Task-11

GO_STP_5247

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 numpy as np
     import pandas as pd
     from matplotlib import pyplot as plt
     import sklearn

     from sklearn import tree
     from sklearn.tree import DecisionTreeClassifier
     from sklearn.experimental import enable_iterative_imputer
     from sklearn.impute import IterativeImputer
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix
     from sklearn.metrics import accuracy_score , f1_score
```

✓ 1s completed at 11:42 AM

```
[2] df = pd.read_csv("/content/DreamHF_data.csv")
```

```
[3] df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
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

```
[5] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
```

[6]
```
df.shape
```

```
(614, 13)
```

[7]
```
df.nunique()
```

```
Loan_ID              614
Gender                 2
Married                2
Dependents             4
Education              2
Self_Employed          2
ApplicantIncome      505
CoapplicantIncome    287
LoanAmount           203
Loan_Amount_Term      10
Credit_History         2
Property_Area          3
Loan_Status            2
dtype: int64
```

[8]
```
df = df.drop(columns=['Loan_ID']) # dropping the unique  Loan ID
enc_df=pd.get_dummies(df,drop_first=True)
enc_df.head()
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Gender_Male	Married_Yes	Dependents_1	Dependents_2	Dependents_3+	Education_Not Graduate	S
0	5849	0.0	NaN	360.0	1.0	1	0	0	0	0	0	
1	4583	1508.0	128.0	360.0	1.0	1	1	1	0	0	0	
2	3000	0.0	66.0	360.0	1.0	1	1	0	0	0	0	
3	2583	2358.0	120.0	360.0	1.0	1	1	0	0	0	1	

```
[9]  X = enc_df.drop(columns='Loan_Status_Y')
     y = enc_df['Loan_Status_Y']
```

```
[10] # Splitting into Train -Test Data
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,stratify =y,random_state =42)
```

```
[11] # Handling/Imputing Missing values
     imp = IterativeImputer(random_state=7)
     imp.fit(X_train)
     X_train = imp.transform(X_train)
     X_test = imp.transform(X_test)
```

```
[12] tree_clf = DecisionTreeClassifier(random_state=7)
     tree_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=7, splitter='best')
```

```
[13] y_pred = tree_clf.predict(X_train)
     print("Training Data Set Accuracy: ", accuracy_score(y_train,y_pred))
     print("Training Data F1 Score ", f1_score(y_train,y_pred))
```

