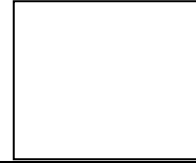




LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : PERCEPTRON
NAMA : DIMAS TRI MUSTAKIM
NIM : 205150200111049
TANGGAL : 26/09/2022
ASISTEN : ANDIKA IRZA PRADANA



A. Praktikum

1. Buka Google Colaboratory melalui [tautan ini](#).
2. Tulis kode berikut ke dalam setiap *cell* pada *notebook* tersebut.
 - a. Fungsi Step Perceptron

```
def percep_step(input, th=0):  
    return 1 if input > th else -1 if input < -th else 0
```

b. Fungsi Training Perceptron

```
def percep_fit(X, target, th=0, a=1, max_epoch=-1, verbose=False,  
draw=False):  
    w = np.zeros(len(X[0]) + 1)  
    bias = np.ones((len(X), 1))  
    X = np.hstack((bias, X))  
    stop = False  
    epoch = 0  
  
    while not stop and (max_epoch == -1 or epoch < max_epoch):  
        stop = True  
        epoch += 1  
  
        if verbose:  
            print('\nEpoch', epoch)  
  
        for r, row in enumerate(X):  
            y_in = np.dot(row, w)  
            y = percep_step(y_in, th)  
  
            if y != target[r]:  
                stop = False  
                w = [w[i] + a * target[r] * row[i] for i in  
range(len(row))]  
  
        if verbose:  
            print('Bobot:', w)  
  
        if draw:  
            plot(line(w, th), line(w, -th), X, target)  
  
    return w, epoch
```

c. Fungsi Testing Perceptron

```
def percep_predict(X, w, th=0):
    Y = []
    for x in X:
        y_in = w[0] + np.dot(x, w[1:])
        y = percep_step(y_in, th)
        Y.append(y)
    return Y
```

d. Fungsi Hitung Akurasi

```
def calc_accuracy(a, b):
    s = [1 if a[i] == b[i] else 0 for i in range(len(a))]
    return sum(s) / len(a)
```

e. Logika AND

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, -1, -1, -1
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

print('Epochs:', epoch)
print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

f. Logika OR

```
train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = 1, 1, 1, -1
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)

print('Epochs:', epoch)
print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)
```

g. Logika AND NOT

```

train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, -1, -1
th = .2
model, epoch = percep_fit(train, target, th, verbose=True,
draw=True)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)
print('Epochs:', epoch)
print('Output:', output)
print('Target:', target)
print('Accuracy:', accuracy)

```

h. Logika XOR

```

train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, 1, -1
th = .2
model, epoch = percep_fit(train, target, th, max_epoch=50,
verbose=True, draw=False)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)
print('Output:', output)
print('Accuracy:', accuracy)

```

i. Load dan plot data

```

import seaborn as sns
import matplotlib.pyplot as plt

iris = sns.load_dataset('iris')

sns.pairplot(iris, hue='species')
plt.show()

```

j. Menghapus Kelas Virginica

```

iris = iris.loc[iris['species'] != 'virginica']

sns.pairplot(iris, hue='species')
plt.show()

```

k. Menghapus ciri sepal_width dan petal_width

```

iris = iris.drop(['sepal_width', 'petal_width'], axis=1)

```

```
sns.pairplot(iris, hue='species')
plt.show()
```

1. Proses Training dan Testing

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale

X = iris[['sepal_length', 'petal_length']].to_numpy()
X = minmax_scale(X)

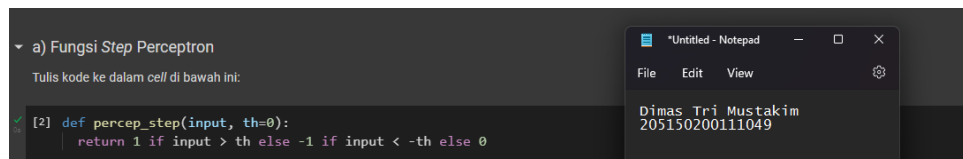
y = iris['species'].to_numpy()
c = {'setosa': -1, 'versicolor': 1}
y = [c[i] for i in y]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=.3)
w, epoch = percep_fit(X_train, y_train, verbose=True, draw=True)
out = percep_predict(X_test, w)
accuracy = calc_accuracy(out, y_test)

print('Epochs:', epoch)
print('Accuracy:', accuracy)
```

B. Screenshot

a. Fungsi Step Perceptron



b. Fungsi training Perceptron

▼ b) Fungsi training Perceptron

Tulis kode ke dalam cell di bawah ini:

```
[3] def percep_fit(X, target, th=0, a=1, max_epoch=-1, verbose=False, draw=False):
    w = np.zeros(len(X[0]) + 1)
    bias = np.ones((len(X), 1))
    X = np.hstack((bias, X))
    stop = False
    epoch = 0

    while not stop and (max_epoch == -1 or epoch < max_epoch):
        stop = True
        epoch += 1

        if verbose:
            print('\nEpoch', epoch)

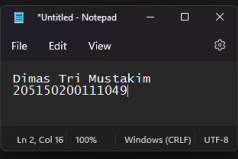
        for r, row in enumerate(X):
            y_in = np.dot(row, w)
            y = percep_step(y_in, th)

            if y != target[r]:
                stop = False
                w = [w[i] + a * target[r] * row[i] for i in range(len(row))]

        if verbose:
            print('Bobot:', w)

        if draw:
            plot(line(w, th), line(w, -th), X, target)

    return w, epoch
```

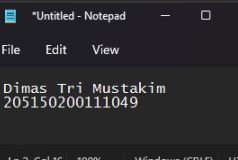


c. Fungsi testing Perceptron

▼ c) Fungsi testing Perceptron

Tulis kode ke dalam cell di bawah ini:

```
[4] def percep_predict(X, w, th=0):
    Y = []
    for x in X:
        y_in = w[0] + np.dot(x, w[1:])
        y = percep_step(y_in, th)
        Y.append(y)
    return Y
```

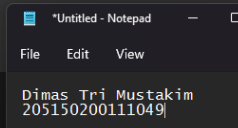


d. Fungsi Hitung Akurasi

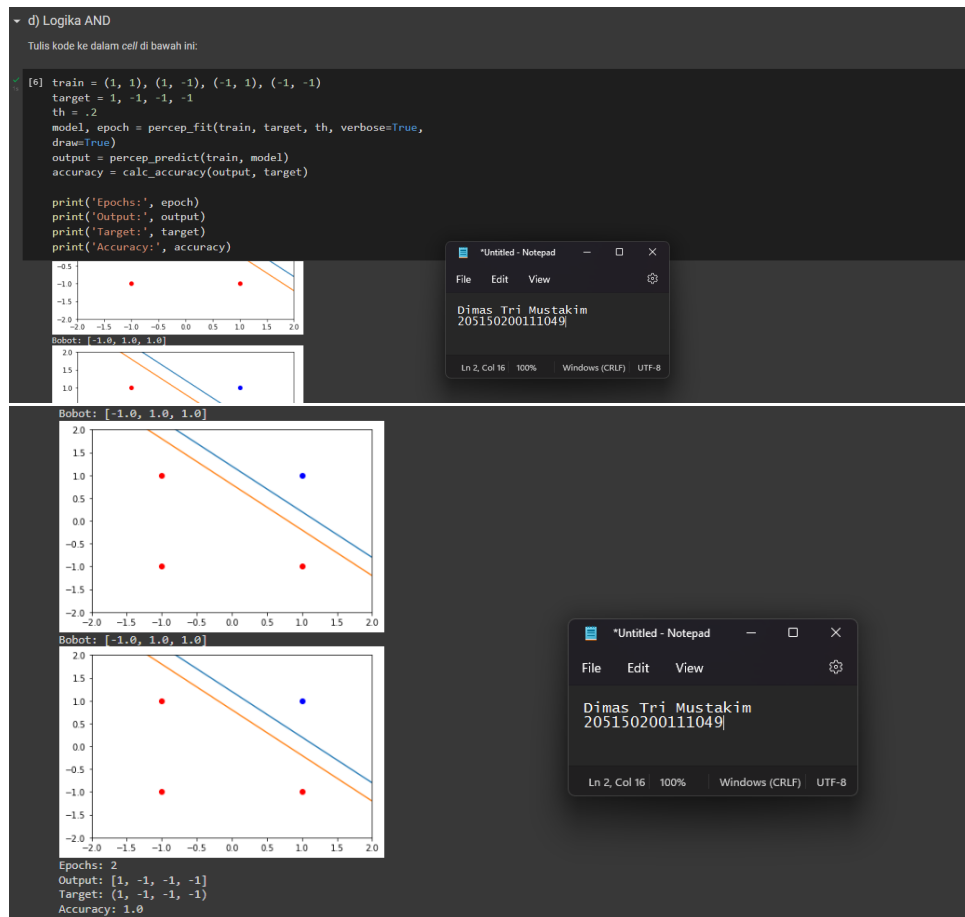
▼ Fungsi Hitung Akurasi

Tulis kode ke dalam cell di bawah ini:

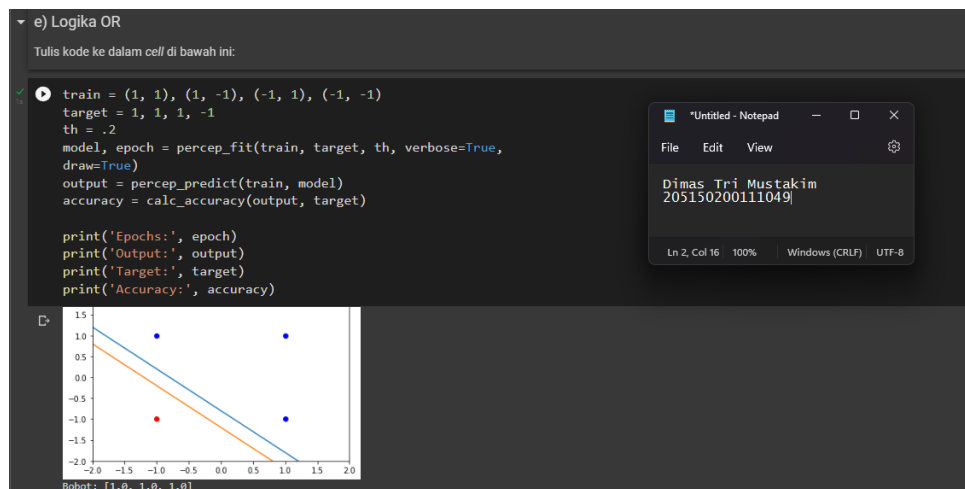
```
[5] def calc_accuracy(a, b):
    s = [1 if a[i] == b[i] else 0 for i in range(len(a))]
    return sum(s) / len(a)
```

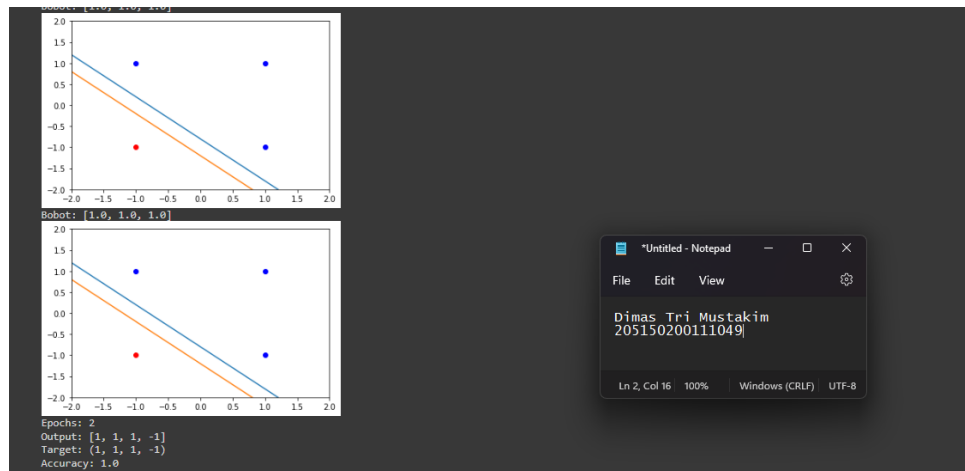


e. Logika AND

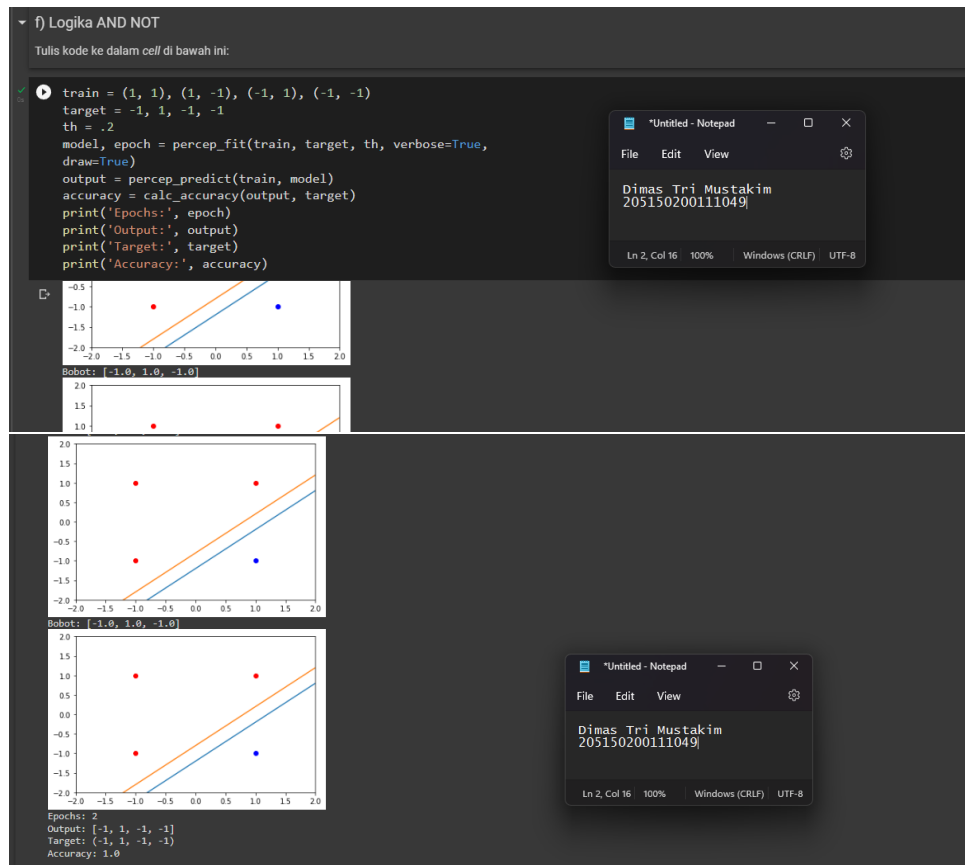


f. Logika OR





g. Logika AND NOT



h. Logika XOR

