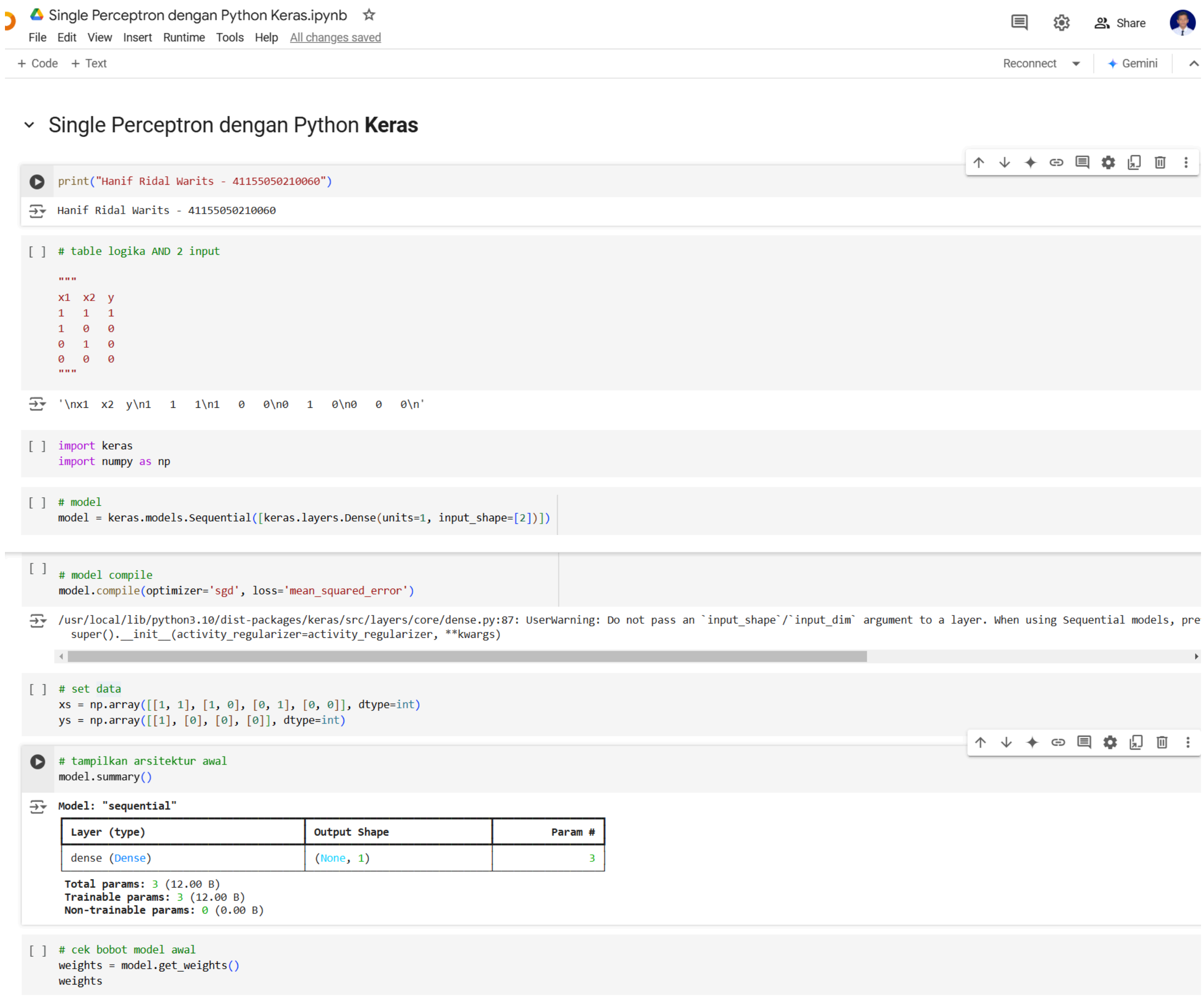
MACHINE LEARNING

Nama : Hanif Ridal Warits

NPM : 41155050210060

Kelas : Informatika A2 - 2021

Tugas Pertemuan 11



Single Perceptron dengan Python Keras.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
Reconnect
Gemini

