2.Random Forest Classification with Artificial Generated Dataset

In [1]:
```
#Import Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
#Generate Dataset
from sklearn.datasets import make_classification
```

In [4]:
```
#without coeficient of underline model
x,y=make_classification(n_samples=1000,n_features=5,n_clusters_per_class=1,n_classes=2,random_state=2529)
```

Get the First Five rows of target Variable(y) and Features(x)

In [5]:
```
x[0:5]
```

Out[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]])
```

In [6]:
```
y[0:5]
```

Out[6]:
```
array([0, 0, 1, 0, 0])
```

Get Shape of DataFrame

In [7]:
```
x.shape,y.shape
```

Out[7]:
```
((1000, 5), (1000,))
```

Get Train Test Split

In [8]:
```
from sklearn.model_selection import train_test_split
```

In [9]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=2529)
```

In [10]:
```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[10]:
```
((700, 5), (300, 5), (700,), (300,))
```

Get Random Forest Classification Model Train

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [13]:
```
model=RandomForestClassifier()
```

In [14]:
```
model.fit(x_train,y_train)
```

Out[14]:
```
RandomForestClassifier()
```

Get Model Prediction

In [15]:
```
y_pred=model.predict(x_test)
```

In [16]:
```
y_pred.shape
```

Out[16]:
```
(300,)
```

In [17]:
```
y_pred
```

Out[17]:
```
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, 0, 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, 0, 1, 0, 1, 0, 1, 0, 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])
```

Get Model Evaluation

In [19]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [20]:
```
accuracy_score(y_test,y_pred)
```

Out[20]:
```
0.99
```

In [21]:
```
confusion_matrix(y_test,y_pred)
```

Out[21]:
```
array([[156,   1],
       [  2, 141]], dtype=int64)
```

In [22]:
```
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
```

Hyperparameter Tunning: Grid Search

In [28]:
```
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]}
gridsearch=GridSearchCV(RandomForestClassifier(),parameters)
gridsearch.fit(x_train,y_train)
```

Out[28]:
```
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]})
```

In [29]:
```
gridsearch.best_params_
```

Out[29]:
```
{'bootstrap': False,
 'max_features': 'auto',
 'min_samples_split': 8,
 'n_estimators': 10}
```

In [30]:
```
gridsearch.best_score_
```

Out[30]:
```
0.9885714285714287
```

In [31]:
```
gridsearch.best_estimator_
```

Out[31]:
```
RandomForestClassifier(bootstrap=False, min_samples_split=8, n_estimators=10)
```

In [32]:
```
gridsearch.best_index_
```

Out[32]:
```
30
```

In [33]:
```
y_pred_grid=gridsearch.predict(x_test)
```

In [34]:
```
confusion_matrix(y_test,y_pred_grid)
```

Out[34]:
```
array([[156,   1],
       [  2, 141]], dtype=int64)
```

In [35]:
```
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
```

In [ ]:
```

```