In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
dataset=pd.read_csv('loan.csv')
```

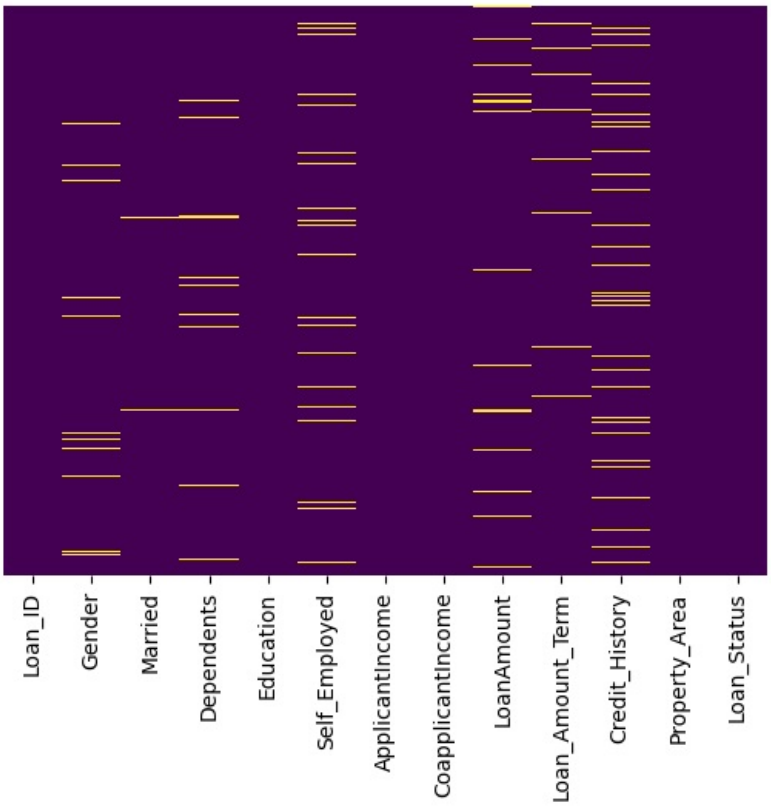
In [5]:
```
dataset
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	
...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	

614 rows × 13 columns

In [9]:
```
sns.heatmap(dataset.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.show()
```



In [11]:
```
dataset.describe()
```

Out[11]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [13]:
```
dataset.isnull().sum()
```

Out[13]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [15]:
```
pd.crosstab(dataset['Credit_History'], dataset['Loan_Status'], margins=True)
```

Out[15]:

Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	97	378	475
All	179	385	564

In [17]:
```
dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12460\2782724874.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  dataset['Gender'].fillna(dataset['Gender'].mode()[0],inplace=True)
```

In [19]:
```
dataset['Married'].fillna(dataset['Married'].mode()[0],inplace=True)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12460\568568310.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  dataset['Married'].fillna(dataset['Married'].mode()[0],inplace=True)
```

In [21]:
```
dataset['Self_Employed'].fillna(dataset['Self_Employed'].mode()[0],inplace=True)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12460\39551059.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  dataset['Self_Employed'].fillna(dataset['Self_Employed'].mode()[0],inplace=True)
```

In [23]:
```
dataset['Dependents'].fillna(dataset['Dependents'].mode()[0],inplace=True)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12460\2291896846.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  dataset['Dependents'].fillna(dataset['Dependents'].mode()[0],inplace=True)
```

In [25]:
```
dataset['Loan_Amount_Term'].fillna(dataset['Loan_Amount_Term'].mode()[0],inplace=True)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12460\3879141361.py:1: FutureWarning: A value is trying to b
e set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  dataset['Loan_Amount_Term'].fillna(dataset['Loan_Amount_Term'].mode()[0],inplace=True)
```

In [27]:
```
dataset['Credit_History'].fillna(dataset['Credit_History'].mode()[0],inplace=True)
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_12460\979352118.py:1: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on w
hich we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)'
or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  dataset['Credit_History'].fillna(dataset['Credit_History'].mode()[0],inplace=True)
```

In [29]:
```
dataset.LoanAmount=dataset.LoanAmount.fillna(dataset.LoanAmount.mean())
```

In [31]:
```
dataset.isnull().sum()
```

Out[31]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

In [33]:
```
dataset
```

Out[33]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	
...	...	...	...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.000000	
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.000000	
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.000000	
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.000000	
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.000000	

614 rows × 13 columns

In [35]:
```
gender=pd.get_dummies(dataset['Gender'],drop_first=True)
married=pd.get_dummies(dataset['Married'],drop_first=True)
education=pd.get_dummies(dataset['Education'],drop_first=True)
self_employed=pd.get_dummies(dataset['Self_Employed'],drop_first=True)
dependents=pd.get_dummies(dataset['Dependents'],drop_first=True)
property_area=pd.get_dummies(dataset['Property_Area'],drop_first=True)
dataset.drop(['Gender','Married','Education','Self_Employed','Dependents','Property_Area'],axis=1,inplace=True)
```

