LABORATORIUM PEMBELAJARAN ILMU KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA

BAB : BACKPROPOGATION (2)

NAMA : RAIKHAN GEZA ALBURAMA
NIM : 225150207111040
TANGGAL : 28/10/2024
ASISTEN : ALIFAH KHAIRUNNISA
ANDHIKA IHSAN CENDEKIA

A. Praktikum

1. Buka Google Collaboratory melalui tautan ini
2. Tulis kode berikut ke dalam setiap *cell* pada *notebook* tersebut.

a. Fungsi Training Backpropagation

```
def bp_fit(X, target, layer_conf, max_epoch,
max_error=0.1, learn_rate=0.1, print_per_epoch=100):
    np.random.seed(1)
    nin = [np.empty(i) for i in layer_conf]
    n = [np.empty(j + 1) if i < len(layer_conf) - 1
else np.empty(j) for i, j in enumerate(layer_conf)]

    w = [np.random.rand(layer_conf[i] + 1, layer_conf[i
+ 1]) for i in range(len(layer_conf) - 1)]

    dw = [np.empty((layer_conf[i] + 1, layer_conf[i +
1])) for i in range(len(layer_conf) - 1)]
    d = [np.empty(s) for s in layer_conf[1:]]
    din = [np.empty(s) for s in layer_conf[1:-1]]
    epoch = 0
    mse = 1
    for i in range(0, len(n) - 1):
        n[i][-1] = 1
    while (max_epoch == -1 or epoch < max_epoch) and
mse > max_error:
        epoch += 1
        mse = 0
        for r in range(len(X)):
            n[0][:-1] = X[r]
            for L in range(1, len(layer_conf)):
                nin[L] = np.dot(n[L - 1], w[L - 1])
                n[L][:len(nin[L])] = sig(nin[L])

            e = target[r] - n[-1]
            mse += sum(e ** 2)
            d[-1] = e * sigd(nin[-1])
```

```
            dw[-1] = learn_rate * d[-1] *
 n[-2].reshape((-1, 1))

            for L in range(len(layer_conf) - 1, 1, -1):
                din[L - 2] = np.dot(d[L - 1],
 np.transpose(w[L - 1][:-1]))
                d[L - 2] = din[L - 2] *
 np.array(sigd(nin[L - 1]))
                dw[L - 2] = learn_rate * d[L - 2] * n[L
 - 2].reshape((-1, 1))
            for L in range(len(dw)):
                w[L] += dw[L]
        mse /= len(X)
        if print_per_epoch > -1 and epoch %
 print_per_epoch == 0:
            print(f'Epoch {epoch}, MSE: {mse}')
    return w, epoch, mse
```


b. Fungsi *Testing* Backpropagation

```
def onehot_enc(lbl, min_val=0):
    mi = min(lbl)
    enc = np.full((len(lbl), max(lbl) - mi + 1), min_val,
np.int8)
    for i, x in enumerate(lbl):
        enc[i, x - mi] = 1
    return enc

def onehot_dec(enc, mi=0):
    return [np.argmax(e) + mi for e in enc]

def sig(X):
    return [1 / (1 + np.exp(-x)) for x in X]

def sigd(X):
    output = []
    for i, x in enumerate(X):
        s = sig([x])[0]
        output.append(s * (1 - s))
    return output
```

c. Percobaan Klasifikasi *Dataset* Iris

```
def onehot_enc(lbl, min_val=0):

    mi = min(lbl)

    enc = np.full((len(lbl), max(lbl) - mi + 1),
min_val, np.int8)

    for i, x in enumerate(lbl):

        enc[i, x - mi] = 1

    return enc

def onehot_dec(enc, mi=0):

    return [np.argmax(e) + mi for e in enc]

def sig(X):

    return [1 / (1 + np.exp(-x)) for x in X]

def sigd(X):

    output = []

    for i, x in enumerate(X):

        s = sig([x])[0]
```

