Objective : WAP to implement the Perceptron Learning Algorithm using numpy in Python. Evaluate performance of a single perceptron for NAND and XOR truth tables as input dataset

Description of the model : A perceptron is a type of artificial neuron that uses a weighted sum of inputs followed by an activation function (step function). It updates its weights iteratively using a learning rule based on the difference between the predicted and actual outputs.

- The NAND function is linearly separable, meaning a perceptron can learn it.
- The XOR function is not linearly separable, meaning a single-layer perceptron cannot learn it.

Python Implementation :

```
import numpy as np

class Perceptron:

    def __init__(self, input_size, learning_rate=0.1, epochs=100):

        self.learning_rate = learning_rate

        self.epochs = epochs

        self.weights = np.random.rand(input_size + 1)

    def activation(self, x):

        return 1 if x >= 0 else 0

    def predict(self, x):

        x = np.insert(x, 0, 1)

        return self.activation(np.dot(self.weights, x))

    def train(self, X, y):

        for epoch in range(self.epochs):
```

```
        for i in range(len(X)):
            x_i = np.insert(X[i], 0, 1)
            y_pred = self.activation(np.dot(self.weights, x_i))
            self.weights += self.learning_rate * (y[i] - y_pred) * x_i
    def evaluate(self, X, y):
        correct = 0
        for i in range(len(X)):
            if self.predict(X[i]) == y[i]:
                correct += 1
        accuracy = correct / len(y) * 100
        return accuracy
X_nand = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_nand = np.array([1, 1, 1, 0])
X_xor = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y_xor = np.array([0, 1, 1, 0])
perceptron_nand = Perceptron(input_size=2)
perceptron_nand.train(X_nand, y_nand)
nand_accuracy = perceptron_nand.evaluate(X_nand, y_nand)
print(f"NAND Perceptron Accuracy: {nand_accuracy}%")


perceptron_xor = Perceptron(input_size=2)
perceptron_xor.train(X_xor, y_xor)
xor_accuracy = perceptron_xor.evaluate(X_xor, y_xor)
print(f"XOR Perceptron Accuracy: {xor_accuracy}%")
```

Description of code :

Class Perceptron: Implements a perceptron model with weight initialization, activation function, training using the perceptron learning rule, and evaluation.

Training (train function): Updates weights using the perceptron learning rule.

Evaluation (evaluate function): Computes the accuracy of the perceptron.

Datasets: NAND and XOR truth tables are used.

Results: The accuracy of the perceptron on NAND and XOR datasets is displayed.

Performance Evaluation :

NAND Perceptron: Achieves 100% accuracy as the NAND function is linearly separable.

XOR Perceptron: Achieves poor accuracy (typically 50%) as a single-layer perceptron cannot solve the XOR problem due to its non-linearly separable nature.