In [37]:
```
dataset=pd.concat([dataset,gender,married,education,self_employed,dependents,property_area],axis=1)
```

In [39]:
```
dataset
```

Out[39]:

	Loan_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status	Male	Yes
0	LP001002	5849	0.0	146.412162	360.0	1.0	Y	True	False
1	LP001003	4583	1508.0	128.000000	360.0	1.0	N	True	True
2	LP001005	3000	0.0	66.000000	360.0	1.0	Y	True	True
3	LP001006	2583	2358.0	120.000000	360.0	1.0	Y	True	True
4	LP001008	6000	0.0	141.000000	360.0	1.0	Y	True	False
...	...	...	...	...	...	...	...	...	...
609	LP002978	2900	0.0	71.000000	360.0	1.0	Y	False	False
610	LP002979	4106	0.0	40.000000	180.0	1.0	Y	True	True
611	LP002983	8072	240.0	253.000000	360.0	1.0	Y	True	True
612	LP002984	7583	0.0	187.000000	360.0	1.0	Y	True	True
613	LP002990	4583	0.0	133.000000	360.0	0.0	N	False	False

614 rows × 16 columns

In [41]:
```
dataset['Male']=dataset['Male'].astype(int)
dataset['Yes']=dataset['Yes'].astype(int)
dataset['Not Graduate']=dataset['Not Graduate'].astype(int)
```

In [43]:
```
dataset
```

Out[43]:

	Loan_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status	Male	Yes	G
0	LP001002	5849	0.0	146.412162	360.0	1.0	Y	1	0	
1	LP001003	4583	1508.0	128.000000	360.0	1.0	N	1	1	
2	LP001005	3000	0.0	66.000000	360.0	1.0	Y	1	1	
3	LP001006	2583	2358.0	120.000000	360.0	1.0	Y	1	1	
4	LP001008	6000	0.0	141.000000	360.0	1.0	Y	1	0	
...	...	...	...	...	...	...	...	...	...	
609	LP002978	2900	0.0	71.000000	360.0	1.0	Y	0	0	
610	LP002979	4106	0.0	40.000000	180.0	1.0	Y	1	1	
611	LP002983	8072	240.0	253.000000	360.0	1.0	Y	1	1	
612	LP002984	7583	0.0	187.000000	360.0	1.0	Y	1	1	
613	LP002990	4583	0.0	133.000000	360.0	0.0	N	0	0	

614 rows × 16 columns

In [45]:
```
dataset['Yes']=dataset['Yes'].astype(int)
dataset['1']=dataset['1'].astype(int)
dataset['2']=dataset['2'].astype(int)
dataset['3+']=dataset['3+'].astype(int)
dataset['Semiurban']=dataset['Semiurban'].astype(int)
dataset['Urban']=dataset['Urban'].astype(int)
```

In [47]:
```
dataset
```

Out[47]:

	Loan_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_Status	Male	Yes	G
0	LP001002	5849	0.0	146.412162	360.0	1.0	Y	1	0	
1	LP001003	4583	1508.0	128.000000	360.0	1.0	N	1	1	
2	LP001005	3000	0.0	66.000000	360.0	1.0	Y	1	1	
3	LP001006	2583	2358.0	120.000000	360.0	1.0	Y	1	1	
4	LP001008	6000	0.0	141.000000	360.0	1.0	Y	1	0	
...	...	...	...	...	...	...	...	...	...	
609	LP002978	2900	0.0	71.000000	360.0	1.0	Y	0	0	
610	LP002979	4106	0.0	40.000000	180.0	1.0	Y	1	1	
611	LP002983	8072	240.0	253.000000	360.0	1.0	Y	1	1	
612	LP002984	7583	0.0	187.000000	360.0	1.0	Y	1	1	
613	LP002990	4583	0.0	133.000000	360.0	0.0	N	0	0	

614 rows × 16 columns

In [49]:
```
loan_status=pd.get_dummies(dataset['Loan_Status'],drop_first=True)
dataset.drop(['Loan_Status'],axis=1,inplace=True)
```

In [51]:
```
dataset=pd.concat([dataset,loan_status],axis=1)
```

In [53]:
```
dataset
```

Out[53]:

