LAPORAN PRAKTIKUM
WORKSHOP KECERDASAN KOMPUTASIONAL
"Classification"

Oleh :

Muhammad Rifqi Aminuddin
NRP. 3123640039

PROGRAM STUDI STrLJ TEKNIK INFORMATIKA
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

1. dataset ← ambil train_data & train_label, tampilkan

   A. Kode

```
# Import library yang dibutuhkan
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
import pandas as pd
import numpy as np

# Baca dataset milk.csv
dataset = pd.read_csv('milk.csv')

# Ambil dan pisahkan train_data dengan train_label
train_data = np.array(dataset)[:,0:-1]
train_label = np.array(dataset)[:,-1]

print(" Data training are: \n", train_data)
print("\n Training label are: \n", train_label)
```

   B. Keluaran

```
Data training are:
[[6.6 35 1 ... 1 0 254]
[6.6 36 0 ... 0 1 253]
[8.5 70 1 ... 1 1 246]
...
[3.0 40 1 ... 1 1 255]
[6.8 43 1 ... 1 0 250]
[8.6 55 0 ... 1 1 255]]

Training label are:
['high' 'high' 'low' ... 'low' 'high' 'low']
```

   C. Analisa

   Kode di atas digunakan untuk melakukan pemanggilan library yang dibutuhkan, serta terdapat kode yang dibutuhkan untuk melakukan pemanggilan dataset dari milk.csv. Dari variabel data milk tersebut dibagi menjadi 2 yaitu training data, dan training label. Training data ialah untuk urutan kolom pertama hingga kolom ke-tujuh, sedangkan untuk training label merupakan kolom paling terakhir yang berisikan label high, low, dan medium. Pemisahan ini nantinya untuk digunakan dalam tahapan normalisasi.

2. Lakukan normalisasi terhadap train_data dengan metode min-max(0-1)

   A. Kode

```
# Normalisasi train_data dengan min-max(0-1)
sc = MinMaxScaler(feature_range=(0,1))
dataNormal = sc.fit_transform(train_data)
print(" Hasil normalisasi data: \n", dataNormal)
```

   B. Keluaran

```
Hasil normalisasi data:
[[0.55384615 0.01785714 1.         ... 1.         0.         0.93333333]
 [0.55384615 0.03571429 0.         ... 0.         1.         0.86666667]
 [0.84615385 0.64285714 1.         ... 1.         1.         0.4       ]
 ...
 [0.         0.10714286 1.         ... 1.         1.         1.        ]
 [0.58461538 0.16071429 1.         ... 1.         0.         0.66666667]
 [0.86153846 0.375      0.         ... 1.         1.         1.        ]]
```

   C. Analisa

   Pada kode di atas dilakukan tahapan normalisasi, yaitu tahapan agar masing masing input atau data hanya terdiri dari data dengan rentang 0 hingga 1. Data ini dibilang cukup efisien dari data semula 10, 50, hingga 255, pada kali ini digunakan min-max (0-1).

3. Lakukan normalisasi & klasifikasi menggunakan k-NN untuk 1 input data test

   A. Kode

```
# Mengambil 1 baris data milk_testing untuk prediksi hasil testing
test_data = np.array(pd.read_csv('milk_testing.csv'))[:,0:-1]

# Melakukan testing data prediksi
kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')
kNN.fit(train_data,train_label)

dataNormalTest = np.array(sc.fit_transform(test_data))[0,:]

print(dataNormalTest)
hasil = kNN.predict(dataNormalTest.reshape(1, -1))
print("Hasil dari k-NN : ",hasil)
```

   B. Keluaran

```
[0.58461538 0.19642857 1.         1.         1.         0.
 0.33333333]
Hasil dari k-NN :  ['medium']
```

   C. Analisa

Kode di atas merupakan kode yang digunakan untuk melakukan uji coba prediksi dengan metode k-NN (Nearest Neighbour). Metode ini menghitung dari tetangga yang paling dekat. Data yang digunakan dalam test kali ini menggunakan data dari dataset milk_testing.csv, yang mana hanya diambil baris pertamanya saja. Dari hasil di atas diperoleh prediksi k-NN atau klasifikasi k-NN dengan label Medium.

