LOAN APPLICATION STATUS PREDICTION PROJECT REPORT

SUBMITTED BY

NEETAL TIWARI

ABSTRACT

Bank are making major part of profits through loans. Though lot of people are applying for loans. It's hard to select the genuine applicant, who will repay the loan. While doing the process manually, lot of misconception may happen to select the genuine applicant. Therefore, we are developing loan prediction system using machine learning, so the system automatically selects the eligible candidate. This is helpful to both bank staff applicant. The time period for the sanction of loan will be drastically reduced. In this report we are predicting the loan data by using some machine learning algorithms.

PROBLAM STATEMENT

A loan is the core business part of banks. The main portion the bank's profit is directly come from the profit earned from the loans. Though bank approves loan after a regress process of verification and testimonial but still there's no surety whether the chosen hopeful is the right hopeful or not. This process takes fresh time while doing it manually. We can prophesy whether that particular hopeful is safe or not and the whole process of testimonial is automated by machine literacy style. Loan prognostic is really helpful for retainer of banks as well as for the hopeful also.

Bank employees check the details of applicant manually and give the loan to eligible applicant. Checking the details of all applicants takes lot of time and efforts. There are chances of human error may occur due checking all details manually. There is possibility of assigning loan to ineligible applicant.

To deal with the problem, we developed automatic loan prediction using machine learning techniques. We will train the machine with previous dataset, so machine can analyse and understand the process. Then machine will check for eligible applicant and give us result.

DATA COLLECTION

To predict loan status first process is load the important modules and then load the data for analyse and prediction.

LOAN APPLICATION STATUS PREDICTION

LOADING THE MODULES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Loading the Dataset

```
las=pd.read_csv("loan_application_status.csv")
```

```
las
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	L
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	

The Dataset of loan application status have 614 rows and 13 attributes… such as

1) Loan_ID – Unique loan id
2) Gender – Male/Female
3) Married – Applicant is married or not
4) Dependents – Numbers of dependents
5) Education – Applicant education (graduate or under graduate)
6) Self_Employed – Self-employed (yes or not)
7) ApplicantIncome – Applicant Income

8) CoapplicantionIncome – Co application Income

9) LoanAmount – Loan amount in thousands

10) Loan_Amount_term – Term of loan in months

11) Credit_History – Credit history meets guidelines

12) Property area – Urban/semi/rural

13) Loan_status – Loan approved (yes or not)

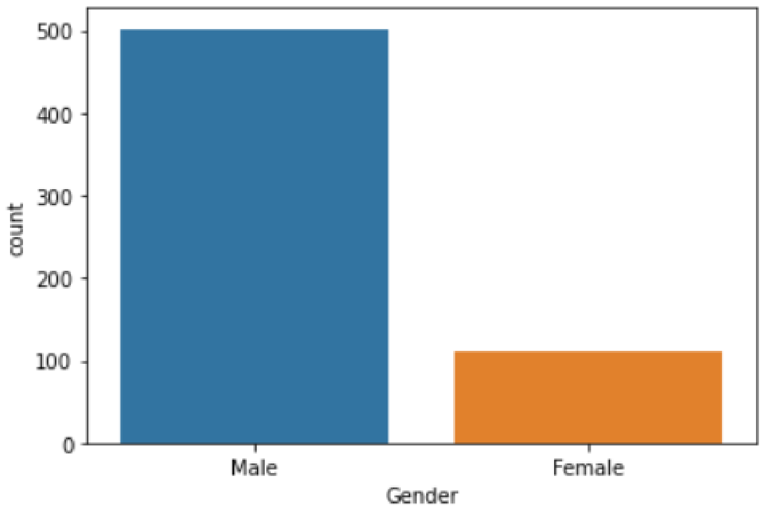
DATA ANALYSIS

To predict the loan status by using machine learning we have to analysis the dataset first, so we can analysis that which attribute affect the target or label.

To analysis the Dataset we use matplotlib and seaborn models. Matplotlib and seaborn have lots of plots so that we can see dataset by graphically…

```
In [50]: sns.countplot("Gender",data=las)

Out[50]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



Male are higher than Female.