	Loan_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Male	Yes	Not Graduate	Yes
0	LP001002	5849	0.0	146.412162	360.0	1.0	1	0	0	0
1	LP001003	4583	1508.0	128.000000	360.0	1.0	1	1	0	0
2	LP001005	3000	0.0	66.000000	360.0	1.0	1	1	0	1
3	LP001006	2583	2358.0	120.000000	360.0	1.0	1	1	1	0
4	LP001008	6000	0.0	141.000000	360.0	1.0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	2900	0.0	71.000000	360.0	1.0	0	0	0	0
610	LP002979	4106	0.0	40.000000	180.0	1.0	1	1	0	0
611	LP002983	8072	240.0	253.000000	360.0	1.0	1	1	0	0
612	LP002984	7583	0.0	187.000000	360.0	1.0	1	1	0	0
613	LP002990	4583	0.0	133.000000	360.0	0.0	0	0	0	1

614 rows × 16 columns

In [55]:
```
dataset['Y']=dataset['Y'].astype(int)
```

In [57]:
```
dataset
```

Out[57]:

	Loan_ID	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Male	Yes	Not Graduate	Yes
0	LP001002	5849	0.0	146.412162	360.0	1.0	1	0	0	0
1	LP001003	4583	1508.0	128.000000	360.0	1.0	1	1	0	0
2	LP001005	3000	0.0	66.000000	360.0	1.0	1	1	0	1
3	LP001006	2583	2358.0	120.000000	360.0	1.0	1	1	1	0
4	LP001008	6000	0.0	141.000000	360.0	1.0	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	2900	0.0	71.000000	360.0	1.0	0	0	0	0
610	LP002979	4106	0.0	40.000000	180.0	1.0	1	1	0	0
611	LP002983	8072	240.0	253.000000	360.0	1.0	1	1	0	0
612	LP002984	7583	0.0	187.000000	360.0	1.0	1	1	0	0
613	LP002990	4583	0.0	133.000000	360.0	0.0	0	0	0	1

614 rows × 16 columns

In [59]:
```
dataset.drop('Loan_ID',axis=1,inplace=True)
```

In [61]:
```
dataset
```

Out[61]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Male	Yes	Not Graduate	Yes	1	2	3+
0	5849	0.0	146.412162	360.0	1.0	1	0	0	0	0	0	0
1	4583	1508.0	128.000000	360.0	1.0	1	1	0	0	1	0	0
2	3000	0.0	66.000000	360.0	1.0	1	1	0	1	0	0	0
3	2583	2358.0	120.000000	360.0	1.0	1	1	1	0	0	0	0
4	6000	0.0	141.000000	360.0	1.0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
609	2900	0.0	71.000000	360.0	1.0	0	0	0	0	0	0	0
610	4106	0.0	40.000000	180.0	1.0	1	1	0	0	0	0	1
611	8072	240.0	253.000000	360.0	1.0	1	1	0	0	1	0	0
612	7583	0.0	187.000000	360.0	1.0	1	1	0	0	0	1	0
613	4583	0.0	133.000000	360.0	0.0	0	0	0	1	0	0	0

614 rows × 15 columns

In [67]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,classification_report,roc_auc_score
from imblearn.over_sampling import SMOTE
```

In [69]:
```
X=dataset.drop('Y',axis=1)
y=dataset['Y']
```

In [71]:
```
print(X.shape)
print(X.isnull().sum())
```

```
(614, 14)
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Male                 0
Yes                  0
Not Graduate         0
Yes                  0
1                    0
2                    0
3+                   0
Semiurban            0
Urban                0
dtype: int64
```

In [73]:
```
print(y.shape)
print(y.isnull().sum())
```

```
(614,)
0
```

In [75]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=101)
```

In [77]:
```
smote=SMOTE(random_state=42)
```

In [79]:
```
X_train_balanced,y_train_balanced=smote.fit_resample(X_train,y_train)
```

In [81]:
```
model=LogisticRegression(class_weight='balanced',max_iter=1000)
```

In [83]:
```
model.fit(X_train_balanced,y_train_balanced)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs fail
ed 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
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[83]:
```
LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000)
```

In [85]:
```
y_pred=model.predict(X_test)
```

In [87]:
```
accuracy=accuracy_score(y_test,y_pred)
```

In [89]:
```
report=classification_report(y_test,y_pred)
```

In [91]:
```
print("ACCURACY:",accuracy)
print("CLASSIFICATION:\n",report)
print("ROC-AUC Score",roc_auc_score(y_test,model.predict_proba(X_test)[:,1]))
print("LOAN STATUS:",y_pred)
```

```
ACCURACY: 0.772972972972973
CLASSIFICATION:
              precision    recall  f1-score   support

           0       0.73      0.55      0.62        64
           1       0.79      0.89      0.84       121

    accuracy                           0.77       185
   macro avg       0.76      0.72      0.73       185
weighted avg       0.77      0.77      0.76       185

ROC-AUC Score 0.750129132231405
LOAN STATUS: [1 1 1 0 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 1 0 0 0 1 0 1 1
 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 1 1 1 0 1 1 0
 0 1 0 1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 0 1 1 1
 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 0 1]
```

In [ ]: