**Description:**

In this project I characterized whether a person is diabetic or not the given dataset has various features that tells if a patient is diabetic or not to do this, I implemented a perceptron model for binary classification using a simple training function with a specified accuracy target. It uses 10-fold cross-validation to evaluate the model's performance on different subsets of the dataset. The perceptron is trained iteratively, adjusting weights based on errors, until a specified number of iterations or until the model achieves perfect accuracy. Additionally, it includes a function to plot confusion matrices for each fold, providing insights into the model's performance.

**perceptron:**

Implements the perceptron training algorithm with a specified number of iterations (100) and learning rate (0.1).

**train\_perceptron:**

Uses the perceptron function for training, iteratively updating weights until a target accuracy (0.75) or more is reached. It also evaluates the accuracy on a test set.

**plot\_confusion\_matrix:**

Generates and displays a confusion matrix heatmap for visualizing the model's performance on the test set.

**Code:**

import numpy as np

import pandas as pd

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import KFold

def perceptron(X, Y, num\_iterations=100, learning\_rate=0.1):

    num\_samples, num\_features = X.shape

    weights = np.zeros(num\_features)

    for iter in range(num\_iterations):

        errors = []

        for idx, sample in enumerate(X):

            c = Y[idx]

            f = sample

            weighted\_sum = np.dot(f, weights)

            output = 1 if weighted\_sum >= 0 else 0

            errors.append(c - output)

            weights += learning\_rate \* errors[-1] \* f

        if all(error == 0 for error in errors):

            break

    return weights

def train\_perceptron(X\_train, Y\_train, X\_test, Y\_test, target\_accuracy=0.75, num\_iterations=100, learning\_rate=0.1):

    X\_train = np.column\_stack([X\_train, np.zeros(len(X\_train))])

    X\_test = np.column\_stack([X\_test, np.zeros(len(X\_test))])

    weights = perceptron(X\_train, Y\_train, num\_iterations, learning\_rate)

    predictions = [1 if np.dot(sample, weights) >= 0 else 0 for sample in X\_test]

    accuracy = accuracy\_score(Y\_test, predictions)

    # Repeat until the training accuracy is achieved

    while accuracy < target\_accuracy:

        weights = perceptron(X\_train, Y\_train, num\_iterations, learning\_rate)

        predictions = [1 if np.dot(sample, weights) >= 0 else 0 for sample in X\_test]

        accuracy = accuracy\_score(Y\_test, predictions)

    return weights, accuracy

def plot\_confusion\_matrix(y\_true, y\_pred):

    conf\_matrix = confusion\_matrix(y\_true, y\_pred)

    sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['0', '1'], yticklabels=['0', '1'])

    plt.xlabel('Predicted')

    plt.ylabel('Actual')

    plt.title('Confusion Matrix')

    plt.show(block=False)

df = pd.read\_csv("/content/drive/MyDrive/ailabmiod/diabetes\_binary\_health\_indicators\_BRFSS2021.csv")

X = df.iloc[:, 1:].values

y = df.iloc[:,0].values

kf = KFold(n\_splits=10, shuffle=True, random\_state=32)

for train\_index, test\_index in kf.split(X):

    x\_train, x\_test = X[train\_index], X[test\_index]

    y\_train, y\_test = y[train\_index], y[test\_index]

    weights, accuracy = train\_perceptron(x\_train, y\_train, x\_test, y\_test)

    print("Weights:", weights)

    print("Accuracy:", accuracy)

    # Use weights and bias to predict the classes for testing data

    x\_test = np.column\_stack([x\_test, np.zeros(len(x\_test))])

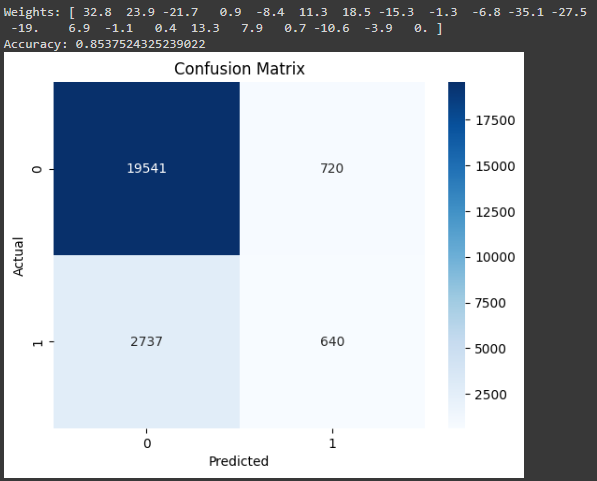
    test\_predictions = [1 if np.dot(sample, weights) >= 0 else 0 for sample in x\_test]

    # Plot confusion matrix for each fold

    plot\_confusion\_matrix(y\_test, test\_predictions)

**Output:**

**Fold 1:**



**Fold 2:**

A screenshot of a graph

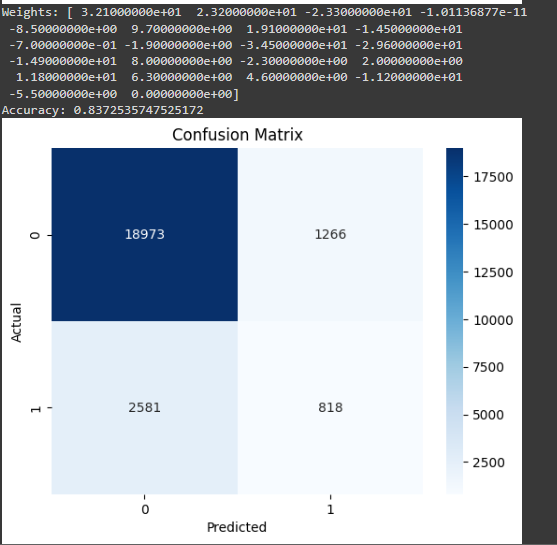
Description automatically generated

**Fold 3:**

A screenshot of a graph

Description automatically generated

**Fold 4:**



**Fold 5:**

A screenshot of a graph

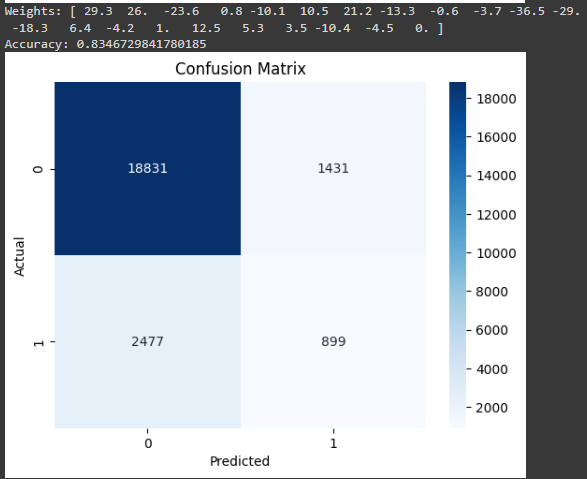
Description automatically generated

**Fold 6:**

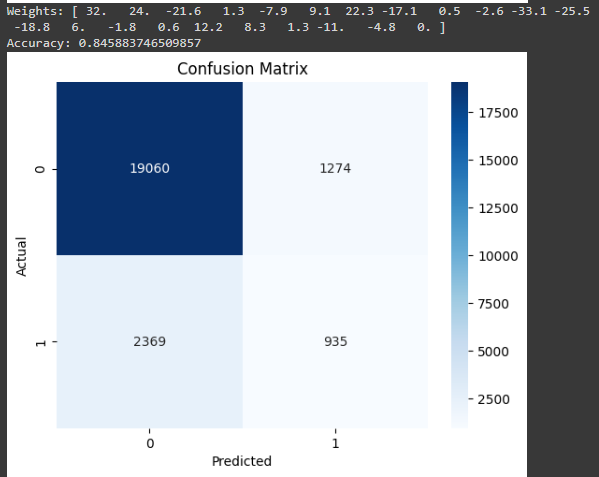
A screenshot of a graph

Description automatically generated

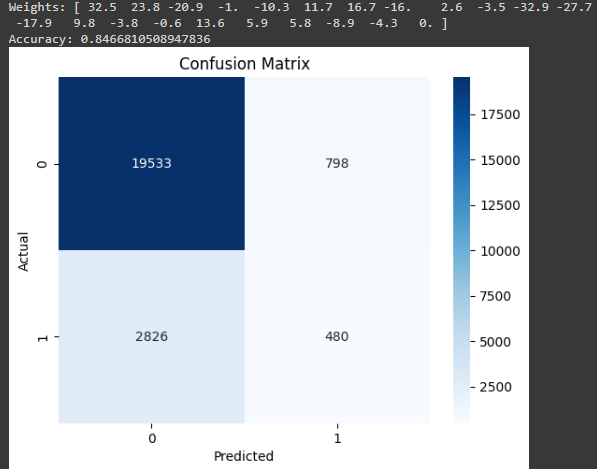
**Fold 7:**



**Fold 8:**



**Fold 9:**



**Fold 10:**

