Name:P.Asha Belcilda ¶

Rollno:225229104

Loan approval classification using SVM

In [31]:
```
import pandas as pd
```

Step 1 :Understand data

In [32]:
```
data=pd.read_csv("train_loan.csv")
data
```

Out[32]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coa
0	LP001002	Male	No	0	Graduate	No	5849	
1	LP001003	Male	Yes	1	Graduate	No	4583	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	
4	LP001008	Male	No	0	Graduate	No	6000	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	
7	LP001014	Male	Yes	3+	Graduate	No	3036	
8	LP001018	Male	Yes	2	Graduate	No	4006	
9	LP001020	Male	Yes	1	Graduate	No	12841	

In [33]:
```
data.head
```

Out[33]:
```
<bound method NDFrame.head of       Loan_ID  Gender Married Dependents
Education Self_Employed  \
0     LP001002    Male      No          0      Graduate            No
1     LP001003    Male     Yes          1      Graduate            No
2     LP001005    Male     Yes          0      Graduate           Yes
3     LP001006    Male     Yes          0  Not Graduate            No
4     LP001008    Male      No          0      Graduate            No
5     LP001011    Male     Yes          2      Graduate           Yes
6     LP001013    Male     Yes          0  Not Graduate            No
7     LP001014    Male     Yes         3+      Graduate            No
8     LP001018    Male     Yes          2      Graduate            No
9     LP001020    Male     Yes          1      Graduate            No
10    LP001024    Male     Yes          2      Graduate            No
11    LP001027    Male     Yes          2      Graduate           NaN
12    LP001028    Male     Yes          2      Graduate            No
13    LP001029    Male      No          0      Graduate            No
14    LP001030    Male     Yes          2      Graduate            No
15    LP001032    Male      No          0      Graduate            No
16    LP001034    Male      No          1  Not Graduate            No
```

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

Out[34]:
```
(614, 13)
```

In [35]:
```
data.columns
```

Out[35]:
```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [36]:
```
data.dtypes
```

Out[36]:
```
Loan_ID               object
Gender                object
Married               object
Dependents            object
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [37]:
```
data.info
```

Out[37]:
```
<bound method DataFrame.info of       Loan_ID  Gender Married Dependents
Education Self_Employed  \
0    LP001002    Male      No          0      Graduate            No
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
5    LP001011    Male     Yes          2      Graduate           Yes
6    LP001013    Male     Yes          0  Not Graduate            No
7    LP001014    Male     Yes         3+      Graduate            No
8    LP001018    Male     Yes          2      Graduate            No
9    LP001020    Male     Yes          1      Graduate            No
10   LP001024    Male     Yes          2      Graduate            No
11   LP001027    Male     Yes          2      Graduate           NaN
12   LP001028    Male     Yes          2      Graduate            No
13   LP001029    Male      No          0      Graduate            No
14   LP001030    Male     Yes          2      Graduate            No
15   LP001032    Male      No          0      Graduate            No
16   LP001034    Male      No          1  Not Graduate            No
```

In [38]:
```
data.LoanAmount.value_counts()
```

Out[38]:
```
120.0    20
110.0    17
100.0    15
187.0    12
160.0    12
128.0    11
113.0    11
130.0    10
96.0      9
95.0      9
70.0      8
115.0     8
112.0     8
150.0     7
135.0     7
136.0     7
132.0     7
125.0     7
104.0     7
80.0      6
81.0      6
138.0     6
90.0      6
158.0     6
116.0     6
175.0     6
144.0     6
155.0     6
180.0     6
152.0     5
         ..
315.0     1
101.0     1
73.0      1
142.0     1
48.0      1
164.0     1
83.0      1
191.0     1
166.0     1
495.0     1
59.0      1
214.0     1
240.0     1
72.0      1
42.0      1
349.0     1
280.0     1
405.0     1
279.0     1
304.0     1
650.0     1
436.0     1
78.0      1
54.0      1
89.0      1
570.0     1
```

```
300.0     1
376.0     1
117.0     1
311.0     1
Name: LoanAmount, Length: 203, dtype: int64
```

Step 2 : Data Cleaning

In [42]:
```
data['Dependents'].dtype
```

Out[42]:
```
dtype('O')
```

In [43]:
```
data["Dependents"].fillna("No_Dep", inplace = True)
data["Dependents"]
```

Out[43]:
```
0          0
1          1
2          0
3          0
4          0
5          2
6          0
7         3+
8          2
9          1
10         2
11         2
12         2
13         0
14         2
15         0
16         1
17         0
18         0
```

In [44]:
```
dep={'0':0,'1':1,'2':2,'3+':3,'No_Dep':0}
data.Dependents= [dep[item] for item in data.Dependents]
```

In [45]:
```
data['Dependents'].astype(int)
```

Out[45]:
```
0      0
1      1
2      0
3      0
4      0
5      2
6      0
7      3
8      2
9      1
10     2
11     2
12     2
13     0
14     2
15     0
16     1
17     0
18     0
```

In [49]:
```
data['Married'].fillna(data['Married'].mode()[0], inplace = True)
data['Gender'].fillna(data['Gender'].mode()[0], inplace = True)
data['Education'].fillna(data['Education'].mode()[0], inplace = True)
data['Self_Employed'].fillna(data['Self_Employed'].mode()[0], inplace = True)
data['Dependents'].fillna(data['Dependents'].mode()[0], inplace = True)
data['Credit_History'].fillna(data['Credit_History'].mode()[0], inplace = True)
```

In [50]:
```
data.isnull().sum()
```

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

In [53]:
```
data['LoanAmount'].fillna(data['LoanAmount'].mean(),inplace = True)
```

In [55]:
```
data['Loan_Amount_Term'].fillna(data['Loan_Amount_Term'].mean(),inplace = True)
```

In [56]:
```
data.drop(['Loan_ID'],axis=1)
```

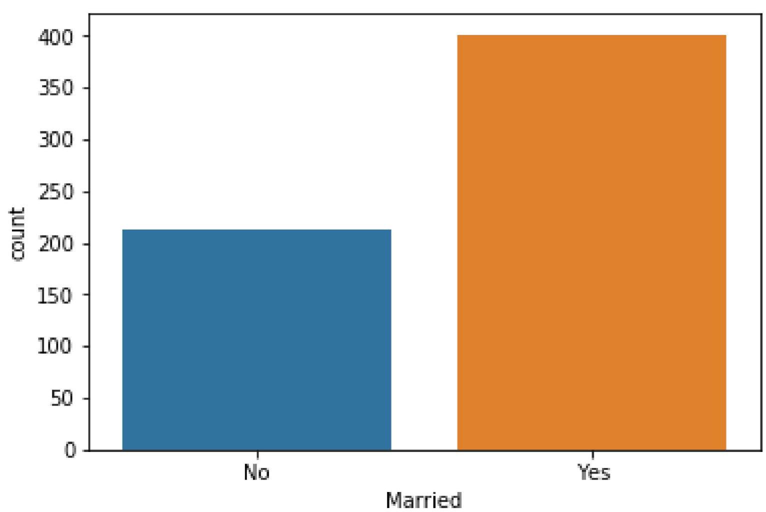
Out[56]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	Male	No	0	Graduate	No	5849	
1	Male	Yes	1	Graduate	No	4583	15
2	Male	Yes	0	Graduate	Yes	3000	
3	Male	Yes	0	Not Graduate	No	2583	23
4	Male	No	0	Graduate	No	6000	
5	Male	Yes	2	Graduate	Yes	5417	41
6	Male	Yes	0	Not Graduate	No	2333	15
7	Male	Yes	3	Graduate	No	3036	25
8	Male	Yes	2	Graduate	No	4006	15
9	Male	Yes	1	Graduate	No	12841	109

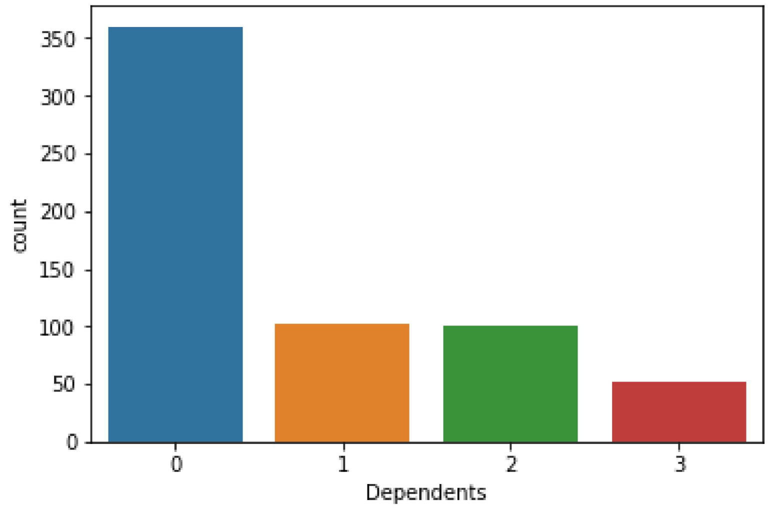
Step 3 : Exploratory data analysis

In [62]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

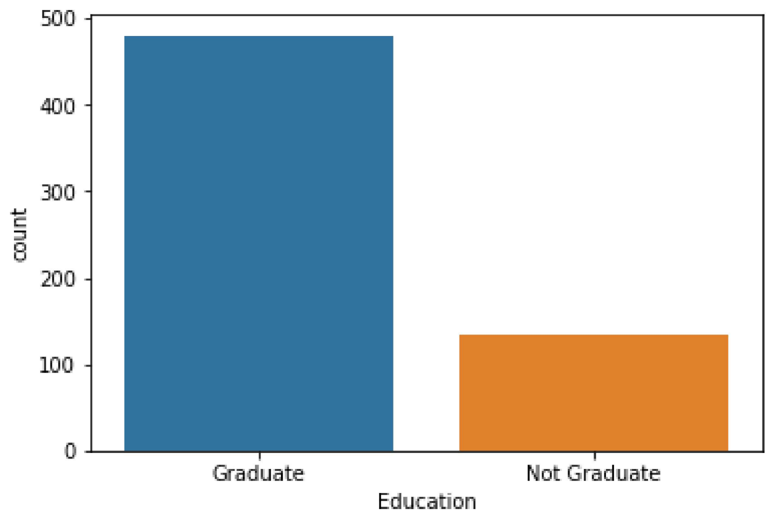
In [64]:
```
sns.countplot(x ='Married', data = data)
plt.show()
```



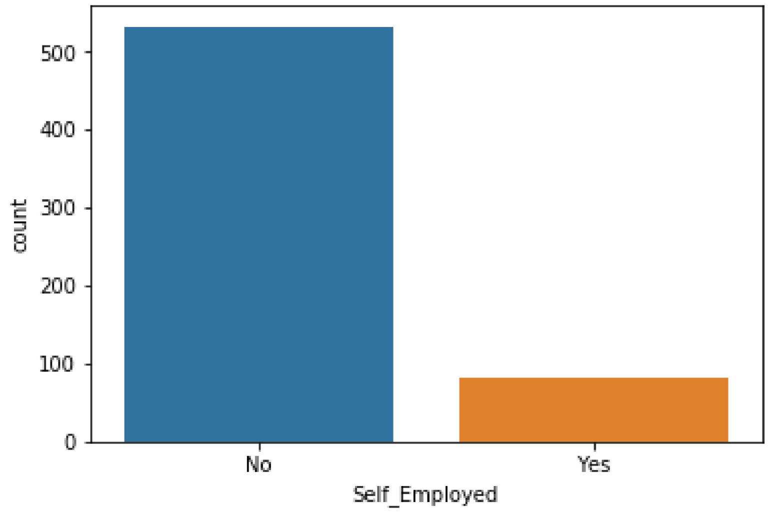
In [65]:
```
sns.countplot(x ='Dependents', data = data)
plt.show()
```



In [67]:
```
sns.countplot(x ='Education', data = data)
plt.show()
```



In [70]:
```
sns.countplot(x ='Self_Employed', data = data)
plt.show()
```



Step 4 : EXtarct X and y

In [71]:
```
X=data.drop(['Loan_Status'],axis=1)
```

In [72]:
```
y=data.pop('Loan_Status')
```

Step 5 : One Hot Encodeing

In [73]:
```
X=pd.get_dummies(X)
```

In [83]:
```
X
```

Out[83]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_
0	0	5849	0.0	146.412162	360.0	
1	1	4583	1508.0	128.000000	360.0	
2	0	3000	0.0	66.000000	360.0	
3	0	2583	2358.0	120.000000	360.0	
4	0	6000	0.0	141.000000	360.0	
5	2	5417	4196.0	267.000000	360.0	
6	0	2333	1516.0	95.000000	360.0	
7	3	3036	2504.0	158.000000	360.0	
8	2	4006	1526.0	168.000000	360.0	
9	1	12841	10968.0	349.000000	360.0	
10	2	3200	700.0	70.000000	360.0	
11	2	2500	1840.0	109.000000	360.0	
12	2	3073	8106.0	200.000000	360.0	
13	0	1853	2840.0	114.000000	360.0	
14	2	1299	1086.0	17.000000	120.0	
15	0	4950	0.0	125.000000	360.0	
16	1	3596	0.0	100.000000	240.0	
17	0	3510	0.0	76.000000	360.0	
18	0	4887	0.0	133.000000	360.0	
19	0	2600	3500.0	115.000000	342.0	
20	0	7660	0.0	104.000000	360.0	
21	1	5955	5625.0	315.000000	360.0	
22	0	2600	1911.0	116.000000	360.0	
23	2	3365	1917.0	112.000000	360.0	
24	1	3717	2925.0	151.000000	360.0	
25	0	9560	0.0	191.000000	360.0	
26	0	2799	2253.0	122.000000	360.0	
27	2	4226	1040.0	110.000000	360.0	
28	0	1442	0.0	35.000000	360.0	
29	2	3750	2083.0	120.000000	360.0	
...	...	...	...	...	...	
584	1	2787	1917.0	146.000000	360.0	
585	1	4283	3000.0	172.000000	84.0	
586	0	2297	1522.0	104.000000	360.0	
587	0	2165	0.0	70.000000	360.0	

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_
588	0	4750	0.0	94.000000	360.0	
589	2	2726	0.0	106.000000	360.0	
590	0	3000	3416.0	56.000000	180.0	
591	2	6000	0.0	205.000000	240.0	
592	3	9357	0.0	292.000000	360.0	
593	0	3859	3300.0	142.000000	180.0	
594	0	16120	0.0	260.000000	360.0	
595	0	3833	0.0	110.000000	360.0	
596	2	6383	1000.0	187.000000	360.0	
597	0	2987	0.0	88.000000	360.0	
598	0	9963	0.0	180.000000	360.0	
599	2	5780	0.0	192.000000	360.0	
600	3	416	41667.0	350.000000	180.0	
601	0	2894	2792.0	155.000000	360.0	
602	3	5703	0.0	128.000000	360.0	
603	0	3676	4301.0	172.000000	360.0	
604	1	12000	0.0	496.000000	360.0	
605	0	2400	3800.0	146.412162	180.0	
606	1	3400	2500.0	173.000000	360.0	
607	2	3987	1411.0	157.000000	360.0	
608	0	3232	1950.0	108.000000	360.0	
609	0	2900	0.0	71.000000	360.0	
610	3	4106	0.0	40.000000	180.0	
611	1	8072	240.0	253.000000	360.0	
612	2	7583	0.0	187.000000	360.0	
613	0	4583	0.0	133.000000	360.0	

614 rows × 631 columns

Step 6 : Model Building

In [79]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.30,random_state=4
```

In [81]:
```
from sklearn.preprocessing import StandardScaler
ss= StandardScaler()
```

In [82]:
```
X_train_ss=ss.fit_transform(X_train)
X_train_ss
```

Out[82]:
```
array([[-0.71703534, -0.50133384,  0.27865737, ..., -0.62317695,
        -0.79056942,  1.40682858],
       [-0.71703534, -0.42803179,  0.45103751, ...,  1.60468065,
        -0.79056942, -0.71081865],
       [-0.71703534, -0.5669725 ,  0.23208844, ..., -0.62317695,
         1.26491106, -0.71081865],
       ...,
       [-0.71703534, -0.37088951, -0.59751445, ..., -0.62317695,
        -0.79056942,  1.40682858],
       [-0.71703534,  0.76362634, -0.59751445, ..., -0.62317695,
         1.26491106, -0.71081865],
       [-0.71703534,  1.36387019, -0.59751445, ..., -0.62317695,
        -0.79056942,  1.40682858]])
```

In [85]:
```
X_test_ss=ss.fit_transform(X_test)
X_test_ss
```

Out[85]:
```
array([[-0.78697069,  0.60310661, -0.4897835 , ..., -0.68429085,
         1.31171195, -0.67579058],
       [-0.78697069, -0.1508012 , -0.4897835 , ..., -0.68429085,
         1.31171195, -0.67579058],
       [ 1.15422368, -0.17338842, -0.07075971, ...,  1.4613669 ,
        -0.7623625 , -0.67579058],
       ...,
       [ 1.15422368,  1.02547189, -0.4897835 , ..., -0.68429085,
        -0.7623625 ,  1.47974835],
       [-0.78697069, -0.34587267,  0.20984434, ...,  1.4613669 ,
        -0.7623625 , -0.67579058],
       [-0.78697069,  0.03716241, -0.4897835 , ...,  1.4613669 ,
        -0.7623625 , -0.67579058]])
```

In [87]:
```
from sklearn.svm import LinearSVC
lvc=LinearSVC()
lvc.fit(X_train_ss,y_train)
l_pred=lvc.predict(X_test_ss)
```

In [88]:
```
l_pred
```

Out[88]:
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N'], dtype=object)
```

In [93]:
```
from sklearn.metrics import accuracy_score
lvc_acc=accuracy_score(y_test,l_pred)
print("lvc_acc_score : ",lvc_acc)
```

```
lvc_acc_score :  0.7513513513513513
```

In [95]:
```
from sklearn.metrics import confusion_matrix
c_mat=confusion_matrix(y_test,l_pred)
c_mat
```

Out[95]:
```
array([[ 21,  44],
       [  2, 118]], dtype=int64)
```

In [97]:
```
from sklearn.metrics import classification_report
c_rep=classification_report(y_test,l_pred)
print(c_rep)
```

```
             precision    recall  f1-score   support

          N       0.91      0.32      0.48        65
          Y       0.73      0.98      0.84       120

avg / total       0.79      0.75      0.71       185
```

Step 7 : Performance Comparison

In [101]:

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train_ss,y_train)
lr_pred=lr.predict(X_test_ss)

from sklearn.svm import LinearSVC
lvc=LinearSVC()
lvc.fit(X_train_ss,y_train)
l_pred=lvc.predict(X_test_ss)

from sklearn.metrics import accuracy_score
lvc_acc=accuracy_score(y_test,l_pred)
print("linear_svc_acc_score : ",lvc_acc)

from sklearn.metrics import accuracy_score
lr_acc=accuracy_score(y_test,lr_pred)
print("linear_reg_acc_score : ",lr_acc)
```

```
linear_svc_acc_score :  0.7513513513513513
linear_reg_acc_score :  0.7783783783783784
```