

LAPORAN PRAKTIKUM KECERDASAN BUATAN
“Basis Data – K-Nearest-Neighbour”



AGUNG ALDI PRASETYA (3122552803)

PJJ D3 TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2023

Tugas dataset milk.csv

- Disediakan oleh Kaggle
- Memberikan informasi kualitas susu dengan target high(baik), medium(sedang), low(buruk).
- Terdiri dari :
 - pH : mendefinisikan PH susu yang berkisar antara 3 hingga 9,5
 - Temperature(Suhu) : mendefinisikan Suhu susu yang berkisar dari 34'C sampai 90'C
 - Taste(Rasa) : mendefinisikan Rasa susu 0 (Buruk) atau 1 (Baik)
 - Odor(Bau) : mendefinisikan Bau susu 0 (Buruk) atau 1 (Baik)
 - Fat (Lemak) : mendefinisikan kadar lemak susu 0 (Rendah) atau 1 (Tinggi)
 - Turbidity (Kekeruhan) :mendefinisikan Kekeruhan susu 0 (Rendah) atau 1 (Tinggi)
 - Colour (Warna) : Kolom ini mendefinisikan Warna susu yang berkisar dari 240 hingga 255



Tugas Klasifikasi dengan k-NN

1. dataset ← milk.csv, ambil train_data & train_label, tampilkan
2. Lakukan normalisasi terhadap train_data dengan metode min-max(0-1)
3. Lakukan klasifikasi menggunakan k-NN untuk 1 input data test (jangan lupa data test juga dinormalisasi)
 - k=1
 - k=2
 - k=3
 - k=4
 - k=5
 - k=6
 - k=7

Bandingkan hasilnya



1. Menampilkan dataset

Pertama-tama, kita perlu memuat dataset milk.csv dan membaginya menjadi train_data dan train_label. Kita dapat menggunakan library pandas untuk melakukan hal ini:

```
KNN.py X
KNN.py > ...
1  import pandas as pd
2
3  # Load dataset
4  dataset = pd.read_csv('milk.csv')
5
6  # Split dataset into train_data and train_label
7  train_data = dataset.drop('Grade', axis=1)
8  train_label = dataset['Grade']
9
10 # Show train_data and train_label
11 print(train_data)
12 print(train_label)
```

```
MINGW64:/c/Users/Administrator/Desktop/test
Administrator@WIN-MULPTT1DLDC MINGW64 ~/Desktop/test
$ python KNN.py
   pH  Temperature  Taste  Odor  Fat  Turbidity  Colour
0   6.6           35      1     0    1           0     254
1   6.6           36      0     1    0           1     253
2   8.5           70      1     1    1           1     246
3   9.5           34      1     1    0           1     255
4   6.6           37      0     0    0           0     255
...  ...         ...    ...    ...    ...         ...    ...
1054 6.7           45      1     1    0           0     247
1055 6.7           38      1     0    1           0     255
1056 3.0           40      1     1    1           1     255
1057 6.8           43      1     0    1           0     250
1058 8.6           55      0     1    1           1     255

[1059 rows x 7 columns]
0      high
1      high
2      low
3      low
4      medium
...
1054  medium
1055   high
1056   low
1057   high
1058   low
Name: Grade, Length: 1059, dtype: object
Administrator@WIN-MULPTT1DLDC MINGW64 ~/Desktop/test
$
```

Penjelasan :

Install dulu library pandas

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Administrator\Desktop\test> pip install pandas
Collecting pandas
  Downloading pandas-2.0.1-cp311-cp311-win_amd64.whl (10.6 MB)
    10.6/10.6 MB 1.8 MB/s eta 0:00:00
Collecting python-dateutil>=2.8.2
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    247.7/247.7 kB 1.9 MB/s eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2023.3-py2.py3-none-any.whl (502 kB)
    502.3/502.3 kB 1.9 MB/s eta 0:00:00
  Downloading tzdata-2023.3-py2.py3-none-any.whl (341 kB)
    341.8/341.8 kB 1.9 MB/s eta 0:00:00
Collecting numpy>=1.21.0
  Downloading numpy-1.24.3-cp311-cp311-win_amd64.whl (14.8 MB)
    14.8/14.8 MB 1.7 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: pytz, tzdata, six, numpy, python-dateutil, pandas
Successfully installed numpy-1.24.3 pandas-2.0.1 python-dateutil-2.8.2 pytz-2023.3 six-1.16.0 tzdata-202
```

2. Melakukan normalisasi menggunakan minMax()

Selanjutnya, kita perlu melakukan normalisasi terhadap train_data menggunakan metode min-max. Kita dapat menggunakan library sklearn untuk melakukan hal ini:

```
from sklearn.preprocessing import MinMaxScaler

# Normalize train_data using MinMaxScaler
scaler = MinMaxScaler()
train_data_normalized = scaler.fit_transform(train_data)

# Show normalized train_data
print(train_data_normalized)
```

```
MINGW64:/c/Users/Administrator/Desktop/test
Administrator@WIN-MULPTT1DLDC MINGW64 ~/Desktop/test
$ python KNN.py
  pH  Temperature  Taste  Odor  Fat  Turbidity  Colour
0   6.6         35     1    0    1         0     254
1   6.6         36     0    1    0         1     253
2   8.5         70     1    1    1         1     246
3   9.5         34     1    1    0         1     255
4   6.6         37     0    0    0         0     255
...  ...         ...    ...    ...    ...     ...
1054 6.7         45     1    1    0         0     247
1055 6.7         38     1    0    1         0     255
1056 3.0         40     1    1    1         1     255
1057 6.8         43     1    0    1         0     250
1058 8.6         55     0    1    1         1     255

[1059 rows x 7 columns]
0      high
1      high
2      low
3      low
4      medium
...
1054    medium
1055    high
1056    low
1057    high
1058    low
Name: Grade, Length: 1059, dtype: object
[[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.         ]]

Administrator@WIN-MULPTT1DLDC MINGW64 ~/Desktop/test
$
```

3. Setelah melakukan normalisasi, kita dapat melakukan klasifikasi menggunakan k-NN dengan memilih nilai k yang berbeda-beda. Kita juga perlu melakukan normalisasi terhadap data test sebelum melakukan klasifikasi. Berikut adalah contoh kode untuk melakukan klasifikasi dengan k-NN dan menampilkan hasilnya:

```

from sklearn.neighbors import KNeighborsClassifier

# Define input data test
test_data = [[7, 50, 1, 1, 1, 0, 245]]

# Normalize test_data using MinMaxScaler
test_data_normalized = scaler.transform(test_data)

# Define k values
k_values = [1, 2, 3, 4, 5, 6, 7]

# Perform k-NN classification for each k value
for k in k_values:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(train_data_normalized, train_label)
    prediction = knn.predict(test_data_normalized)
    print('k =', k, '->', prediction)

```

```

MINGW64/c/Users/Administrator/Desktop/test

Administrator@WIN-MULPTT1DLDC MINGW64 ~/Desktop/test
$ python KNN.py
E:\python\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but MinMaxScaler was fitted with feature names
  warnings.warn(
   pH  Temperature  Taste  Odor  Fat  Turbidity  Colour
0    6.6          35      1    0    1           0    254
1    6.6          36      0    1    0           1    253
2    8.5          70      1    1    1           1    246
3    9.5          34      1    1    0           1    255
4    6.6          37      0    0    0           0    255
...
1054  6.7          45      1    1    0           0    247
1055  6.7          38      1    0    1           0    255
1056  3.0          40      1    1    1           1    255
1057  6.8          43      1    0    1           0    250
1058  8.6          55      0    1    1           1    255

[1059 rows x 7 columns]
0      high
1      high
2      low
3      low
4      medium
...
1054     medium
1055      high
1056      low
1057      high
1058      low
Name: Grade, Length: 1059, dtype: object
[[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.       ]]
k = 1 -> ['low']
k = 2 -> ['low']
k = 3 -> ['low']
k = 4 -> ['low']
k = 5 -> ['low']
k = 6 -> ['low']
k = 7 -> ['low']

