Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load Data

In [2]:
```
data=pd.read_csv(r"C:\Users\Hp\Downloads\Loan Status Prediction.csv")
```

Understanding the data

In [3]:
```
data.shape
```

Out[3]:
```
(381, 13)
```

In [4]:
```
data
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
0	LP001003	Male	Yes	1	Graduate	No	4583	
1	LP001005	Male	Yes	0	Graduate	Yes	3000	
2	LP001006	Male	Yes	0	Not Graduate	No	2583	
3	LP001008	Male	No	0	Graduate	No	6000	
4	LP001013	Male	Yes	0	Not Graduate	No	2333	
...	...	...	...	...	...	...	...	
376	LP002953	Male	Yes	3+	Graduate	No	5703	
377	LP002974	Male	Yes	0	Graduate	No	3232	
378	LP002978	Female	No	0	Graduate	No	2900	
379	LP002979	Male	Yes	3+	Graduate	No	4106	
380	LP002990	Female	No	0	Graduate	Yes	4583	

381 rows × 13 columns

In [5]:
```
data.head()
```

Out[5]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
0	LP001003	Male	Yes	1	Graduate	No	4583	
1	LP001005	Male	Yes	0	Graduate	Yes	3000	
2	LP001006	Male	Yes	0	Not Graduate	No	2583	
3	LP001008	Male	No	0	Graduate	No	6000	
4	LP001013	Male	Yes	0	Not Graduate	No	2333	

In [6]:
```
data.tail()
```

Out[6]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
376	LP002953	Male	Yes	3+	Graduate	No	5703	
377	LP002974	Male	Yes	0	Graduate	No	3232	
378	LP002978	Female	No	0	Graduate	No	2900	
379	LP002979	Male	Yes	3+	Graduate	No	4106	
380	LP002990	Female	No	0	Graduate	Yes	4583	

In [7]:
```
data.sample(5)
```

Out[7]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	C
63	LP001319	Male	Yes	2	Not Graduate	No	3273	
199	LP002008	Male	Yes	2	Graduate	Yes	5746	
337	LP002732	Male	No	0	Not Graduate	NaN	2550	
92	LP001520	Male	Yes	0	Graduate	No	4860	
53	LP001250	Male	Yes	3+	Not Graduate	No	4755	

In [8]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 381 entries, 0 to 380
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            381 non-null    object 
 1   Gender             376 non-null    object 
 2   Married            381 non-null    object 
 3   Dependents         373 non-null    object 
 4   Education          381 non-null    object 
 5   Self_Employed      360 non-null    object 
 6   ApplicantIncome    381 non-null    int64  
 7   CoapplicantIncome  381 non-null    float64
 8   LoanAmount         381 non-null    float64
 9   Loan_Amount_Term   370 non-null    float64
 10  Credit_History     351 non-null    float64
 11  Property_Area      381 non-null    object 
 12  Loan_Status        381 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 38.8+ KB
```

In [9]:

```
data.describe()
```

Out[9]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	381.000000	381.000000	381.000000	370.000000	351.000000
mean	3579.845144	1277.275381	104.986877	340.864865	0.837607
std	1419.813818	2340.818114	28.358464	68.549257	0.369338
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2600.000000	0.000000	90.000000	360.000000	1.000000
50%	3333.000000	983.000000	110.000000	360.000000	1.000000
75%	4288.000000	2016.000000	127.000000	360.000000	1.000000
max	9703.000000	33837.000000	150.000000	480.000000	1.000000

In [10]:

```
data.describe(include='object')
```

Out[10]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Property_Area
count	381	376	381	373	381	360	381
unique	381	2	2	4	2	2	3
top	LP001003	Male	Yes	0	Graduate	No	Semiurban
freq	1	291	228	234	278	325	149

Treating Null Values

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

Out[11]:
```
Loan_ID               0
Gender                5
Married               0
Dependents            8
Education             0
Self_Employed        21
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     11
Credit_History       30
Property_Area         0
Loan_Status           0
dtype: int64
```

In [12]:
```
data['Gender'].fillna(data['Gender'].mode(),inplace=True)
```

In [13]:
```
data['Dependents'].value_counts()
```

Out[13]:
```
0     234
2      59
1      52
3+     28
Name: Dependents, dtype: int64
```

In [14]:
```
data['Dependents'].fillna(data['Dependents'].mode(),inplace=True)
```

In [15]:
```
data['Self_Employed'].value_counts()
```

Out[15]:
```
No     325
Yes     35
Name: Self_Employed, dtype: int64
```

In [16]:
```
data['Self_Employed'].fillna(data['Self_Employed'].mode(),inplace=True)
```

In [17]:
```
data['Loan_Amount_Term'].value_counts()
```

Out[17]:
```
360.0    312
180.0     29
480.0     11
300.0      7
120.0      3
84.0       3
240.0      2
60.0       1
12.0       1
36.0       1
Name: Loan_Amount_Term, dtype: int64
```

In [18]:
```
data['Loan_Amount_Term'].skew()
```

Out[18]:
```
-2.2049305975078237
```

In [19]:
```
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].median(),inplace=T
```

In [20]:
```
data['Credit_History'].value_counts()
```

Out[20]:
```
1.0    294
0.0     57
Name: Credit_History, dtype: int64
```

In [21]:
```
data['Credit_History'].fillna(data['Credit_History'].median(),inplace=True)
```

In [22]:
```
data['Credit_History'].values
```

Out[22]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0.,
       1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1.,
       1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0.,
       0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1.,
       1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0.,
       1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.,
       1., 0., 1., 1., 1., 1., 0.])
```

In [23]:
```
data.isnull().sum()
```

Out[23]:
```
Loan_ID               0
Gender                5
Married               0
Dependents            8
Education             0
Self_Employed        21
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term      0
Credit_History        0
Property_Area         0
Loan_Status           0
dtype: int64
```

In [24]:
```
data['Credit_History'].values
```

Out[24]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0.,
       1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 1.,
       0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
       0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1.,
       1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0., 0.,
       1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 0.,
       0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 0., 1.,
       1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
       1., 0., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0.,
       1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.,
       1., 0., 1., 1., 1., 1., 0.])
```

In [25]:
```
data['Gender'].value_counts()
```

Out[25]:
```
Male      291
Female     85
Name: Gender, dtype: int64
```

In [26]:
```
data['Gender'].fillna('Male',inplace=True)
```

In [27]:
```
data['Dependents'].value_counts()
```

Out[27]:
```
0     234
2      59
1      52
3+     28
Name: Dependents, dtype: int64
```

In [28]:
```
data['Dependents'].fillna('0',inplace=True)
```

In [29]:
```
data['Self_Employed'].value_counts()
```

Out[29]:
```
No     325
Yes     35
Name: Self_Employed, dtype: int64
```

In [30]:
```
data['Self_Employed'].fillna('No',inplace=True)
```

In [31]:
```
data.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 [32]:
```
data.shape
```

Out[32]:
```
(381, 13)
```

In [33]:
```
data.skew()
```

```
C:\Users\Hp\AppData\Local\Temp\ipykernel_29688\1188251951.py:1: FutureWarn
ing: The default value of numeric_only in DataFrame.skew is deprecated. In
a future version, it will default to False. In addition, specifying 'numer
ic_only=None' is deprecated. Select only valid columns or specify the valu
e of numeric_only to silence this warning.
  data.skew()
```

Out[33]:
```
ApplicantIncome      1.119751
CoapplicantIncome    8.660692
LoanAmount          -0.804282
Loan_Amount_Term    -2.253633
Credit_History      -1.972497
dtype: float64
```

In [34]:
```
data.columns
```

Out[34]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
s'],
      dtype='object')
```

TREATING OUTLIERS

In [35]:
```
numcol=data[['ApplicantIncome', 'CoapplicantIncome', 'LoanAmount','Loan_Amo
```

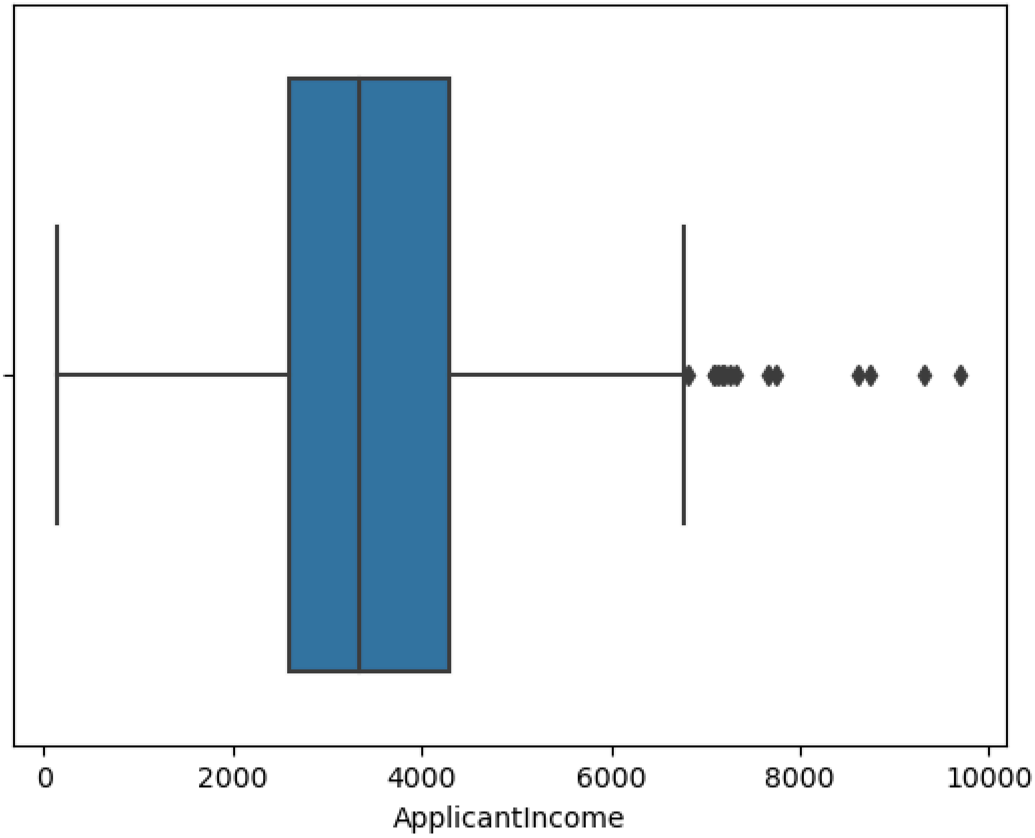
In [36]:
```
numcol
```

Out[36]:

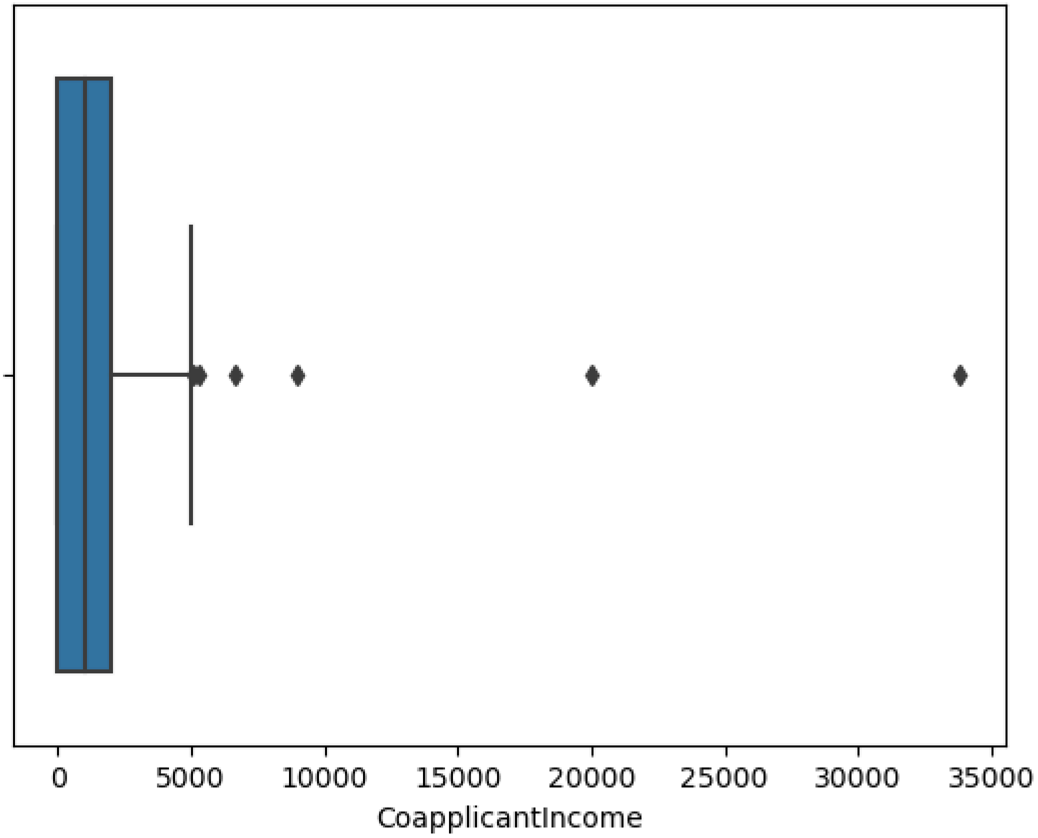
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	4583	1508.0	128.0	360.0	1.0
1	3000	0.0	66.0	360.0	1.0
2	2583	2358.0	120.0	360.0	1.0
3	6000	0.0	141.0	360.0	1.0
4	2333	1516.0	95.0	360.0	1.0
...	...	...	...	...	...
376	5703	0.0	128.0	360.0	1.0
377	3232	1950.0	108.0	360.0	1.0
378	2900	0.0	71.0	360.0	1.0
379	4106	0.0	40.0	180.0	1.0
380	4583	0.0	133.0	360.0	0.0

381 rows × 5 columns

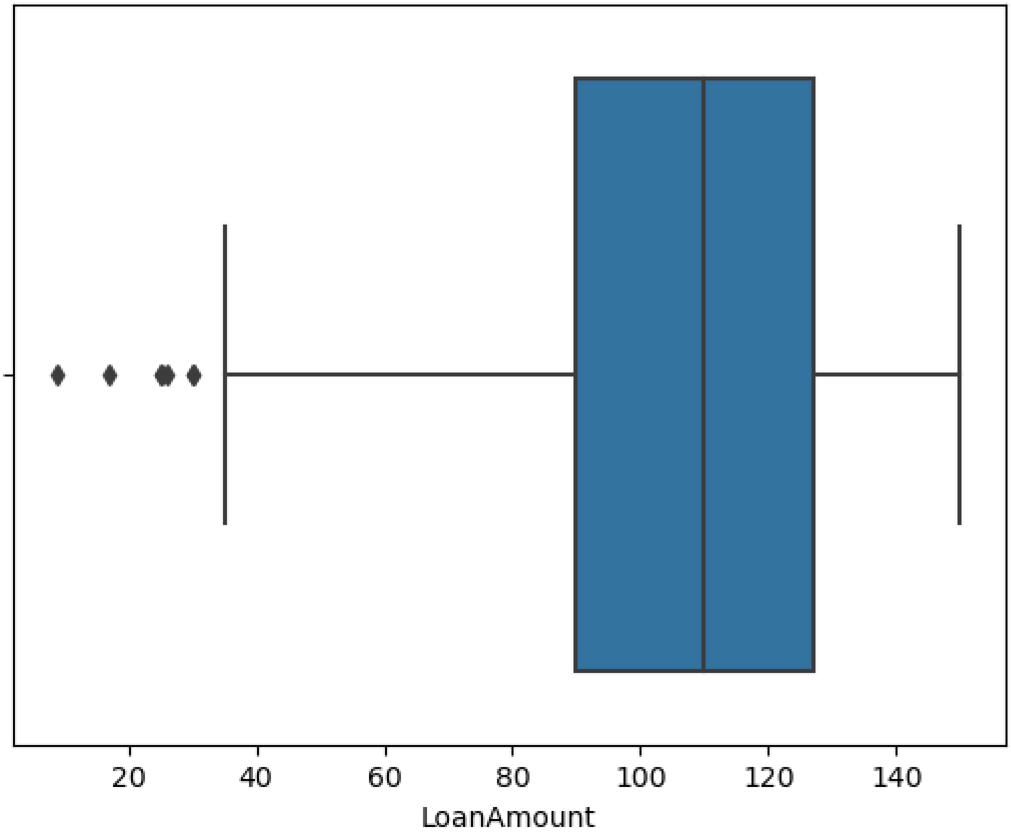
In [37]:
```
sns.boxplot(data=data, x=data['ApplicantIncome'])
plt.show()
```



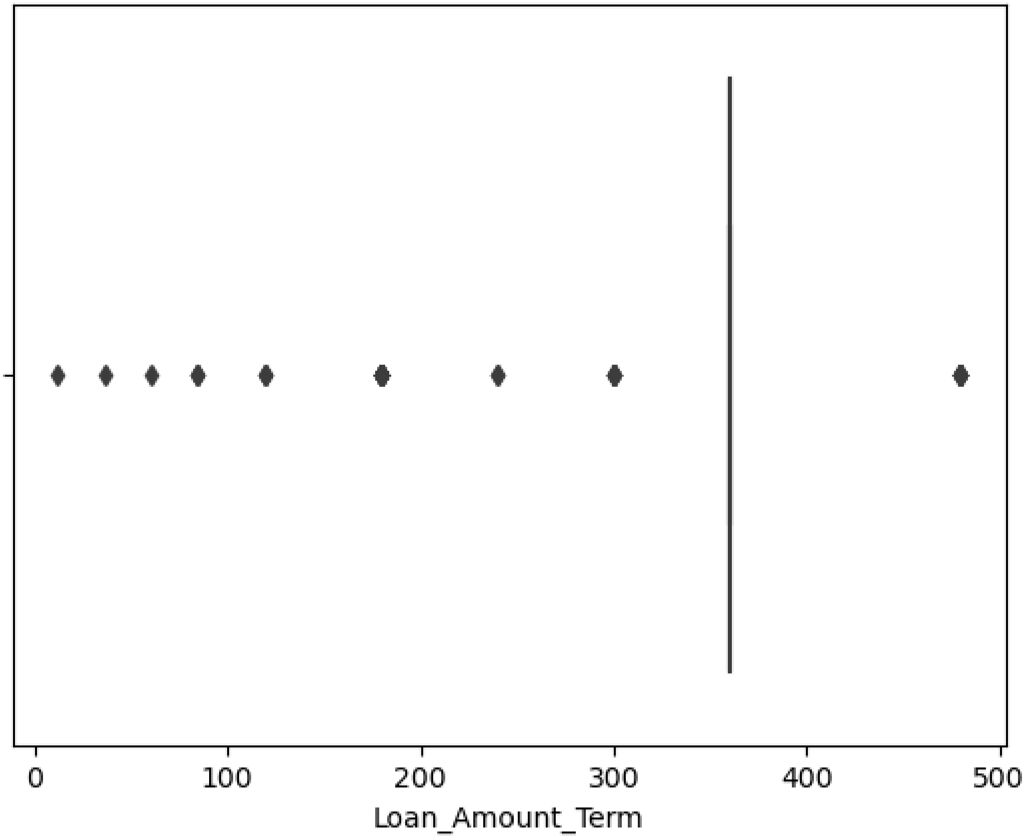
In [38]:
```
sns.boxplot(data=data, x=data['CoapplicantIncome'])
plt.show()
```



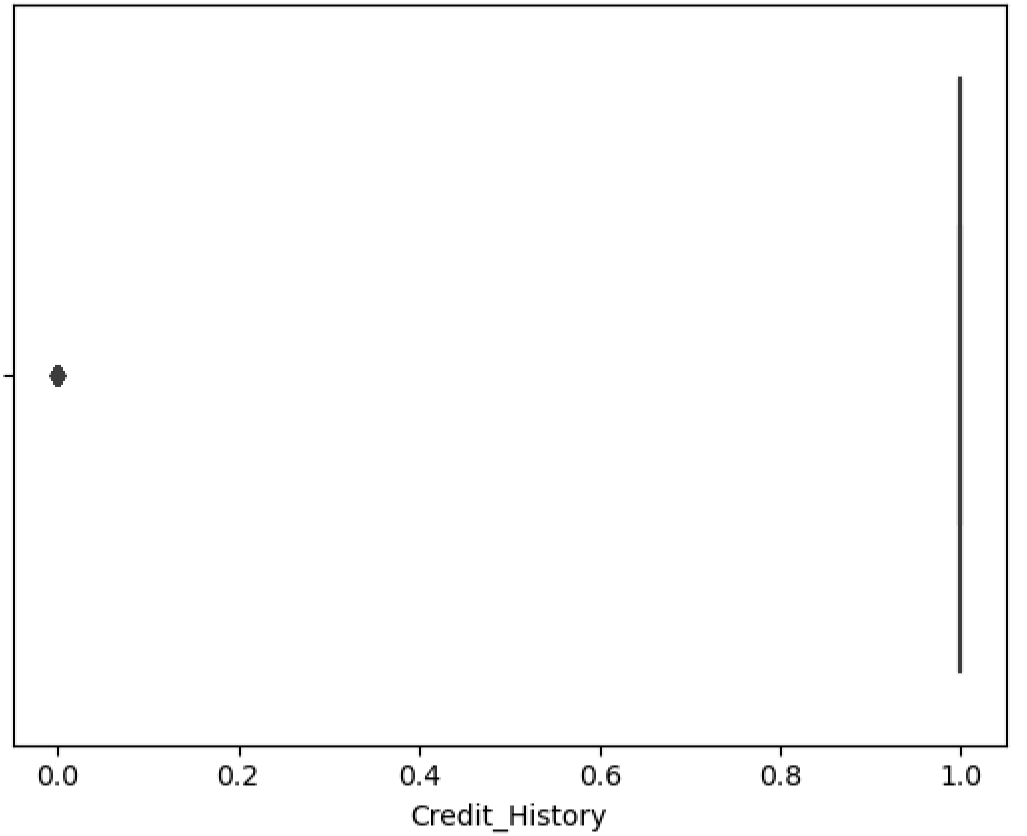
In [39]:
```
sns.boxplot(data=data, x=data['LoanAmount'])
plt.show()
```



In [40]:
```
sns.boxplot(data=data, x=data['Loan_Amount_Term'])
plt.show()
```



In [41]:
```
sns.boxplot(data=data, x=data['Credit_History'])
plt.show()
```



In [42]:

```
def treatoutlier2(col):
    Q1=data[col].quantile(0.25)
    Q3=data[col].quantile(0.75)
    IQR=Q3-Q1
    UL=Q3+1.5*IQR
    LL=Q1-1.5*IQR
    upperoutlier=data[col]>UL
    loweroutlier=data[col]<LL
    median=data[col].median()
    data.loc[upperoutlier,col]=median
    data.loc[loweroutlier,col]=median
    return data
```

In [43]:

```
for i in data.select_dtypes(include=['int', 'float']):
    treatoutlier2(i)
```

In [44]:

```
data.kurt()
```

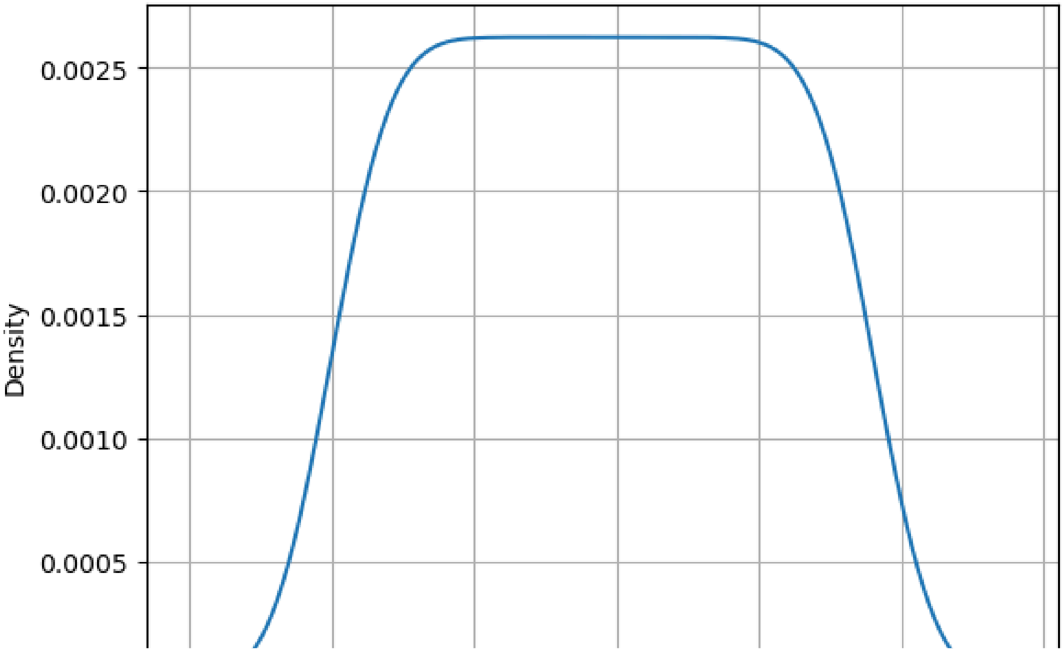
```
C:\Users\Hp\AppData\Local\Temp\ipykernel_29688\2907027414.py:1: FutureWarn
ing: The default value of numeric_only in DataFrame.kurt is deprecated. In
a future version, it will default to False. In addition, specifying 'numer
ic_only=None' is deprecated. Select only valid columns or specify the valu
e of numeric_only to silence this warning.
  data.kurt()
```

Out[44]:

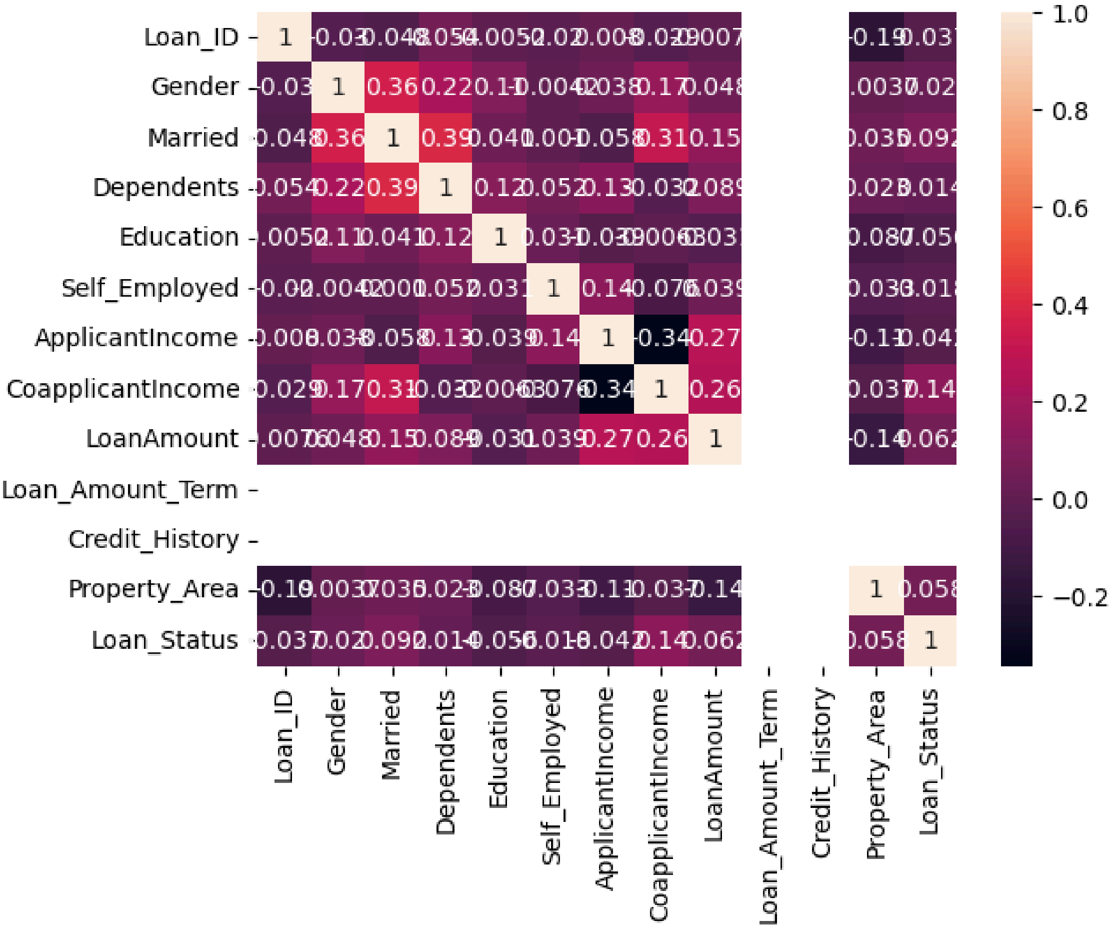
```
ApplicantIncome      0.241663
CoapplicantIncome   -0.172952
LoanAmount          -0.230708
Loan_Amount_Term     0.000000
Credit_History       0.000000
dtype: float64
```

In [ ]:

In [60]:
```
for i in data:
    sns.kdeplot(data=data,x=i)
    plt.grid()
    plt.show()
```

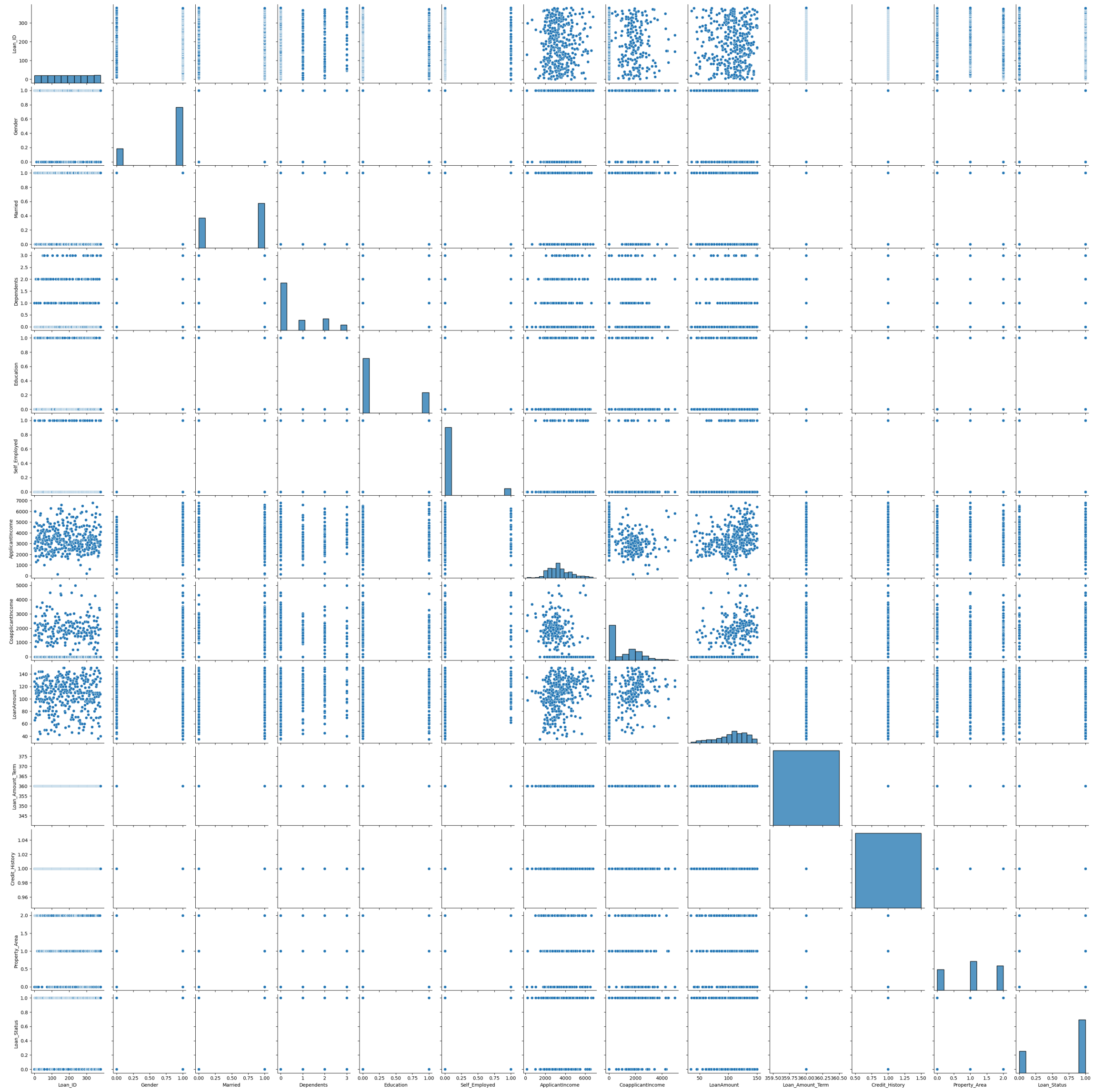


In [61]:
```
sns.heatmap(data.corr(),annot=True)
plt.show()
```



In [72]:
```
sns.pairplot(data=data)

plt.show()
```



In [70]:

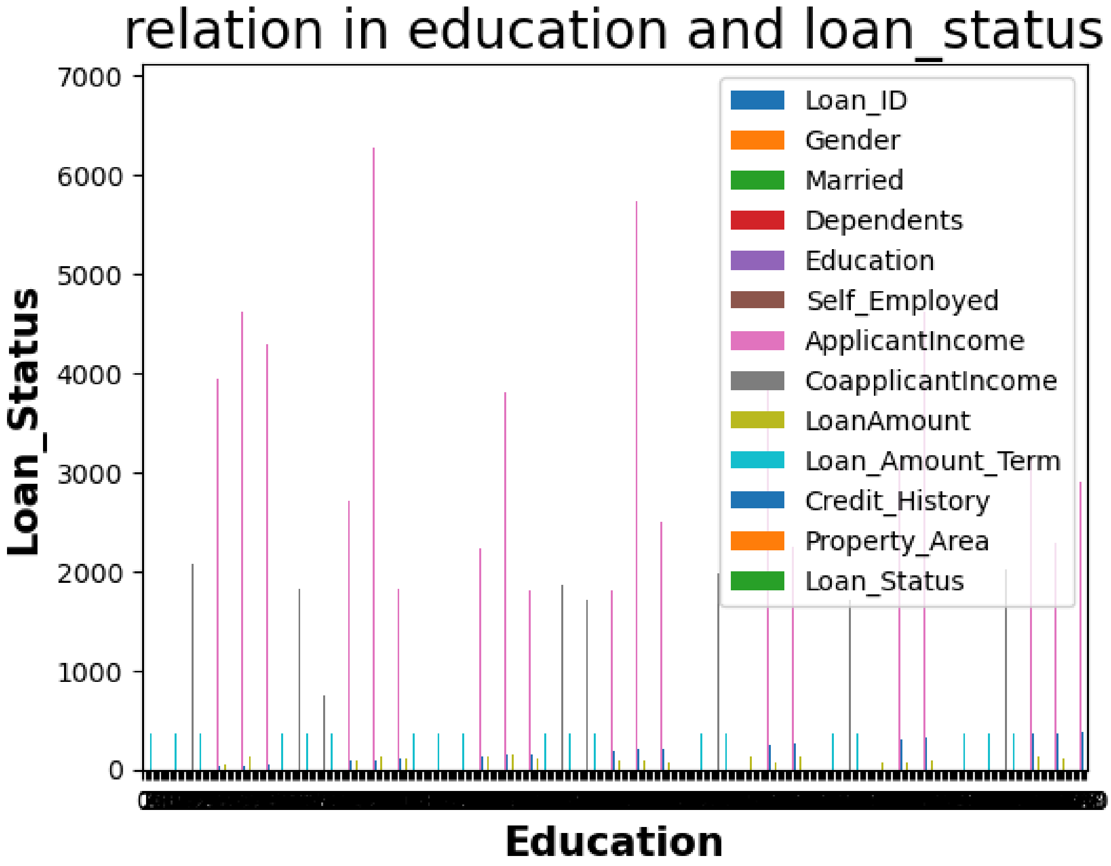
```
data.plot(kind='bar')

plt.title('relation in education and loan_status',size='20')

plt.xticks(color='k',rotation='horizontal')
plt.yticks(color='k')

plt.xlabel('Education',size=15,fontweight='bold')
plt.ylabel('Loan_Status',size=15,fontweight='bold')

plt.show()
```

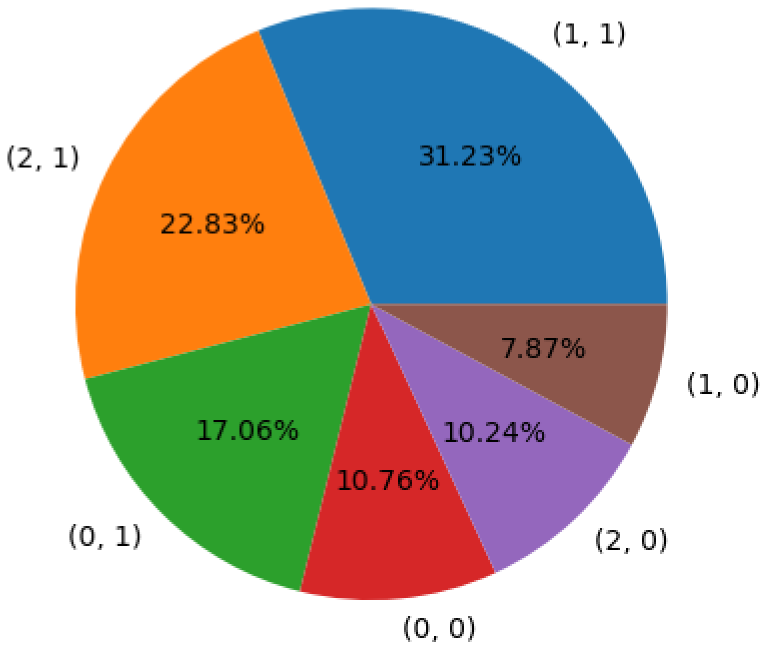


In [66]:

```
data[['Property_Area','Loan_Status']].value_counts().plot(kind='pie',autopc

plt.grid()

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

transforming data

In [56]:

```
from sklearn.preprocessing import LabelEncoder
```

In [57]:

```
LE=LabelEncoder()
```

In [58]:

```
for i in data.select_dtypes(include=['object']):
    data[i]=LE.fit_transform(data[i])
```

In [59]:
```
data
```

Out[59]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Cc
0	0	1	1	1	0	0	4583	
1	1	1	1	0	0	1	3000	
2	2	1	1	0	1	0	2583	
3	3	1	0	0	0	0	6000	
4	4	1	1	0	1	0	2333	
...	...	...	...	...	...	...	...	
376	376	1	1	3	0	0	5703	
377	377	1	1	0	0	0	3232	
378	378	0	0	0	0	0	2900	
379	379	1	1	3	0	0	4106	
380	380	0	0	0	0	1	4583	

381 rows × 13 columns

In [52]:
```
from sklearn.model_selection import train_test_split
```

In [53]:
```
X=data.drop('Loan_Status',axis=1)
y=data['Loan_Status']
```

In [54]:
```
X
```

Out[54]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Cc
0	0	1	1	1	0	0	4583	
1	1	1	1	0	0	1	3000	
2	2	1	1	0	1	0	2583	
3	3	1	0	0	0	0	6000	
4	4	1	1	0	1	0	2333	
...	...	...	...	...	...	...	...	
376	376	1	1	3	0	0	5703	
377	377	1	1	0	0	0	3232	
378	378	0	0	0	0	0	2900	
379	379	1	1	3	0	0	4106	
380	380	0	0	0	0	1	4583	

381 rows × 12 columns

In [55]:
```
y
```

Out[55]:
```
0      0
1      1
2      1
3      1
4      1
      ..
376    1
377    1
378    1
379    1
380    0
Name: Loan_Status, Length: 381, dtype: int32
```

In [108]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_sta
```

USING KNN CLASSIFIER

In [109]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [173]:
```
KNN=KNeighborsClassifier(n_neighbors=25)
```

In [174]:
```
KNN.fit(x_train,y_train)
```

Out[174]:
```
KNeighborsClassifier(n_neighbors=25)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [175]:
```
knn_pred=KNN.predict(x_test)
```

In [176]:
```
knn_pred
```

Out[176]:
```
array([1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

In [177]:
```
from sklearn.metrics import confusion_matrix,classification_report,accuracy
```

In [178]:
```
accuracy_score(y_test,knn_pred)
```

Out[178]:
```
0.7272727272727273
```

In [180]:
```
print(classification_report(y_test,knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.60      0.14      0.22        22
           1       0.74      0.96      0.83        55

    accuracy                           0.73        77
   macro avg       0.67      0.55      0.53        77
weighted avg       0.70      0.73      0.66        77
```

In [181]:
```
f1_score(y_test,knn_pred)
```

Out[181]:
```
0.8346456692913387
```

In [ ]:

USING RANDOM FOREST CLASSIFIER

In [121]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [122]:
```
from sklearn.model_selection import GridSearchCV
```

In [123]:
```
Parameters={'n_estimators':range(100,200),'criterion':['gini','entropt'],'m
```

In [124]:
```
gscv=GridSearchCV(estimator=RandomForestClassifier(),param_grid=Parameters,
```

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

In [ ]:
```
gscv.best_params_
```

In [ ]:
```
RFC=RandomForestClassifier(n_estimators=101)
```

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

In [ ]:
```
modelpred=RFC.predict(x_test)
```

In [ ]:
```
modelpred
```

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

In [83]:
```
RFC.score(x_test,y_test)
```

Out[83]:
```
0.6875
```

In [56]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_
```

In [85]:

```
accuracy_score(y_test,modelpred)
```

Out[85]:

```
0.6875
```

In [ ]:

In [ ]:

In [ ]: