LAPORAN PRAKTIKUM
WORKSHOP KECERDASAN KOMPUTASIONAL
"Validation"

Oleh :

Muhammad Rifqi Aminuddin
NRP. 3123640039

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

1. dataset ← milk.csv

   - 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

dataset = pd.read_csv('../Dataset/milk.csv')
print(dataset)
```

   - Keluaran

```
        pH  Temprature  Taste  Odor  Fat   Turbidity  Colour   Grade
0      6.6          35      1     0     1          0     254    high
1      6.6          36      0     1     0          1     253    high
2      8.5          70      1     1     1          1     246     low
3      9.5          34      1     1     0          1     255     low
4      6.6          37      0     0     0          0     255  medium
...    ...         ...    ...   ...   ...        ...     ...     ...
1054   6.7          45      1     1     0          0     247  medium
1055   6.7          38      1     0     1          0     255    high
1056   3.0          40      1     1     1          1     255     low
1057   6.8          43      1     0     1          0     250    high
1058   8.6          55      0     1     1          1     255     low

[1059 rows x 8 columns]
```

   - Analisa

     Kode di atas digunakan untuk menampilkan dataset dari file dataset berupa milk.csv, yang mana berisi data data susu mulai dari pH, suhu, rasa, indeks lemak, dll.

2. Lakukan validation Model dengan metode:

   a. Hold-out Method (70%-30%)

   - Kode

```
# Validation Model Metode Hold-out
from sklearn.model_selection import train_test_split
datalabel = dataset.loc[:,['Grade']]
xtrainHold, xtestHold, ytrainHold, ytestHold = train_test_split(dataset.loc[:,
                                                dataset.columns != 'Grade'],
                                                datalabel, test_size = 0.30,
                                                random_state=100)
```

- Keluaran

```
Ytrain
       Grade
255  medium
662     low
899  medium
380    high
954    high
..      ...
802    high
53      low
350  medium
79   medium
792     low

[741 rows x 1 columns]
```

- Analisa

  Kode di atas merupakan kode untuk melakukan validasi dengan metode Hold-out dengan (70%-30%). Metode ini diperlukan untuk melakukan dan memisahkan antara data_test dengan data_train secara otomatis berdasar algoritma masing masing metode.

b. K-Fold (k=10)

- Kode

```
# Validation Model Metode k-Fold
from sklearn.model_selection import KFold

kf=KFold(n_splits=10, random_state=0, shuffle=True)
p=0
for train_index, test_index in kf.split(dataset):
    p=p+1
    xtrain=dataset.loc[train_index]
    xtest=dataset.loc[test_index]
    xtrainFold=xtrain.loc[:, dataset.columns != 'Grade']
    xtestFold=xtest.loc[:, dataset.columns != 'Grade']
    ytrainFold=xtrain.loc[:,['Grade']]
    ytestFold=xtest.loc[:,['Grade']]
```

- Keluaran

```
Ytrain
        Grade
0        high
1        high
2         low
3         low
4      medium
...       ...
1054   medium
1055     high
1056      low
1057     high
1058      low

[954 rows x 1 columns]
```

- Analisa

  Kode di atas merupakan kode untuk melakukan validasi dengan metode k-Fold dengan k=10. Metode ini diperlukan untuk melakukan dan memisahkan antara data_test dengan data_train secara otomatis berdasar algoritma masing masing metode.

c. LOO

- Kode

```
# Validation Model Metode LOO
from sklearn.model_selection import LeaveOneOut

loo=LeaveOneOut()
n=loo.get_n_splits(dataset)
for xtrainLoo, xtestLoo in loo.split(dataset):
    xtrainLoo = dataset.filter(items=xtrainLoo, axis=0)
    xtestLoo = dataset.filter(items=xtestLoo, axis=0)
ytrainLoo=xtrainLoo.loc[:,['Grade']]
ytestLoo=xtestLoo.loc[:,['Grade']]

xtrainLoo = xtrainLoo.loc[:, dataset.columns != 'Grade']
xtestLoo = xtestLoo.loc[:, dataset.columns != 'Grade']
```

- Keluaran

```
Ytrain
        Grade
0        high
1        high
2         low
3         low
4      medium
...       ...
1053      low
1054   medium
1055     high
1056      low
1057     high

[1058 rows x 1 columns]
```

- Analisa

  Kode di atas merupakan kode untuk melakukan validasi dengan metode LOO, yang mana metode mengeluarkan satu data_test dan ini diperlukan untuk melakukan dan memisahkan antara data_test dengan data_train secara otomatis berdasar algoritma masing masing metode.

3. train_data ← lakukan normalisasi pada train_datadengan Min-Max 0-1 (catat nilai min dan max setiap atribut)

- Kode

```
# Normalisasi train_data test_data dengan min-max(0-1)
sc = MinMaxScaler(feature_range=(0,1))

# Metode Hold-out
datatrainnormalHold = sc.fit_transform(xtrainHold)
datatestnormalHold = sc.fit_transform(xtestHold)
print(" Hasil normalisasi test_data Hold-out: \n", datatestnormalHold)

# Metode K-Fold
datatrainnormalFold = sc.fit_transform(xtrainFold)
datatestnormalFold = sc.fit_transform(xtestFold)
print("\n Hasil normalisasi test_data K-Fold: \n",
      datatestnormalFold[:3], "\n...\n",datatestnormalFold[-3:])

# Metode LOO
datatrainnormalLoo = sc.fit_transform(xtrainLoo)
mindata = xtrainLoo.min().min()
maxdata = xtrainLoo.max().max()
datatestnormalLoo = (np.array(xtestLoo)[:,:] - mindata) / (maxdata - mindata)
print("\n Hasil normalisasi test_data LOO: \n", datatestnormalLoo)
# print('\nxtest\n', xtestFold)
# print('\nxtest\n', xtestLoo)
print('\nYtest\n', ytestLoo)
```

- Keluaran

```
Hasil normalisasi test_data Hold-out:
[[0.55384615 0.19642857 0.         ... 0.         1.         0.66666667]
[0.4        0.28571429 0.         ... 1.         1.         1.        ]
[0.58461538 0.19642857 0.         ... 1.         1.         1.        ]
...
[0.58461538 0.19642857 0.         ... 0.         1.         1.        ]
[0.92307692 0.16071429 1.         ... 1.         1.         0.66666667]
[0.78461538 0.57142857 1.         ... 1.         1.         1.        ]]

Hasil normalisasi test_data K-Fold:
[[0.53846154 0.05357143 0.         0.         0.         0.
 0.33333333]
[0.92307692 0.16071429 1.         1.         1.         1.
 0.53333333]
[0.58461538 0.19642857 1.         1.         1.         0.
 0.33333333]]
...
[[0.55384615 0.07142857 0.         0.         0.         0.
 1.        ]
[0.55384615 0.28571429 0.         0.         0.         1.
 0.66666667]
[0.53846154 0.03571429 0.         0.         0.         0.
 0.46666667]]

Hasil normalisasi test_data LOO:
[[0.03372549 0.21568627 0.         0.00392157 0.00392157 0.00392157
 1.        ]]
```

- Analisa

  Kode di atas dilakukan untuk melakukan normalisasi pada data train yang mana diperlukan untuk membuat data dengan indeks 0 hingga 1

4. test_data ← lakukan normalisasi pada train_data dengan min-max

- Kode

```
# Normalisasi train_data test_data dengan min-max(0-1)
sc = MinMaxScaler(feature_range=(0,1))

# Metode Hold-out
datatrainnormalHold = sc.fit_transform(xtrainHold)
datatestnormalHold = sc.fit_transform(xtestHold)
print(" Hasil normalisasi test_data Hold-out: \n", datatestnormalHold)

# Metode K-Fold
datatrainnormalFold = sc.fit_transform(xtrainFold)
datatestnormalFold = sc.fit_transform(xtestFold)
print("\n Hasil normalisasi test_data K-Fold: \n",
      datatestnormalFold[:3], "\n...\n",datatestnormalFold[-3:])

