Logistic regression Classifier with Arificial Generated

Import Library

In [1]:

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

Generate Dataset

In [2]:

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

In [3]:

```
# without coeficient of underline model
X,y = make_classification(n_samples=1000,n_features=5,n_clusters_per_class=1,n_classes=2,ra
```

Get the First Five Rows of Target Variable(y) and Features(X)

In [4]:

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

Out[4]:

```
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 [5]:

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

Out[5]:

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

Get Shape of Dataframe

In [6]:

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

Out[6]:

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

Get Train Test Split

In [7]:

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

In [8]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=2529)
```

In [9]:

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

Out[9]:

```
((700, 5),
 (300, 5),
 array([1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,
        1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,
        0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1,
        0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0,
        1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1,
        1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
        1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1,
        0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1,
        1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0,
        0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1,
        0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1,
        0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
        1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
        1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
        1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1,
        0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1,
        0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
        0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
        1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0,
        1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0]),
 (300,))
```

Get Logistic Regression classification Model Train

LogisticRegression(penalty='l2', dual = False, tol=0.0001,C=1.0,fit_intercept =True,intercept_scaling=1,class_weight=None,random_state=None,solver='lbfgs',max_iter=100,multi_clas

In [10]:

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

In [11]:

```
model = LogisticRegression()
```

In [12]:

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

Out[12]:

```
LogisticRegression()
```

Get Model Prediction

In [13]:

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

In [14]:

```
y_pred.shape
```

Out[14]:

```
(300,)
```

In [15]:

```
y_pred
```

Out[15]:

```
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 [16]:

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

In [17]:

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

Out[17]:

```
0.99
```

In [18]:

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

Out[18]:

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

In [19]:

```
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

GridSearchCV(estimator, param_grid,scoring=None,n_jobs=None,refit=True,cv=None,verbose=0,pre_dispatch='2n_jobs',error_scor

In [20]:

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

Out[20]:

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

In [21]:

```
gridsearch.best_params_
```

Out[21]:

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

In [22]:

```
gridsearch.best_estimator_
```

Out[22]:

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

In [23]:

```
gridsearch.best_index_
```

Out[23]:

```
6
```

In [24]:

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

In [25]:

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

Out[25]:

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

In [26]:

```
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 [ ]:

```

```