Decision Tree

(Code: Subhajit Das)

What is Decision Tree:

A Decision Tree is a supervised learning technique used for both classification and regression problems. It's a tree-structured classifier where internal nodes represent the features of a dataset, branches represent decision rules, and each leaf node represents the outcome.

The decision tree starts with a root node, expands on further branches, and constructs a tree-like structure. The decisions or tests are performed based on the features of the given dataset.

Here are some terminologies used in Decision Trees:

1. **Root Node**: It represents the entire dataset, which further gets divided into two or more homogeneous sets.
2. **Leaf Node**: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
3. **Splitting**: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
4. **Branch/Sub Tree**: A tree formed by splitting the tree.
5. **Pruning**: Pruning is the process of removing the unwanted branches from the tree.
6. **Parent/Child node**: The root node of the tree is called the parent node, and other nodes are called the child nodes.

The decision tree algorithm works by comparing the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch corresponding to the condition and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and moves further.

Where we can use Decision Tree:

1. **Marketing**: Businesses can use decision trees to enhance the accuracy of their promotional campaigns by observing the performance of their competitors' products and services. Decision trees can help in audience segmentation and support businesses in producing better-targeted advertisements that have higher conversion rates.
2. **Retention of Customers**: Companies use decision trees for customer retention through analyzing their behaviors and releasing new offers or products to suit those behaviors. By using decision tree models, companies can figure out the satisfaction levels of their customers as well.
3. **Diagnosis of Diseases and Ailments**: Decision trees can help physicians and medical professionals in identifying patients that are at a higher risk of developing serious (or preventable) conditions such as diabetes or dementia. The ability of decision trees to narrow down possibilities according to specific variables is quite helpful in such cases.
4. **Detection of Frauds**: Companies can prevent fraud by using decision trees to identify fraudulent behavior beforehand. It can save companies a lot of resources, including time and money.

Use of Decision Tree in Supervised Learning:

Supervised Learning: Decision Trees are one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks1. They build a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met.

In [ ]:
```
import pandas as pd
import numpy as np
```

In [ ]:
```
tennis_df = pd.read_csv("/content/drive/MyDrive/ML and DL DataSets/6_Play_Te
tennis_df
```

Out[56]:

	outlook	temp	humidity	wind	play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes
5	Rain	Cool	Normal	Strong	No
6	Overcast	Cool	Normal	Strong	Yes
7	Sunny	Mild	High	Weak	No
8	Sunny	Cool	Normal	Weak	Yes
9	Rain	Mild	Normal	Weak	Yes
10	Sunny	Mild	Normal	Strong	Yes
11	Overcast	Mild	High	Strong	Yes
12	Overcast	Hot	Normal	Weak	Yes
13	Rain	Mild	High	Strong	No

LabelEncoder

In [ ]:
```
'''LabelEncoder is a technique used to convert categorical columns into nume
from sklearn.preprocessing import LabelEncoder # creating an instance of the
```

In [ ]:
```
Le = LabelEncoder()
```

In [ ]:
```
# 1. fit the encoder using the '.fit()' method on your data. This determines
# 2. transform your data into numerical labels using the '.transform()' meth

#  A categorical feature with values ‘low’, ‘medium’, and ‘high’, LabelEncod
tennis_df['outlook'] = Le.fit_transform(tennis_df['outlook'])
tennis_df['temp'] = Le.fit_transform(tennis_df['temp'])
tennis_df['humidity'] = Le.fit_transform(tennis_df['humidity'])
tennis_df['wind'] = Le.fit_transform(tennis_df['wind'])
tennis_df['play'] = Le.fit_transform(tennis_df['play'])
tennis_df
```

Out[5]:

	outlook	temp	humidity	wind	play
0	2	1	0	1	0
1	2	1	0	0	0
2	0	1	0	1	1
3	1	2	0	1	1
4	1	0	1	1	1
5	1	0	1	0	0
6	0	0	1	0	1
7	2	2	0	1	0
8	2	0	1	1	1
9	1	2	1	1	1
10	2	2	1	0	1
11	0	2	0	0	1
12	0	1	1	1	1
13	1	2	0	0	0

In [ ]:
```
''' Drawback of LabelEncoder: A label with a high value may be considered to
```

Out[6]:
```
' Drawback of LabelEncoder: A label with a high value may be considered to
have high priority than a label having a lower value '
```

Separating Dependent and Independent variables

In [ ]:
```
features = ['outlook', 'temp', 'humidity', 'wind']
x = tennis_df[features]
y = tennis_df.play
```

In [ ]:
```
x # Viewing the x variable(features)
```

Out[8]:

	outlook	temp	humidity	wind
0	2	1	0	1
1	2	1	0	0
2	0	1	0	1
3	1	2	0	1
4	1	0	1	1
5	1	0	1	0
6	0	0	1	0
7	2	2	0	1
8	2	0	1	1
9	1	2	1	1
10	2	2	1	0
11	0	2	0	0
12	0	1	1	1
13	1	2	0	0

In [ ]:
```
y # Viewing the y variable(class)
```

Out[9]:
```
0     0
1     0
2     1
3     1
4     1
5     0
6     1
7     0
8     1
9     1
10    1
11    1
12    1
13    0
Name: play, dtype: int64
```

Spliting train and test datasets

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x_le_train, x_le_test, y_le_train, y_le_test = train_test_split(x, y, test_s
```

Using Decision Tree Classifier

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
Decision_le = DecisionTreeClassifier(criterion = 'gini')
```

In [ ]:
```
Decision_le.fit(x_le_train, y_le_train)
```

Out[14]:
```
DecisionTreeClassifier()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
Decision_le.predict(x_le_test)
```

Out[15]:
```
array([1, 1, 1])
```

In [ ]:
```
x_le_test
```

Out[16]:

	outlook	temp	humidity	wind
1	2	1	0	0
4	1	0	1	1
7	2	2	0	1

In [ ]:
```
y_le_test
```

Out[17]:
```
1    0
4    1
7    0
Name: play, dtype: int64
```

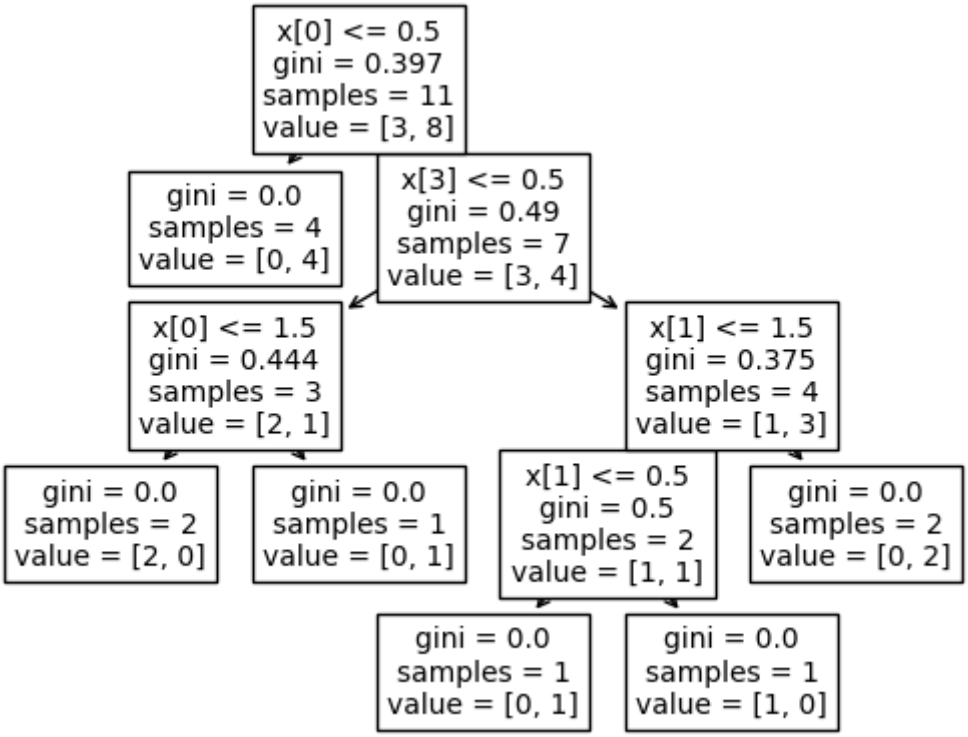
In [ ]:
```
Decision_le.score(x_le_test, y_le_test)
```

Out[18]:
```
0.3333333333333333
```

Printing leaf and nodes

In [ ]:
```
from sklearn import tree
tree.plot_tree(Decision_le)
```

Out[19]:
```
[Text(0.375, 0.9, 'x[0] <= 0.5\ngini = 0.397\nsamples = 11\nvalue = [3,
8]'),
 Text(0.25, 0.7, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.5, 0.7, 'x[3] <= 0.5\ngini = 0.49\nsamples = 7\nvalue = [3, 4]'),
 Text(0.25, 0.5, 'x[0] <= 1.5\ngini = 0.444\nsamples = 3\nvalue = [2,
1]'),
 Text(0.125, 0.3, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.375, 0.3, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.75, 0.5, 'x[1] <= 1.5\ngini = 0.375\nsamples = 4\nvalue = [1,
3]'),
 Text(0.625, 0.3, 'x[1] <= 0.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.5, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.75, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.875, 0.3, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]')]
```



OneHotEncoder ((For the drawback of LabelEncoder we are using this)

In [ ]:
```
''' One-hot encoding is a process of converting categorical data variables s
    we convert each categorical value into a new categorical value and assig

    For ex: if we have a “color” feature which can have values “red”, “green
    features - ‘is_red’, ‘is_green’ and ‘is_blue’. If the color is red, ther

from sklearn.preprocessing import OneHotEncoder
```

In [ ]:
```
Ohe = OneHotEncoder(drop = 'first')
```

In [ ]:
```
''' Reason of removing 1st column in OneHotEncoder: The idea of dropping the
such as when feeding the resulting data into an unregularized linear model.
```

Out[22]:
```
' Reason of removing 1st column in OneHotEncoder: The idea of dropping the
first column is useful in situations where perfectly collinear features ca
use problems,\nsuch as when feeding the resulting data into an unregulariz
ed linear model. However, a principle of machine learning is to build a hi
ghly predictive model'
```

In [ ]:
```
# 1. Fit the encoder using the .fit() method on your data. This determines t
# 2. Transform your data into one-hot encoded vectors using the .transform()

# A categorical feature with values ‘low’, ‘medium’, and ‘high’, OneHotEncod
tennis_df['outlook'] = Ohe.fit_transform(tennis_df[['outlook']]).toarray()
tennis_df['temp'] = Ohe.fit_transform(tennis_df[['temp']]).toarray()
tennis_df['humidity'] = Ohe.fit_transform(tennis_df[['humidity']]).toarray()
tennis_df['wind'] = Ohe.fit_transform(tennis_df[['wind']]).toarray()
tennis_df['play'] = Ohe.fit_transform(tennis_df[['play']]).toarray()
tennis_df
```

Out[24]:

	outlook	temp	humidity	wind	play
0	0.0	1.0	0.0	1.0	0.0
1	0.0	1.0	0.0	0.0	0.0
2	0.0	1.0	0.0	1.0	1.0
3	1.0	0.0	0.0	1.0	1.0
4	1.0	0.0	1.0	1.0	1.0
5	1.0	0.0	1.0	0.0	0.0
6	0.0	0.0	1.0	0.0	1.0
7	0.0	0.0	0.0	1.0	0.0
8	0.0	0.0	1.0	1.0	1.0
9	1.0	0.0	1.0	1.0	1.0
10	0.0	0.0	1.0	0.0	1.0
11	0.0	0.0	0.0	0.0	1.0
12	0.0	1.0	1.0	1.0	1.0
13	1.0	0.0	0.0	0.0	0.0

In [ ]:
```
pd.get_dummies(tennis_df['outlook']).head() # Viewing the data in One Hot Er
```

Out[54]:

	0.0	1.0
0	1	0
1	1	0
2	1	0
3	0	1
4	0	1
5	0	1
6	1	0
7	1	0
8	1	0
9	0	1
10	1	0
11	1	0
12	1	0
13	0	1

Separating Dependent and Independent variables

In [ ]:
```
features = ['outlook', 'temp', 'humidity', 'wind']
x = tennis_df[features]
y = tennis_df.play
```

In [ ]:
```
x # Viewing the x variable(features)
```

Out[26]:

	outlook	temp	humidity	wind
0	0.0	1.0	0.0	1.0
1	0.0	1.0	0.0	0.0
2	0.0	1.0	0.0	1.0
3	1.0	0.0	0.0	1.0
4	1.0	0.0	1.0	1.0
5	1.0	0.0	1.0	0.0
6	0.0	0.0	1.0	0.0
7	0.0	0.0	0.0	1.0
8	0.0	0.0	1.0	1.0
9	1.0	0.0	1.0	1.0
10	0.0	0.0	1.0	0.0
11	0.0	0.0	0.0	0.0
12	0.0	1.0	1.0	1.0
13	1.0	0.0	0.0	0.0

In [ ]:
```
y # Viewing the y variable(class)
```

Out[27]:
```
0     0.0
1     0.0
2     1.0
3     1.0
4     1.0
5     0.0
6     1.0
7     0.0
8     1.0
9     1.0
10    1.0
11    1.0
12    1.0
13    0.0
Name: play, dtype: float64
```

Spliting train and test datasets

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
x_ohe_train, x_ohe_test, y_ohe_train, y_ohe_test = train_test_split(x, y, te
```

Using Decision Tree Classifier

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [ ]:
```
Decision_ohe = DecisionTreeClassifier(criterion = 'gini')
```

In [ ]:
```
Decision_ohe.fit(x_ohe_train, y_ohe_train)
```

Out[32]:
```
DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
Decision_ohe.predict(x_ohe_test)
```

Out[33]:
```
array([1., 1., 0.])
```

In [ ]:
```
x_ohe_test
```

Out[34]:

	outlook	temp	humidity	wind
7	0.0	0.0	0.0	1.0
8	0.0	0.0	1.0	1.0
11	0.0	0.0	0.0	0.0

In [ ]:
```
y_ohe_test
```

Out[35]:
```
7     0.0
8     1.0
11    1.0
Name: play, dtype: float64
```

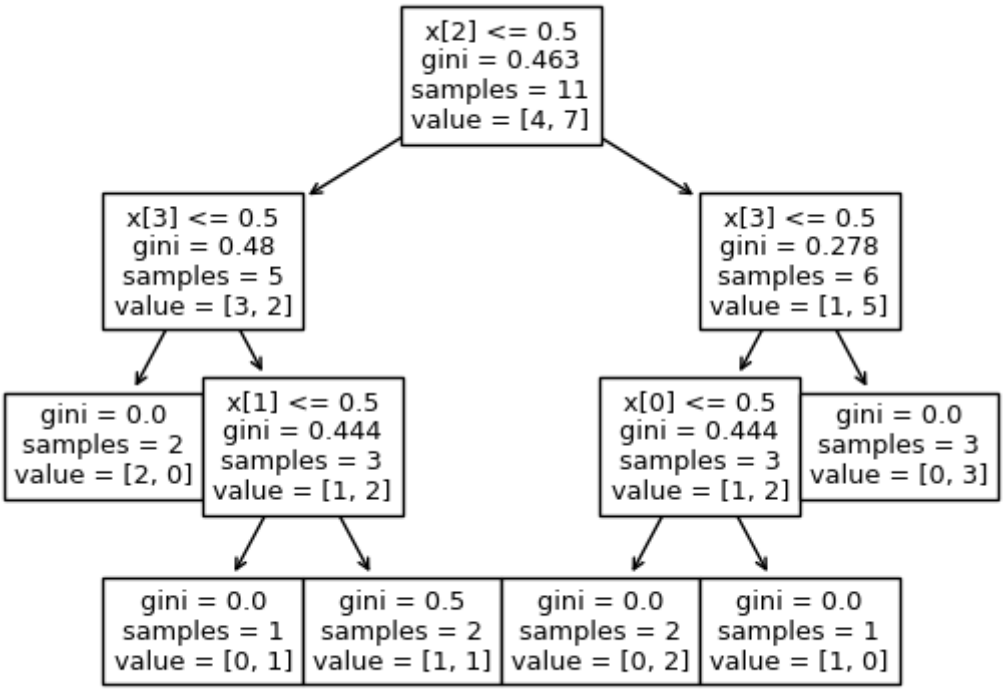
In [ ]:
```
Decision_ohe.score(x_ohe_test, y_ohe_test)
```

Out[36]:
```
0.3333333333333333
```

Printing leaf and nodes

In [ ]:
```
from sklearn import tree
tree.plot_tree(Decision_ohe)
```

Out[37]:
```
[Text(0.5, 0.875, 'x[2] <= 0.5\ngini = 0.463\nsamples = 11\nvalue = [4,
7]'),
 Text(0.2, 0.625, 'x[3] <= 0.5\ngini = 0.48\nsamples = 5\nvalue = [3,
2]'),
 Text(0.1, 0.375, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.3, 0.375, 'x[1] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [1,
2]'),
 Text(0.2, 0.125, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.4, 0.125, 'gini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.8, 0.625, 'x[3] <= 0.5\ngini = 0.278\nsamples = 6\nvalue = [1,
5]'),
 Text(0.7, 0.375, 'x[0] <= 0.5\ngini = 0.444\nsamples = 3\nvalue = [1,
2]'),
 Text(0.6, 0.125, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.8, 0.125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.9, 0.375, 'gini = 0.0\nsamples = 3\nvalue = [0, 3]')]
```



Using Gini Index in Decision Tree with LabelEncoder

In [ ]:
```
Decision_le_ent = DecisionTreeClassifier(criterion = 'entropy')
```

In [ ]:
```
Decision_le_ent.fit(x_le_train, y_le_train)
```

Out[39]:
```
DecisionTreeClassifier(criterion='entropy')
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
Decision_le_ent.predict(x_le_test)
```

Out[40]:
```
array([1, 1, 1])
```

In [ ]:
```
x_le_test
```

Out[41]:

	outlook	temp	humidity	wind
1	2	1	0	0
4	1	0	1	1
7	2	2	0	1

In [ ]:
```
y_le_test
```

Out[42]:
```
1    0
4    1
7    0
Name: play, dtype: int64
```

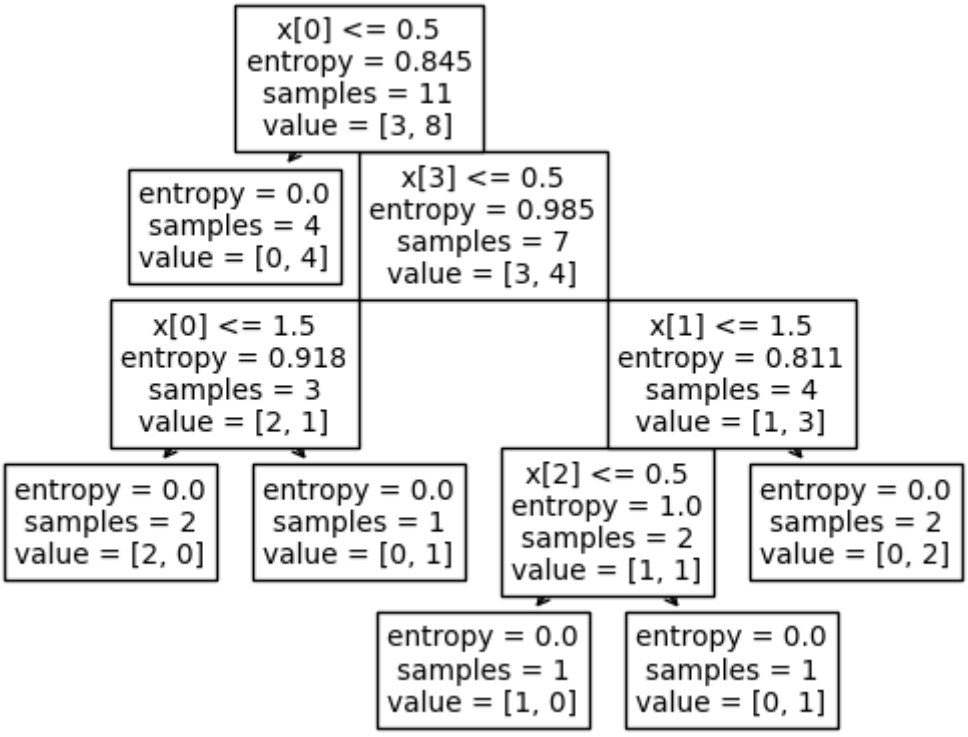
In [ ]:
```
Decision_le_ent.score(x_le_test, y_le_test)
```

Out[43]:
```
0.3333333333333333
```

Printing leaf and nodes

In [ ]:
```
from sklearn import tree
tree.plot_tree(Decision_le_ent)
```

Out[44]:
```
[Text(0.375, 0.9, 'x[0] <= 0.5\nentropy = 0.845\nsamples = 11\nvalue = [3,
8]'),
 Text(0.25, 0.7, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.5, 0.7, 'x[3] <= 0.5\nentropy = 0.985\nsamples = 7\nvalue = [3,
4]'),
 Text(0.25, 0.5, 'x[0] <= 1.5\nentropy = 0.918\nsamples = 3\nvalue = [2,
1]'),
 Text(0.125, 0.3, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.375, 0.3, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.75, 0.5, 'x[1] <= 1.5\nentropy = 0.811\nsamples = 4\nvalue = [1,
3]'),
 Text(0.625, 0.3, 'x[2] <= 0.5\nentropy = 1.0\nsamples = 2\nvalue = [1,
1]'),
 Text(0.5, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.75, 0.1, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.875, 0.3, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]')]
```



Predict Tennis Play with user input

In [ ]:
```
enc_outlook = LabelEncoder()
categories = ['sunny', 'rainy', 'overcast']
enc_outlook.fit(categories)
```

Out[45]:
```
LabelEncoder()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
enc_temp = LabelEncoder()
categories = ['mild', 'hot', 'cool']
enc_temp.fit(categories)
```

Out[46]:
```
LabelEncoder()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
enc_humidity = LabelEncoder()
categories = ['high', 'normal']
enc_humidity.fit(categories)
```

Out[47]:
```
LabelEncoder()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
enc_wind = LabelEncoder()
categories = ['strong', 'weak']
enc_wind.fit(categories)
```

Out[48]:
```
LabelEncoder()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:
```
enc_predict = LabelEncoder()
categories = ['yes', 'no']
enc_predict.fit(categories)
```

Out[49]:
```
LabelEncoder()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [ ]:

```
outlook = input("Enter the Outlook: ") # 2: Sunny, 1: Rainy, 0: Overcast
outlook_var = enc_outlook.transform([outlook])[0] # The [0] at the end is ar

temp = input("Enter the Temperature: ") # 2: Mild, 1: Hot, 0: Cool
temp_var = enc_temp.transform([temp])[0]

humidity = input("Enter the Humidity: ") # 1: High, 0: Normal
humidity_var = enc_humidity.transform([humidity])[0]

wind = input("Enter the Wind: ") # 1: Strong, 0: Weak
wind_var = enc_wind.transform([wind])[0]

label_map = {0: 'no', 1: 'yes'}
play = Decision_le_ent.predict([[outlook_var, temp_var, humidity_var, wind_v

# If we use .ravel() insted of [0] then we have to use, (.ravel() function i
# predict_play = Decision_le_ent.predict([np.concatenate((outlook_var, temp_

# Map the numerical prediction back to a string label
play_label = label_map[play[0]]

print(play_label) # 1: yes, 0: no
```

```
Enter the Outlook: sunny
Enter the Temperature: mild
Enter the Humidity: normal
Enter the Wind: weak
yes

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning:
X does not have valid feature names, but DecisionTreeClassifier was fitted
with feature names
  warnings.warn(
```