letsgrowmore

NOV 2021

Intermediate Level Task 2 : Prediction Using Decision Tree Algorithm

AIM: Task Goal: Create the Decision Tree Classifier and visualize it graphically

importing libraries

In [2]:

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

load the given dataset

In [3]:

```
data=pd.read_csv('irisdata.csv')
```

In [4]:

```
data
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [5]:

```
data.head()
```

Out[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [6]:

```
data.head(10)
```

Out[6]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa
6	7	4.6	3.4	1.4	0.3	Iris-setosa
7	8	5.0	3.4	1.5	0.2	Iris-setosa
8	9	4.4	2.9	1.4	0.2	Iris-setosa
9	10	4.9	3.1	1.5	0.1	Iris-setosa

In [7]:

```
data.tail()
```

Out[7]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [8]:

```
data.tail(10)
```

Out[8]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
140	141	6.7	3.1	5.6	2.4	Iris-virginica
141	142	6.9	3.1	5.1	2.3	Iris-virginica
142	143	5.8	2.7	5.1	1.9	Iris-virginica
143	144	6.8	3.2	5.9	2.3	Iris-virginica
144	145	6.7	3.3	5.7	2.5	Iris-virginica
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

In [9]:

```
data.drop('Id',axis=1)
```

Out[9]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 5 columns

In [10]:
```
data.shape
```
Out[10]:
```
(150, 6)
```

checking the given data with NaN and unknown values (preprocessing)

In [11]:
```
data.isnull()
```
Out[11]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
145	False	False	False	False	False	False
146	False	False	False	False	False	False
147	False	False	False	False	False	False
148	False	False	False	False	False	False
149	False	False	False	False	False	False

150 rows × 6 columns

In [12]:
```
data.isnull().sum()
```
Out[12]:
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

saperating the target values from given data

In [15]:
```
x=data.iloc[:, [0, 1, 2, 3]].values
y=data.iloc[:,-1].values
```

In [16]:
```
print(x)
```
```
[[  1.    5.1   3.5   1.4]
 [  2.    4.9   3.    1.4]
 [  3.    4.7   3.2   1.3]
 [  4.    4.6   3.1   1.5]
 [  5.    5.    3.6   1.4]
 [  6.    5.4   3.9   1.7]
 [  7.    4.6   3.4   1.4]
 [  8.    5.    3.4   1.5]
 [  9.    4.4   2.9   1.4]
 [ 10.    4.9   3.1   1.5]
 [ 11.    5.4   3.7   1.5]
 [ 12.    4.8   3.4   1.6]
 [ 13.    4.8   3.    1.4]
 [ 14.    4.3   3.    1.1]
 [ 15.    5.8   4.    1.2]
 [ 16.    5.7   4.4   1.5]
 [ 17.    5.4   3.9   1.3]
 [ 18.    5.1   3.5   1.4]
 [ 19.    5.7   3.8   1.7]
 [ 20.    5.1   3.8   1.5]
```

