Decision_Tree

In [ ]:

```
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.datasets import load_iris
iris=load_iris()
```

In [ ]:

```
#dimensions of dataset
X,y=iris.data,iris.target
print(X.shape)
print(y.shape)
```

```
(150, 4)
(150,)
```

In [ ]:

```
# Using gini-index
def train_using_gini(X_train, y_train):
  clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100,
                                    max_depth=3, min_samples_leaf=4)
  clf_gini.fit(X_train, y_train)
  return clf_gini
```

In [ ]:

```
#Using Entropy
def train_using_entropy(X_train,y_train):
  #Creating a classifier object
  clf_entropy = DecisionTreeClassifier(criterion="entropy",random_state = 100,
                                       max_depth=3,min_samples_leaf=4)
  #Training
  clf_entropy.fit(X_train,y_train)
  return clf_entropy
```

In [ ]:

```
#Function to make predictions
def prediction(X_test,clf_object):
  y_pred=clf_object.predict(X_test)
  print("Predicted values:",y_pred)
  return y_pred
```

In [ ]:

```
#Function to calculate accuracy
def cal_accuracy(y_test,y_pred):
  print("Confusion Matrix: ",confusion_matrix(y_test,y_pred))
  print("Accuracy:",accuracy_score(y_test,y_pred)*100)
  print("Report :",classification_report(y_test,y_pred))
```

In [ ]:

```
#Splitting training and testing data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.3, random_state = 100)
print("Dimensions for training data",X_train.shape)
print("Dimensions for testing data",y_train.shape)
```

```
Dimensions for training data (105, 4)
Dimensions for testing data (105,)
```

In [ ]:

```
#Gini Index
clf_gini = train_using_gini(X_train, y_train)
print("Results Using Gini Index:")
# Prediction using gini
y_pred_gini = prediction(X_test, clf_gini)
cal_accuracy(y_test, y_pred_gini)
```

```
Results Using Gini Index:
Predicted values:
[2 0 2 0 2 2 0 0 2 0 0 2 0 0 2 1 1 2 2 2 2 0 2 0 1 2 1 0 1 2 1 1 1 0 0 1 0
 1 2 2 0 1 2 2 0]
Confusion Matrix:  [[16  0  0]
 [ 0 10  1]
 [ 0  1 17]]
Accuracy: 95.55555555555556
Report :               precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.91      0.91      0.91        11
           2       0.94      0.94      0.94        18

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45
```

In [ ]:

```
#Analysing Metrics using entropy
clf_entropy = train_using_entropy(X_train,y_train)
# Prediction using entropy
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)
```

```
Predicted values:
[2 0 2 0 2 2 0 0 2 0 0 2 0 0 2 1 1 2 2 2 2 0 2 0 1 2 1 0 1 2 1 1 1 0 0 1 0
 1 2 2 0 1 2 2 0]
Confusion Matrix:  [[16  0  0]
 [ 0 10  1]
 [ 0  1 17]]
Accuracy: 95.55555555555556
Report :               precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.91      0.91      0.91        11
           2       0.94      0.94      0.94        18

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45
```