Single Perceptron dengan Python **Keras**

```
print("Hanif Ridal Warits - 41155050210060")
```

```
Hanif Ridal Warits - 41155050210060
```

```
# table logika AND 2 input

"""
x1  x2  y
1   1   1
1   0   0
0   1   0
0   0   0
"""
```

```
'\nx1  x2  y\n1   1   1\n1   0   0\n0   1   0\n0   0   0\n'
```

```
import keras
import numpy as np
```

```
# model
model = keras.models.Sequential([keras.layers.Dense(units=1, input_shape=[2])])
```

```
# model compile
model.compile(optimizer='sgd', loss='mean_squared_error')
```

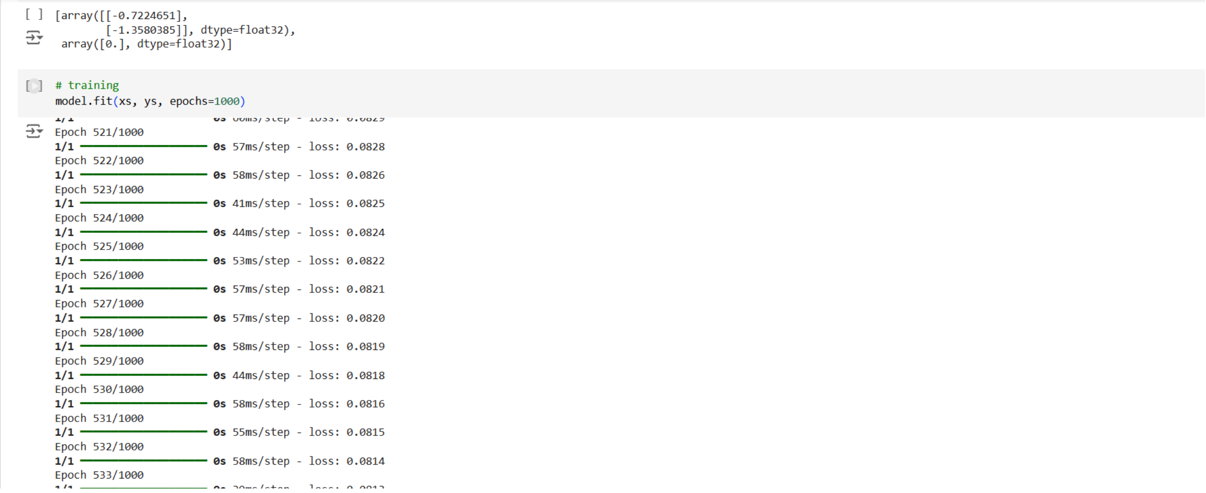
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, pre
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# set data
xs = np.array([[1, 1], [1, 0], [0, 1], [0, 0]], dtype=int)
ys = np.array([[1], [0], [0], [0]], dtype=int)
```

