# Mini Project 4: Classification Toy Dataset

# 1. Decision Tree Classifier

## Step 1:Import Library

[ ] ↳ *1 cell hidden*

## Step 2: Generating Dataset

```
from sklearn.datasets import make_classification
```

```
X,y=make_classification(n_samples=1000,n_features=5,n_clusters_per_class=1,n_classes=2,random_state=2529)
```

```
X[0:5]
```

```
     array([[ 1.54701705,  0.84770596, -0.41725021, -0.62356778, -0.19388577],
            [ 0.80633556,  0.40985594, -0.45641095, -0.3052022 ,  0.50935923],
            [ 0.94390268,  0.70041038,  1.11385452, -0.49394417,  1.42305455],
            [ 1.92091517,  0.95815739, -1.2235022 , -0.71578154,  0.66588981],
            [ 1.45270369,  0.69035375, -1.18119669, -0.52009219, -0.22745417]])
```

```
y[0:5]
```

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

```
X.shape,y.shape
```

```
((1000, 5), (1000,))
```

## Step 3: Splitting Data

```
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,random_state=2529)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((700, 5), (300, 5), (700,), (300,))
```

## Step 4: Creating Model

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

```
model=DecisionTreeClassifier()
```

## Step 5: Training Model

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

```
DecisionTreeClassifier()
```

## Step 6: Prediction Model

```
y_pred=model.predict(X_test)
```

```
y_pred.shape
```

```
(300,)
```

```
y_pred
```

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

## Step 7: Accuracy

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
accuracy_score(y_test,y_pred)
```

```
0.9866666666666667
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[156,   1],
       [  3, 140]])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       157
           1       0.99      0.98      0.99       143

    accuracy                           0.99       300
   macro avg       0.99      0.99      0.99       300
weighted avg       0.99      0.99      0.99       300
```

## Step 8: Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {'criterion':['gini','entropy'],'max_depth':[2,3,4,5,6,7,8,9,10,11,12,15,20,30,40,50,70,90,120,150]}
```

```
gs = GridSearchCV(DecisionTreeClassifier(), parameters)
gs.fit(X_train, y_train)
```

```
GridSearchCV(estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15,
                                       20, 30, 40, 50, 70, 90, 120, 150]})
```

```
gs.best_params_
```

```
{'criterion': 'entropy', 'max_depth': 3}
```

```
gs.best_score_
```

```
0.9885714285714287
```

```
gs.best_estimator_
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=3)
```

```
gs.best_index_
```

```
21
```

## Step 9: Re-Prediction and Re-Evaluation

```
y_pred_grid = gs.predict(X_test)
```

```
confusion_matrix(y_test,y_pred_grid)
```

```
array([[156,   1],
       [  1, 142]])
```

```
print(classification_report(y_test,y_pred_grid))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       157
```

```
              1       0.99      0.99      0.99       143

       accuracy                           0.99       300
      macro avg       0.99      0.99      0.99       300
   weighted avg       0.99      0.99      0.99       300
```

# 2. Random Forest Classifier

## Step 1:Import Library

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

## Step 2: Generating Dataset

```
from sklearn.datasets import make_classification
```

```
X,y = make_classification(n_samples=1000,n_features=5,n_clusters_per_class=1,n_classes=2,random_state=2529)
```

```
X[0:5]
```

```
     array([[ 1.54701705,  0.84770596, -0.41725021, -0.62356778, -0.19388577],
            [ 0.80633556,  0.40985594, -0.45641095, -0.3052022 ,  0.50935923],
            [ 0.94390268,  0.70041038,  1.11385452, -0.49394417,  1.42305455],
            [ 1.92091517,  0.95815739, -1.2235022 , -0.71578154,  0.66588981],
            [ 1.45270369,  0.69035375, -1.18119669, -0.52009219, -0.22745417]])
```

```
y[0:5]

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

```
X.shape

     (1000, 5)
```

```
y.shape

     (1000,)
```

## Step 3: Splitting Data

```
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,random_state=2529)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape

     ((700, 5), (300, 5), (700,), (300,))
```

## Step 4: Creating Model

```
from sklearn.ensemble import RandomForestClassifier
```

```
model=RandomForestClassifier()
```

## Step 5: Training Model

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

```
    RandomForestClassifier()
```

## Step 6: Prediction Model

```
y_pred=model.predict(X_test)
```

```
y_pred.shape
```

```
    (300,)
```

```
y_pred
```

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

## Step 7: Accuracy

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
accuracy_score(y_test,y_pred)
```

```
    0.99
```

```
confusion_matrix(y_test,y_pred)
```

```
    array([[156,   1],
           [  2, 141]])
```

```
print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.99      0.99      0.99       157
               1       0.99      0.99      0.99       143

        accuracy                           0.99       300
       macro avg       0.99      0.99      0.99       300
    weighted avg       0.99      0.99      0.99       300
```

## Step 8: Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
parameters={'n_estimators':[10,20,30,100,200,500],'max_features':['auto','sqrt'],'min_samples_split':[4,8],'bootstrap':[True,False]}
```

```
gs=GridSearchCV(RandomForestClassifier(),parameters)
gs.fit(X_train,y_train)

     GridSearchCV(estimator=RandomForestClassifier(),
                  param_grid={'bootstrap': [True, False],
                              'max_features': ['auto', 'sqrt'],
                              'min_samples_split': [4, 8],
                              'n_estimators': [10, 20, 30, 100, 200, 500]})
```

```
gs.best_params_

     {'bootstrap': False,
      'max_features': 'sqrt',
      'min_samples_split': 8,
      'n_estimators': 30}
```

```
gs.best_score_

     0.99
```

```
gs.best_estimator_

     RandomForestClassifier(bootstrap=False, max_features='sqrt',
                            min_samples_split=8, n_estimators=30)
```

```
gs.best_index_

     44
```

## Step 9: Re-Prediction and Re-Evaluation

```
y_pred_grid=gs.predict(X_test)
```

```
confusion_matrix(y_test,y_pred_grid)
```

```
array([[156,   1],
       [  1, 142]])
```

```
print(classification_report(y_test,y_pred_grid))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       157
           1       0.99      0.99      0.99       143

    accuracy                           0.99       300
   macro avg       0.99      0.99      0.99       300
weighted avg       0.99      0.99      0.99       300
```

## Link of the same:

## https://colab.research.google.com/drive/1P2fK5P_8VA6x48Y1PPBwf5W79t_VF1O8?usp=sharing