Aufgabe: Implementiere ein Perzeptron

Du wirst ein einfaches **Perzeptron** implementieren, das die logische **AND-Funktion** erlernt. Ein Perzeptron ist ein einfaches neuronales Netz mit einer Eingabeschicht und einer Ausgabeschicht.

Aktivierungsfunktionen (Einsteiger)

Implementiere die folgenden Aktivierungsfunktionen:

- **Step-Funktion** (binäre Entscheidung)
- **Sigmoid-Funktion** (weiche Übergänge)
- **ReLU-Funktion** (nur positive Werte)
- **Tanh-Funktion** (Werte zwischen -1 und 1)

Deine Aufgabe: Ergänze die fehlenden Aktivierungsfunktionen.

```
import numpy as np

# Aktivierungsfunktionen
def step_function(x):
        # TODO: Implementiere die Step-Funktion

def sigmoid(x):
    # TODO: Implementiere die Sigmoid-Funktion

def relu(x):
    # TODO: Implementiere die ReLU-Funktion

def tanh(x):
    # TODO: Implementiere die Tanh-Funktion
```

Perzeptron-Klasse erstellen

Das Perzeptron benötigt:

- Gewichte und Bias, die zufällig initialisiert werden
- Eine Methode `predict(x)`, um Vorhersagen zu treffen
- Eine Methode `train(X, y, epochs)`, um das Modell zu trainieren

Deine Aufgabe: Implementiere die `predict()` -Methode.

```
class Perceptron:
    def __init__(self, input_size, learning_rate=0.1, activation='step'):
        self.weights = np.random.rand(input_size) - 0.5
        self.bias = np.random.rand() - 0.5
        self.learning_rate = learning_rate
        self.activation_function = get_activation_function(activation)

    def predict(self, x):
        # TODO: Berechne die gewichtete Summe der Eingaben und wende die Ak
```

Training implementieren

Beim Training werden:

- Fehler berechnet: **Fehlervariable = Soll - Ist**
- Gewichte und Bias aktualisiert:
  $w_i = w_i + \eta(y_{target} - y_{pred})x_i$
  $b = b + \eta(y_{target} - y_{pred})$

Deine Aufgabe: Implementiere das Training.

```
    def train(self, X, y, epochs=10):
        for _ in range(epochs):
            for i in range(len(X)):
                prediction = self.predict(X[i])
                error = y[i] - prediction
                # TODO: Aktualisiere die Gewichte und den Bias
```

Teste dein Modell

Nutze die logische **AND-Funktion** als Trainingsdaten:

Deine Aufgabe: Erstelle das Perzeptron und trainiere es.

```
# Beispiel-Daten für AND-Gatter
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
y = np.array([0, 0, 0, 1])

# TODO: Erstelle ein Perzeptron mit 2 Eingaben und trainiere es
```

```
perceptron = Perceptron(input_size=2, learning_rate=0.1, activation='step')

# TODO: Trainiere das Modell
perceptron.train(X, y, epochs=10)

# TODO: Teste das Modell mit Vorhersagen
predictions = [perceptron.predict(x) for x in X]
print("Vorhersagen:", predictions)
```

Bonusaufgaben

1. **Verändere die Lernrate und beobachte, wie sich das Training verhält.**
2. **Teste andere Aktivierungsfunktionen wie Sigmoid , Tanh oder ReLU.**
3. **Erstelle ein Perzeptron für das OR-Gatter.**