4. dataset ← milk_training.csv, ambil train_data & train_label, tampilkan

A. Kode

```
# Baca dataset dari milk_training.csv
dataset = pd.read_csv('milk_training.csv')

# Pisahkan antara train_data dengan train_label
train_data = np.array(dataset)[:,0:-1]
train_label = np.array(dataset)[:,-1]

print(" Data training are: \n", train_data)
print("\n Training label are: \n", train_label)
```

B. Keluaran

```
Data training are:
[[6.6 35 1 ... 1 0 254]
[6.6 36 0 ... 0 1 253]
[6.6 37 1 ... 1 1 255]
...
[6.7 41 1 ... 0 0 247]
[6.8 41 0 ... 0 0 255]
[6.8 38 0 ... 0 0 255]]

Training label are:
['high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
```

```
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium']
```

C. Analisa

Kode di atas digunakan untuk membaca data dari dataset milk_training.csv untuk kemudian dipanggil kedalam dataset. Data dari variabel dataset tersebut dibagi menjadi 2 yaitu training data, dan training label. Training data ialah untuk urutan kolom pertama hingga kolom ke-tujuh, sedangkan untuk training label merupakan kolom paling terakhir yang berisikan label high, low, dan medium. Pemisahan ini nantinya untuk digunakan dalam tahapan normalisasi terhadap train_data.

5. Lakukan normalisasi terhadap train_data dengan metode min-max(0-1)

A. Kode

```
# Normalisasi terhadap train_data
dataNormalTrain = sc.fit_transform(train_data)
print(" Hasil normalisasi data: \n", dataNormalTrain)
```

B. Keluaran

```
Hasil normalisasi data:
[[0.55384615 0.01785714 1.         ... 1.         0.         0.93333333]
[0.55384615 0.03571429 0.         ... 0.         1.         0.86666667]
[0.55384615 0.05357143 1.         ... 1.         1.         1.        ]
...
[0.56923077 0.125      1.         ... 0.         0.         0.46666667]
[0.58461538 0.125      0.         ... 0.         0.         1.        ]
[0.58461538 0.07142857 0.         ... 0.         0.         1.        ]]
```

C. Analisa

Kode di atas digunakan untuk melakukan normalisasi data dari train_data yang telah dipisah sebelumnya dari dataset milk_training.csv, tentunya agar masing-masing data hanya terdiri dari data dengan rentang 0 hingga 1. Data ini dibilang cukup efisien dari data semula 10, 50, hingga 255, pada kali ini digunakan min-max (0-1).

6. test_dataset ← milk_testing.csv, ambil test_data & test_label, tampilkan

A. Kode

```
# Baca dataset testing dari milk_testing.csv
test_dataset = pd.read_csv('milk_testing.csv')

test_data = np.array(test_dataset)[:,0:-1]
test_label = np.array(test_dataset)[:,-1]

print(" Data testing are: \n", train_data)
print("\n Testing label are: \n", train_label)
```

B. Keluaran

```
Data testing are:
[[6.6 35 1 ... 1 0 254]
[6.6 36 0 ... 0 1 253]
[6.6 37 1 ... 1 1 255]
...
[6.7 41 1 ... 0 0 247]
[6.8 41 0 ... 0 0 255]
[6.8 38 0 ... 0 0 255]]

Testing label are:
['high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
```

```
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium']
```

C. Analisa

Kode di atas digunakan untuk membaca data dari dataset milk_testing.csv untuk kemudian dipanggil kedalam dataset. Data dari variabel dataset tersebut dibagi menjadi 2 yaitu testing_data, dan testing_label. Testing data ialah untuk urutan kolom pertama hingga kolom ke-tujuh, sedangkan untuk testing label merupakan kolom paling terakhir yang berisikan label high, low, dan medium. Pemisahan ini nantinya untuk digunakan dalam tahapan normalisasi terhadap testing_data.

7. Lakukan normalisasi terhadap test_data dengan metode min-max(0-1)

A. Kode

```
# Lakukan normalisasi terhadap test_data
sc = MinMaxScaler(feature_range=(0,1))
dataNormalTest = sc.fit_transform(test_data)
print(" Hasil normalisasi data: \n", dataNormalTest)
```

B. Keluaran

```
Hasil normalisasi data:
[[0.58461538 0.19642857 1.         ... 1.         0.         0.33333333]
 [0.55384615 0.05357143 1.         ... 1.         1.         1.        ]
 [0.56923077 0.07142857 1.         ... 1.         0.         1.        ]
 ...
 [0.53846154 0.05357143 0.         ... 0.         0.         1.        ]
 [0.53846154 0.10714286 1.         ... 0.         0.         0.66666667]
 [0.56923077 0.19642857 1.         ... 0.         0.         0.46666667]]
```

C. Analisa

Kode di atas merupakan kode yang digunakan untuk menormalisasi data, pada tahap ini dilakukan pengecilan data agar masing-masing input atau data hanya terdiri dari data dengan rentang 0 hingga 1. Data ini dibilang cukup efisien dari data semula 10, 50, hingga 255, pada kali ini digunakan min-max (0-1).

8. Lakukan klasifikasi menggunakan k-NN untuk test_data, bandingkan hasilnya dengan test_label

A. Kode

```
# Normalisasi dengan data test
kNN.fit(dataNormalTrain,train_label)
hasilNormalisasi = kNN.predict(dataNormalTest)

# Lihat akurasi
def cek(test_label, hasil):
    error = 0
    for i in range(len(test_label)):
        if test_label[i] != hasil[i]:
            error = error+1
    akurasi = (len(test_label)-error)/len(test_label)*100
    return akurasi

print("Akurasi sebelum normalisasi : ", cek(test_label, hasil), "%")
print("Akurasi setelah normalisasi : ", cek(test_label, hasilNormalisasi), "%")

print("\nTest Label :\n",test_label)
print("\nHasil k-NN Normalisasi : \n",hasilNormalisasi)
```

B. Keluaran

```
Akurasi sebelum normalisasi :  98.73817034700315 %
Akurasi setelah normalisasi :  99.36908517350159 %

Test Label :
 ['high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
```

```
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium']

Hasil k-NN Normalisasi :
 ['high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'high' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'high' 'low' 'low' 'low' 'low' 'low' 'low' 'low'
 'low' 'low' 'low' 'low' 'low' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
```

```
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium' 'medium'
'medium' 'medium' 'medium']
```

C. Analisa

Pada kode di atas merupakan kode yang digunakan untuk melakukan uji coba prediksi dengan metode k-NN (Nearest Neighbour). Metode ini menghitung dari tetangga yang paling dekat. Uji prediksi ini dengan menggunakan training_data dan training_label untuk diuji cobakan memprediksi label keluaran dari test_data. Test_data sendiri ialah data yang diambil dari dataset milk_testing yang telah dipisahkan dari labelnya. Data hasil k-NN yang disajikan diatas ada yang menggunakan data yang telah dinormalisasi atau data biasa yang tanpa proses normalisasi, sehingga dari data di atas tersebut dapat dianalisa jika normalisasi meningkatkan keakurasian dalam proses k-NN. Hal ini karena jika sebelum dilakukan normalisasi persentase keakurasian ialah sebesar 98,73817034700315%, sedangkan jika dengan normalisasi persentase keakurasiannya ialah sebesar 99,36908517350159 %.

Kesimpulan

Dari praktikum di atas, dapat disimpulkan jika proses normalisasi ialah proses untuk mengefektifkan dan mengefisiensikan data dari input yang tersedia, entah dari input manual ataupun dari dataset yang tersedia. Data tersebut kemudian disederhanakan dalam rentang angka 0 hingga 1. Tentunya hal ini berdampak kepada keakurasian dari proses k-NN, peningkatan keakurasian tersebut ialah sebesar 0,63091482649844%. k-NN tersebut digunakan untuk menganalisa ataupun memprediksi hasil keluaran yang kira-kira akan diperoleh dari jika dimasukkan input atau data masukan tertentu.