```
#import modules
import numpy as np
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#load data
dataset_path = '/content/diabetes.csv'
data = pd.read_csv(dataset_path)

data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigre
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

```
# extract features and labeles from data
X = data.drop('Outcome', axis=1)
y = data['Outcome'].values
```

```
type(X),type(y)
```

```
(pandas.core.frame.DataFrame, numpy.ndarray)
```

```
# split data into train & test sets
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.5)
```

```
X_train.shape,X_test.shape
```

```
((384, 8), (384, 8))
```

```
X.shape[1]
```

```
8
```

```
#converting the input features to a binary format

X_binarised_train = X_train.apply(pd.cut, bins=2, labels=[1,0]).values
X_binarised_test = X_test.apply(pd.cut, bins=2, labels=[1,0]).values
```

```
X_binarised_train.shape,X_binarised_test.shape
```

```
((384, 8), (384, 8))
```

```
# define MP Neuron Class with necessary methods
class MPNeuron:

    def __init__(self):
        self.b = None

    def model(self, x):
        return(sum(x) >= self.b)

    def predict(self, X):
        Y = []
        for x in X:
            result = self.model(x)
            Y.append(result)
        return np.array(Y)

    def accuracy_score(self,pred,actual):

        score=0
        num_samples = len(pred)
```

```
        for i in range(num_samples):
            if(pred[i]==actual[i]):
                score+=1
        return (score/num_samples)


    def fit(self, X, Y):
        accuracy = {}

        for b in range(X.shape[1] + 1):
            self.b = b
            Y_pred = self.predict(X)
            accuracy[b] = self.accuracy_score(Y_pred, Y)

        best_b = max(accuracy, key = accuracy.get)
        self.b = best_b

        print('Optimal value of b is', best_b)
        print('Highest accuracy is', accuracy[best_b])
```

```
#Calling the class MPNeuron
mp_neuron = MPNeuron()
#Calling the fit method inside the class on the training data
mp_neuron.fit(X_binarised_train, Y_train)
```

```
Optimal value of b is 8
Highest accuracy is 0.6614583333333334
```

```
#Calling the predict method inside the class on the testing data to make predictions
pred = mp_neuron.predict(X_binarised_test)
c = 0
for i in range(len(Y_test)):
    #print(pred[i],bool(Y_test[i]),sep='-->')
    if(pred[i]==bool(Y_test[i])):
        c+=1
print("Test Accuracy: ", (c/len(Y_test)))
```

```
Test Accuracy:  0.5677083333333334
```

Perceptron

```
#load data
dataset_path = '/content/diabetes.csv'
data = pd.read_csv(dataset_path)

data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	
4	0	137	40	35	168	43.1	2.288	33	

```
# extract features and labels from data
X = data.iloc[:,:-1].values
y = data.iloc[:,8].values
```

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

```
((768, 8), (768,))
```

```
# split data into train and test sets
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.15,random_state=0, stratify = y)
```

```
X_train[0]
```

```
array([  6.   , 125.   ,  68.   ,  30.   , 120.   ,  30.   ,   0.464,
        32.   ])
```

```
X_train.shape, X_test.shape
```

```
((652, 8), (116, 8))
```

```
W = np.ones((1,X_train.shape[1])) # Initialize weights with all ones
epochs = 100
```

```
W
```

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

```
X_train[0]
```

```
array([  6.   , 125.   ,  68.   ,  30.   , 120.   ,  30.   ,   0.464,
        32.   ])
```

```
# perceptron learning algorithm

for epoch in range(epochs):
    for i in range(X_train.shape[0]):
        g = np.multiply(W,X_train[i])

        if(y_train[i]==1 and np.sum(g)<0):
            W = np.add(W,X_train[i])
        if(y_train[i]==0 and np.sum(g)>=0):
            W = np.subtract(W,X_train[i])
```

```
W
```

```
array([[1759.   ,   81.   , -238.   ,  120.   ,  -49.   , -257.4  ,
         888.159, -136.   ]])
```

```
test = X_test[0]
test
```

```
array([ 4.   , 76.   , 62.   ,  0.   ,  0.   , 34.   ,  0.391, 25.   ])
```

```
g = np.sum(np.multiply(W,test))
```

```
g
```

```
-13368.329830999768
```

```
y_test[0]
```

```
0
```

```
y_pred = []
for i in range(X_test.shape[0]):
    g = np.multiply(W,X_test[i])
    if np.sum(g)>=0:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

```
score = 0
print("Actual-->Predicted\n")
for i in range(X_test.shape[0]):
    print(y_test[i],'-->',y_pred[i])

    if(y_test[i]==y_pred[i]):
        score+=1

acc = (score/X_test.shape[0])*100
print('\nAccuracy %.2f' % acc,"%")
```

```
 Actual-->Predicted

 0 --> 0
 0 --> 0
 0 --> 0
 1 --> 0
 1 --> 0
 0 --> 0
 0 --> 0
 0 --> 1
 0 --> 0
```

```
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
1 --> 0
0 --> 0
1 --> 0
1 --> 0
1 --> 0
0 --> 0
1 --> 0
1 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
1 --> 0
1 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
```

```
# Train Data

y_pred = []
for i in range(X_train.shape[0]):
    g = np.multiply(W,X_train[i])
    if np.sum(g)>=0:
        y_pred.append(1)
    else:
        y_pred.append(0)


print("Actual-->Predicted\n")

score=0
for i in range(X_train.shape[0]):
    print(y_train[i],'-->',y_pred[i])

    if(y_train[i]==y_pred[i]):
        score+=1

acc = (score/X_train.shape[0])*100
print('\nAccuracy %.2f' % acc,"%")
```

```
Actual-->Predicted

0 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
1 --> 0
0 --> 0
```

```
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
0 --> 1
0 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
1 --> 1
0 --> 0
0 --> 1
1 --> 0
0 --> 0
1 --> 0
0 --> 0
1 --> 0
1 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
0 --> 0
0 --> 0
0 --> 0
1 --> 0
```

```
from sklearn.linear_model import Perceptron

model = Perceptron(max_iter=100, random_state=5)
model.fit(X_train,y_train)
pred = model.predict(X_test)

acc = model.score(X_test,y_test)
print(acc*100)

y_pred = model.predict(X_test)
```

```
60.3448275862069
```

```
y_pred
```

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

```
score = 0
#print("Actual-->Predicted\n")
for i in range(X_test.shape[0]):
    #print(y_test[i],'-->',pred[i])

    if(y_test[i]==pred[i]):
        score+=1

acc = (score/X_test.shape[0])*100
print('\nAccuracy %.2f' % acc,"%")
```

```
Accuracy 60.34 %
```