1. Create a data-frame with three columns (1) Goal (2) num_donors and (3) funding_status

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_excel("Crowdfunding_data_1000_projects (3).xlsx")
```

```
df1 = df.loc[:,["Goal","num_donors","funding_status"]]
```

```
df1.head()
```

	Goal	num_donors	funding_status
0	887.15	7	completed
1	761.52	3	NotCompleted
2	266.55	6	completed
3	808.15	1	NotCompleted
4	1296.65	1	NotCompleted

Convert values in column funding_status from text to integers (completed=1; NotCompleted=0)

```
df1["funding_status"]=np.where(df1.funding_status=="completed",1,0)
df1.head()
```

	Goal	num_donors	funding_status
0	887.15	7	1
1	761.52	3	0
2	266.55	6	1
3	808.15	1	0
4	1296.65	1	0

```
x = df1.iloc[:,[0,1]]
y = df1.iloc[:,2]
```

Perform 70:30 (i.e., 70% training data and remaining test data) split and create two data-frames: (1) train and (2) test. The rows must be selected randomly (2 points)

```
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)
```

2. Use train data-frame to train a decision tree model (2 points).

```
from sklearn.tree import DecisionTreeClassifier
```

```
model =  DecisionTreeClassifier()
```

```
model.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=None, splitter='best')
```

3. Plot the tree (2 points).

```
from sklearn import tree
```

```
decisions = tree.export_text(model)
print(decisions)
```

```
|--- feature_1 <= 0.50
|   |--- class: 0
|--- feature_1 >  0.50
|   |--- feature_1 <= 3.50
|   |   |--- feature_0 <= 243.96
|   |   |   |--- feature_0 <= 204.20
|   |   |   |   |--- feature_0 <= 156.93
|   |   |   |   |   |--- feature_0 <= 156.36
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_0 >  156.36
```

```
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_0 >  156.93
|   |   |   |   |   |--- feature_0 <= 188.02
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_0 >  188.02
|   |   |   |   |   |   |--- feature_0 <= 188.14
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_0 >  188.14
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |--- feature_0 >  204.20
|   |   |   |   |--- feature_0 <= 206.04
|   |   |   |   |   |--- feature_1 <= 2.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_1 >  2.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  206.04
|   |   |   |   |   |--- feature_0 <= 228.75
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- feature_0 >  228.75
|   |   |   |   |   |   |--- feature_0 <= 230.63
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_0 >  230.63
|   |   |   |   |   |   |   |--- feature_0 <= 234.99
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- feature_0 >  234.99
|   |   |   |   |   |   |   |   |--- feature_0 <= 237.99
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- feature_0 >  237.99
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |--- feature_0 >  243.96
|   |   |   |--- feature_1 <= 1.50
|   |   |   |   |--- feature_0 <= 4345.76
|   |   |   |   |   |--- feature_0 <= 320.23
|   |   |   |   |   |   |--- feature_0 <= 275.63
|   |   |   |   |   |   |   |--- feature_0 <= 272.98
|   |   |   |   |   |   |   |   |--- feature_0 <= 264.19
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 254.57
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- feature_0 >  254.57
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 258.43
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  258.43
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- feature_0 >  264.19
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- feature_0 >  272.98
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- feature_0 >  275.63
```

```
1 #fig = plt.figure(figsize=(25,20))
2 #a= tree.plot_tree(model)
```

```
1 import graphviz
2 re = tree.export_graphviz(model)
```

```
graph = graphviz.Source(re, format="png")
graph
```

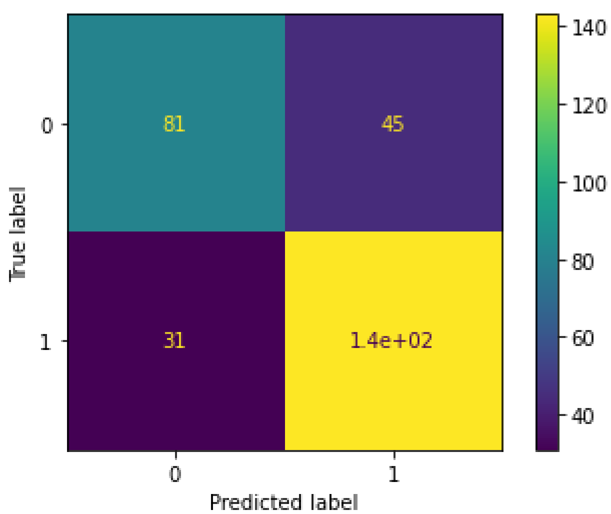
4. Use test data-frame to show confusion matrix and model accuracy (2 points).

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

```
confusion_matrix(y_test,model.predict(x_test))
```

```
array([[ 81,  45],
       [ 31, 143]])
```

```
plot_confusion_matrix(model, x_test, y_test)
plt.show()
```




```
model.score(x_test,y_test)# model accuracy on test data
```

```
0.7466666666666667
```

5. Perform steps 1-4 with two columns: (1) Goal and (2) funding_status, and document the change in accuracy as a comment (2 points).

```
df2 = df1.iloc[:,[0,2]]
df2.head()
```

	Goal	funding_status
0	887.15	1
1	761.52	0
2	266.55	1
3	808.15	0
4	1296.65	0

```
x = df2.iloc[:,[0]]
y = df2.iloc[:,[1]]
```

```
x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(x,y,test_size=0.3)
```

```
model_1 =  DecisionTreeClassifier()
```

```
model_1.fit(x_train_1, y_train_1)
```

```
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=None, splitter='best')
```

```
decisions_1 = tree.export_text(model_1)
print(decisions_1)
```

```
|--- feature_0 <= 611.35
|   |--- feature_0 <= 299.12
|   |   |--- feature_0 <= 185.70
|   |   |   |--- feature_0 <= 156.91
|   |   |   |   |--- feature_0 <= 156.36
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  156.36
|   |   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  156.91
|   |   |   |   |--- feature_0 <= 177.64
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  177.64
|   |   |   |   |   |--- feature_0 <= 178.25
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_0 >  178.25
|   |   |   |   |   |   |--- class: 1
```

```
|   |   |--- feature_0 >  185.70
|   |   |   |--- feature_0 <= 186.25
|   |   |   |   |--- class: 0
|   |   |   |--- feature_0 >  186.25
|   |   |   |   |--- feature_0 <= 265.91
|   |   |   |   |   |--- feature_0 <= 243.96
|   |   |   |   |   |   |--- feature_0 <= 213.50
|   |   |   |   |   |   |   |--- feature_0 <= 210.58
|   |   |   |   |   |   |   |   |--- feature_0 <= 206.04
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 204.20
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 198.43
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 6
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  198.43
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- feature_0 >  204.20
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 204.66
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  204.66
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |--- feature_0 >  206.04
|   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- feature_0 >  210.58
|   |   |   |   |   |   |   |   |--- feature_0 <= 210.99
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- feature_0 >  210.99
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 211.68
|   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |--- feature_0 >  211.68
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 212.35
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  212.35
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |--- feature_0 >  213.50
|   |   |   |   |   |   |   |--- feature_0 <= 230.91
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- feature_0 >  230.91
|   |   |   |   |   |   |   |   |--- feature_0 <= 237.78
|   |   |   |   |   |   |   |   |   |--- feature_0 <= 235.23
|   |   |   |   |   |   |   |   |   |   |--- feature_0 <= 232.28
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |   |--- feature_0 >  232.28
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
```

```
import graphviz
re = tree.export_graphviz(model_1)
```

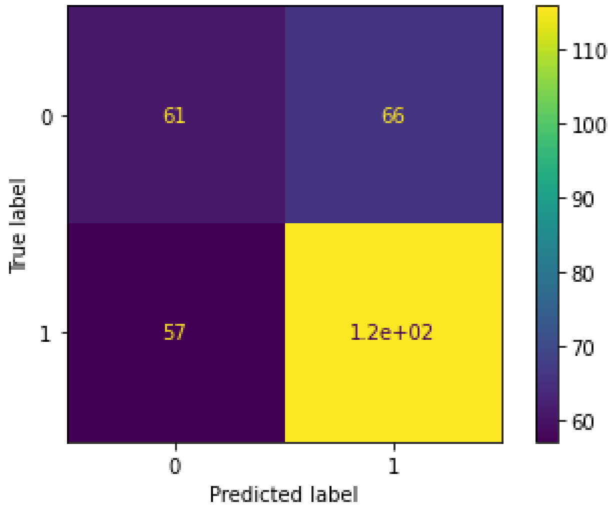
```
graph = graphviz.Source(re, format="png")
graph
```

```
confusion_matrix(y_test_1,model_1.predict(x_test_1))
```

```
array([[ 61,  66],
```

```
       [ 57, 116]])
```

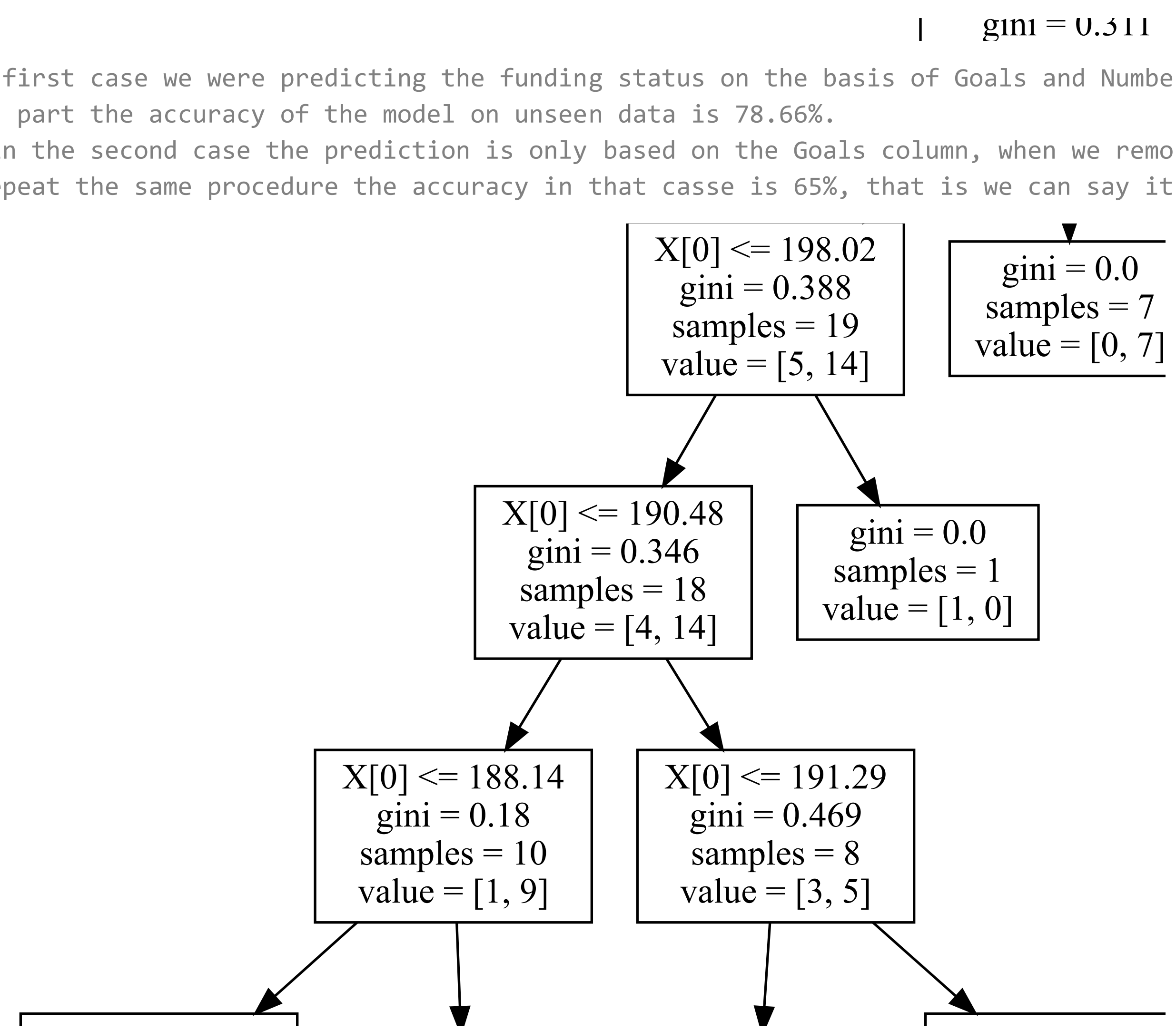
```
plot_confusion_matrix(model_1, x_test_1, y_test_1)
plt.show()
```

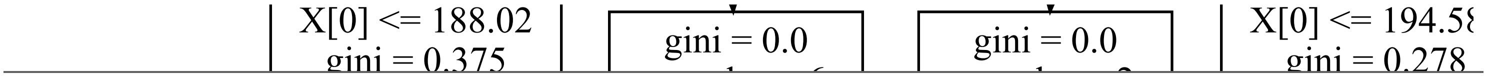



```
model_1.score(x_test_1,y_test_1)# model accuracy on test data
```

```
0.59
```

```
#  In the first case we were predicting the funding status on the basis of Goals and Numbe
#  in this part the accuracy of the model on unseen data is 78.66%.
#  while in the second case the prediction is only based on the Goals column, when we remo
#  then repeat the same procedure the accuracy in that casse is 65%, that is we can say it
```




X[0] <= 188.02
gini = 0.375
gini = 0.0
gini = 0.0
gini = 0.278