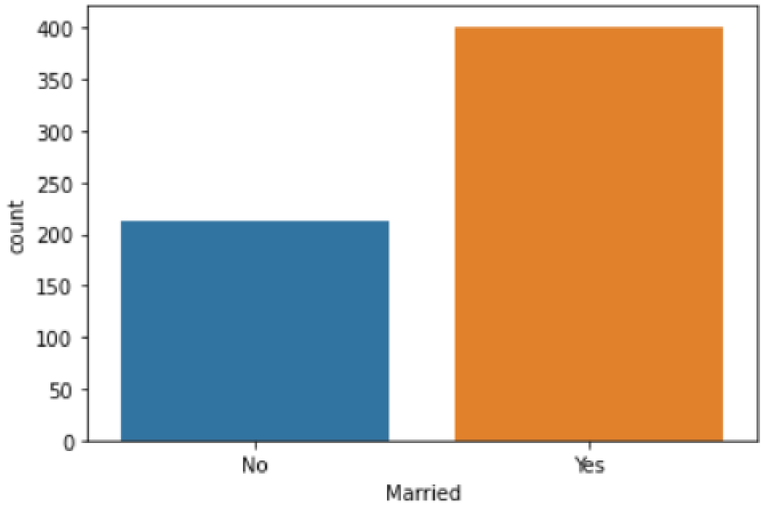
Here we can see by graph that out of 614 applicants, 502 applicants are Male and only 112 applicants are Female. So, we can say easily that Male applicants are higher than Female applicants.

In [51]:
```
las["Married"].value_counts().sort_values(ascending=True)
```

Out[51]:
```
No     213
Yes    401
Name: Married, dtype: int64
```

In [52]:
```
sns.countplot("Married",data=las)
```

Out[52]:
```
<AxesSubplot:xlabel='Married', ylabel='count'>
```



Observation;

out of 614 people 401 people are Married,it means Married peoples are higher.

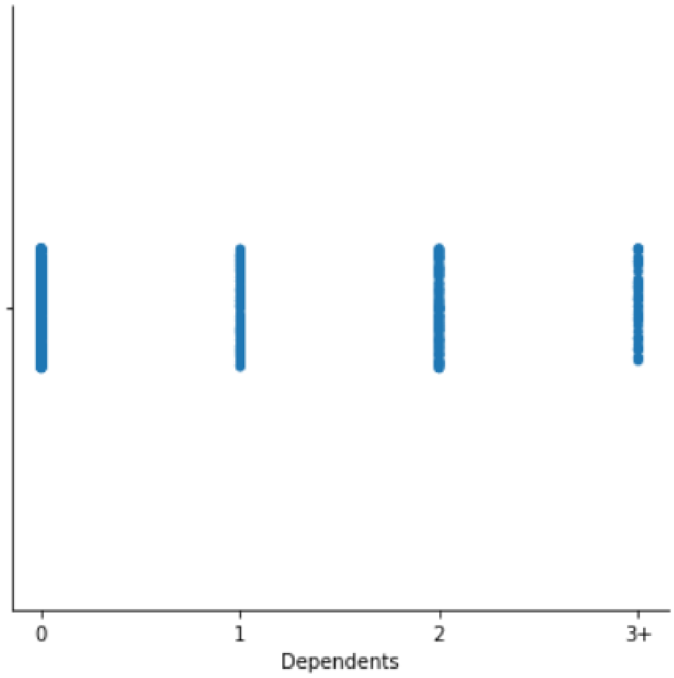
We can analysis by graph that Married applicants are higher than Unmarried.

In [53]:
```
las["Dependents"].value_counts().sort_values(ascending=False)
```

Out[53]:
```
0     360
1     102
2     101
3+     51
Name: Dependents, dtype: int64
```

