Lavina Golani

Batch A

16

**Experiment:3,4**

Aim: To implement multilayer perceptron

Code:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPClassifier

from sklearn.metrics import classification\_report,confusion\_matrix

file = '/home/lavina/Desktop/pima-indians-diabetes-database/diabetes.csv'

# load data set into frame

dataFrame = pd.read\_csv(file, names=['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'])

print(dataFrame.describe())

# splitting into training and test

X = dataFrame.loc[:, ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age']]

y = dataFrame.loc[:, ['Outcome']]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30)

# scaling data

scaler = StandardScaler()

scaler.fit(X\_train)

StandardScaler(copy=True, with\_mean=True, with\_std=True)

X\_train = scaler.transform(X\_train)

X\_test = scaler.transform(X\_test)

# creating the model

mlp = MLPClassifier(alpha=0.0001, hidden\_layer\_sizes=(20, 20, 20), solver='lbfgs', learning\_rate='constant', learning\_rate\_init= 0.0001, max\_iter= 100)

mlp.fit(X\_train, y\_train)

# predicting from the model

predictions = mlp.predict(X\_test)

#accuracy

score = mlp.score(X\_test, y\_test)

print(score)

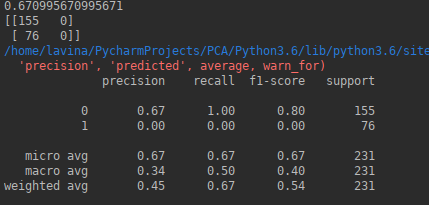
#analysing output

print(confusion\_matrix(y\_test, predictions))

print(classification\_report(y\_test, predictions))

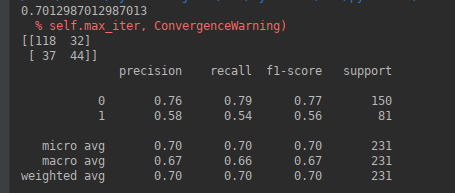
Output with different models:

mlp= MLPClassifier(alpha=0.0001, hidden\_layer\_sizes=(20, 20, 20), learning\_rate='constant', learning\_rate\_init= 0.0001, max\_iter= 100, early\_stopping=True)

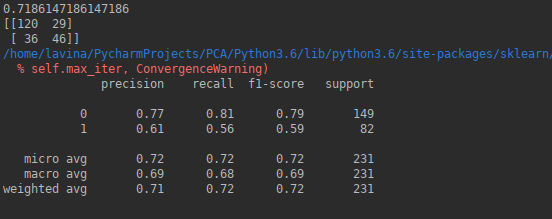


mlp = MLPClassifier(alpha=0.0001,hidden\_layer\_sizes=(20, 20), learning\_rate='constant', learning\_rate\_init=0.001, verbose=True, max\_iter= 200)

Score , COnfusion Matrix , Summary



mlp = MLPClassifier(alpha=0.0001, hidden\_layer\_sizes=(20, 20, 20, 20), learning\_rate='constant', learning\_rate\_init= 0.0001, verbose=True, max\_iter= 1000)



mlp = MLPClassifier(alpha=0.0001, hidden\_layer\_sizes=(20, 20, 20), solver='lbfgs', learning\_rate='constant', learning\_rate\_init= 0.0001, max\_iter= 100)

Conclusion: Hence we have successfully implemented multi layer perceptron.