# Metode LOO
datatrainnormalLoo = sc.fit_transform(xtrainLoo)
mindata = xtrainLoo.min().min()
maxdata = xtrainLoo.max().max()
datatestnormalLoo = (np.array(xtestLoo)[:,:] - mindata) / (maxdata - mindata)
print("\n Hasil normalisasi test_data LOO: \n", datatestnormalLoo)
# print('\nxtest\n', xtestFold)
# print('\nxtest\n', xtestLoo)
print('\nYtest\n', ytestLoo)
```

- Keluaran

```
 Hasil normalisasi test_data Hold-out:
 [[0.55384615 0.19642857 0.         ... 0.         1.         0.66666667]
 [0.4        0.28571429 0.         ... 1.         1.         1.        ]
 [0.58461538 0.19642857 0.         ... 1.         1.         1.        ]
 ...
 [0.58461538 0.19642857 0.         ... 0.         1.         1.        ]
 [0.92307692 0.16071429 1.         ... 1.         1.         0.66666667]
 [0.78461538 0.57142857 1.         ... 1.         1.         1.        ]]

 Hasil normalisasi test_data K-Fold:
 [[0.53846154 0.05357143 0.         0.         0.         0.
  0.33333333]
 [0.92307692 0.16071429 1.         1.         1.         1.
  0.53333333]
 [0.58461538 0.19642857 1.         1.         1.         0.
  0.33333333]]
...
 [[0.55384615 0.07142857 0.         0.         0.         0.
  1.        ]
 [0.55384615 0.28571429 0.         0.         0.         1.
  0.66666667]
 [0.53846154 0.03571429 0.         0.         0.         0.
  0.46666667]]

 Hasil normalisasi test_data LOO:
 [[0.03372549 0.21568627 0.         0.00392157 0.00392157 0.00392157
  1.        ]]
```

- Analisa

  Kode di atas dilakukan untuk melakukan normalisasi pada data train yang mana diperlukan untuk membuat data dengan indeks 0 hingga 1

5. Lakukan klasifikasi k-NN (k=3) untuk masing-masing pendekatan validasi dan hitunglah error ratio-nya

   - Kode

```
# Klasifikasi k-NN (k=3)
kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')

#5.A. k-NN dengan metode Hold-out
kNN.fit(datatrainnormalHold,ytrainHold)
hasilKlasifikasiHold = kNN.predict(datatestnormalHold)

precision_rasioHold=kNN.score(datatestnormalHold, ytestHold)
error_ratioHold = (1 - precision_rasioHold)*100

print("5.A.Error rasio dengan Hold-out", error_ratioHold, "%")
print("    Hasil klasifikasi k-NN dengan Hold-out\n", hasilKlasifikasiHold)

#5.B. k-NN dengan metode k-Fold
kNN.fit(datatrainnormalFold,ytrainFold)
hasilKlasifikasiFold = kNN.predict(datatestnormalFold)

precision_rasioFold=kNN.score(datatestnormalFold, ytestFold)
error_ratioFold = (1 - precision_rasioFold)*100

print("5.B.Error rasio dengan k-Fold", error_ratioFold, "%")
print("    Hasil klasifikasi k-NN dengan k-Fold\n", hasilKlasifikasiFold)

#5.C. k-NN dengan metode LOO
kNN.fit(datatrainnormalLoo,ytrainLoo)
hasilKlasifikasiLoo = kNN.predict(datatestnormalLoo)

precision_rasioLoo=kNN.score(datatestnormalLoo, ytestLoo)
error_ratioLoo = (1 - precision_rasioLoo)*100

print("5.C.Error rasio dengan LOO", error_ratioLoo, "%")
print("    Hasil klasifikasi k-NN dengan LOO\n", hasilKlasifikasiLoo)
```

   - Keluaran

```
5.A.Error rasio dengan Hold-out 0.6289308176100628 %
5.B.Error rasio dengan k-Fold 0.0 %
5.C.Error rasio dengan LOO 0.0 %
```

   - Analisa

     Berdasarkan perhitungan di atas k-Fold memiliki error rasio yang sama dengan LOO yang mana sempurna yaitu 0 % sedangkan Hold-Out masih memiliki 0.628% rasio errornya