In [54]:
```
sns.catplot("Dependents",data=las)
```

Out[54]:
```
<seaborn.axisgrid.FacetGrid at 0x1c02ea46af0>
```



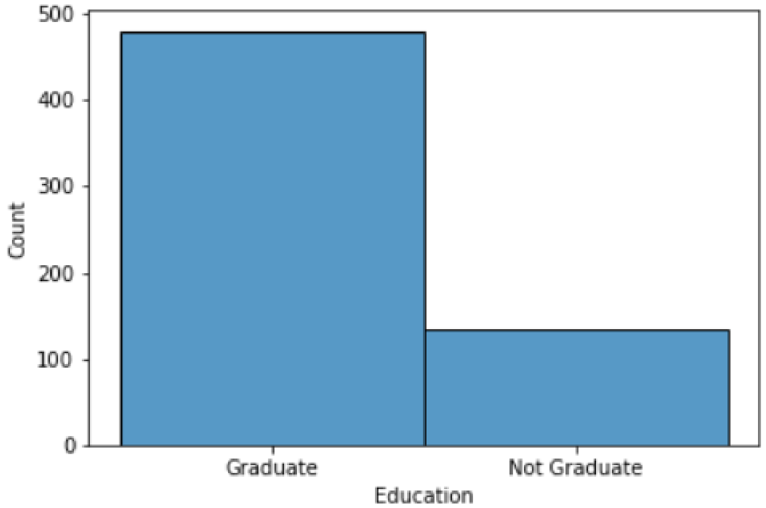
Out of 614 applicants 360 applicants have 0 dependents mean there is no dependent for 360 applicants, 102 applicants have only 1 dependent, 101 applicants have 2 dependents and 51 applicants have more than 3 dependents.

In [56]:
```
las["Education"].value_counts().sort_values(ascending=False)
```

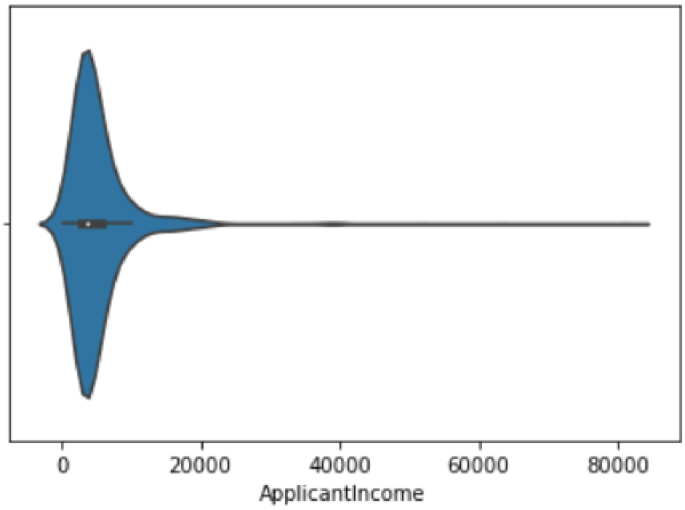
Out[56]:
```
Graduate        480
Not Graduate    134
Name: Education, dtype: int64
```

In [57]:
```
sns.histplot(las["Education"])
```

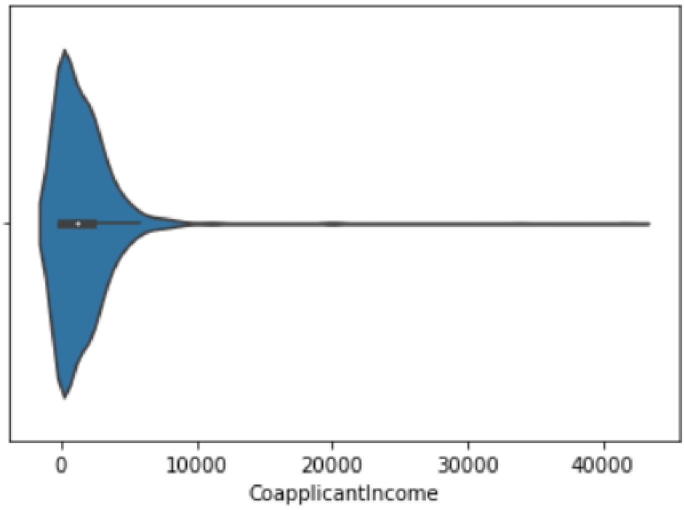
Out[57]:
```
<AxesSubplot:xlabel='Education', ylabel='Count'>
```



480 applicants are Graduate and only 134 applicants are not graduate or undergraduate.

In [62]:
```
sns.violinplot("ApplicantIncome",data=las)
```
Out[62]:
```
<AxesSubplot:xlabel='ApplicantIncome'>
```


Applicants Income is higher in between 2500 to 7000.

