

# **LAPORAN UTS KECERDASAN BUATAN**



Disusun oleh :

Muhammad Zidan Alif Oktavian (21091397045)

---

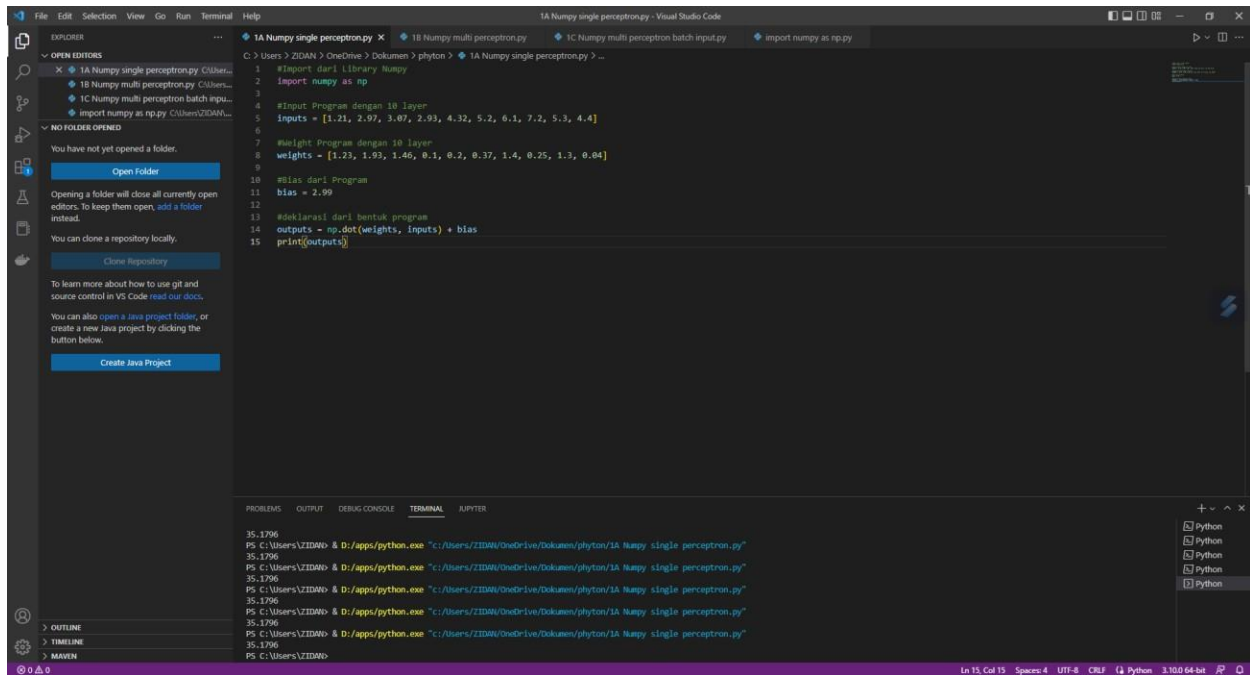
**PROGRAM STUDI D4 MANAJEMEN INFORMATIKA**

**FAKULTAS VOKASI**

**UNIVERSITAS NEGERI SURABAYA**

**2022**

## No. 1A



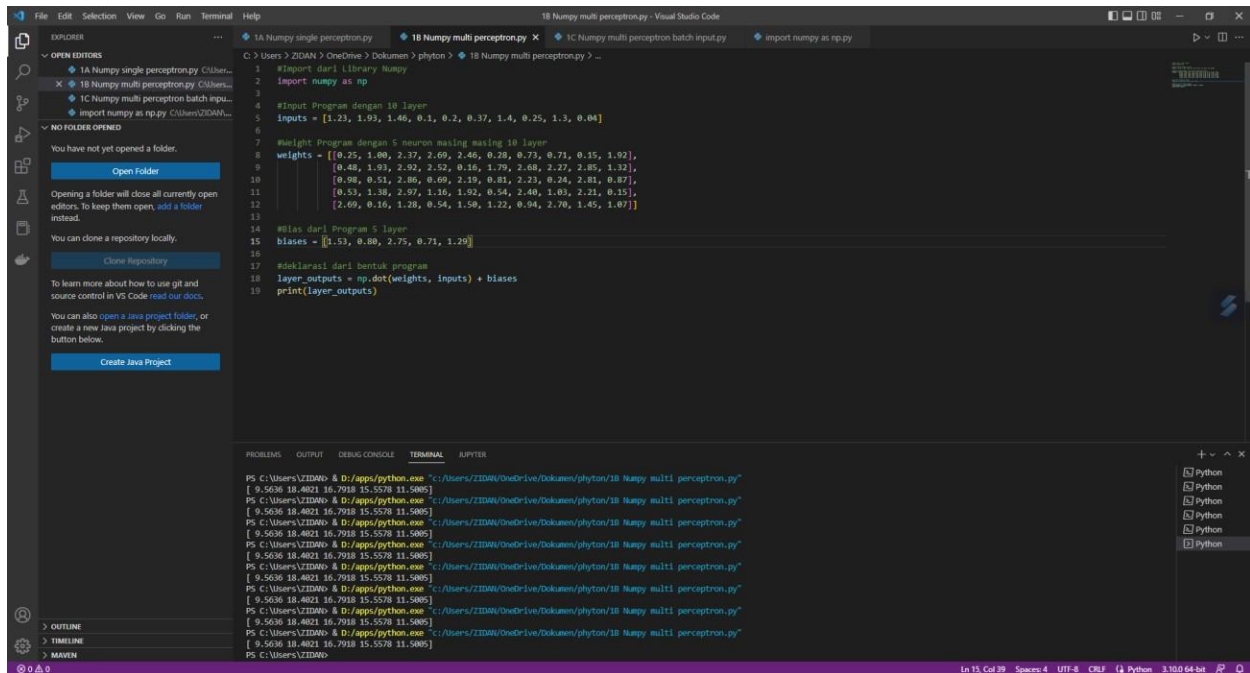
```
1 #Import dari Library Numpy
2 import numpy as np
3
4 #Input Program dengan 10 layer
5 inputs = [1.21, 2.97, 3.07, 2.93, 4.32, 5.2, 6.1, 7.2, 5.3, 4.4]
6
7 #Weight Program dengan 10 layer
8 weights = [1.23, 1.93, 1.46, 0.1, 0.2, 0.37, 1.4, 0.25, 1.3, 0.84]
9
10 #bias dari Program
11 bias = 2.99
12
13 #Kalkulasi dari bentuk program
14 outputs = np.dot(weights, inputs) + bias
15 print(outputs)
```

35.1796  
PS C:\Users\ZIDAN> & D:/apps/python.exe "c:/Users/ZIDAN/OneDrive/Dokumen/python/1A Numpy single perceptron.py"  
35.1796  
PS C:\Users\ZIDAN> & D:/apps/python.exe "c:/Users/ZIDAN/OneDrive/Dokumen/python/1A Numpy single perceptron.py"  
35.1796  
PS C:\Users\ZIDAN> & D:/apps/python.exe "c:/Users/ZIDAN/OneDrive/Dokumen/python/1A Numpy single perceptron.py"  
35.1796  
PS C:\Users\ZIDAN> & D:/apps/python.exe "c:/Users/ZIDAN/OneDrive/Dokumen/python/1A Numpy single perceptron.py"  
35.1796  
PS C:\Users\ZIDAN> & D:/apps/python.exe "c:/Users/ZIDAN/OneDrive/Dokumen/python/1A Numpy single perceptron.py"  
35.1796  
PS C:\Users\ZIDAN>

### Analisis:

Single Neuron Perceptron adalah unit jaringan saraf tiruan paling dasar yang melakukan perhitungan yang tepat untuk mendeteksi fitur dalam data input dengan menambahkan weight dan bias.

## No. 1B



```
1 #Import dari library Numpy
2 import numpy as np
3
4 #Input Program dengan 10 layer
5 inputs = [1.23, 1.93, 1.46, 0.1, 0.2, 0.37, 1.4, 0.25, 1.3, 0.04]
6
7 #Weight Program dengan 5 neuron masing masing 10 layer
8 weights = [[0.25, 1.00, 2.37, 2.69, 2.46, 0.20, 0.79, 0.71, 0.15, 1.92],
9            [0.48, 1.93, 2.92, 2.52, 0.16, 1.79, 2.66, 2.27, 2.85, 1.32],
10           [0.98, 0.51, 2.86, 0.69, 2.19, 0.81, 2.23, 0.24, 2.81, 0.87],
11           [0.53, 1.38, 2.97, 1.16, 1.92, 0.54, 2.40, 1.03, 2.21, 0.15],
12           [2.69, 0.16, 1.28, 0.54, 1.50, 1.22, 0.94, 2.70, 1.45, 1.07]]
13
14 #Bias dari Program 5 layer
15 biases = [1.53, 0.88, 2.75, 0.71, 1.29]
16
17 #deklarasi dari bentuk program
18 layer_outputs = np.dot(weights, inputs) + biases
19 print(layer_outputs)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN > D:/apps/python.exe "C:/Users/ZIDAN/OneDrive/Dokumen/python/10_Numpy_multi_perceptron.py"
[ 9.5636 18.4021 16.7918 15.5578 11.5085]
PS C:\Users\ZIDAN >
```

### Analisis:

Cara kerja multi layer adalah input layer menyuplai input vektor pada jaringan, kemudian input yang dimasukkan melakukan komputasi pada layer yang kedua, lalu output dari layer yang kedua digunakan sebagai input dari layer yang ketiga dan seterusnya.

## No. 1C

```
#Import dari Library Numpy
import numpy as np

#Input Program 10 batch dengan 6 layer
inputs = [[2.03, 1.68, 2.41, 2.20, 2.71, 0.82, 2.43, 0.76, 2.31, 0.93],
          [2.40, 1.75, 0.37, 1.60, 2.26, 1.45, 2.96, 0.73, 1.57, 0.92],
          [1.51, 2.39, 0.97, 1.90, 0.91, 1.49, 1.30, 2.58, 0.68, 0.27],
          [2.74, 0.11, 0.70, 1.61, 2.97, 1.16, 1.05, 2.33, 0.51, 1.08],
          [1.19, 0.18, 3.00, 0.96, 0.64, 2.91, 1.93, 1.50, 1.29, 1.94],
          [1.18, 1.43, 1.33, 2.90, 2.25, 1.15, 0.56, 2.46, 1.25, 2.66]]

#Weight dari program 5 Batch
weights = [[2.71, 0.40, 0.04, 0.55, 2.74, 0.90, 0.86, 1.35, 3.08, 2.20],
           [2.43, 0.48, 0.61, 1.04, 2.64, 1.89, 1.68, 1.74, 1.03, 2.28],
           [1.80, 0.43, 2.39, 1.92, 2.18, 2.19, 2.71, 2.59, 0.39, 1.61],
           [1.46, 0.49, 1.74, 2.20, 1.86, 1.68, -3.92, 1.33, 3.23, 3.22],
           [0.51, 2.11, 1.22, 1.65, 1.06, 2.67, 1.49, 1.63, 2.90, -0.38,]]

#Bias program tersebut
biases = [1.98, 2.86, 1.95, 0.62, 2.28]

#deklarasi dari bentuk program tersebut
layer_outputs = np.dot(inputs, np.array(weights).T) + biases

#untuk mencetak hasil program
print(layer_outputs)

[[29.8997 30.9661 34.9658 21.7997 29.6974]
 [27.9669 30.0863 31.0869 13.1861 26.3587]
 [19.2357 23.4519 27.814 15.4605 25.3661]
 [27.54 30.5115 31.5264 21.0151 20.0964]
 [22.2235 27.4603 33.6925 20.6997 25.2833]
 [28.1026 30.9281 33.5182 31.534 25.2206]]
```

### Analisa:

Sebuah Perceptron Multilayer memiliki lapisan input dan output, dan satu atau lebih lapisan tersembunyi dengan banyak neuron ditumpuk bersama-sama. Dan sementara di Perceptron neuron harus memiliki fungsi aktivasi yang memberlakukan ambang batas, seperti ReLU atau sigmoid, neuron di Multilayer Perceptron dapat menggunakan fungsi aktivasi arbitrer apa pun.

## No. 2

### Source Code:

```
#Import dari Library Numpy
import numpy as np

#Input pertama Program 10 batch dengan 6 layer
inputs1 = [[2.03, 1.68, 2.41, 2.20, 2.71, 0.82, 2.43, 0.76, 2.31, 0.93],
            [2.40, 1.75, 0.37, 1.60, 2.26, 1.45, 2.96, 0.73, 1.57, 0.92],
            [1.51, 2.39, 0.97, 1.90, 0.91, 1.49, 1.30, 2.58, 0.68, 0.27],
            [2.74, 0.11, 0.70, 1.61, 2.97, 1.16, 1.05, 2.33, 0.51, 1.08],
            [1.19, 0.18, 3.00, 0.96, 0.64, 2.91, 1.93, 1.50, 1.29, 1.94],
            [1.18, 1.43, 1.33, 2.90, 2.25, 1.15, 0.56, 2.46, 1.25, 2.66]]

#Weight pertama dari program 5 Batch
weights1 = [[2.71, 0.40, 0.04, 0.55, 2.74, 0.90, 0.86, 1.35, 3.08, 2.20],
             [2.43, 0.48, 0.61, 1.04, 2.64, 1.89, 1.68, 1.74, 1.03, 2.28],
             [1.80, 0.43, 2.39, 1.92, 2.18, 2.19, 2.71, 2.59, 0.39, 1.61],
             [1.46, 0.49, 1.74, 2.20, 1.86, 1.68, -3.92, 1.33, 3.23, 3.22],
             [0.51, 2.11, 1.22, 1.65, 1.06, 2.67, 1.49, 1.63, 2.90, -0.38]]

#Bias pertama program tersebut
biases1 = [1.98, 2.86, 1.95, 0.62, 2.28]

#Weight kedua dari program
weights2 = [[-3.93, 0.14, 0.86, 0.48, -1.27],
            [0.36, 3.30, 3.65, 0.26, -2.60],
            [-0.12, -0.29, -0.73, 1.70, -0.01]]

#Bias kedua dari program
biases2 = [-3.83, -1.16, -2.86]

#deklarasi dari bentuk program untuk menghitung layer 1
layer1_outputs = np.dot(inputs1, np.array(weights1).T) + biases1

#deklarasi dari bentuk program untuk menghitung layer 2 dari hasil layer 1
layer2_outputs = np.dot(layer1_outputs, np.array(weights2).T) + biases2

#untuk mencetak hasil program
print(layer2_outputs)
```

### OutPut:

```
[[[-114.181821  167.871874  -4.190651]
  [-109.939322  156.555825 -15.481709]
  [-77.016902  122.745092  -6.244366]
  [-96.113066  177.726996  -2.502701]
  [-80.522298  160.082417  -3.149175]
  [-98.011474  175.986376  13.685847]]]
```

### Analisa:

Program tersebut adalah versi advance dari program 1 dimana hasil dari program 1 di berikan Bobot lagi di Weight2 lalu di tranpose dan ditambahkan biaskan lagi oleh biases2