Administrator@WIN-MULPTT1DLDC MINGW64 ~/Desktop/test
$ |

```

Penjelasan :

Install modul scikit-learn

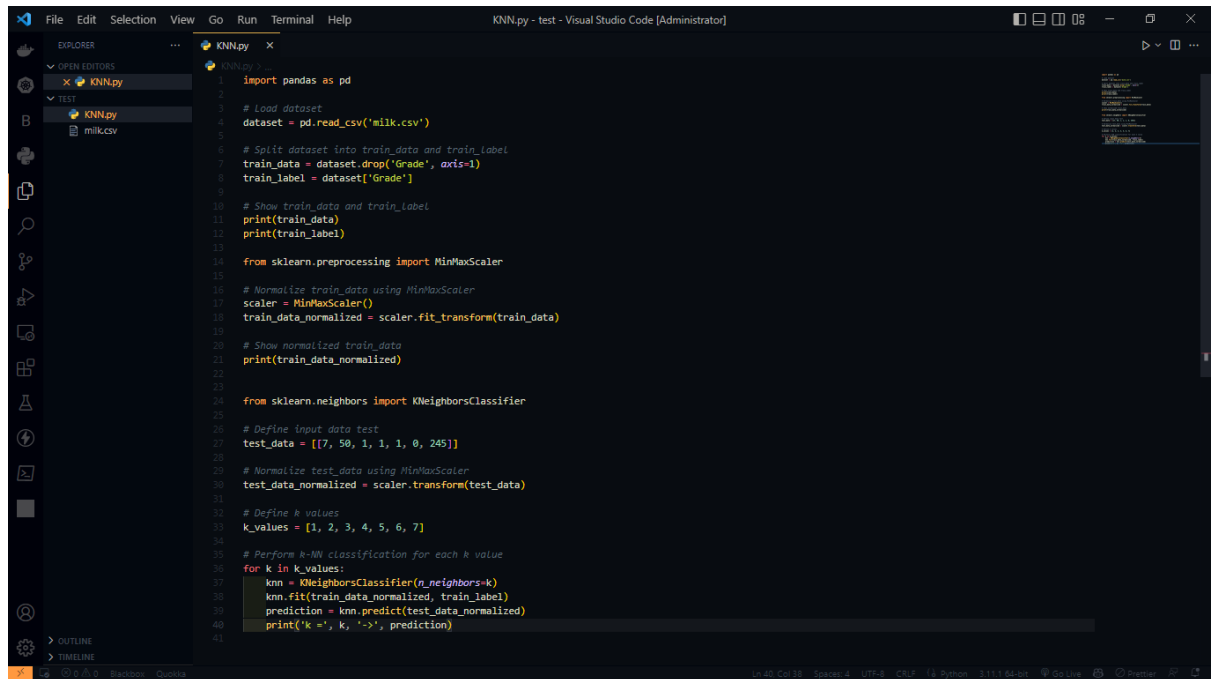
```
PS C:\Users\Administrator\Desktop\test> pip install scikit-learn
Collecting scikit-learn
  Collecting scikit-learn
    Downloading scikit_learn-1.2.2-cp311-cp311-win_amd64.whl (8.3 MB)
      8.3/8.3 MB 2.1 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17.3 in e:\python\lib\site-packages (from scikit-learn) (1.24.3)
Collecting scipy>=1.3.2 (from scikit-learn)
  Downloading scipy-1.10.1-cp311-cp311-win_amd64.whl (42.2 MB)
      42.2/42.2 MB 2.0 MB/s eta 0:00:00
Collecting joblib>=1.1.1 (from scikit-learn)
  Downloading joblib-1.2.0-py3-none-any.whl (297 kB)
      298.0/298.0 kB 2.0 MB/s eta 0:00:00
Collecting threadpoolctl>=2.0.0 (from scikit-learn)
  Downloading threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.2.0 scikit-learn-1.2.2 scipy-1.10.1 threadpoolctl-3.1.0
PS C:\Users\Administrator\Desktop\test>
```

Dalam contoh di atas, kita menggunakan input data test dengan nilai pH 7, suhu 50°C, rasa baik, bau baik, lemak tinggi, kekeruhan rendah, dan warna 245.

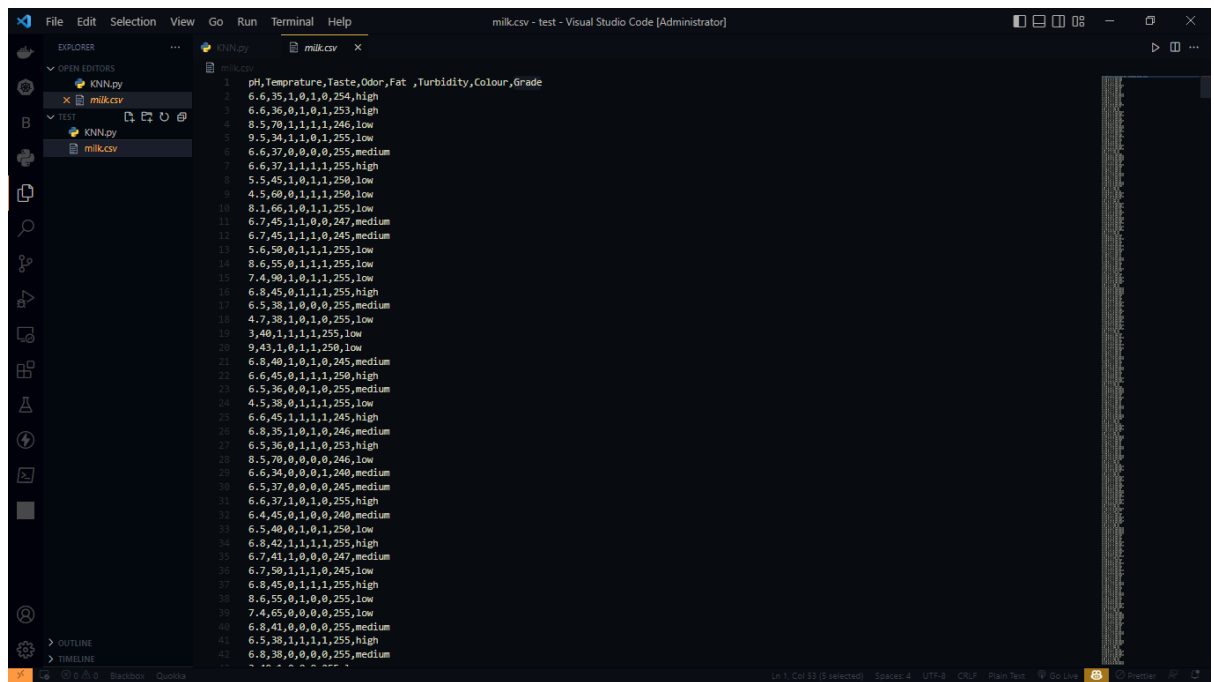
Selanjutnya, kita melakukan normalisasi terhadap data test menggunakan MinMaxScaler, lalu melakukan klasifikasi dengan k-NN untuk nilai k yang berbeda-beda (k = 1 hingga 7). Hasil klasifikasi akan ditampilkan pada layar.

Demikianlah contoh kode untuk melakukan klasifikasi dengan k-NN pada dataset milk.csv dan membandingkan hasilnya untuk nilai k yang berbeda-beda.

Full code :



