TUGUME JAMES 2023-U-MMU-BCS-01680

In [63]:
```
# import library libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [64]:
```
# loading the datasets
data =pd.read_csv("C:\\Users\\tugumejames\\Desktop\\Logistic_dataset.csv")
```

In [65]:
```
data
```

Out[65]:

	Feature 1	Feature 2	Label
0	1.764052	4.764052	0
1	0.978738	-2.021262	1
2	1.867558	4.867558	0
3	0.950088	-2.049912	1
4	-0.103219	2.896781	0
...	...	...	...
95	-1.292857	-4.292857	1
96	-0.039283	-3.039283	1
97	0.523277	-2.476723	1
98	0.771791	3.771791	0
99	2.163236	5.163236	0

100 rows × 3 columns

In [66]:
```
X = np.array(data[["Feature 1","Feature 2"]])
y = np.array(data["Label"])
```

In [67]:
```
X
```

Out[67]:
```
array([[ 1.76405235,  4.76405235],
       [ 0.97873798, -2.02126202],
       [ 1.86755799,  4.86755799],
       [ 0.95008842, -2.04991158],
       [-0.10321885,  2.89678115],
       [ 0.14404357, -2.85595643],
       [ 0.76103773,  3.76103773],
       [ 0.44386323, -2.55613677],
       [ 1.49407907, -1.50592093],
       [ 0.3130677 ,  3.3130677 ],
       [-2.55298982,  0.44701018],
       [ 0.8644362 , -2.1355638 ],
       [ 2.26975462,  5.26975462],
       [ 0.04575852, -2.95424148],
       [ 1.53277921,  4.53277921],
       [ 0.15494743, -2.84505257],
       [-0.88778575,  2.11221425],
       [-0.34791215,  2.65208785],
       [ 1.23029068,  4.23029068],
       [-0.38732682,  2.61267318],
       [-1.04855297, -4.04855297],
       [-1.70627019, -4.70627019],
       [-0.50965218,  2.49034782],
       [-1.25279536, -4.25279536],
       [-1.61389785,  1.38610215],
       [-0.89546656, -3.89546656],
       [-0.51080514, -3.51080514],
       [-0.02818223,  2.97181777],
       [ 0.06651722, -2.93348278],
       [-0.63432209,  2.36567791],
       [-0.67246045,  2.32753955],
       [-0.81314628,  2.18685372],
       [ 0.17742614,  3.17742614],
       [-1.63019835,  1.36980165],
       [-0.90729836, -3.90729836],
       [ 0.72909056, -2.27090944],
       [ 1.13940068,  4.13940068],
       [ 0.40234164,  3.40234164],
       [-0.87079715,  2.12920285],
       [-0.31155253,  2.68844747],
       [-1.16514984, -4.16514984],
       [ 0.46566244, -2.53433756],
       [ 1.48825219,  4.48825219],
```

```
[ 1.17877957,  4.17877957],
[-1.07075262, -4.07075262],
[-0.40317695,  2.59682305],
[ 0.20827498, -2.79172502],
[ 0.3563664 , -2.6436336 ],
[ 0.01050002, -2.98949998],
[ 0.12691209, -2.87308791],
[ 1.8831507 , -1.1168493 ],
[-1.270485  , -4.270485  ],
[-1.17312341, -4.17312341],
[-0.41361898, -3.41361898],
[ 1.92294203, -1.07705797],
[ 1.86755896, -1.13244104],
[-0.86122569, -3.86122569],
[-0.26800337,  2.73199663],
[ 0.94725197,  3.94725197],
[ 0.61407937, -2.38592063],
[ 0.37642553, -2.62357447],
[ 0.29823817, -2.70176183],
[-0.69456786, -3.69456786],
[-0.43515355,  2.56484645],
[ 0.67229476,  3.67229476],
[-0.76991607, -3.76991607],
[-0.67433266,  2.32566734],
[-0.63584608,  2.36415392],
[ 0.57659082, -2.42340918],
[ 0.39600671,  3.39600671],
[-1.49125759, -4.49125759],
[ 0.1666735 ,  3.1666735 ],
[ 2.38314477, -0.61685523],
[-0.91282223,  2.08717777],
[-1.31590741,  1.68409259],
[-0.06824161, -3.06824161],
[-0.74475482,  2.25524518],
[-0.09845252,  2.90154748],
[ 1.12663592,  4.12663592],
[-1.14746865,  1.85253135],
[-0.49803245,  2.50196755],
[ 0.94942081, -2.05057919],
[-1.22543552,  1.77456448],
[-1.00021535, -4.00021535],
[ 1.18802979,  4.18802979],
[ 0.92085882,  3.92085882],
```

