Decision Tree Clasifier with Artificial Generated Dataset

Importing the libraries

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
```

Generate Dataset

```
1   asets import make_classification
2   ification(n_samples=1000,n_features=5,n_classes=2,n_clusters_per_class=1,random_sta
```

Get the first 5 rows of y and X

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

```
array([[-1.05725592, -1.30225279, -1.46889882, -1.1266874 ,  1.24135436],
       [ 2.02829207, -1.92846587, -0.9868359 , -1.81084964,  2.0087454 ],
       [ 0.3263314 , -1.01864946, -0.92874165, -0.90770617,  1.00260713],
       [-0.33353226,  0.01998436, -0.69739414,  0.10353936, -0.12231527],
       [ 1.08320214,  2.11279242,  3.03617595,  1.74972077, -1.92032348]])
```

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

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

Get shape of DataFrame

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

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

Splitting the dataset into the Training set and Test set

```
1 from sklearn.model_selection import train_test_split
```

```
2 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_stat
```

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

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

Get Decision Tree

```
1 from sklearn.tree import DecisionTreeClassifier
2 model=DecisionTreeClassifier()
```

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

```
DecisionTreeClassifier()
```

Get the first 5 rows of y and X

Model Prediction

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

```
1 y_pred.shape
```

```
(300,)
```

```
1 y_pred
```

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

Model Evaluation

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

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

```
0.9633333333333334
```

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

```
array([[150,   6],
       [  5, 139]])
```

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

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       156
           1       0.96      0.97      0.96       144

    accuracy                           0.96       300
   macro avg       0.96      0.96      0.96       300
weighted avg       0.96      0.96      0.96       300
```

Hyperparameter Tuning

```
1 from  sklearn.model_selection import GridSearchCV
2 parameter={'criterion':['gini','entropy'],'max_depth':[2,3,4,5,6,7,8,9,10,11,12,15,20,
3 gridsearch=GridSearchCV(DecisionTreeClassifier(),parameter)
4 gridsearch.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]})
```

```
1   gridsearch.best_params_
```

```
{'criterion': 'gini', 'max_depth': 11}
```

```
1   gridsearch.best_score_
```

```
0.9814285714285715
```

```
1   gridsearch.best_estimator_
```

```
DecisionTreeClassifier(max_depth=11)
```

```
1   gridsearch.best_index_
```

```
9
```

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

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

```
array([[150,   6],
       [  5, 139]])
```

+ Code + Text

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

```
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       156
           1       0.96      0.97      0.96       144

    accuracy                           0.96       300
   macro avg       0.96      0.96      0.96       300
weighted avg       0.96      0.96      0.96       300
```