Predict Loan Eligibility for Dream Housing Finance company

Dream Housing Finance company deals in all kinds of home loans. They have presence across all urban, semi urban and rural areas. Customer first applies for home loan and after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have provided a dataset to identify the customers segments that are eligible for loan amount so that they can specifically target these customers.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
```

```
df=pd.read_csv("/content/training.csv")
```

```
df.head(10)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Co
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	

```
df.shape
```

```
(614, 13)
```

```
df.dtypes
```

```
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
```

```
df.corr()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cr
ApplicantIncome	1.000000	-0.116605	0.570909	-0.045306	
CoapplicantIncome	-0.116605	1.000000	0.188619	-0.059878	
LoanAmount	0.570909	0.188619	1.000000	0.039447	
Loan_Amount_Term	-0.045306	-0.059878	0.039447	1.000000	
Credit_History	-0.014715	-0.002056	-0.008433	0.001470	

```
a= df['Property_Area'].values
a
```

```
array(['Urban', 'Rural', 'Urban', 'Urban', 'Urban', 'Urban', 'Urban',
       'Semiurban', 'Urban', 'Semiurban', 'Urban', 'Urban', 'Urban',
       'Rural', 'Urban', 'Urban', 'Urban', 'Urban', 'Rural', 'Urban',
       'Urban', 'Urban', 'Semiurban', 'Rural', 'Semiurban', 'Semiurban',
       'Semiurban', 'Urban', 'Urban', 'Semiurban', 'Urban', 'Urban',
       'Rural', 'Semiurban', 'Rural', 'Urban', 'Urban', 'Semiurban',
       'Urban', 'Semiurban', 'Urban', 'Urban', 'Urban', 'Semiurban',
       'Urban', 'Urban', 'Urban', 'Urban', 'Urban', 'Semiurban',
       'Semiurban', 'Semiurban', 'Semiurban', 'Urban', 'Urban',
       'Semiurban', 'Semiurban', 'Rural', 'Urban', 'Urban', 'Urban',
       'Urban', 'Rural', 'Rural', 'Semiurban', 'Semiurban', 'Urban',
       'Urban', 'Urban', 'Semiurban', 'Urban', 'Semiurban', 'Semiurban',
       'Semiurban', 'Semiurban', 'Urban', 'Urban', 'Urban', 'Semiurban',
       'Semiurban', 'Semiurban', 'Semiurban', 'Urban', 'Semiurban',
       'Urban', 'Semiurban', 'Semiurban', 'Semiurban', 'Urban',
       'Semiurban', 'Semiurban', 'Semiurban', 'Urban', 'Semiurban',
       'Semiurban', 'Urban', 'Semiurban', 'Semiurban', 'Semiurban',
       'Semiurban', 'Urban', 'Semiurban', 'Urban', 'Semiurban', 'Urban',
       'Urban', 'Urban', 'Rural', 'Urban', 'Semiurban', 'Urban',
```

```
'Semiurban', 'Rural', 'Semiurban', 'Semiurban', 'Rural',
'Semiurban', 'Urban', 'Rural', 'Urban', 'Rural', 'Semiurban',
'Semiurban', 'Semiurban', 'Rural', 'Rural', 'Rural', 'Rural',
'Urban', 'Rural', 'Urban', 'Urban', 'Semiurban', 'Semiurban',
'Semiurban', 'Semiurban', 'Rural', 'Urban', 'Semiurban', 'Rural',
'Rural', 'Urban', 'Semiurban', 'Semiurban', 'Urban', 'Semiurban',
'Urban', 'Urban', 'Rural', 'Semiurban', 'Rural', 'Rural', 'Urban',
'Rural', 'Urban', 'Semiurban', 'Rural', 'Urban', 'Rural',
'Semiurban', 'Semiurban', 'Urban', 'Semiurban', 'Rural', 'Urban',
'Rural', 'Rural', 'Rural', 'Semiurban', 'Semiurban', 'Rural',
'Urban', 'Rural', 'Semiurban', 'Semiurban', 'Rural', 'Rural',
'Semiurban', 'Semiurban', 'Urban', 'Urban', 'Rural', 'Semiurban',
'Semiurban', 'Semiurban', 'Semiurban', 'Rural', 'Rural', 'Rural',
'Rural', 'Rural', 'Semiurban', 'Urban', 'Semiurban', 'Rural',
'Semiurban', 'Rural', 'Urban', 'Semiurban', 'Urban', 'Semiurban',
'Semiurban', 'Urban', 'Urban', 'Semiurban', 'Semiurban', 'Urban',
'Rural', 'Urban', 'Semiurban', 'Semiurban', 'Semiurban', 'Urban',
'Rural', 'Urban', 'Semiurban', 'Rural', 'Semiurban', 'Semiurban',
'Semiurban', 'Urban', 'Semiurban', 'Semiurban', 'Semiurban',
'Semiurban', 'Rural', 'Urban', 'Semiurban', 'Semiurban', 'Rural',
'Semiurban', 'Rural', 'Rural', 'Semiurban', 'Semiurban', 'Rural',
'Urban', 'Urban', 'Rural', 'Semiurban', 'Rural', 'Urban', 'Urban',
'Rural', 'Semiurban', 'Urban', 'Urban', 'Urban', 'Semiurban',
'Urban', 'Semiurban', 'Urban', 'Rural', 'Semiurban', 'Urban',
'Rural', 'Rural', 'Urban', 'Rural', 'Semiurban', 'Urban',
'Semiurban', 'Semiurban', 'Rural', 'Semiurban', 'Rural',
'Semiurban', 'Urban', 'Rural', 'Urban', 'Urban', 'Urban', 'Rural',
'Semiurban', 'Semiurban', 'Semiurban', 'Semiurban', 'Urban',
'Semiurban', 'Rural', 'Urban', 'Semiurban', 'Urban', 'Urban',
'Rural', 'Rural', 'Semiurban', 'Rural', 'Semiurban', 'Rural',
'Rural', 'Semiurban', 'Urban', 'Urban', 'Semiurban', 'Urban',
'Semiurban', 'Urban', 'Rural', 'Urban', 'Urban', 'Semiurban',
'Rural', 'Urban', 'Rural', 'Urban', 'Rural', 'Urban', 'Rural',
'Rural', 'Semiurban', 'Semiurban', 'Rural', 'Rural', 'Rural',
'Urban', 'Semiurban', 'Urban', 'Semiurban', 'Rural', 'Semiurban',
'Semiurban', 'Rural', 'Rural', 'Rural', 'Rural', 'Rural',
'Semiurban', 'Urban', 'Urban', 'Urban', 'Semiurban', 'Urban',
'Urban', 'Urban', 'Semiurban', 'Rural', 'Rural', 'Urban',
'Semiurban', 'Rural', 'Rural', 'Urban', 'Semiurban', 'Rural',
'Semiurban', 'Rural', 'Urban', 'Semiurban', 'Rural', 'Semiurban',
```

```
from sklearn.preprocessing import LabelEncoder
label=LabelEncoder()
```

```
df.Property_Area=label.fit_transform(df.Property_Area)
df.Property_Area.head()
```

```
0    2
1    0
2    2
3    2
4    2
Name: Property_Area, dtype: int64
```

```
df.Loan_Status=label.fit_transform(df.Loan_Status)
```

```
df.Loan_Status=label.fit_transform(df.Loan_Status)
df.Loan_Status.head()
```

```
0    1
1    0
2    1
3    1
4    1
Name: Loan_Status, dtype: int64
```

```
newdf=df.replace(np.NAN,{'LoanAmount':100,'Loan_Amount_Term':360.0,'Credit_History':1.0})
newdf
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
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
...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900
610	LP002979	Male	Yes	3+	Graduate	No	4106
611	LP002983	Male	Yes	1	Graduate	No	8072
612	LP002984	Male	Yes	2	Graduate	No	7583
613	LP002990	Female	No	0	Graduate	Yes	4583

614 rows × 13 columns

```
sns.relplot(x='ApplicantIncome',y='LoanAmount',hue="Credit_History",data=newdf)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f3e80769ad0>
```


```
x=newdf.drop(['Loan_ID','Gender','Married','Dependents','Education','Self_Employed','Loan_Sta
x
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	5849	0.0	100.0	360.0	1.0
1	4583	1508.0	128.0	360.0	1.0
2	3000	0.0	66.0	360.0	1.0
3	2583	2358.0	120.0	360.0	1.0
4	6000	0.0	141.0	360.0	1.0
...	...	...	...	...	...
609	2900	0.0	71.0	360.0	1.0
610	4106	0.0	40.0	180.0	1.0
611	8072	240.0	253.0	360.0	1.0
612	7583	0.0	187.0	360.0	1.0
613	4583	0.0	133.0	360.0	0.0

614 rows × 6 columns

```
y=newdf['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: 614, dtype: int64
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=1)
print(len(x_train))
print(len(x_test))
```

```
429
185
```

```
from sklearn.tree import DecisionTreeClassifier
clf=DecisionTreeClassifier(random_state=5)
clf.fit(x_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=5, splitter='best')
```

```
y_pred=clf.predict(x_test)
y_pred
```

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

```
from sklearn.metrics import accuracy_score
Accuracy=accuracy_score(y_test,y_pred)
print("Accuracy is",Accuracy*100,'%')
```

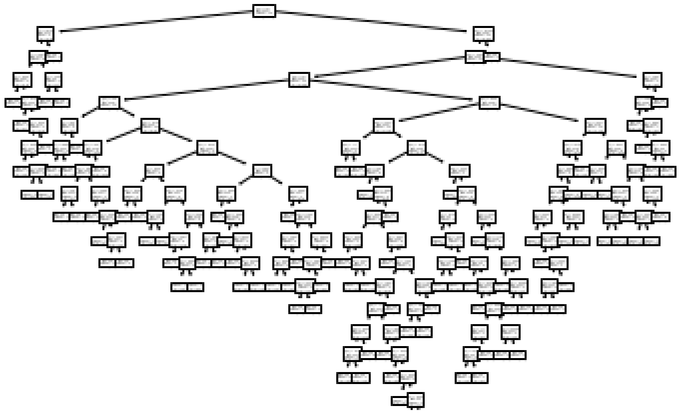
```
Accuracy is 72.97297297297297 %
```

```
from sklearn.metrics import confusion_matrix
cm=np.array(confusion_matrix(y_test,y_pred))
cm
```

```
array([[ 35,  26],
       [ 24, 100]])
```

```
from sklearn import tree
tree.plot_tree(clf)
plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```



```
plt.figure()
tree.plot_tree(clf,filled=True)
plt.savefig('tree.jpg',format='jpg',bbox_inches = "tight")
```