```
       [ 0.85683061, -2.14316939],
       [-1.03424284, -4.03424284],
       [-0.80340966, -3.80340966],
       [-0.4555325 , -3.4555325 ],
       [-0.35399391,  2.64600609],
       [-0.6436184 ,  2.3563816 ],
       [ 0.62523145, -2.37476855],
       [-1.10438334,  1.89561666],
       [-0.739563  , -3.739563  ],
       [-1.29285691, -4.29285691],
       [-0.03928282, -3.03928282],
       [ 0.52327666, -2.47672334],
       [ 0.77179055,  3.77179055],
       [ 2.16323595,  5.16323595]])
```

In [68]:
```
y
```

Out[68]:
```
array([0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0], dtype=int64)
```

In [69]:
```
# spliting data int train and test
X_train, X_test, y_train,y_test = train_test_split(X,y ,test_size=0.2, random_state = 42)
```

In [70]:
```
# building the model
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[70]:
```
▾ LogisticRegression

LogisticRegression()
```

In [71]:
```
y_pred = model.predict(X_test)
```

In [72]:
```
y_pred
```

Out[72]:
```
array([1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [73]:
```
accuracy = accuracy_score(y_test, y_pred)
```

In [74]:
```
accuracy
```

Out[74]:
```
1.0
```

In [75]:
```
coefficient = model.coef_
intercept  = model.intercept_
```

In [76]:
```
coefficient
```

Out[76]:
```
array([[ 0.66645107, -1.67786247]])
```

In [77]:
```
intercept
```

Out[77]:
```
array([0.28951949])
```

model optimaization on logostic regression

In [78]:
```
# importing the necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [88]:
```
# loading the datasets
data =pd.read_csv("C:\\Users\\tugumejames\\Desktop\\Logistic_dataset.csv")
```

In [89]:
```
data
```

Out[89]:

	Feature 1	Feature 2	Label
0	1.764052	4.764052	0
1	0.978738	-2.021262	1
2	1.867558	4.867558	0
3	0.950088	-2.049912	1
4	-0.103219	2.896781	0
...	...	...	...
95	-1.292857	-4.292857	1
96	-0.039283	-3.039283	1
97	0.523277	-2.476723	1
98	0.771791	3.771791	0
99	2.163236	5.163236	0

100 rows × 3 columns

In [80]:
```
X = np.array(data[["Feature 1","Feature 2"]])
y = np.array(data["Label"])
```