```
# tampilkan arsitektur awal
model.summary()
```

```
Model: "sequential"
┌──────────────────────────────┬──────────────────────┬──────────┐
│ Layer (type)                 │ Output Shape         │  Param # │
├──────────────────────────────┼──────────────────────┼──────────┤
│ dense (Dense)                │ (None, 1)            │        3 │
└──────────────────────────────┴──────────────────────┴──────────┘
 Total params: 3 (12.00 B)
 Trainable params: 3 (12.00 B)
 Non-trainable params: 0 (0.00 B)
```

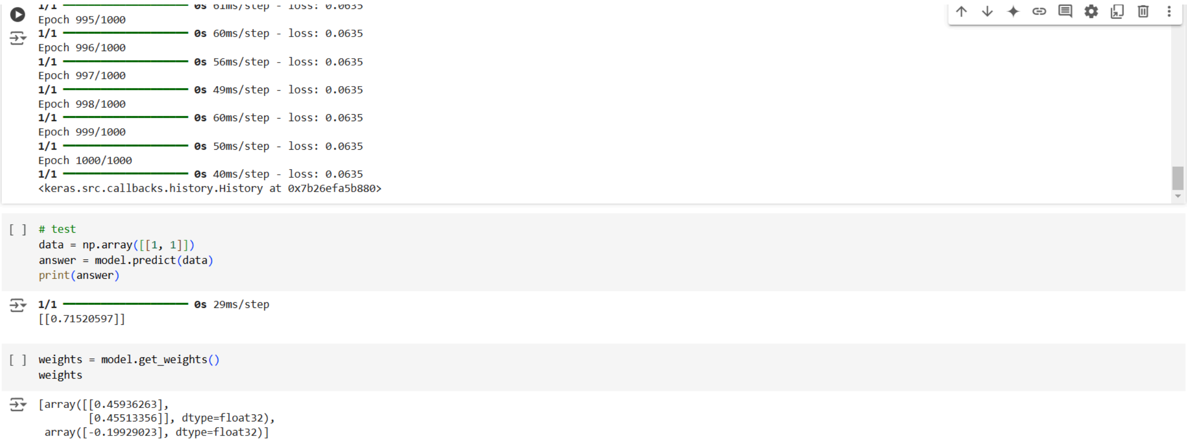
```
# cek bobot model awal
weights = model.get_weights()
weights
```



```
[array([[-0.7224651],
        [-1.3580385]], dtype=float32),
 array([0.], dtype=float32)]
```

```
# training
model.fit(xs, ys, epochs=1000)
```

```
1/1                          0s 60ms/step - loss: 0.0829
Epoch 521/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 0.0828
Epoch 522/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.0826
Epoch 523/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.0825
Epoch 524/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 0.0824
Epoch 525/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - loss: 0.0822
Epoch 526/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 0.0821
Epoch 527/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 0.0820
Epoch 528/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.0819
Epoch 529/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - loss: 0.0818
Epoch 530/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.0816
Epoch 531/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 0.0815
Epoch 532/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.0814
Epoch 533/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step - loss: 0.0813
```



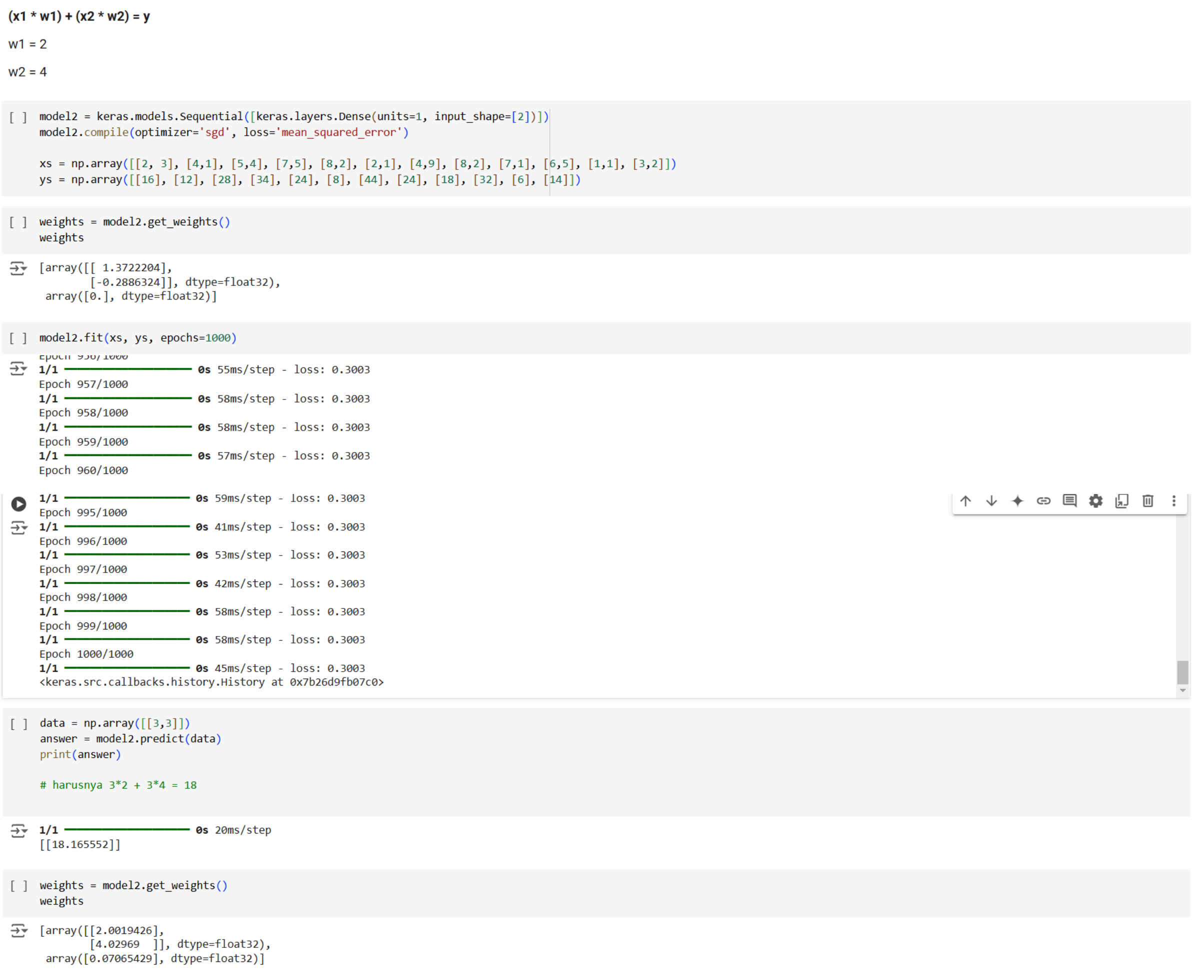
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 0.0635
Epoch 995/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 0.0635
Epoch 996/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 56ms/step - loss: 0.0635
Epoch 997/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - loss: 0.0635
Epoch 998/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 0.0635
Epoch 999/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step - loss: 0.0635
Epoch 1000/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 40ms/step - loss: 0.0635
<keras.src.callbacks.history.History at 0x7b26efa5b880>
```

```
# test
data = np.array([[1, 1]])
answer = model.predict(data)
print(answer)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
[[0.71520597]]
```

```
weights = model.get_weights()
weights
```

```
[array([[0.45936263],
        [0.45513356]], dtype=float32),
 array([-0.19929023], dtype=float32)]
```



(x1 * w1) + (x2 * w2) = y

w1 = 2

w2 = 4

```
[ ] model2 = keras.models.Sequential([keras.layers.Dense(units=1, input_shape=[2])])
    model2.compile(optimizer='sgd', loss='mean_squared_error')

    xs = np.array([[2, 3], [4,1], [5,4], [7,5], [8,2], [2,1], [4,9], [8,2], [7,1], [6,5], [1,1], [3,2]])
    ys = np.array([[16], [12], [28], [34], [24], [8], [44], [24], [18], [32], [6], [14]])
```

```
[ ] weights = model2.get_weights()
    weights
```

```
[array([[ 1.3722204],
        [-0.2886324]], dtype=float32),
 array([0.], dtype=float32)]
```

```
[ ] model2.fit(xs, ys, epochs=1000)
```

```
Epoch 956/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 0.3003
Epoch 957/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.3003
Epoch 958/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.3003
Epoch 959/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 0.3003
Epoch 960/1000

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - loss: 0.3003
Epoch 995/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 41ms/step - loss: 0.3003
Epoch 996/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 53ms/step - loss: 0.3003
Epoch 997/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step - loss: 0.3003
Epoch 998/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.3003
Epoch 999/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - loss: 0.3003
Epoch 1000/1000
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 0.3003
<keras.src.callbacks.history.History at 0x7b26d9fb07c0>
```

```
[ ] data = np.array([[3,3]])
    answer = model2.predict(data)
    print(answer)

    # harusnya 3*2 + 3*4 = 18
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
[[18.165552]]
```

```
[ ] weights = model2.get_weights()
    weights
```

```
[array([[2.0019426],
        [4.02969  ]], dtype=float32),
 array([0.07065429], dtype=float32)]
```