```
1 import pandas as pd
2
3 # Load dataset
4 dataset = pd.read_csv('milk.csv')
5
6 # Split dataset into train_data and train_label
7 train_data = dataset.drop('Grade', axis=1)
8 train_label = dataset['Grade']
9
10 # Show train_data and train_label
11 print(train_data)
12 print(train_label)
13
14 from sklearn.preprocessing import MinMaxScaler
15
16 # Normalize train_data using MinMaxScaler
17 scaler = MinMaxScaler()
18 train_data_normalized = scaler.fit_transform(train_data)
19
20 # Show normalized train_data
21 print(train_data_normalized)
22
23 from sklearn.neighbors import KNeighborsClassifier
24
25 # Define input data test
26 test_data = [[7, 50, 1, 1, 1, 0, 245]]
27
28 # Normalize test_data using MinMaxScaler
29 test_data_normalized = scaler.transform(test_data)
30
31 # Define k values
32 k_values = [1, 2, 3, 4, 5, 6, 7]
33
34 # Perform k-NN classification for each k value
35 for k in k_values:
36     knn = KNeighborsClassifier(n_neighbors=k)
37     knn.fit(train_data_normalized, train_label)
38     prediction = knn.predict(test_data_normalized)
39     print('k =', k, '->', prediction)
```



```
1 pH,Temperature,Taste,Odor,Fat ,Turbidity,Colour,Grade
2 6.6,35,1,0,1,0,254,high
3 6.6,36,0,1,0,1,253,high
4 8.5,70,1,1,1,1,246,low
5 9.5,34,1,1,0,1,255,low
6 6.6,37,0,0,0,0,255,medium
7 6.6,37,1,1,1,1,255,high
8 5.5,45,1,0,1,1,250,low
9 4.5,60,0,1,1,1,250,low
10 8.1,66,1,0,1,1,255,low
11 6.7,45,1,1,0,0,247,medium
12 6.7,45,1,1,1,0,245,medium
13 5.6,50,0,1,1,1,255,low
14 8.6,55,0,1,1,1,255,low
15 7.4,50,1,0,1,1,255,low
16 6.8,45,0,1,1,1,255,high
17 6.5,38,1,0,0,0,255,medium
18 4.7,38,1,0,1,0,255,low
19 3.40,1,1,1,1,255,low
20 9.43,1,0,1,1,250,low
21 6.8,40,1,0,1,0,245,medium
22 6.6,45,0,1,1,1,250,high
23 6.5,36,0,0,1,0,255,medium
24 4.5,38,0,1,1,1,255,low
25 6.6,45,1,1,1,1,245,high
26 6.8,35,1,0,1,0,246,medium
27 6.5,36,0,1,1,0,253,high
28 8.5,70,0,0,0,0,246,low
29 6.6,34,0,0,0,1,240,medium
30 6.5,37,0,0,0,0,245,medium
31 6.6,37,1,0,1,0,255,high
32 6.4,45,0,1,0,0,240,medium
33 6.5,40,0,1,0,1,250,low
34 6.8,42,1,1,1,1,255,high
35 6.7,41,1,0,0,0,247,medium
36 6.7,50,1,1,1,0,245,low
37 6.8,45,0,1,1,1,255,high
38 8.6,55,0,1,0,0,255,low
39 7.4,65,0,0,0,0,255,low
40 6.8,41,0,0,0,0,255,medium
41 6.5,38,1,1,1,1,255,high
42 6.8,38,0,0,0,0,255,medium
```