Prac3

import numpy as np

# Simplified 3x3 representations of digits 0–9

digits\_3x3 = {

0: ["111", "101", "111"],

1: ["010", "010", "111"],

2: ["111", "011", "110"],

3: ["111", "011", "111"],

4: ["101", "111", "001"],

5: ["111", "100", "111"],

6: ["110", "111", "111"],

7: ["111", "001", "001"],

8: ["111", "111", "111"],

9: ["111", "101", "011"]

}

# Convert to flat numpy vectors

def digit\_to\_vector(digit):

lines = digits\_3x3[digit]

return np.array([int(pixel) for row in lines for pixel in row])

X = np.array([digit\_to\_vector(i) for i in range(10)]) # Features

y = np.array([1 if i % 2 == 0 else 0 for i in range(10)]) # Even: 1, Odd: 0

# Perceptron training

def train\_perceptron(X, y, epochs=100, lr=0.1):

weights = np.zeros(X.shape[1])

bias = 0

for \_ in range(epochs):

for xi, target in zip(X, y):

activation = np.dot(xi, weights) + bias

prediction = 1 if activation >= 0 else 0

update = lr \* (target - prediction)

weights += update \* xi

bias += update

return weights, bias

# Train

weights, bias = train\_perceptron(X, y)

# Predict

def predict(vec):

return 1 if (np.dot(vec, weights) + bias) >= 0 else 0

# Test

print("Digit | Even?")

print("--------------")

for i in range(10):

result = predict(digit\_to\_vector(i))

print(f" {i} | {'Yes' if result == 1 else 'No'}")