```
g) Logika XOR
Tulis kode ke dalam cell/ di bawah ini:

[9] train = (1, 1), (1, -1), (-1, 1), (-1, -1)
target = -1, 1, 1, -1
th = .2
model, epoch = percep_fit(train, target, th, max_epoch=50,
verbose=True, draw=False)
output = percep_predict(train, model)
accuracy = calc_accuracy(output, target)
print('Output:', output)
print('Accuracy:', accuracy)

Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

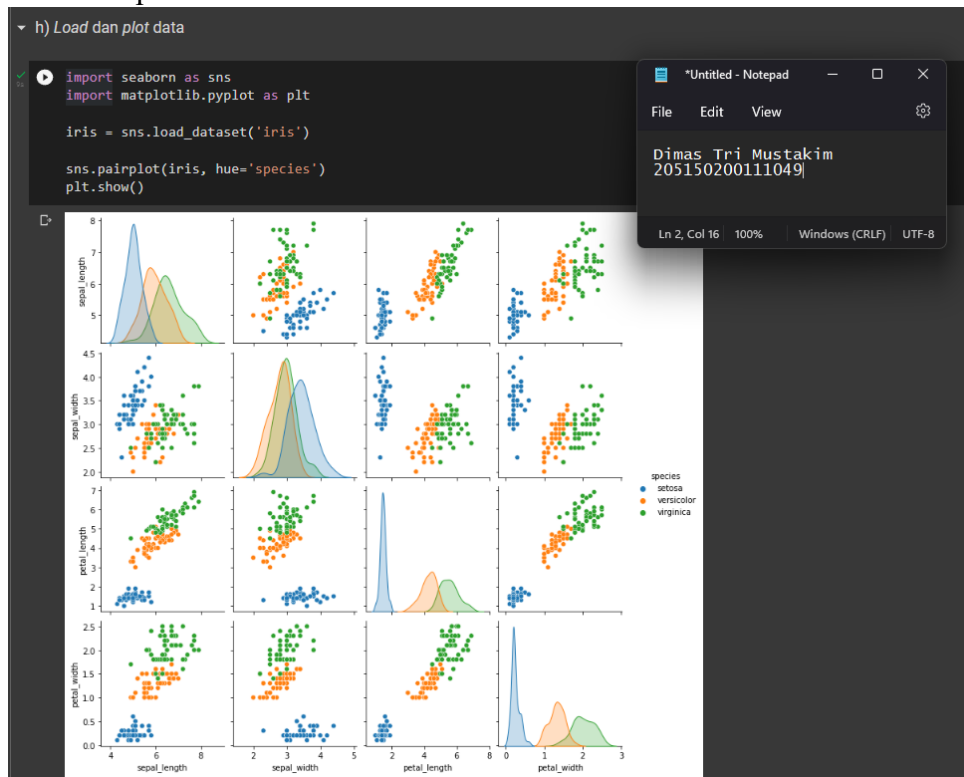
Epoch 42
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 48
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 49
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]

Epoch 50
Bobot: [-1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, -2.0]
Bobot: [1.0, -1.0, -1.0]
Bobot: [0.0, 0.0, 0.0]
Output: [0, 0, 0, 0]
Accuracy: 0.0
```

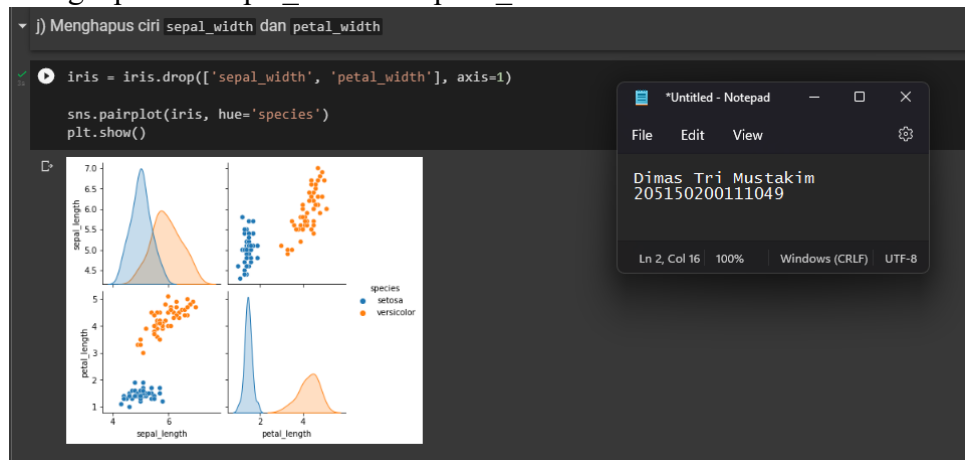
i. Load dan plot data



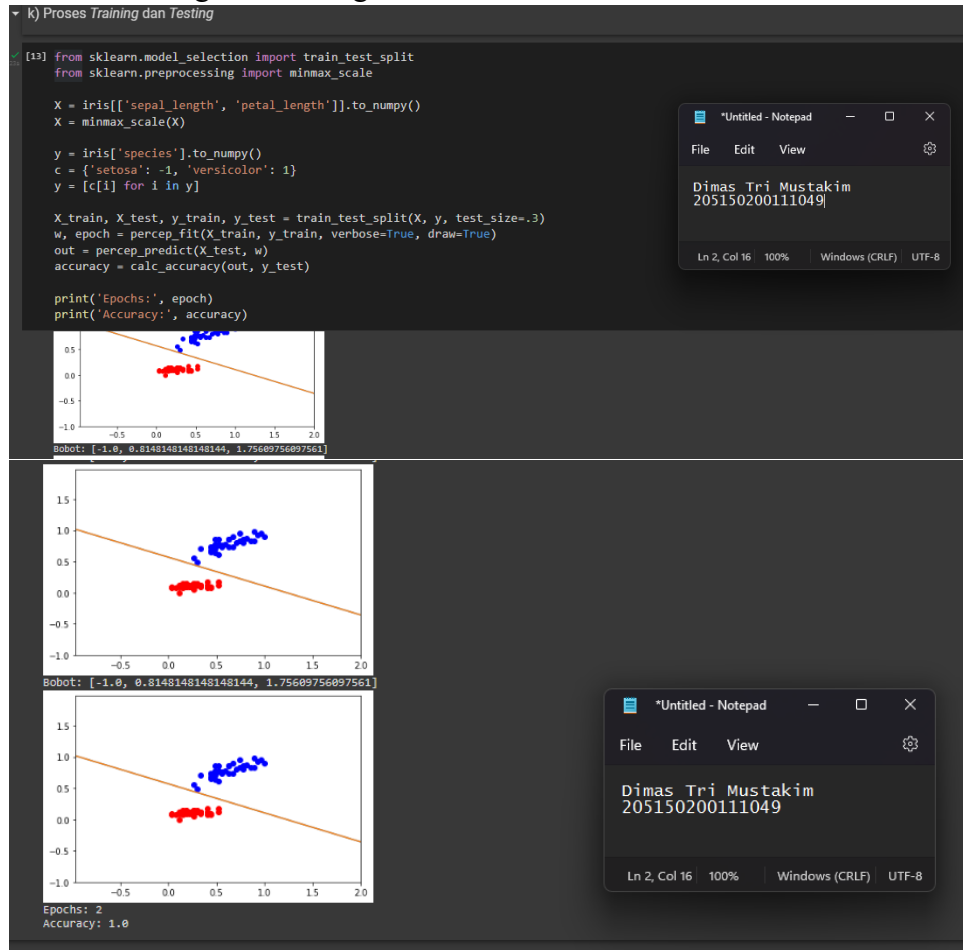
j. Menghapus Kelas Virginica



k. Menghapus ciri sepal_width dan petal_width



1. Proses Training dan Testing



C. Analisis

1. Mengapa perceptron gagal dalam melakukan proses training menggunakan data logika XOR? Jelaskan.

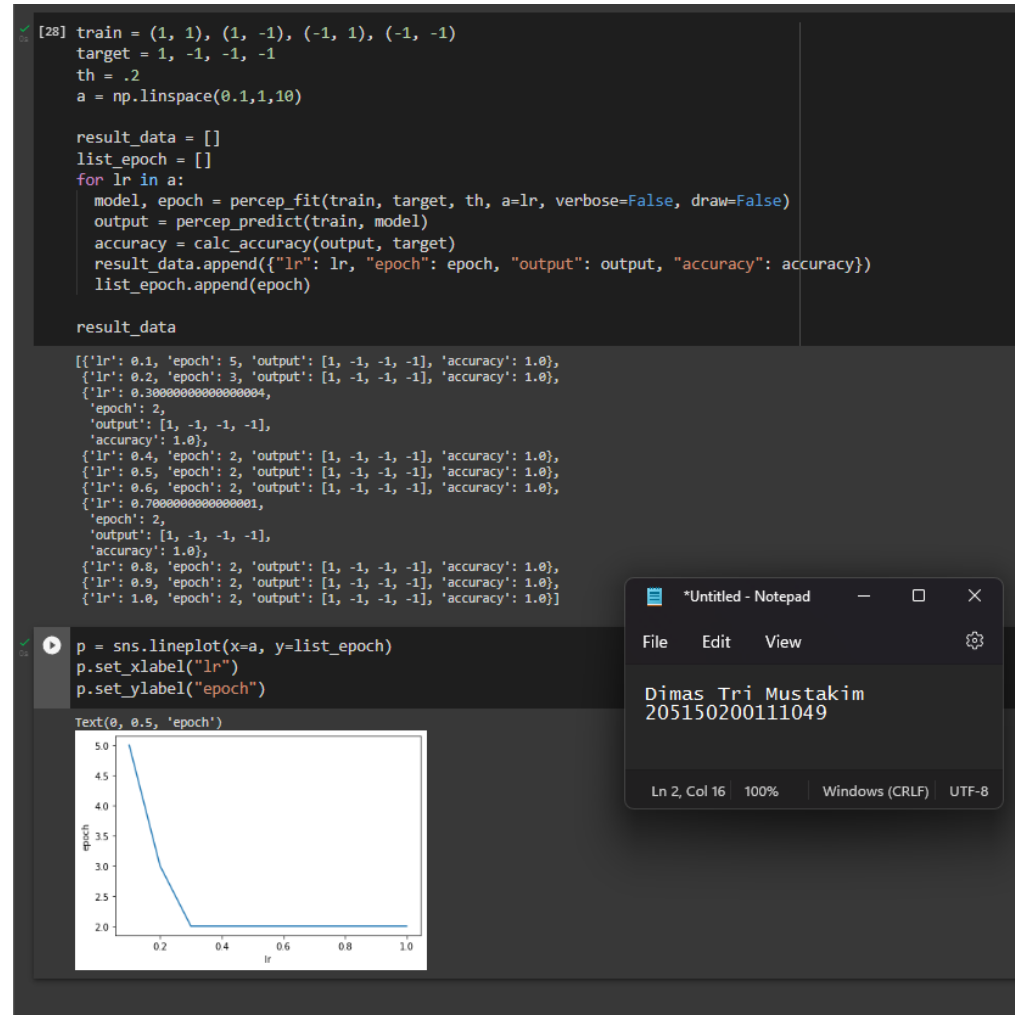
Jawab:

Karena logika XOR merupakan permasalahan yang termasuk *not linearly separable*. Perceptron hanya bisa memisahkan data dengan satu buah garis linear, yang membuatnya tidak bisa melakukan klasifikasi pada permasalahan XOR. Berdasarkan pengamatan saya di dalam hasil training, setiap akhir epoch, semua bobot selalu kembali menjadi 0 dan tidak ada perubahan bobot dari awal hingga epoch terakhir.

2. Lakukan pelatihan data logika AND dengan learning rate yang berbeda-beda. Amati jumlah epoch yang dilakukan. Bagaimanakah efeknya pada proses pelatihan?

Jawab:

Efeknya adalah ketika jumlah learning rate ditambah, maka jumlah epoch akan menurun. Di dalam percobaan yang saya lakukan, epoch paling kecil ada di angka 2. Angka learning rate yang optimal menurut saya ada di angka 0.3 karena setelah learning rate 0.3 nilai epoch tidak lagi berkurang.



D. Kesimpulan

Beberapa perbedaan antara Perceptron dengan Hebb Net yaitu bahwa pada Perceptron terdapat parameter learning rate sedangkan Hebb Net tidak. Perceptron juga bisa dilatih dengan nilai epoch lebih dari satu sedangkan Hebb Net hanya dilatih dengan satu epoch. Perbedaan selanjutnya adalah bahwa perceptron menggunakan fungsi aktivasi yang dapat menghasilkan nilai 1, 0, dan -1 sedangkan Hebb Net menggunakan fungsi bipolar.

Learning rate merupakan parameter yang digunakan dalam proses training yang berfungsi untuk mengontrol seberapa banyak bobot dari jaringan disesuaikan untuk setiap proses training. Pada data yang kompleks, learning rate sangat penting agar pergeseran decision boundary tidak terlalu besar yang menyebabkan posisi decision boundary tersebut tidak baik.

Menurut saya, kasus yang bisa diselesaikan dengan perceptron adalah kasus klasifikasi yang *linearly separable* dan digunakan untuk memisahkan dua kelas. Selain itu, perceptron tidak bisa digunakan.