```
     Training Data Set Accuracy:  1.0
     Training Data F1 Score  1.0
```

```
[14] y_test_pred = tree_clf.predict(X_test)
     print("Test Data Set Accuracy: ", accuracy_score(y_test,y_test_pred))
     print("Test Data F1 Score ", f1_score(y_test,y_test_pred))
```

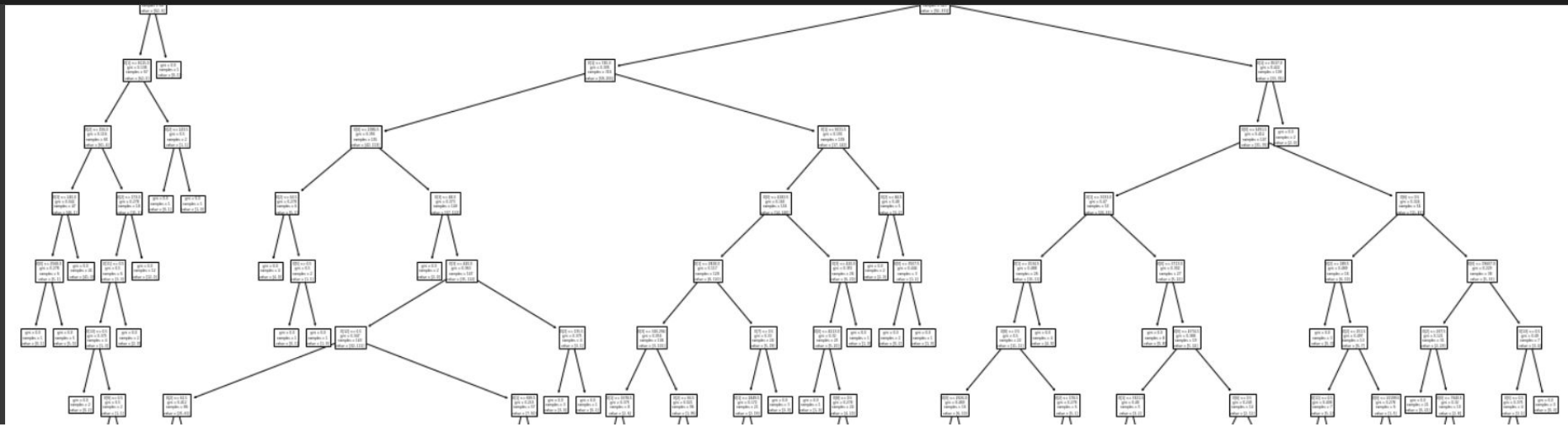
```
[15] tree_clf.feature_importances_
```

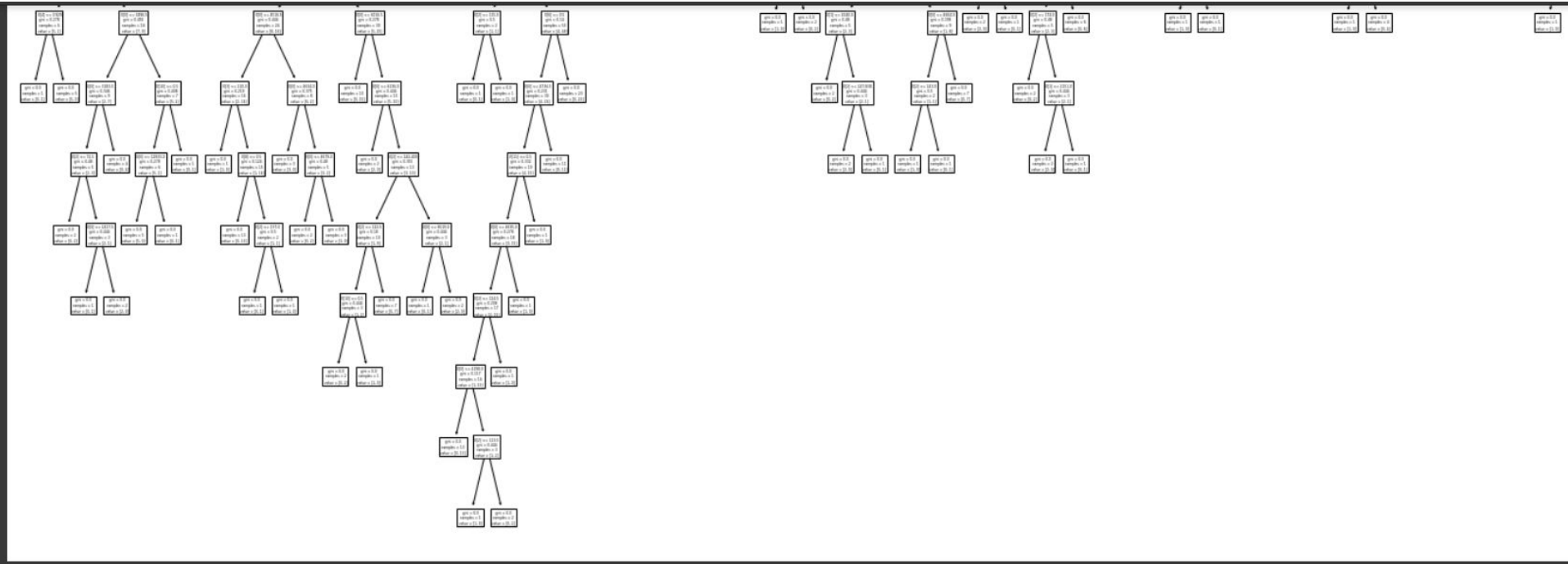
```
array([0.20953886, 0.13484385, 0.1828354 , 0.04264554, 0.28356323,
       0.01182608, 0.0162204 , 0.02663684, 0.02286376, 0.00473043,
       0.01193871, 0.02052818, 0.01387201, 0.0179567 ])
```