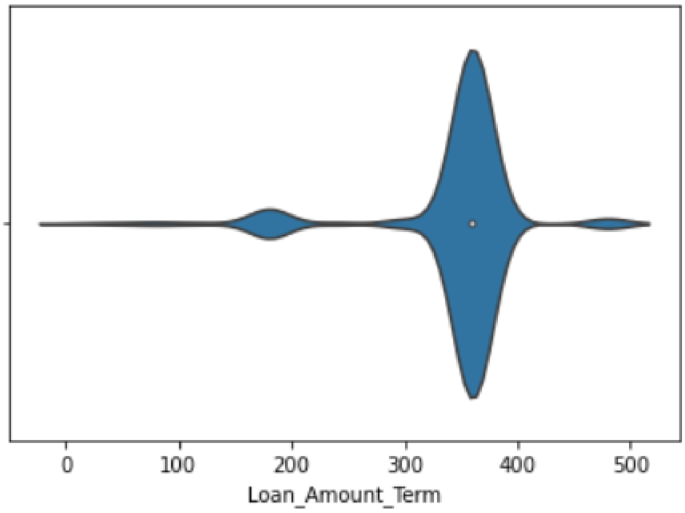
In [66]:
```
sns.violinplot("CoapplicantIncome",data=las)
```
Out[66]:
```
<AxesSubplot:xlabel='CoapplicantIncome'>
```


Co-applicants Income is higher in between 0 to 5000.

```
----        -
36.0        2
12.0        1
Name: Loan_Amount_Term, dtype: int64
```

In [68]:
```
sns.violinplot("Loan_Amount_Term",data=las)
```

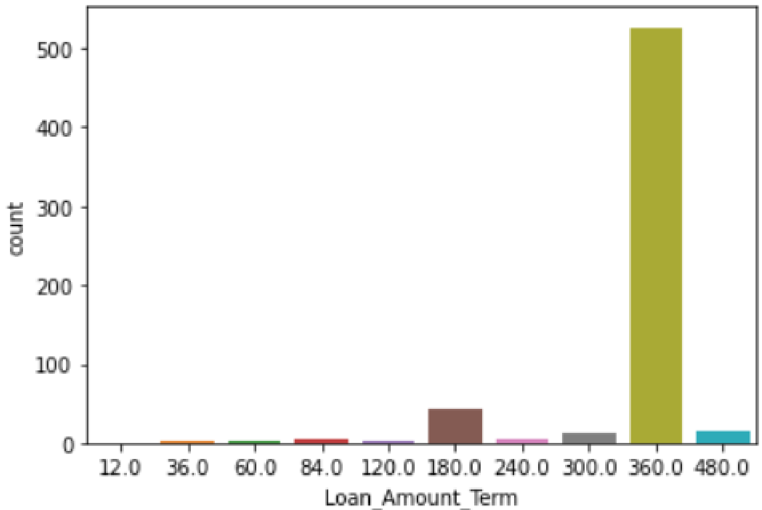
Out[68]: <AxesSubplot:xlabel='Loan_Amount_Term'>



---

n [69]:
```
sns.countplot("Loan_Amount_Term",data=las)
```

ut[69]: <AxesSubplot:xlabel='Loan_Amount_Term', ylabel='count'>

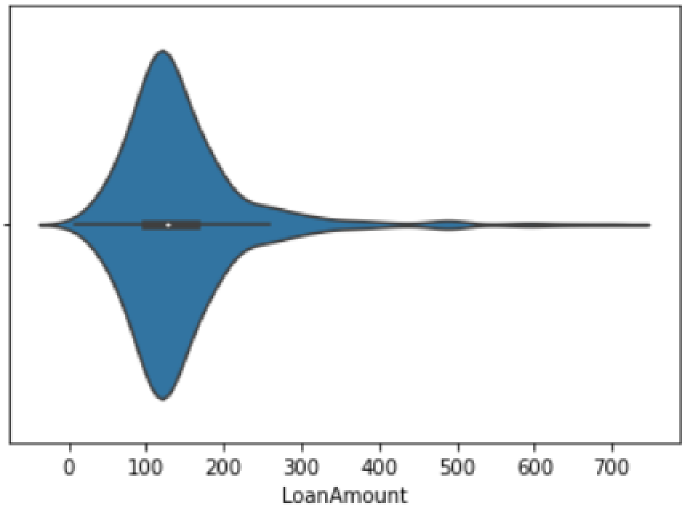


Loan_Amount_Term is highest in 360.

