	457	1
13	14100000	1	8	780	0
14	35700000	0	10	607	1

In [200]:
```
df[" loan_status"].head(10)
```

Out[200]:
```
0     Approved
1     Rejected
2     Rejected
3     Rejected
4     Rejected
5     Rejected
6     Approved
7     Rejected
8     Approved
9     Rejected
Name:  loan_status, dtype: object
```

In [ ]:
```

```