```
 [ 21.    5.4   3.4   1.7]
 [ 22.    5.1   3.7   1.5]
 [ 23.    4.6   3.6   1. ]
 [ 24.    5.1   3.3   1.7]
 [ 25.    4.8   3.4   1.9]
 [ 26.    5.    3.    1.6]
 [ 27.    5.    3.4   1.6]
 [ 28.    5.2   3.5   1.5]
 [ 29.    5.2   3.4   1.4]
 [ 30.    4.7   3.2   1.6]
 [ 31.    4.8   3.1   1.6]
 [ 32.    5.4   3.4   1.5]
 [ 33.    5.2   4.1   1.5]
 [ 34.    5.5   4.2   1.4]
 [ 35.    4.9   3.1   1.5]
 [ 36.    5.    3.2   1.2]
 [ 37.    5.5   3.5   1.3]
 [ 38.    4.9   3.1   1.5]
 [ 39.    4.4   3.    1.3]
 [ 40.    5.1   3.4   1.5]
 [ 41.    5.    3.5   1.3]
 [ 42.    4.5   2.3   1.3]
 [ 43.    4.4   3.2   1.3]
 [ 44.    5.    3.5   1.6]
 [ 45.    5.1   3.8   1.9]
 [ 46.    4.8   3.    1.4]
 [ 47.    5.1   3.8   1.6]
 [ 48.    4.6   3.2   1.4]
 [ 49.    5.3   3.7   1.5]
 [ 50.    5.    3.3   1.4]
 [ 51.    7.    3.2   4.7]
 [ 52.    6.4   3.2   4.5]
 [ 53.    6.9   3.1   4.9]
 [ 54.    5.5   2.3   4. ]
 [ 55.    6.5   2.8   4.6]
 [ 56.    5.7   2.8   4.5]
 [ 57.    6.3   3.3   4.7]
 [ 58.    4.9   2.4   3.3]
 [ 59.    6.6   2.9   4.6]
 [ 60.    5.2   2.7   3.9]
 [ 61.    5.    2.    3.5]
 [ 62.    5.9   3.    4.2]
 [ 63.    6.    2.2   4. ]
 [ 64.    6.1   2.9   4.7]
 [ 65.    5.6   2.9   3.6]
 [ 66.    6.7   3.1   4.4]
 [ 67.    5.6   3.    4.5]
 [ 68.    5.8   2.7   4.1]
 [ 69.    6.2   2.2   4.5]
 [ 70.    5.6   2.5   3.9]
 [ 71.    5.9   3.2   4.8]
 [ 72.    6.1   2.8   4. ]
 [ 73.    6.3   2.5   4.9]
 [ 74.    6.1   2.8   4.7]
 [ 75.    6.4   2.9   4.3]
 [ 76.    6.6   3.    4.4]
 [ 77.    6.8   2.8   4.8]
 [ 78.    6.7   3.    5. ]
 [ 79.    6.    2.9   4.5]
 [ 80.    5.7   2.6   3.5]
 [ 81.    5.5   2.4   3.8]
 [ 82.    5.5   2.4   3.7]
 [ 83.    5.8   2.7   3.9]
 [ 84.    6.    2.7   5.1]
 [ 85.    5.4   3.    4.5]
 [ 86.    6.    3.4   4.5]
 [ 87.    6.7   3.1   4.7]
 [ 88.    6.3   2.3   4.4]
 [ 89.    5.6   3.    4.1]
 [ 90.    5.5   2.5   4. ]
 [ 91.    5.5   2.6   4.4]
 [ 92.    6.1   3.    4.6]
 [ 93.    5.8   2.6   4. ]
 [ 94.    5.    2.3   3.3]
 [ 95.    5.6   2.7   4.2]
 [ 96.    5.7   3.    4.2]
 [ 97.    5.7   2.9   4.2]
 [ 98.    6.2   2.9   4.3]
 [ 99.    5.1   2.5   3. ]
 [100.    5.7   2.8   4.1]
 [101.    6.3   3.3   6. ]
 [102.    5.8   2.7   5.1]
 [103.    7.1   3.    5.9]
```

```
 [104.    6.3   2.9   5.6]
 [105.    6.5   3.    5.8]
 [106.    7.6   3.    6.6]
 [107.    4.9   2.5   4.5]
 [108.    7.3   2.9   6.3]
 [109.    6.7   2.5   5.8]
 [110.    7.2   3.6   6.1]
 [111.    6.5   3.2   5.1]
 [112.    6.4   2.7   5.3]
 [113.    6.8   3.    5.5]
 [114.    5.7   2.5   5. ]
 [115.    5.8   2.8   5.1]
 [116.    6.4   3.2   5.3]
 [117.    6.5   3.    5.5]
 [118.    7.7   3.8   6.7]
 [119.    7.7   2.6   6.9]
 [120.    6.    2.2   5. ]
 [121.    6.9   3.2   5.7]
 [122.    5.6   2.8   4.9]
 [123.    7.7   2.8   6.7]
 [124.    6.3   2.7   4.9]
 [125.    6.7   3.3   5.7]
 [126.    7.2   3.2   6. ]
 [127.    6.2   2.8   4.8]
 [128.    6.1   3.    4.9]
 [129.    6.4   2.8   5.6]
 [130.    7.2   3.    5.8]
 [131.    7.4   2.8   6.1]
 [132.    7.9   3.8   6.4]
 [133.    6.4   2.8   5.6]
 [134.    6.3   2.8   5.1]
 [135.    6.1   2.6   5.6]
 [136.    7.7   3.    6.1]
 [137.    6.3   3.4   5.6]
 [138.    6.4   3.1   5.5]
 [139.    6.    3.    4.8]
 [140.    6.9   3.1   5.4]
 [141.    6.7   3.1   5.6]
 [142.    6.9   3.1   5.1]
 [143.    5.8   2.7   5.1]
 [144.    6.8   3.2   5.9]
 [145.    6.7   3.3   5.7]
 [146.    6.7   3.    5.2]
 [147.    6.3   2.5   5. ]
 [148.    6.5   3.    5.2]
 [149.    6.2   3.4   5.4]
 [150.    5.9   3.    5.1]]
```

In [17]:
```
x.shape
```

Out[17]: (150, 4)

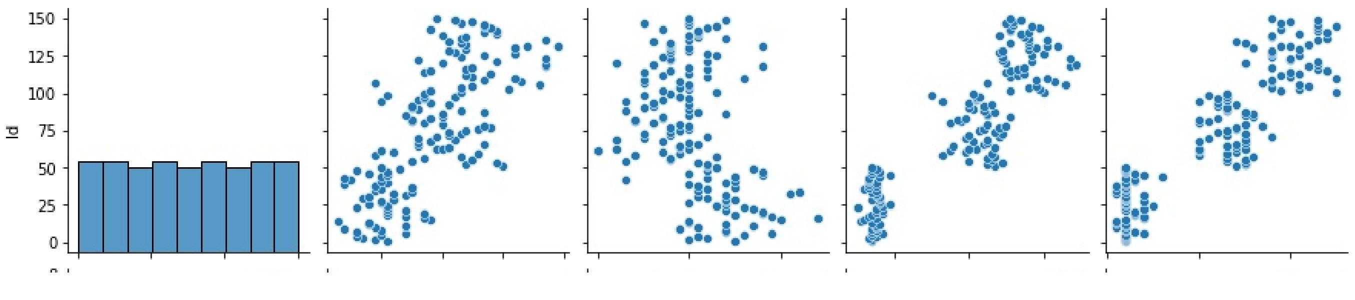
In [18]:
```
y.shape
```

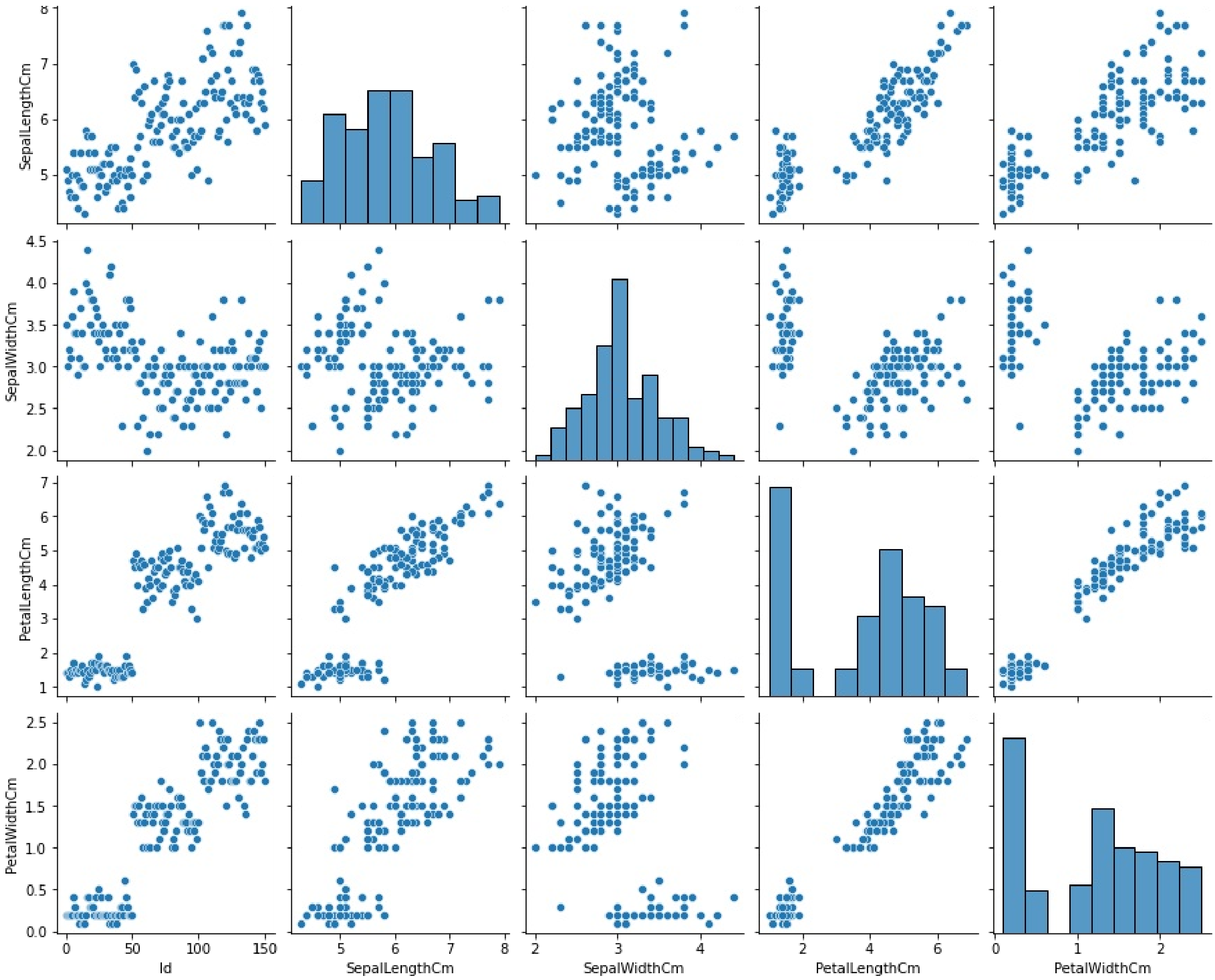
Out[18]: (150,)

analysing data

In [20]:
```
sns.pairplot(data)
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x1cd016627f0>





Splitting given dataset into training data and test data

In [49]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
```

In [22]:

```
L=LabelEncoder()
y=L.fit_transform(y)
y
```

Out[22]:

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

In [24]:

```
X_train, X_test, y_train, y_test = train_test_split(x,y,random_state=0,test_size=0.2)
```

Decision Tree Algorithm

In [25]:

```
dt=DecisionTreeClassifier()
```

In [27]:

```
#Train our model
```

```
dt.fit(X_train,y_train)
```

Out[27]: DecisionTreeClassifier()

In [28]:
```
#Training score
dt.score(X_train,y_train)
```

Out[28]: 1.0

Making Predictions of given dataset

In [29]:
```
pred=dt.predict(X_test)
```

In [30]:
```
print(pred)
```

```
[2 1 0 2 0 1 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0]
```

comparing Actual Vs predicted values of data

In [31]:
```
c=pd.DataFrame({"Actual":y_test,"Predicted":pred})
```

In [32]:
```
print(c)
```

```
    Actual  Predicted
0        2          2
1        1          1
2        0          0
3        2          2
4        0          0
5        2          1
6        0          0
7        1          1
8        1          1
9        1          1
10       2          2
11       1          1
12       1          1
13       1          1
14       1          1
15       0          0
16       1          1
17       1          1
18       0          0
19       0          0
20       2          2
21       1          1
22       0          0
23       0          0
24       2          2
25       0          0
26       0          0
27       1          1
28       1          1
29       0          0
```

checking the Accuracy of model

In [34]:
```
from sklearn.metrics import accuracy_score
```

In [40]:
```
print("the score on test data {}".format(accuracy_score(y_test,pred)))
```

```
the score on test data 0.9666666666666667
```

Visualising the Decision Tree

In [43]:
```
!pip install pydotplus
!apt-get install graphviz -y
```

```
Collecting pydotplus
  Downloading pydotplus-2.0.2.tar.gz (278 kB)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\korra srinu\anaconda3\lib\site-packages (from pydotpl
us) (2.4.7)
Building wheels for collected packages: pydotplus
  Building wheel for pydotplus (setup.py): started
  Building wheel for pydotplus (setup.py): finished with status 'done'
  Created wheel for pydotplus: filename=pydotplus-2.0.2-py3-none-any.whl size=24566 sha256=9f084a0eb9cccca4435637
301af5ebb35d38fd643c1e7bb4b582b7e306acd8ba
  Stored in directory: c:\users\korra srinu\appdata\local\pip\cache\wheels\fe\cd\78\a7e873cc049759194f8271f780640
cf96b35e5a48bef0e2f36
Successfully built pydotplus
Installing collected packages: pydotplus
Successfully installed pydotplus-2.0.2
```

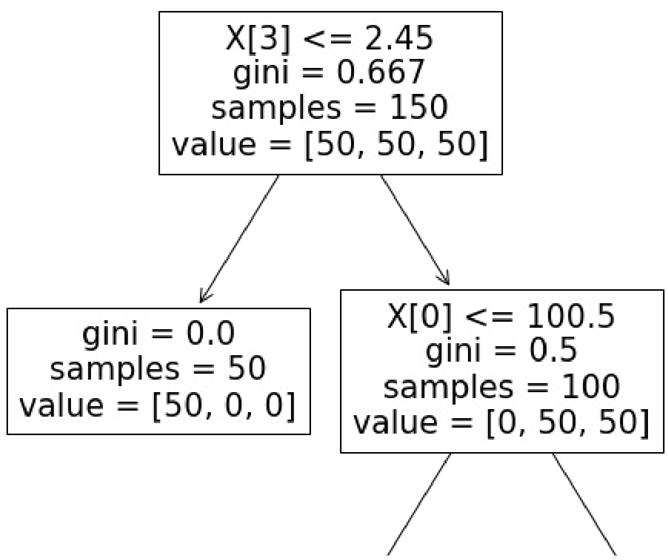
```
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken b
y 'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An existing connection was forcibly closed b
y the remote host', None, 10054, None))': /simple/pydotplus/
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, status=None)) after connection broken b
y 'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An existing connection was forcibly closed b
y the remote host', None, 10054, None))': /simple/pydotplus/
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, status=None)) after connection broken b
y 'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An existing connection was forcibly closed b
y the remote host', None, 10054, None))': /simple/pydotplus/
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, status=None)) after connection broken b
y 'ProtocolError('Connection aborted.', ConnectionResetError(10054, 'An existing connection was forcibly closed b
y the remote host', None, 10054, None))': /simple/pydotplus/
'apt-get' is not recognized as an internal or external command,
operable program or batch file.
```

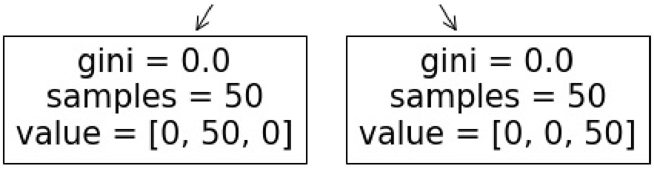
In [44]:
```
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

In [58]:
```
treestr=DecisionTreeClassifier()
treestr.fit(x,y)
```

Out[58]:
```
DecisionTreeClassifier()
```

In [75]:
```
fig,ax = plt.subplots(figsize=(10, 10))
tree.plot_tree(treestr)
plt.show()
```





In [61]:
```
#predicting of new entry
```

In [70]:
```
X_new=np.array([2,4,2,0]).reshape(1,-1)
```

In [71]:
```
pred=dt.predict(X_new)
```

In [72]:
```
print(pred)
```

```
[0]
```

In [73]:
```
print("the new entry belong to {}".format(pred))
```

```
the new entry belong to [0]
```

the accuracy score on test data is 0.9666666666666667

In [ ]:
```
```