Decision tree learning is one of the most widely adopted algorithms for classification.

As the name indicates, it builds a model in the form of a tree structure

There are many implementations of decision tree, the most prominent ones being

C5.0, CART (Classification and Regression Tree), CHAID (Chi-square Automatic Interaction Detector)

and ID3 (Iterative Dichotomiser 3) algorithms.

Entropy is a measure of impurity of an attribute or feature adopted by many algorithms such as ID3 and C5.0. Let us say S is the sample set of training examples. Then, Entropy (S) measuring the impurity of S is defined as

$$\textbf{Entropy}(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Information Gain:

```
The information gain is calculated on the basis of the decrease in entropy (S)
after a data set is split according to a particular attribute (A).
Constructing a decision tree is all about finding an attribute that returns the
highest information gain (i.e. the most homogeneous branches).
Information gain for a particular feature A is calculated by the difference in
entropy before a split (or Sbs ) with the entropy after the split (Sas ).
```

$$\textbf{Information Gain (S, A)} = \textbf{Entropy } (S_{bs}) - \textbf{Entropy } (S_{as})$$

criterion : {"gini", "entropy"}, default="gini" | The function to measure the quality of a split. Supported criteria are | "gini" for the Gini impurity and "entropy" for the information gain.

max_depth : int, default=None | The maximum depth of the tree. If None, then nodes are expanded until | all leaves are pure or until all leaves contain less than | min_samples_split samples.

get_depth(self) | Return the depth of the decision tree. |
| The depth of a tree is the maximum distance between the root | and any leaf.

In [17]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
data = pd.read_csv('https://raw.githubusercontent.com/Jovita7/datasets/main/DecisionTreeDataset%20-Num.csv')
data.head()
dtree_entropy = DecisionTreeClassifier(criterion = 'entropy',max_depth=3)
x = data.drop('Job Offered', axis = 1)
y = data['Job Offered']
model = dtree_entropy.fit(x,y)
dtree_entropy.get_depth()
```

Out[17]: 3

In [18]:
```
from sklearn import tree
text_representation = tree.export_text(dtree_entropy)
print(text_representation)
```

```
|--- feature_2 <= 0.50
|   |--- class: 0
|--- feature_2 >  0.50
|   |--- feature_1 <= 0.50
|   |   |--- feature_0 <= 1.50
|   |   |   |--- class: 0
|   |   |--- feature_0 >  1.50
|   |   |   |--- class: 1
|   |--- feature_1 >  0.50
|   |   |--- class: 1
```

In [19]:
```
prediction = dtree_entropy.predict(x)
print(prediction)
```

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

In [20]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y, prediction)
cm
```

Out[20]:
```
array([[10,  0],
       [ 0,  8]], dtype=int64)
```

In [21]:
```
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]
print(TP, FN, TN, FP)
accuracy = (TP + TN) / (TP + FP + FN + TN)
accuracy
```

```
8 0 10 0
```

Out[21]:
```
1.0
```

#Prediction with Actual Dataset

In [22]:
```
data = pd.read_csv('https://raw.githubusercontent.com/Jovita7/datasets/main/diabetes.csv')
data.head()
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [23]:
```
#Segregating predictor variables
x = data.iloc[:, 0:8]

#Segregating the target/class variable
y = data.iloc[:, 8]
```

In [24]:
```
#split into training and test datasets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3)
```

In [25]:
```
#Entropy Calculation
dtree_entropy = DecisionTreeClassifier(criterion='entropy',max_depth=26)
```

In [26]:
```
#Train the model on training data
model = dtree_entropy.fit(x_train, y_train)
dtree_entropy.get_depth()
```

Out[26]: 16

In [27]:
```
text_representation = tree.export_text(dtree_entropy)
print(text_representation)
```

```
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- feature_3 >  28.50
|   |   |   |   |   |   |   |--- feature_2 <= 89.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_2 >  89.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_4 >  36.50
|   |   |   |   |   |--- feature_5 <= 45.05
|   |   |   |   |   |   |--- feature_0 <= 13.00
|   |   |   |   |   |   |   |--- feature_2 <= 66.00
|   |   |   |   |   |   |   |   |--- feature_5 <= 33.80
|   |   |   |   |   |   |   |   |   |--- feature_3 <= 31.00
|   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |   |--- feature_3 >  31.00
|   |   |   |   |   |   |   |   |   |   |--- feature_7 <= 36.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |   |   |   |--- feature_7 >  36.00
|   |   |   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |   |--- feature_5 >  33.80
|   |   |   |   |   |   |   |   |   |--- class: 0
```

In [28]:
```
#Predictions
prediction = dtree_entropy.predict(x_test)
```

In [29]:
```
#Metric Confusion Matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, prediction)
cm
```

Out[29]:
```
array([[117,  41],
       [ 29,  44]], dtype=int64)
```

In [30]:
```
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]
print(TP, FN, TN, FP)
```

```
44 29 117 41
```

In [31]:
```
accuracy = (TP + TN) / (TP + FP + FN + TN)
accuracy
```

Out[31]:
```
0.696969696969697
```

In [32]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, prediction)
```

Out[32]:
```
0.696969696969697
```

In [33]:
```
sensitivity = TP / (TP + FN)
sensitivity
```

Out[33]:
```
0.6027397260273972
```

In [ ]:
```

```