In [81]:
```
X
```

Out[81]:
```
array([[ 1.76405235,  4.76405235],
       [ 0.97873798, -2.02126202],
       [ 1.86755799,  4.86755799],
       [ 0.95008842, -2.04991158],
       [-0.10321885,  2.89678115],
       [ 0.14404357, -2.85595643],
       [ 0.76103773,  3.76103773],
       [ 0.44386323, -2.55613677],
       [ 1.49407907, -1.50592093],
       [ 0.3130677 ,  3.3130677 ],
       [-2.55298982,  0.44701018],
       [ 0.8644362 , -2.1355638 ],
       [ 2.26975462,  5.26975462],
       [ 0.04575852, -2.95424148],
       [ 1.53277921,  4.53277921],
       [ 0.15494743, -2.84505257],
       [-0.88778575,  2.11221425],
       [-0.34791215,  2.65208785],
       [ 1.23029068,  4.23029068],
       [-0.38732682,  2.61267318],
       [-1.04855297, -4.04855297],
       [-1.70627019, -4.70627019],
       [-0.50965218,  2.49034782],
       [-1.25279536, -4.25279536],
       [-1.61389785,  1.38610215],
       [-0.89546656, -3.89546656],
       [-0.51080514, -3.51080514],
       [-0.02818223,  2.97181777],
       [ 0.06651722, -2.93348278],
       [-0.63432209,  2.36567791],
       [-0.67246045,  2.32753955],
       [-0.81314628,  2.18685372],
       [ 0.17742614,  3.17742614],
       [-1.63019835,  1.36980165],
       [-0.90729836, -3.90729836],
       [ 0.72909056, -2.27090944],
       [ 1.13940068,  4.13940068],
       [ 0.40234164,  3.40234164],
       [-0.87079715,  2.12920285],
       [-0.31155253,  2.68844747],
       [-1.16514984, -4.16514984],
       [ 0.46566244, -2.53433756],
       [ 1.48825219,  4.48825219],
```

```
[ 1.17877957,  4.17877957],
[-1.07075262, -4.07075262],
[-0.40317695,  2.59682305],
[ 0.20827498, -2.79172502],
[ 0.3563664 , -2.6436336 ],
[ 0.01050002, -2.98949998],
[ 0.12691209, -2.87308791],
[ 1.8831507 , -1.1168493 ],
[-1.270485  , -4.270485  ],
[-1.17312341, -4.17312341],
[-0.41361898, -3.41361898],
[ 1.92294203, -1.07705797],
[ 1.86755896, -1.13244104],
[-0.86122569, -3.86122569],
[-0.26800337,  2.73199663],
[ 0.94725197,  3.94725197],
[ 0.61407937, -2.38592063],
[ 0.37642553, -2.62357447],
[ 0.29823817, -2.70176183],
[-0.69456786, -3.69456786],
[-0.43515355,  2.56484645],
[ 0.67229476,  3.67229476],
[-0.76991607, -3.76991607],
[-0.67433266,  2.32566734],
[-0.63584608,  2.36415392],
[ 0.57659082, -2.42340918],
[ 0.39600671,  3.39600671],
[-1.49125759, -4.49125759],
[ 0.1666735 ,  3.1666735 ],
[ 2.38314477, -0.61685523],
[-0.91282223,  2.08717777],
[-1.31590741,  1.68409259],
[-0.06824161, -3.06824161],
[-0.74475482,  2.25524518],
[-0.09845252,  2.90154748],
[ 1.12663592,  4.12663592],
[-1.14746865,  1.85253135],
[-0.49803245,  2.50196755],
[ 0.94942081, -2.05057919],
[-1.22543552,  1.77456448],
[-1.00021535, -4.00021535],
[ 1.18802979,  4.18802979],
[ 0.92085882,  3.92085882],
```

```
       [ 0.85683061, -2.14316939],
       [-1.03424284, -4.03424284],
       [-0.80340966, -3.80340966],
       [-0.4555325 , -3.4555325 ],
       [-0.35399391,  2.64600609],
       [-0.6436184 ,  2.3563816 ],
       [ 0.62523145, -2.37476855],
       [-1.10438334,  1.89561666],
       [-0.739563  , -3.739563  ],
       [-1.29285691, -4.29285691],
       [-0.03928282, -3.03928282],
       [ 0.52327666, -2.47672334],
       [ 0.77179055,  3.77179055],
       [ 2.16323595,  5.16323595]])
```

In [82]:
```
y
```

Out[82]:
```
array([0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0], dtype=int64)
```

In [83]:
```
# spliting the data into training and testing
X_train,X_test, y_train,y_test = train_test_split(X,y,test_size=0.2, random_state=42)
```

In [84]:
```
# building the logistic regression model
model = LogisticRegression()
```

In [85]:
```
#Implementing the hyperparamter tuning using gridsearchCV


param_grid = {

             "C":[0.001,0.01,0.1,1,10,100],

              "penalty":["l1","l2",'none'],
              "solver":[ 'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga'],
              "max_iter":[100,1000,]
             }
```

In [86]:
```
# perform a gridsearch (cross validation)

grid_search= GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train )
```

```
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:1182: FutureWarning: `penalty
='none'`has been deprecated in 1.2 and will be removed in 1.4. To keep the past behaviour, set `penalty=Non
e`.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:1192: UserWarning: Setting pen
alty=None will ignore the C and l1_ratio parameters
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:1182: FutureWarning: `penalty
='none'`has been deprecated in 1.2 and will be removed in 1.4. To keep the past behaviour, set `penalty=Non
e`.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py:1192: UserWarning: Setting pen
alty=None will ignore the C and l1_ratio parameters
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\linear_model\_sag.py:350: ConvergenceWarning: The max_it
er was reached which means the coef_ did not converge
```

In [87]:
```
#Geting the best paramters found from Grid Search

best_params = grid_search.best_params_
print("Best parameters: ", best_params)

# use the best parameters to train the model

best_model = LogisticRegression(**best_params)
best_model.fit(X_train,y_train)
y_pred = best_model.predict(X_test)

#Evaluating the performance of the model
accuracy = accuracy_score(y_test,y_pred)
print("accuracy: ", accuracy)

coefficient = best_model.coef_
intercept  = best_model.intercept_
print("coefficient:", coefficient)
print("intercept:", intercept)
```

```
Best parameters:  {'C': 0.001, 'max_iter': 100, 'penalty': 'l2', 'solver': 'liblinear'}
accuracy:  1.0
coefficient: [[ 0.00112375 -0.10074539]]
intercept: [0.0052003]
```

In [ ]: