Double-click (or enter) to edit

PROPERTY LOAN PREDICTION

DATA COLLECTION

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('/content/loan_sanction_train.csv')
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Appli
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	
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	
610	LP002979	Male	Yes	3+	Graduate	No	
611	LP002983	Male	Yes	1	Graduate	No	
612	LP002984	Male	Yes	2	Graduate	No	
613	LP002990	Female	No	0	Graduate	Yes	

614 rows × 13 columns

```
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applica
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	

✓ 0s completed at 9:26 PM

3	LP001006	Male	Yes	0	Not Graduate	No
4	LP001008	Male	No	0	Graduate	No

```
df.describe()
```

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

```
df.shape
```

```
(614, 13)
```

```
df.isna().sum()
```

```
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
```

Dropping the missing values

```
dfe.isna().sum()
```

```
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
```

```
dfe.shape
```

```
(599, 13)
```

Label_Encoding

```
df.replace({"Loan_Status":{'N':0,'Y':1}},inplace=True)
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Appli
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	
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	
610	LP002979	Male	Yes	3+	Graduate	No	
611	LP002983	Male	Yes	1	Graduate	No	
612	LP002984	Male	Yes	2	Graduate	No	
613	LP002990	Female	No	0	Graduate	Yes	

614 rows × 13 columns

```
df.value_counts(['Dependents'])
```

```
Dependents
0             345
1             102
2             101
3+             51
dtype: int64
```

```
#replaceing 3+ values to 4
df.replace({'Dependents':{'3+':4}},inplace=True)
df
```

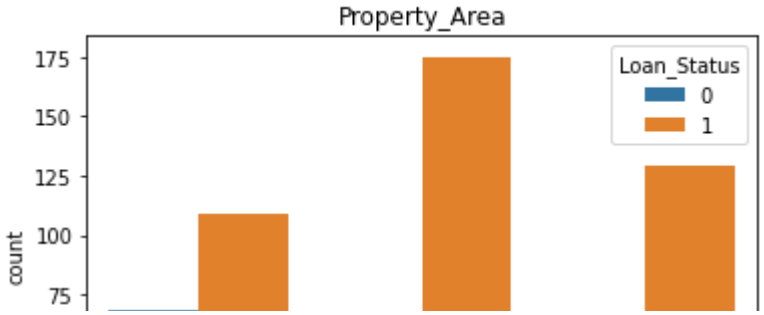
	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Appli
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	
...	...	...	...	...	...	...	
609	LP002978	Female	No	0	Graduate	No	
610	LP002979	Male	Yes	4	Graduate	No	
611	LP002983	Male	Yes	1	Graduate	No	
612	LP002984	Male	Yes	2	Graduate	No	
613	LP002990	Female	No	0	Graduate	Yes	

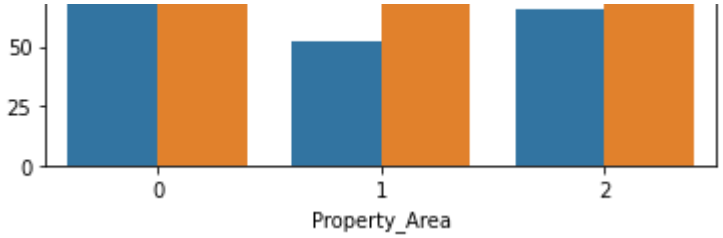
614 rows × 13 columns

DATA VISUALIZATION

```
sns.countplot(x='Property_Area',hue='Loan_Status',data=dfe)
plt.title('Property_Area')
```

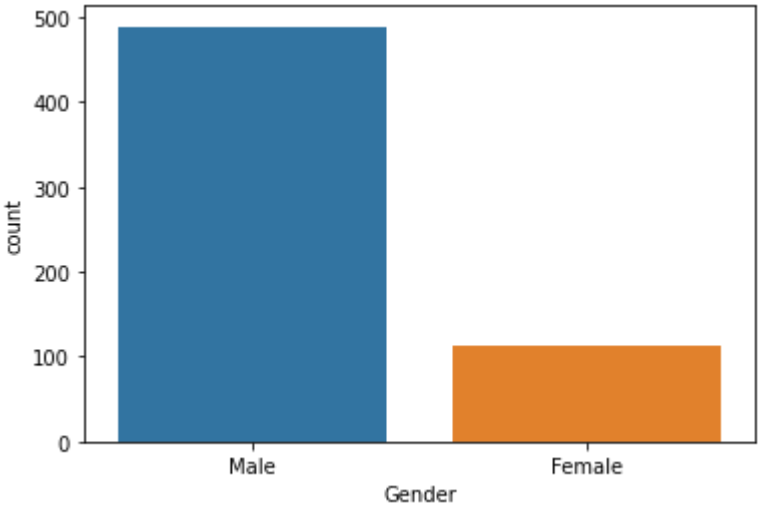
```
Text(0.5, 1.0, 'Property_Area')
```






```
sns.countplot(df['Gender'])
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWa
  warnings.warn(
<matplotlib.axes._subplots.AxesSubplot at 0x7fb846b879a0>
```




```
df.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Empl
                    'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Educa
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Appli
0	LP001002	1.0	0.0	0	1	0.0	
1	LP001003	1.0	1.0	1	1	0.0	
2	LP001005	1.0	1.0	0	1	1.0	
3	LP001006	1.0	1.0	0	0	0.0	
4	LP001008	1.0	0.0	0	1	0.0	
...	...	...	...	...	...	...	
609	LP002978	0.0	0.0	0	1	0.0	
610	LP002979	1.0	1.0	4	1	0.0	
611	LP002983	1.0	1.0	1	1	0.0	
612	LP002984	1.0	1.0	2	1	0.0	
613	LP002990	0.0	0.0	0	1	1.0	

614 rows × 13 columns

```
df['Gender']=df['Gender'].fillna(df['Gender'].mean())
df['Married']=df['Married'].fillna(df['Married'].mean())
df['Self_Employed']=df['Self_Employed'].fillna(df['Self_Employed'].mean())
df['LoanAmount']=df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term']=df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean
df['Credit_History']=df['Credit_History'].fillna(df['Credit_History'].mean())
```

```
dfe=df.dropna()
dfe
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Appli
0	LP001002	1.0	0.0	0	1	0.0	
1	LP001003	1.0	1.0	1	1	0.0	
2	LP001005	1.0	1.0	0	1	1.0	
3	LP001006	1.0	1.0	0	0	0.0	
4	LP001008	1.0	0.0	0	1	0.0	
...	...	...	...	...	...	...	
609	LP002978	0.0	0.0	0	1	0.0	
610	LP002979	1.0	1.0	4	1	0.0	
611	LP002983	1.0	1.0	1	1	0.0	
612	LP002984	1.0	1.0	2	1	0.0	
613	LP002990	0.0	0.0	0	1	1.0	

599 rows × 13 columns

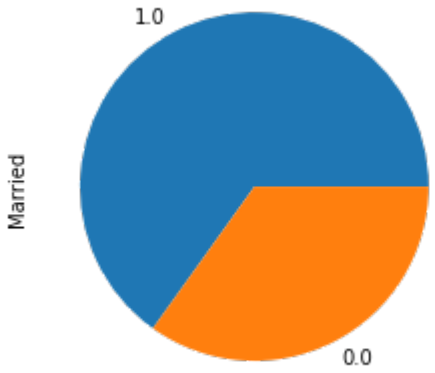
```
dfe.isna().sum()
```

```
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
```

```
dfe['Married'].value_counts().plot(kind='pie')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb846b49730>
```




```
dfe.tail()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Appli
609	LP002978	0.0	0.0	0	1	0.0	
610	LP002979	1.0	1.0	4	1	0.0	
611	LP002983	1.0	1.0	1	1	0.0	
612	LP002984	1.0	1.0	2	1	0.0	
613	LP002990	0.0	0.0	0	1	1.0	

```
x=dfe.drop(['Loan_ID','Loan_Status'],axis=1)
x
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	1.0	0.0	0	1	0.0	5849
1	1.0	1.0	1	1	0.0	4583
2	1.0	1.0	0	1	1.0	3000
3	1.0	1.0	0	0	0.0	2583
4	1.0	0.0	0	1	0.0	6000
...	...	...	...	...	...	...
609	0.0	0.0	0	1	0.0	2900

609	0.0	0.0	0	1	0.0	2900
610	1.0	1.0	4	1	0.0	4106
611	1.0	1.0	1	1	0.0	8072
612	1.0	1.0	2	1	0.0	7583
613	0.0	0.0	0	1	1.0	4583

599 rows × 11 columns

```
y=dfe['Loan_Status']
y
```

```
0      1
1      0
2      1
3      1
4      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 599, dtype: int64
```

```
from sklearn.model_selection import train_test_split
xtest,xtrain,ytest,ytrain=train_test_split(x,y,test_size=0.30,random_state=42)
xtrain
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
112	1.0	1.0	0	0	0.0	3572
430	0.0	0.0	1	1	1.0	8624
578	1.0	1.0	1	1	0.0	1782
77	1.0	1.0	1	1	1.0	1000
184	0.0	1.0	0	1	0.0	3625
...	...	...	...	...	...	...
54	0.0	1.0	1	1	1.0	11500
505	1.0	1.0	2	1	0.0	3510
46	1.0	1.0	1	1	0.0	5649
93	1.0	0.0	0	1	0.0	4133
269	0.0	0.0	1	1	0.0	2876

180 rows × 11 columns

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(xtrain)
xtrain=scaler.transform(xtrain)
xtest=scaler.transform(xtest)
xtrain
```

```
array([[ 0.4679807 ,  0.74278135, -0.75854143, ..., -0.00726213,
        -2.41267568, -1.27064737],
       [-2.14916944, -1.3462912 ,  0.035281  , ...,  0.27284874,
         0.4573293 , -0.01396316],
       [ 0.4679807 ,  0.74278135,  0.035281  , ...,  0.27284874,
         0.4573293 , -1.27064737],
       ...,
       [ 0.4679807 ,  0.74278135,  0.035281  , ...,  0.27284874,
         0.4573293 ,  1.24272106],
       [ 0.4679807 , -1.3462912 , -0.75854143, ...,  0.27284874,
         0.4573293 , -0.01396316],
       [-2.14916944, -1.3462912 ,  0.035281  , ...,  0.27284874,
         0.4573293 ,  1.24272106]])
```

KNN MODEL

```
from sklearn.neighbors import KNeighborsClassifier
model=KNeighborsClassifier(n_neighbors=5)
model.fit(xtrain,ytrain)
ypred=model.predict(xtest)
ypred
```

```
array([1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 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,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 0, 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, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 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, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
       1])
```

Performance Evaluation

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matri
result=confusion_matrix(ytest,ypred)
result
```

```
array([[ 55,  70],
       [ 18, 276]])
```

```
score=accuracy_score(ytest,ypred)
score*100
```

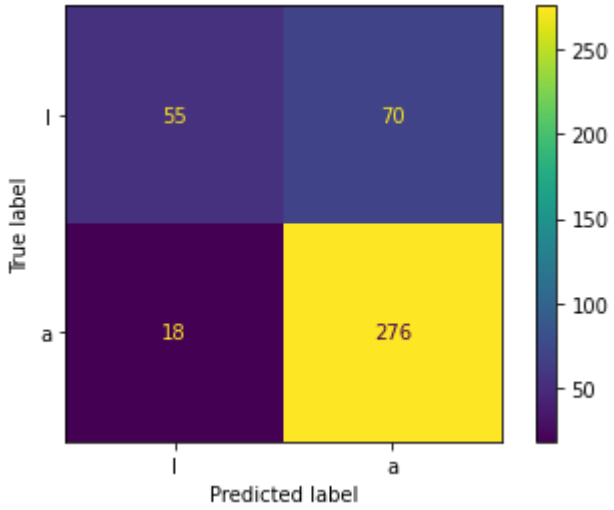
```
78.99761336515513
```

```
report=classification_report(ytest,ypred)
report
```

```
'              precision    recall  f1-score   support\n\n           0
0.75      0.44      0.56       125\n           1       0.80      0.94
0.86       294\n\n    accuracy                           0.79       419\n
macro avg       0.78      0.69      0.71       419\nweighted avg       0.7
```

```
from sklearn.metrics import ConfusionMatrixDisplay
cmd=ConfusionMatrixDisplay(result,display_labels='label')
cmd.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fb80c71e520>
```




```
from sklearn.metrics import mean_absolute_error,mean_absolute_percentage_error,m
print('mean_absolute_error',mean_absolute_percentage_error(ytest,ypred))
print('MAE % :',mean_absolute_percentage_error(ytest,ypred))
print('mean squared error : ',np.sqrt(mean_squared_error(ytest,ypred)))
print('r2_score:',r2_score(ytest,ypred))
```

```
mean_absolute_error 752391345861419.4
MAE % : 752391345861419.4
```

```
mean squared error :  0.458283609076790 67
r2_score: -0.003319727891156532
```