```
[16] cm=np.array(confusion_matrix(y_test,y_test_pred))
     cm
```

```
array([[23, 15],
       [18, 67]])
```

```
[17] fig = plt.figure(figsize=(25,20))
     _=tree.plot_tree(tree_clf)
```





```
# taking max_depth as 5
tree_clf = DecisionTreeClassifier(max_depth = 5,random_state=7)
tree_clf.fit(X_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, 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',
```

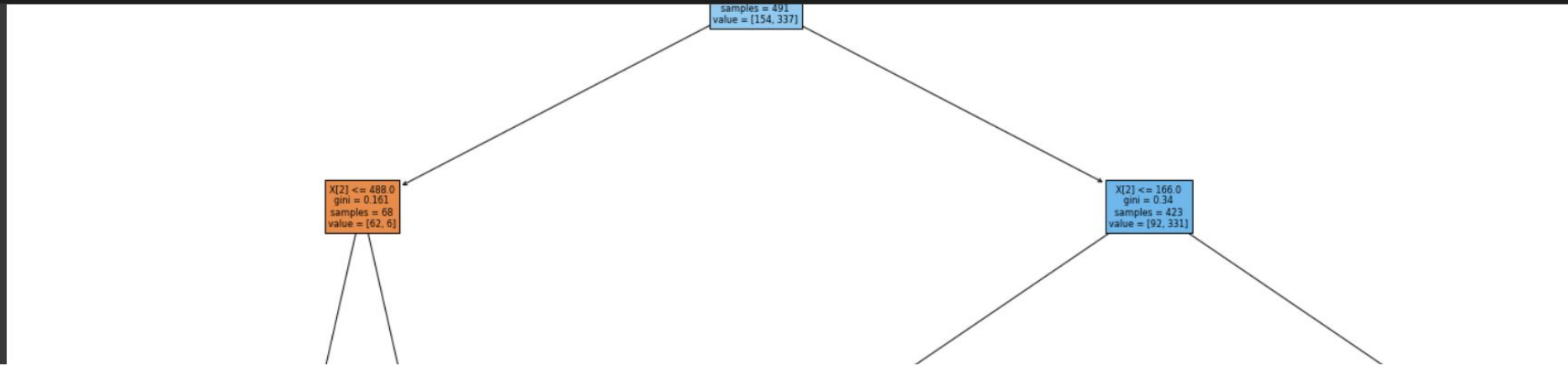
```
[19] # adding the max_depth parameter has given 5 % increase in accuracy
     y_test_pred = tree_clf.predict(X_test)
     print("Test Data Set Accuracy: ", accuracy_score(y_test,y_test_pred))
     print("Test Data F1 Score ", f1_score(y_test,y_test_pred))
```

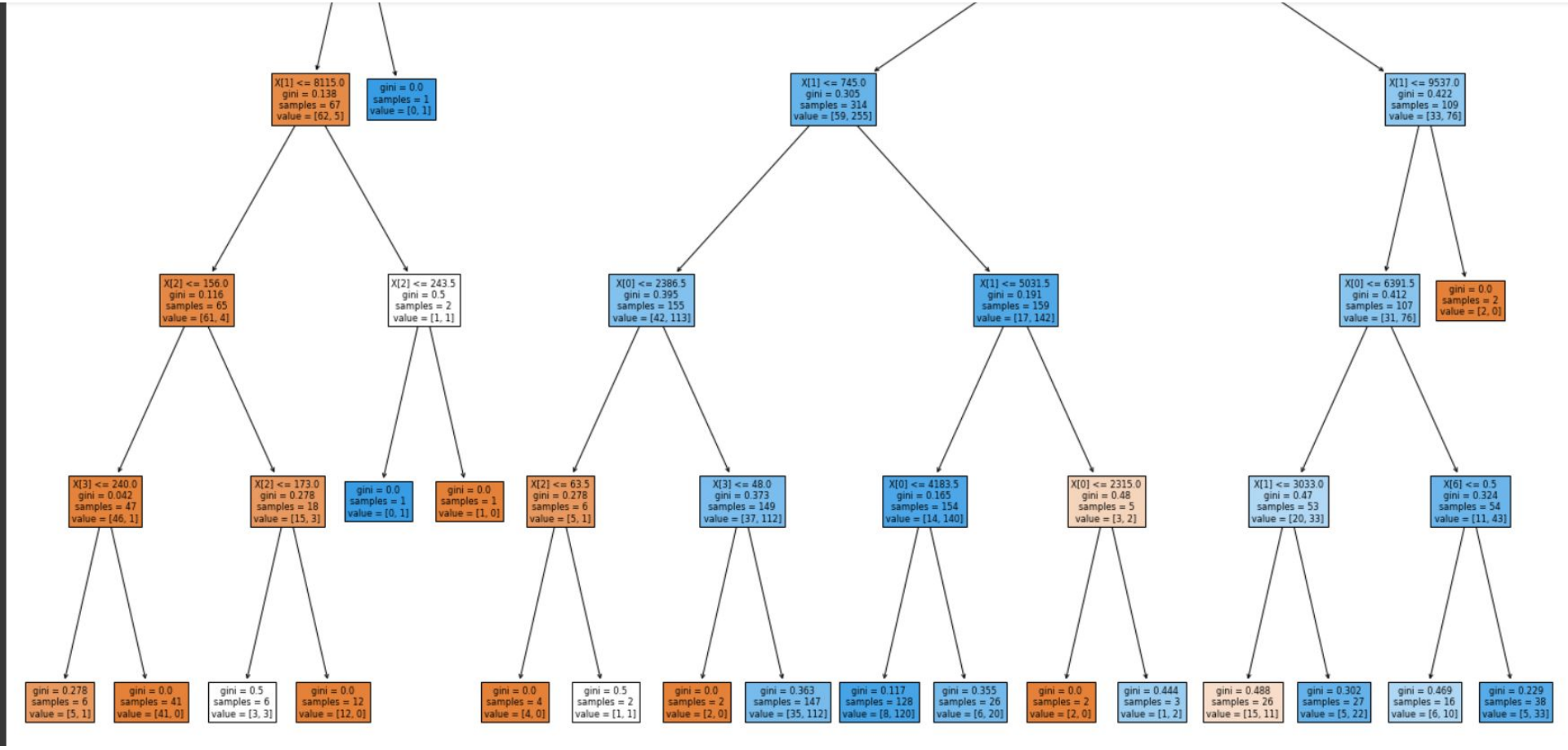
```
Test Data Set Accuracy:  0.7967479674796748
Test Data F1 Score  0.8571428571428571
```

```
[20] cm=np.array(confusion_matrix(y_test,y_test_pred))
     cm
```

```
array([[23, 15],
       [10, 75]])
```

```
[21] fig = plt.figure(figsize=(25,20))
     _=tree.plot_tree(tree_clf,filled=True)
```




X[1] <= 8115.0
gini = 0.138
samples = 67
value = [62, 5]
gini = 0.0
samples = 1
value = [0, 1]
X[2] <= 156.0
gini = 0.116
samples = 65
value = [61, 4]
X[2] <= 243.5
gini = 0.5
samples = 2
value = [1, 1]
X[3] <= 240.0
gini = 0.042
samples = 47
value = [46, 1]
X[2] <= 173.0
gini = 0.278
samples = 18
value = [15, 3]
gini = 0.0
samples = 1
value = [0, 1]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.278
samples = 6
value = [5, 1]
gini = 0.0
samples = 41
value = [41, 0]
gini = 0.5
samples = 6
value = [3, 3]
gini = 0.0
samples = 12
value = [12, 0]
X[1] <= 745.0
gini = 0.305
samples = 314
value = [59, 255]
X[1] <= 9537.0
gini = 0.422
samples = 109
value = [33, 76]
X[0] <= 2386.5
gini = 0.395
samples = 155
value = [42, 113]
X[1] <= 5031.5
gini = 0.191
samples = 159
value = [17, 142]
X[0] <= 6391.5
gini = 0.412
samples = 107
value = [31, 76]
gini = 0.0
samples = 2
value = [2, 0]
X[2] <= 63.5
gini = 0.278
samples = 6
value = [5, 1]
X[3] <= 48.0
gini = 0.373
samples = 149
value = [37, 112]
X[0] <= 4183.5
gini = 0.165
samples = 154
value = [14, 140]
X[0] <= 2315.0
gini = 0.48
samples = 5
value = [3, 2]
X[1] <= 3033.0
gini = 0.47
samples = 53
value = [20, 33]
X[6] <= 0.5
gini = 0.324
samples = 54
value = [11, 43]
gini = 0.0
samples = 4
value = [4, 0]
gini = 0.5
samples = 2
value = [1, 1]
gini = 0.0
samples = 2
value = [2, 0]
gini = 0.363
samples = 147
value = [35, 112]
gini = 0.117
samples = 128
value = [8, 120]
gini = 0.355
samples = 26
value = [6, 20]
gini = 0.0
samples = 2
value = [2, 0]
gini = 0.444
samples = 3
value = [1, 2]
gini = 0.488
samples = 26
value = [15, 11]
gini = 0.302
samples = 27
value = [5, 22]
gini = 0.469
samples = 16
value = [6, 10]
gini = 0.229
samples = 38
value = [5, 33]

[22]
```
# taking max_depth as 3 gave a 5 % increase in accuracy
tree_clf = DecisionTreeClassifier(max_depth = 3,random_state=7)
tree_clf.fit(X_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, 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=7, splitter='best')
```

[23]
```
y_test_pred = tree_clf.predict(X_test)
print("Test Data Set Accuracy: ", accuracy_score(y_test,y_test_pred))
print("Test Data F1 Score ", f1_score(y_test,y_test_pred))
```

```
Test Data Set Accuracy:  0.8455284552845529
Test Data F1 Score  0.8983957219251337
```

[24]
```
cm=np.array(confusion_matrix(y_test,y_test_pred))
cm
```

```
array([[20, 18],
       [ 1, 84]])
```

[25]
```
# min_samples_leaf=40 - giving a miniscule increase in accuracy of 1 % only!
tree_clf = DecisionTreeClassifier(max_depth = 3,random_state=7,min_samples_leaf=40)
tree_clf.fit(X_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=40, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=7, splitter='best')
```

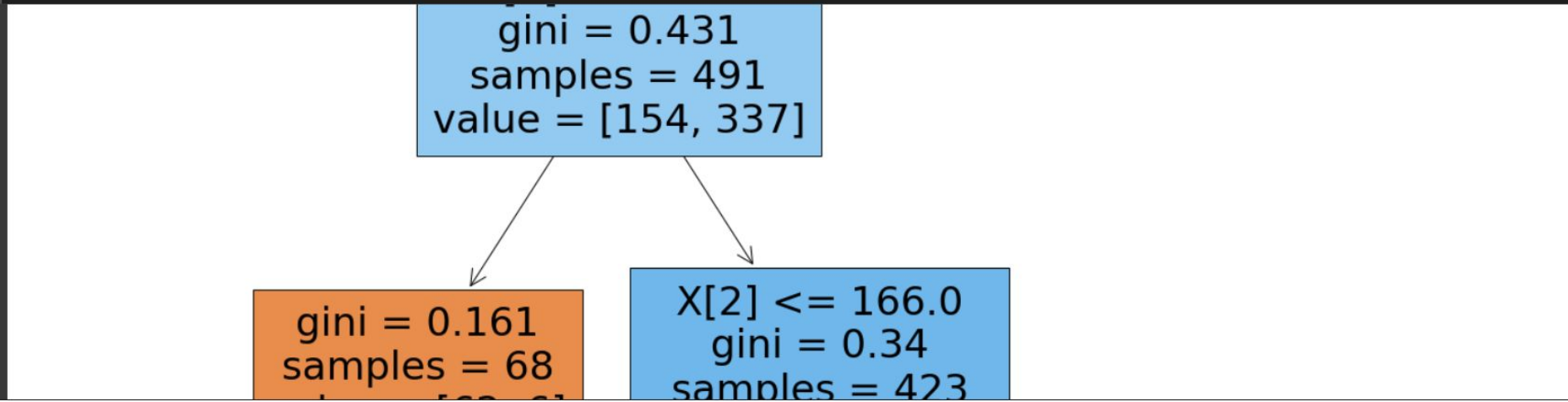
```
[26] y_test_pred = tree_clf.predict(X_test)
     print("Test Data Set Accuracy: ", accuracy_score(y_test,y_test_pred))
     print("Test Data F1 Score ", f1_score(y_test,y_test_pred))
```

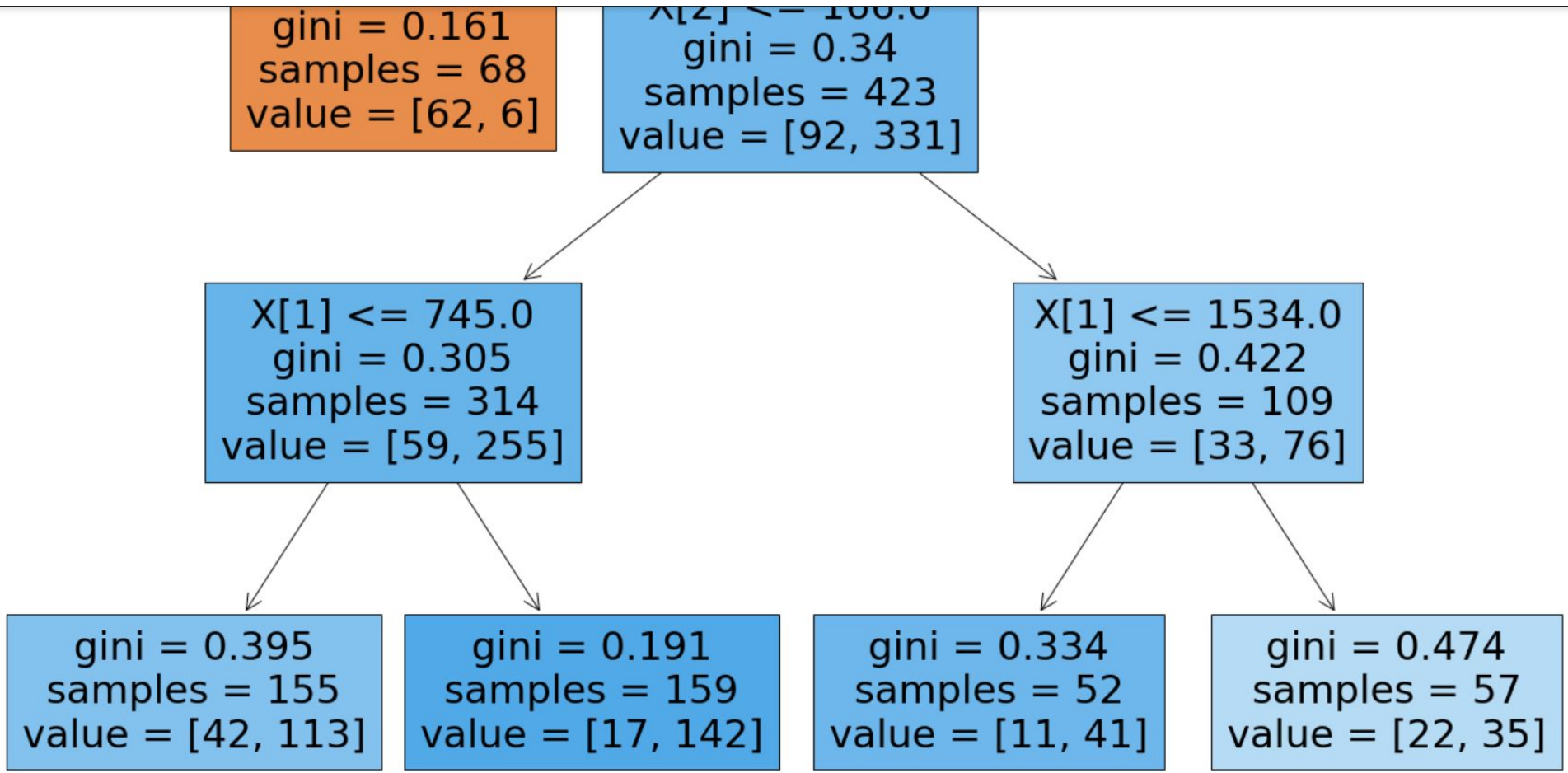
```
Test Data Set Accuracy:  0.8536585365853658
Test Data F1 Score  0.903225806451613
```

```
[27] cm=np.array(confusion_matrix(y_test,y_test_pred))
     cm
```

```
array([[21, 17],
       [ 1, 84]])
```

```
fig = plt.figure(figsize=(25,20))
_=tree.plot_tree(tree_clf,filled=True)
```




gini = 0.161
samples = 68
value = [62, 6]
X[2] <= 166.0
gini = 0.34
samples = 423
value = [92, 331]
X[1] <= 745.0
gini = 0.305
samples = 314
value = [59, 255]
X[1] <= 1534.0
gini = 0.422
samples = 109
value = [33, 76]
gini = 0.395
samples = 155
value = [42, 113]
gini = 0.191
samples = 159
value = [17, 142]
gini = 0.334
samples = 52
value = [11, 41]
gini = 0.474
samples = 57
value = [22, 35]