In [2]:

```
import pandas as pd
```

In [3]:

```
import numpy as np
```

In [4]:

```
import matplotlib.pyplot as plt
```

GENERATE DATASET

In [6]:

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

In [8]:

```
#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 [9]:

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

Out[9]:

```
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 [10]:

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

Out[10]:

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

Get Shape of DataFrame

In [11]:

```
x.shape,y.shape
```

Out[11]:

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

Get Train Test Split

In [43]:

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

In [44]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=2529)
```

In [45]:

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out[45]:

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

Get Logistic Regression Classification Model Train

In [46]:

```
from sklearn.linear_model import LogisticRegression
```

In [47]:

```
model=LogisticRegression()
```

In [48]:

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

Out[48]:

```
LogisticRegression()
```

Get Model Prediction

In [49]:

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

In [50]:

```
y_pred.shape
```

Out[50]:

```
(300,)
```

In [51]:

```
y_pred
```

Out[51]:

```
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])
```

Get Model Evaluation

In [52]:

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

In [53]:

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

Out[53]:

```
0.99
```

In [54]:

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

Out[54]:

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

In [28]:

```
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 [59]:

```
from sklearn.model_selection import GridSearchCV
parameters={'penalty':['l1','l2'],'C':[0.001,.009,0.01,.09,1,5,10,25],'solver':['liblinear']}
gridsearch=GridSearchCV(LogisticRegression(),parameters)
gridsearch.fit(x_train,y_train)
```

Out[59]:

```
GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.009, 0.01, 0.09, 1, 5, 10, 25],
                         'penalty': ['l1', 'l2'], 'solver': ['liblinear']})
```

In [60]:

```
gridsearch.best_params_
```

Out[60]:

```
{'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
```

In [61]:

```
gridsearch.best_score_
```

Out[61]:

```
0.9914285714285714
```

In [62]:

```
gridsearch.best_estimator_
```

Out[62]:

```
LogisticRegression(C=1, penalty='l1', solver='liblinear')
```

In [63]:

```
gridsearch.best_index_
```

Out[63]:

```
8
```

In [64]:

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

In [65]:

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

Out[65]:

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

In [67]:

```
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
```