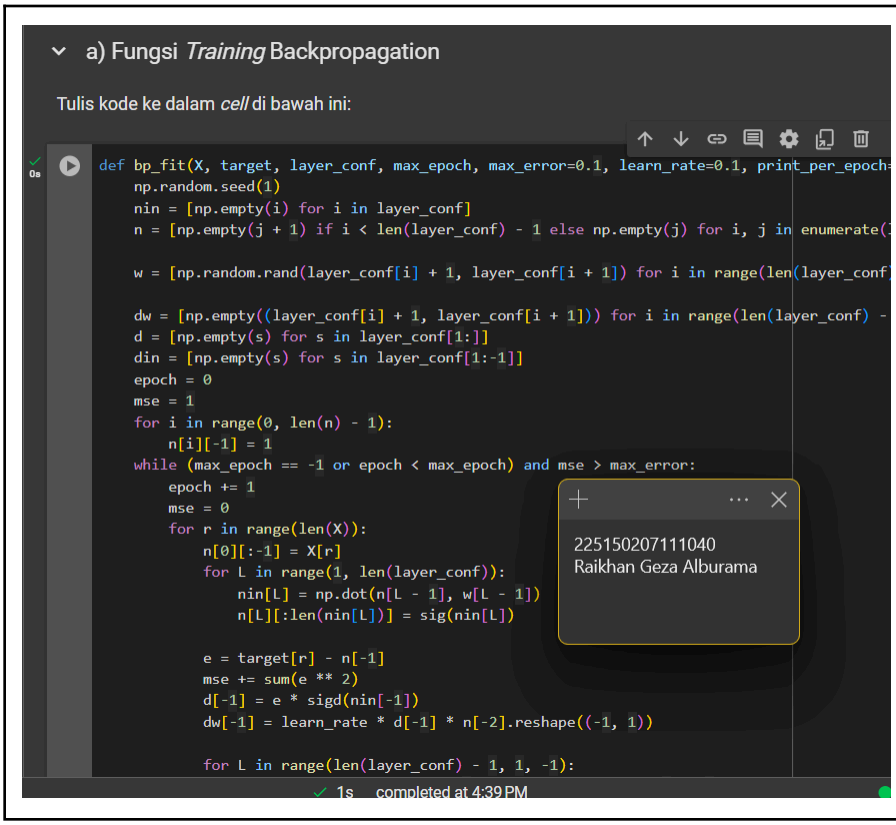
```
        output.append(s * (1 - s))

    return output
```

```
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import minmax_scale
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
X = minmax_scale(iris.data)

def onehot_enc(labels, min_val=0):
    mi = min(labels)
    enc = np.full((len(labels), max(labels) - mi + 1),
min_val, dtype=np.int8)
    for i, x in enumerate(labels):
        enc[i, x - mi] = 1
    return enc

def onehot_dec(encoded_labels):
    return np.argmax(encoded_labels, axis=1)

Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X,
Y, test_size=0.3, random_state=1)

w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 3,
3), learn_rate=0.1, max_epoch=1000, max_error=0.1,
print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')

predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)

accuracy = accuracy_score(predict, y_test)
```