n [72]:

```
sns.violinplot("LoanAmount",data=las)
```

ut[72]: <AxesSubplot:xlabel='LoanAmount'>

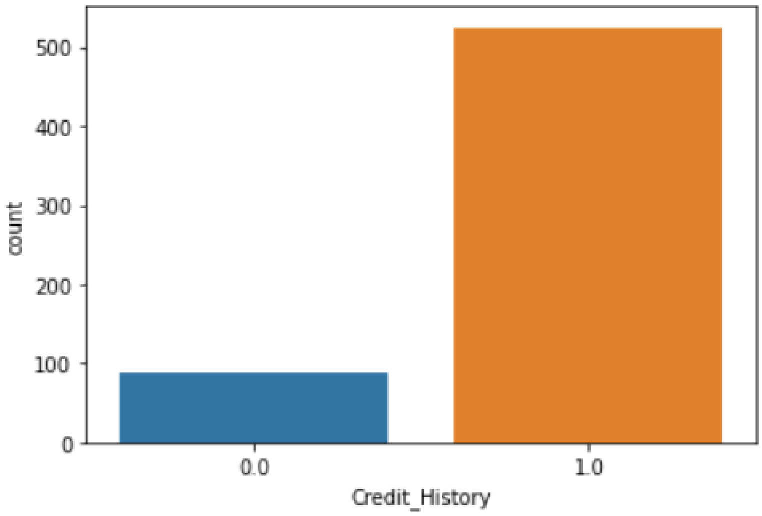


LoanAmount data is higher in between 50 to 200.

In [74]:

```
sns.countplot("Credit_History",data=las)
```

Out[74]: <AxesSubplot:xlabel='Credit_History', ylabel='count'>



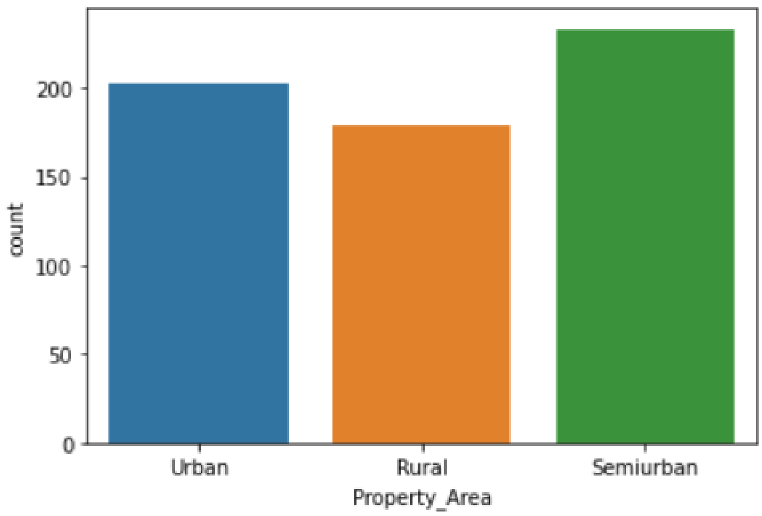
1 type of credit history is higher than 0, so we can analysis that credit history no is higher than yes.

In [77]:
```
las["Property_Area"].value_counts().sort_values(ascending=False)
```

Out[77]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

In [78]:
```
sns.countplot("Property_Area",data=las)
```

Out[78]:
```
<AxesSubplot:xlabel='Property_Area', ylabel='count'>
```



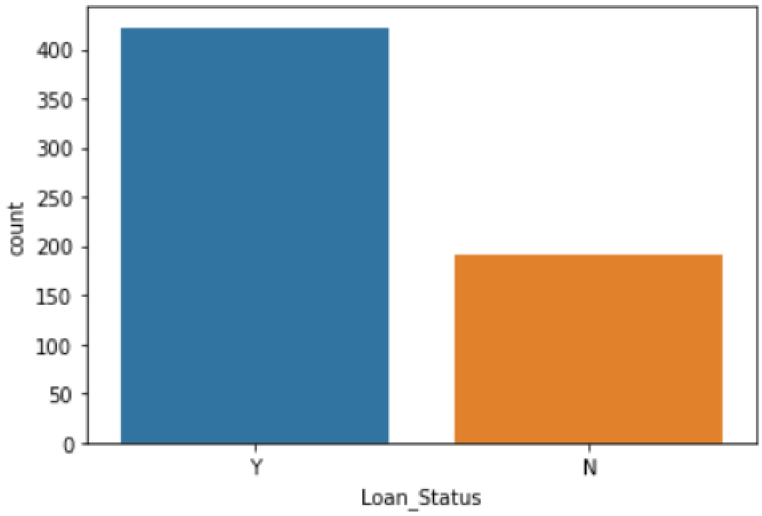
We can see that all type of Property_Area data are almost same.

In [80]:
```
las["Loan_Status"].value_counts().sort_values(ascending=False)
```

Out[80]:
```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [81]:
```
sns.countplot("Loan_Status",data=las)
```

Out[81]:
```
<AxesSubplot:xlabel='Loan_Status', ylabel='count'>
```



Out of 614 applicants, 422 applicants get the loan and out of 614 applicants, only 192 applicants do not get the loan. So, we can say that Loan status yes is higher than no.

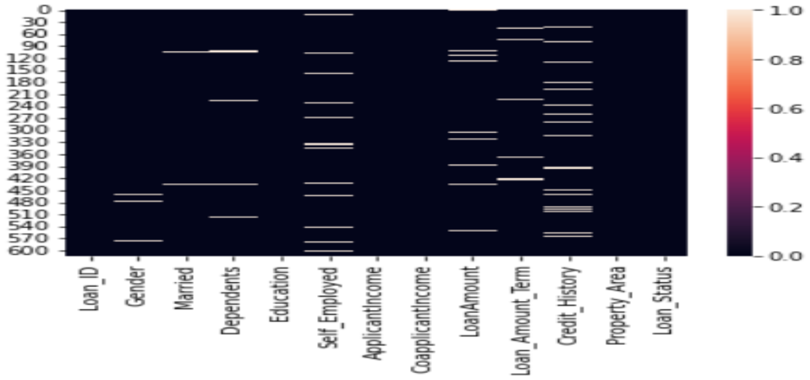
DATA PREPROCESSING

The collect data may contain missing values that may lead to inconsistency. To gain better results data need to be pre-processed and so it'll better the effectiveness of the algorithm and we should remove the outliers.

Checking the null values--

In [10]:
```
sns.heatmap(las.isnull())
```

Out[10]: <AxesSubplot:>



Blanks in Heatmap it means null values are there

In [11]:
```
las.isnull().sum()
```

Out[11]:
```
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
```

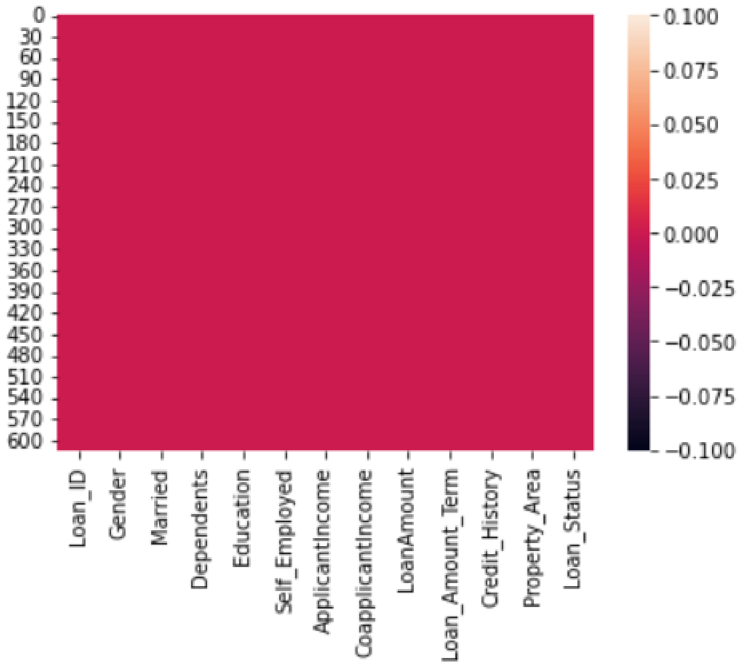
There are 13 null values in Gender,3 null values in Married,15 null values in Dependents,32 null values in Self_Employed,22 null values in LoanAmount,14 null values in Loan_Amount_term and 50 null values in Credit_History.

We can use fillna method to fill the null values.

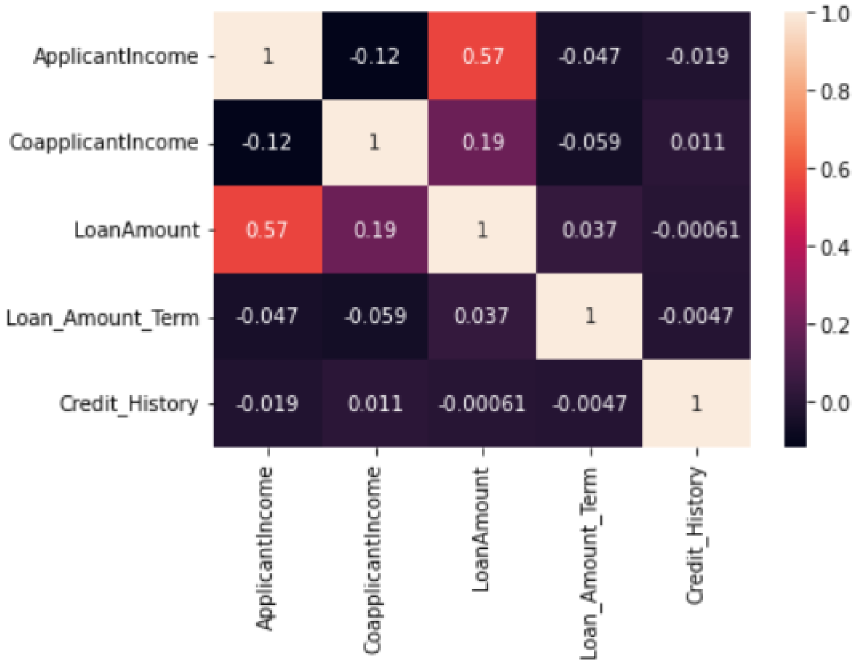
In [40]:

```
sns.heatmap(las.isnull())
```

Out[40]: <AxesSubplot:>



After using fillna method Now there is no null values in Loan application status dataset.

In [42]:
```
las.corr().sum().sort_values(ascending=False)
```
Out[42]:
```
LoanAmount           1.790751
ApplicantIncome      1.383429
CoapplicantIncome    1.024364
Credit_History       0.987206
Loan_Amount_Term     0.926341
dtype: float64
```
In [43]:
```
sns.heatmap(las.corr(),annot=True)
```
Out[43]:
```
<AxesSubplot:>
```


We can see by using heatmap that which of the attribute is high correlated with the y label or target. So, we can see that LoanAmount and ApplicantIncome is high correlated.

After that we should remove the skewness and outlier so that data can be clean for the prediction.

Checking The Skewness

In [105…

```
las.skew()
```

Out[105…

```
ApplicantIncome        6.539513
CoapplicantIncome      7.491531
LoanAmount             2.743053
Loan_Amount_Term      -2.402112
Credit_History        -2.021971
dtype: float64
```

Obsevartion;

To remove the skewness we can use log transform and power transform techniques….

In [122…

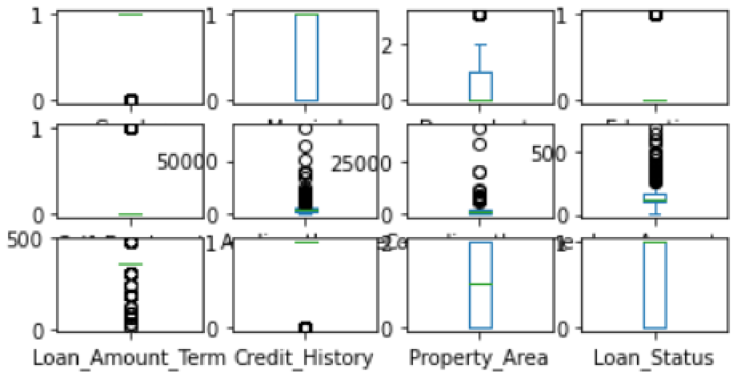
```
x
```

Out[122…

```
Self_Employed         2.159796
Education             1.367622
Dependents            0.441404
Loan_Amount_Term      0.392571
LoanAmount            0.020831
ApplicantIncome      -0.092946
CoapplicantIncome    -0.145646
Property_Area        -0.158267
Married              -0.644850
Gender               -1.648795
Credit_History       -2.021971
dtype: float64
```

Now Skewness has been removed.

```
Property_Area          AxesSubplot(0.529348,0.321957;0.168478›
Loan_Status            AxesSubplot(0.731522,0.321957;0.168478›
dtype: object
```



Outliers are present in loan application dataset.

There are some outliers in loan application status dataset.

In [131…
```
new_las.shape
```
Out[131…
```
(577, 12)
```

In [132…
```
las.shape
```
Out[132…
```
(614, 12)
```

37 rows has been removed. it means outliers has been removed.

Outliers presents in 37 rows…

TRAIN MODEL ON TRAINING DATASET

Now we should train the models on the training dataset and make soothsaying for the test dataset. We can divide our train dataset into two track train and test. We can train the models on this training part and using that make soothsaying for the test part. In this way, we can validate our soothsaying as we have the true soothsaying for the testimony part (which we don't have for the test dataset.)

I used RandomForestCalssifier, DecisionTreeClassifier, KNeighboursClassifier, and SVC for the prediction.

In [156…

```
rfc=RandomForestClassifier()
rfc.fit(x_train,y_train)
rfcpred=rfc.predict(x_test)
print(accuracy_score(y_test,rfcpred)*100)
print(classification_report(y_test,rfcpred))
print(confusion_matrix(y_test,rfcpred))
```

```
78.84615384615384
              precision    recall  f1-score   support

           0       0.71      0.41      0.52        29
           1       0.80      0.93      0.86        75

    accuracy                           0.79       104
   macro avg       0.76      0.67      0.69       104
weighted avg       0.78      0.79      0.77       104

[[12 17]
 [ 5 70]]
```

The accuracy of the RandomForestClassifier is 78.85%

In [158…

```
knc=KNeighborsClassifier()
knc.fit(x_train,y_train)
kncpred=knc.predict(x_test)
print(accuracy_score(y_test,kncpred)*100)
print(classification_report(y_test,kncpred))
print(confusion_matrix(y_test,kncpred))
```

```
65.38461538461539
              precision    recall  f1-score   support

           0       0.29      0.17      0.22        29
           1       0.72      0.84      0.78        75

    accuracy                           0.65       104
   macro avg       0.51      0.51      0.50       104
weighted avg       0.60      0.65      0.62       104

[[ 5 24]
 [12 63]]
```

The accuracy of the KNeighborsClassifier is 65.38%.

In [159…

```
svc=SVC()
svc.fit(x_train,y_train)
svcpred=svc.predict(x_test)
print(accuracy_score(y_test,svcpred)*100)
print(classification_report(y_test,svcpred))
print(confusion_matrix(y_test,svcpred))
```

```
72.11538461538461
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        29
           1       0.72      1.00      0.84        75

    accuracy                           0.72       104
   macro avg       0.36      0.50      0.42       104
weighted avg       0.52      0.72      0.60       104

[[ 0 29]
 [ 0 75]]
```

The accuracy of the SVC is 72.12%.

```
In [178…
dtc=DecisionTreeClassifier(criterion="gini",max_leaf_nodes=None,min_impurity_decrease=0.1,splitter="best"
dtc.fit(x_train,y_train)
dtcpred=dtc.predict(x_test)
print(accuracy_score(y_test,dtcpred)*100)
print(classification_report(y_test,dtcpred))
print(confusion_matrix(y_test,dtcpred))
```

```
80.76923076923077
              precision    recall  f1-score   support

           0       0.85      0.38      0.52        29
           1       0.80      0.97      0.88        75

    accuracy                           0.81       104
   macro avg       0.82      0.68      0.70       104
weighted avg       0.81      0.81      0.78       104

[[11 18]
 [ 2 73]]
```

The accuracy of the DecisionTreeClassifier is 80.77%

By using all the algorithms DecisionTreeClassifier is working well.

<u>CONCLUSION</u>

From a proper analysis of positive points and constraints on the member, it can be safely concluded that the product is a considerably productive member. This use is working duly and meeting to all Banker requisites. This member can be freely plugged in numerous other systems. There have been mathematics cases of computer glitches, violations in content and most important weight of features is fixed in automated prophecy system, so in the near future the software could be made more secure, trustworthy and dynamic weight conformation. In near future this module of prophecy can be integrated with the module of automated processing system. The system is trained on old training dataset in future software can be made resembling that new testing data should also take part in training data after some fix time.