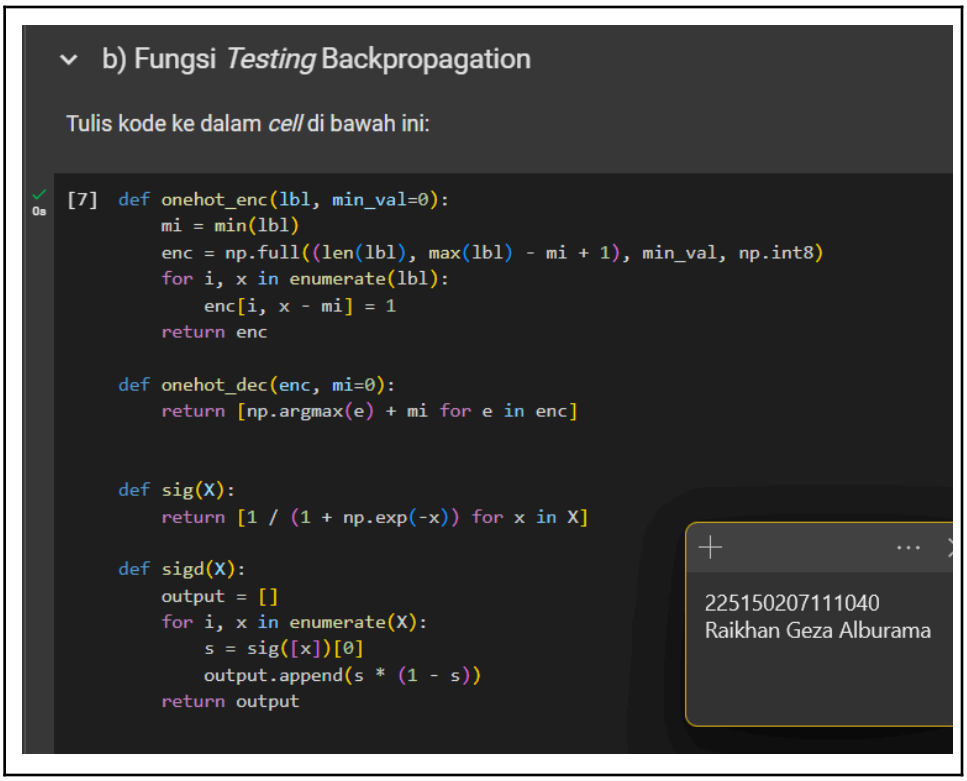
```
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

B. Screenshot

a. Fungsi *Training* Backpropagation

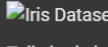


b. Fungsi *Testing* Backpropagation



c. Percobaan Klasifikasi *Dataset* Iris

c) Percobaan Klasifikasi Dataset Iris



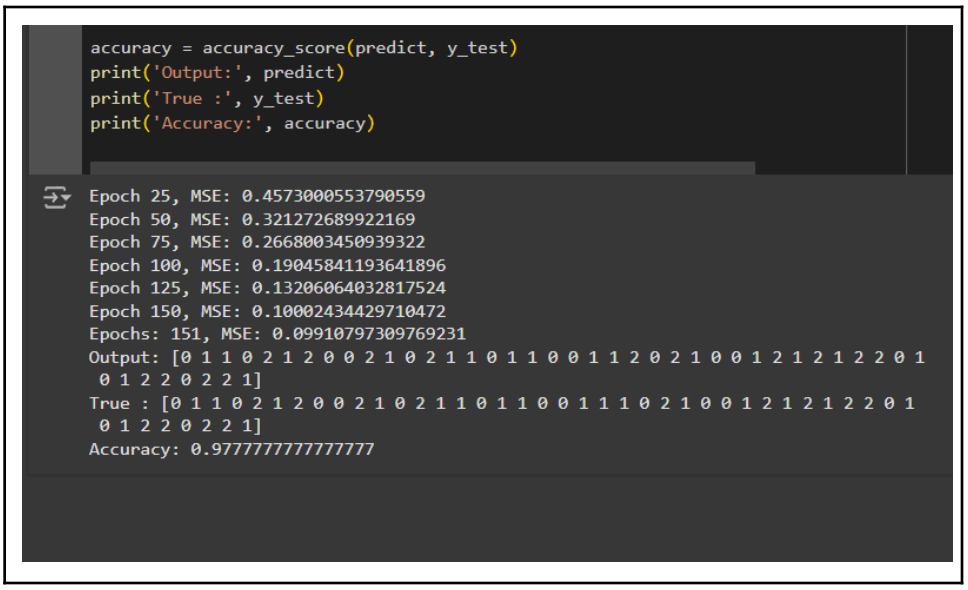
Tulis kode ke dalam *cell* di bawah ini:

```python
def onehot_enc(lbl, min_val=0):
    mi = min(lbl)
    enc = np.full((len(lbl), max(lbl) - mi + 1), min_val, np.int8)
    for i, x in enumerate(lbl):
        enc[i, x - mi] = 1
    return enc

def onehot_dec(enc, mi=0):
    return [np.argmax(e) + mi for e in enc]


def sig(X):
    return [1 / (1 + np.exp(-x)) for x in X]

def sigd(X):
    output = []
    for i, x in enumerate(X):
        s = sig([x])[0]
        output.append(s * (1 - s))
    return output
```

225150207111040
Raikhan Geza Alburama

```python
[13] import numpy as np
     from sklearn import datasets
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import minmax_scale
     from sklearn.metrics import accuracy_score

     iris = datasets.load_iris()
     X = minmax_scale(iris.data)

     def onehot_enc(labels, min_val=0):
         mi = min(labels)
         enc = np.full((len(labels), max(labels) - mi + 1), min_val, dtype=np.int8)
         for i, x in enumerate(labels):
             enc[i, x - mi] = 1
         return enc

     def onehot_dec(encoded_labels):
         return np.argmax(encoded_labels, axis=1)

     Y = onehot_enc(iris.target)
     X_train, X_test, y_train, y_test = train_test_split(X                    tate=1)

     w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 3, 3), learn_rate=0.1, max_epoch=1000
     print(f'Epochs: {ep}, MSE: {mse}')

     predict = bp_predict(X_test, w)
     predict = onehot_dec(predict)
     y_test = onehot_dec(y_test)
```

225150207111040
Raikhan Geza Alburama

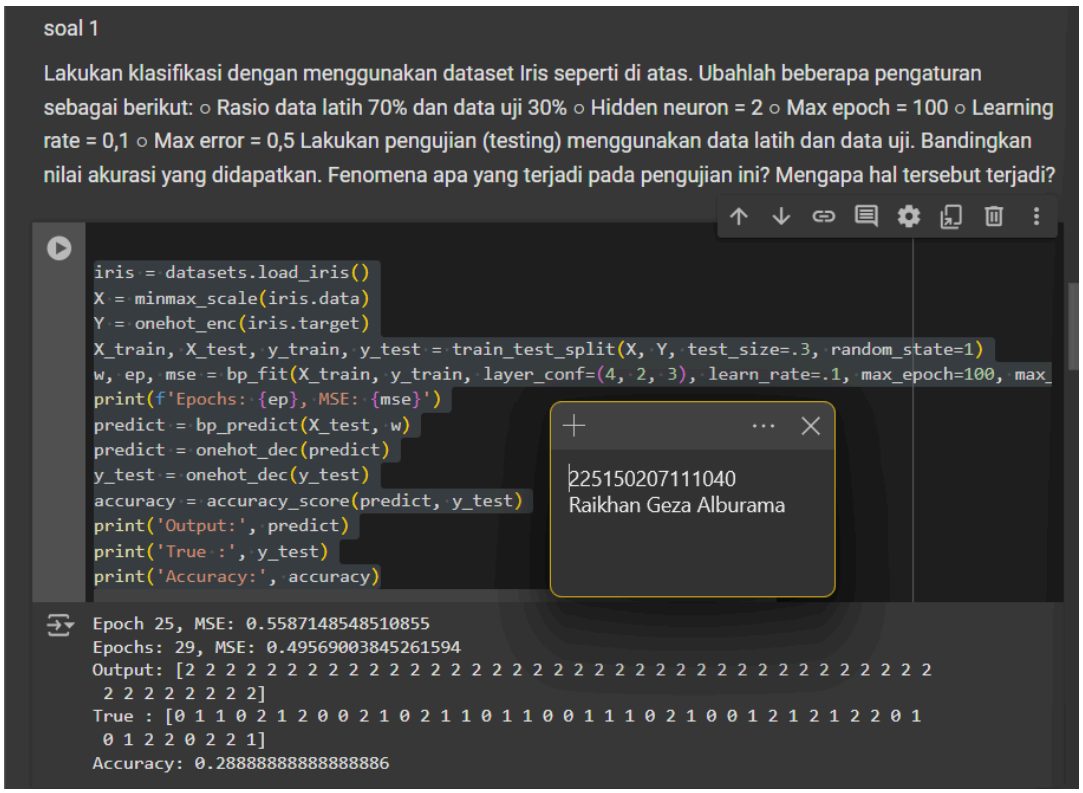


C. Soal Soal

1.Lakukan klasifikasi dengan menggunakan dataset Iris seperti di atas. Ubahlah beberapa pengaturan sebagai berikut: ○ Rasio data latih 70% dan data uji 30% ○ Hidden neuron = 2 ○ Max epoch = 100 ○ Learning rate = 0,1 ○ Max error = 0,5 Lakukan pengujian (testing) menggunakan data latih dan data uji. Bandingkan nilai akurasi yang didapatkan. Fenomena apa yang terjadi pada pengujian ini? Mengapa hal tersebut terjadi

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X,
Y, test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 2,
3), learn_rate=.1, max_epoch=100, max_error=.5,
print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
```

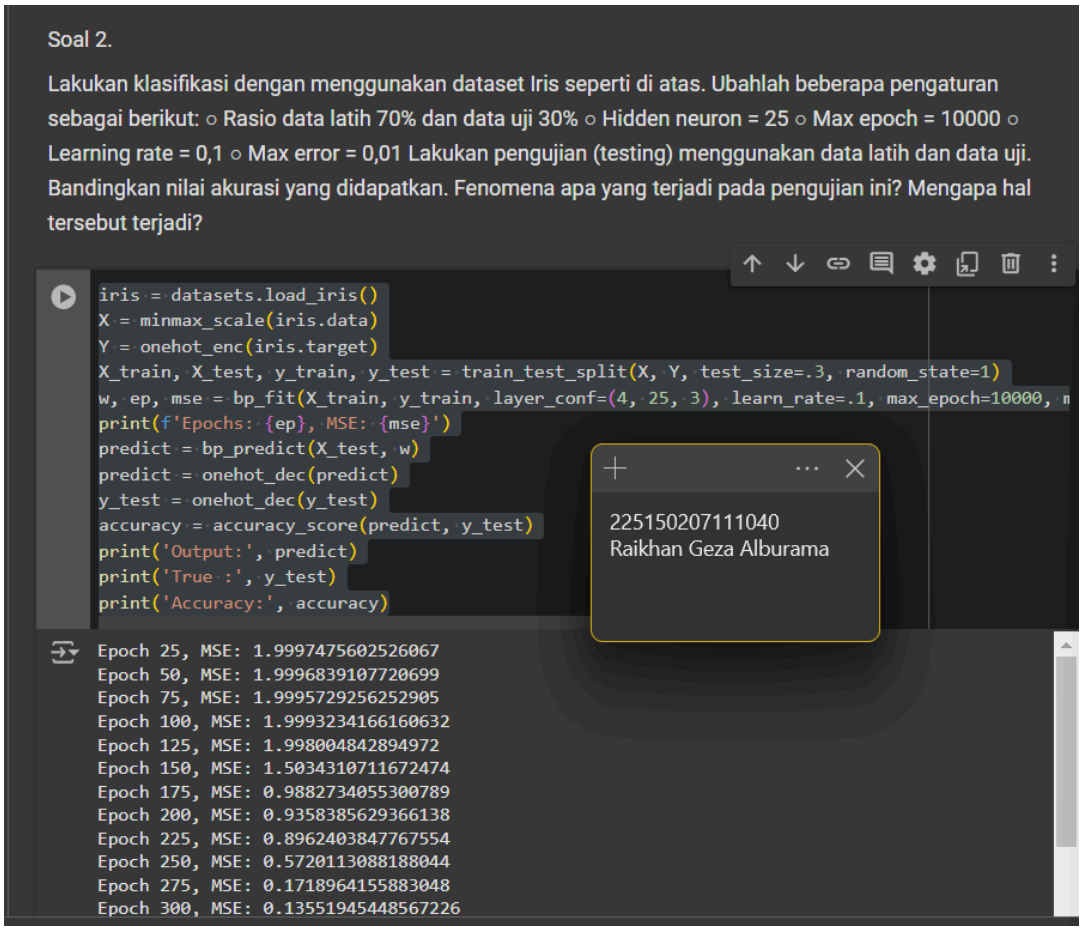
```
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```



2. Lakukan klasifikasi dengan menggunakan dataset Iris seperti di atas. Ubahlah beberapa pengaturan sebagai berikut: ○ Rasio data latih 70% dan data uji 30% ○ Hidden neuron = 25 ○ Max epoch = 10000 ○ Learning rate = 0,1 ○ Max error = 0,01 Lakukan pengujian (testing) menggunakan data latih dan data uji. Bandingkan nilai akurasi yang didapatkan. Fenomena apa yang terjadi pada pengujian ini? Mengapa hal tersebut terjadi?

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X,
```

```
Y, test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4,
25, 3), learn_rate=.1, max_epoch=10000, max_error=.01,
print_per_epoch=25)
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```



3. Buatlah fungsi bernama bin_enc_str yang berfungsi untuk melakukan binary encoding pada string. Fungsi ini menerima input berupa list of string dan

menghasilkan output berupa representasi binary encoding dari list tersebut. Jangan lupa membuat fungsi decodernya juga dengan nama bin_dec_str

```
import pandas as pd

def bin_enc_str(kelas):
    lbl_num = pd.factorize(kelas)[0]
    lbl_rev = pd.factorize(kelas)[1]
    mi = min(lbl_num)
    length = len(bin(max(lbl_num) - mi + 1)[2:])
    enc = []
    for i in lbl_num:
        b = bin(i - mi)[2:].zfill(length)
        enc.append([int(n) for n in b])
    return enc, lbl_rev

def bin_dec_str(enc, lbl_rev, mi=0):
    lbl = []
    lbl_str = []
    for en in enc:
        rounded = [int(round(x)) for x in en]
        string = ''.join(str(x) for x in rounded)
        num = int(string, 2) + mi
        lbl.append(num)
    for xx in range(len(lbl)):

lbl_str.append(pd.factorize(lbl_rev)[1][lbl[xx]])
    return lbl_str


kelas = ['A', 'B', 'A', 'C']
enc, lbl_rev = bin_enc_str(kelas)
print('Encoding (Binary):', enc)

dec = bin_dec_str(enc, lbl_rev)
print('Decoding (Binary):', dec)
```

SOAL 3

Ubahlah parameter berikut agar mendapatkan akurasi tertinggi saat melakukan testing menggunakan uji :

Hidden neuron Max epoch Max error Berapakah nilai akurasi tertinggi yang dapat Anda peroleh? Berap nilai masing-masing parameter tersebut?**bold text**

```
iris = datasets.load_iris()
X = minmax_scale(iris.data)
Y = onehot_enc(iris.target)
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=.3, random_state=1)
w, ep, mse = bp_fit(X_train, y_train, layer_conf=(4, 10, 3), learn_rate=.1, max_epoch=500
print(f'Epochs: {ep}, MSE: {mse}')
predict = bp_predict(X_test, w)
predict = onehot_dec(predict)
y_test = onehot_dec(y_test)
accuracy = accuracy_score(predict, y_test)
print('Output:', predict)
print('True :', y_test)
print('Accuracy:', accuracy)
```

```
Epoch 25, MSE: 0.383776096514961
Epoch 50, MSE: 0.2938290580342846
Epoch 75, MSE: 0.23051680773921684
Epoch 100, MSE: 0.1641746565691314
Epoch 125, MSE: 0.12054245285529827
Epoch 150, MSE: 0.09528092568898881
Epoch 175, MSE: 0.08037329840572681
Epoch 200, MSE: 0.07110175933017954
Epoch 225, MSE: 0.06496455009618983
Epoch 250, MSE: 0.060650291850340744
Epoch 275, MSE: 0.057455956978168486
Epoch 300, MSE: 0.05498915006560867
Epoch 325, MSE: 0.05302027463743097
```

225150207111040
Raikhan Geza Alburama

```
Epoch 4700, MSE: 0.03838782508066817
Epoch 4725, MSE: 0.03837776371719042
Epoch 4750, MSE: 0.038367656439496044
Epoch 4775, MSE: 0.03835750412327031
Epoch 4800, MSE: 0.038347307659300306
Epoch 4825, MSE: 0.03833706794682804
Epoch 4850, MSE: 0.03832678588720755
Epoch 4875, MSE: 0.03831646237790476
Epoch 4900, MSE: 0.03830609830687769
Epoch 4925, MSE: 0.03829569454736284
Epoch 4950, MSE: 0.0382852519530897
Epoch 4975, MSE: 0.038274771353936386
Epoch 5000, MSE: 0.03826425355203584
Epochs: 5000, MSE: 0.03826425355203584
Output: [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 2 2 1]
True : [0 1 1 0 2 1 2 0 0 2 1 0 2 1 1 0 1 1 0 0 1 1 1 0 2 1 0 0 1 2 1 2 1 2 2 0 1
 0 1 2 2 0 2 2 1]
Accuracy: 1.0
```

225150207111040
Raikhan Geza Alburama

D. Kesimpulan

Overfitting adalah kondisi di mana model machine learning belajar terlalu detail dari data latih, sehingga performanya sangat baik pada data latih namun buruk pada data baru. Hal ini terjadi karena model "menghafal" data latih alih-alih belajar pola yang dapat digeneralisasi. Untuk mengatasi overfitting, kita bisa menggunakan teknik seperti regularisasi, menambah data latih, atau menggunakan model yang lebih sederhana.

Underfitting terjadi ketika model tidak mampu menangkap pola yang cukup dari data, sehingga performanya rendah baik pada data latih maupun data baru. Penyebabnya biasanya adalah model yang terlalu sederhana atau data yang kurang memadai. Untuk mengatasi underfitting, kita dapat meningkatkan kompleksitas model, menambah fitur yang relevan, atau memperpanjang waktu pelatihan.