LAPORAN PRAKTIKUM
WORKSHOP KECERDASAN KOMPUTASIONAL
"Naive Bayes Classification"

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. Lakukan klasifikasi Naïve Bayes untuk masing-masing pendekatan validasi dan hitunglah akurasi untuk masing-masing metode validation

- Kode

```
#Klasifikasi Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

#A. Hold-Method (70%-30%)
classifierHold = GaussianNB()
classifierHold.fit(xtrainHold, ytrainHold)
ypredHold = classifierHold.predict(xtestHold)

# Menghitung akurasi Hold-Method tanpa normalisasi
acHold = accuracy_score(ytestHold,ypredHold)
print("A. Akurasi Hold-Method tanpa normalisasi",acHold*100," %")
print("   Prediksi Hold-Method\n  ",ypredHold)

#B. k-Fold (k=10)
classifierFold = GaussianNB()
classifierFold.fit(xtrainFold, ytrainFold)
ypredFold = classifierFold.predict(xtestFold)

# Menghitung akurasi k-Fold tanpa normalisasi
acFold = accuracy_score(ytestFold,ypredFold)
print("\nB. Akurasi k-Fold tanpa normalisasi",acFold*100," %")
print("   Prediksi k-Fold\n  ",ypredFold)
```

```
#C. LOO
classifierLoo = GaussianNB()
classifierLoo.fit(xtrainLoo, ytrainLoo)
ypredLoo = classifierLoo.predict(xtestLoo)

# Menghitung akurasi LOO tanpa normalisasi
acLoo = accuracy_score(ytestLoo,ypredLoo)
print("\nC. Akurasi LOO tanpa normalisasi", acLoo*100, " %")
print("   Prediksi LOO\n  ",ypredLoo)
```

- Keluaran

```
A. Akurasi Hold-Method tanpa normalisasi 91.82389937106919  %
   Prediksi Hold-Method
   ['medium' 'low' 'high' 'low' 'high' 'medium' 'low' 'high' 'high' 'high'
 'high' 'medium' 'high' 'medium' 'medium' 'medium' 'medium' 'medium'
 'medium' 'low' 'low' 'low' 'medium' 'low' 'low' 'medium' 'medium' 'low'
 'low' 'low' 'low' 'medium' 'low' 'high' 'high' 'medium' 'low' 'medium'
 'low' 'high' 'medium' 'high' 'high' 'high' 'medium' 'high' 'high'
 'medium' 'high' 'medium' 'high' 'low' 'low' 'low' 'medium' 'medium' 'low'
 'medium' 'low' 'low' 'medium' 'high' 'low' 'low' 'low' 'medium' 'high'
 'low' 'high' 'low' 'medium' 'low' 'low' 'high' 'low' 'high' 'high'
 'medium' 'high' 'high' 'low' 'high' 'medium' 'medium' 'low' 'low' 'low'
 'low' 'high' 'medium' 'high' 'low' 'low' 'medium' 'medium' 'low' 'medium'
 'low' 'low' 'high' 'high' 'high' 'low' 'low' 'high' 'low' 'medium'
 'medium' 'high' 'medium' 'high' 'medium' 'medium' 'medium' 'low' 'medium'
 'low' 'high' 'low' 'medium' 'low' 'high' 'medium' 'low' 'high' 'low'
 'low' 'low' 'medium' 'low' 'medium' 'medium' 'low' 'high' 'high' 'high'
 'low' 'low' 'high' 'low' 'medium' 'high' 'low' 'high' 'medium' 'high'
 'medium' 'medium' 'medium' 'low' 'high' 'low' 'low' 'high' 'high' 'low'
 'high' 'low' 'low' 'medium' 'high' 'low' 'high' 'low' 'high' 'high' 'low'
 'low' 'medium' 'high' 'high' 'medium' 'high' 'high' 'low' 'low' 'medium'
 'high' 'low' 'high' 'high' 'medium' 'low' 'high' 'medium' 'high' 'medium'
 'medium' 'high' 'high' 'medium' 'high' 'low' 'medium' 'high' 'low' 'low'
 'high' 'low' 'medium' 'low' 'low' 'low' 'low' 'high' 'medium' 'low'
 'medium' 'medium' 'medium' 'high' 'medium' 'high' 'low' 'high' 'high'
 'high' 'medium' 'medium' 'high' 'medium' 'high' 'medium' 'high' 'low'
 'low' 'medium' 'medium' 'high' 'medium' 'high' 'low' 'medium' 'medium'
 'low' 'high' 'medium' 'medium' 'high' 'high' 'low' 'high' 'high' 'medium'
 'low' 'low' 'low' 'medium' 'low' 'low' 'medium' 'high' 'medium' 'high'
 'low' 'medium' 'high' 'high' 'medium' 'low' 'low' 'low' 'low' 'high'
 'high' 'low' 'high' 'medium' 'low' 'high' 'low' 'high' 'high' 'high'
 'low' 'low' 'medium' 'medium' 'medium' 'medium' 'low' 'medium' 'medium'
 'medium' 'low' 'low' 'low' 'high' 'low' 'high' 'high' 'high' 'low' 'high'
 'high' 'low' 'low' 'medium' 'low' 'medium' 'medium' 'high' 'medium' 'low'
 'low' 'high' 'medium' 'low' 'low' 'high' 'high' 'medium' 'high' 'low'
 'high' 'medium' 'low' 'low']

B. Akurasi k-Fold tanpa normalisasi 96.19047619047619  %
   Prediksi k-Fold
   ['medium' 'low' 'high' 'low' 'low' 'high' 'medium' 'high' 'low' 'medium'
 'medium' 'low' 'medium' 'medium' 'high' 'high' 'low' 'medium' 'low' 'low'
 'low' 'low' 'medium' 'high' 'medium' 'medium' 'high' 'high' 'high'
 'medium' 'medium' 'high' 'low' 'low' 'high' 'low' 'low' 'high' 'low'
 'medium' 'high' 'low' 'high' 'medium' 'low' 'low' 'medium' 'high' 'low'
 'low' 'medium' 'medium' 'high' 'low' 'medium' 'low' 'high' 'low' 'medium'
 'medium' 'high' 'low' 'medium' 'medium' 'low' 'high' 'medium' 'low'
```

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

C. Akurasi LOO tanpa normalisasi 100.0  %
   Prediksi LOO
   ['low']
```

- Analisa

  Kode di atas digunakan untuk melakukan klasifikasi naïve bayes, metode ini merupakan metode klasifikasi yang biasanya digunakan untuk memprediksi hasil keluaran dari model yang telah dilatih dengan data_training. Pada praktiknya model akan dilatih menggunakan data dari data_training dengan jumlah data yang beragam yaitu tergantung pada metode validasi apa yang akan digunakan. Pada studi kasus di atas digunakanlah metode validasi Hold-out, k-Fold, dan LOO, yang mana ketiganya memiliki standar dan aturan sendiri dalam memisahkan antara data_training dengan data_test. Apabila model telah dilatih dengan semua data yang ada, maka ia akan diujicobakan untuk diprediksi hasil keluaran seperti apa yang akan keluar apabila diberikan nilai tertentu. Sehingga dari proses di atas diperoleh beberapa hasil akurasi yang diantaranya Hold-Out dengan 91,823%, k-Fold dengan 96,1904% dan LOO dengan hasil yang sempurna yaitu 100%.

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
normalTrainHold = sc.fit_transform(xtrainHold)
normalTestHold = sc.fit_transform(xtestHold)
print(" Hasil normalisasi test_data Hold-out: \n", normalTestHold)

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

# Metode LOO
mindata = xtrainLoo.min().min()
maxdata = xtrainLoo.max().max()
normalTrainLoo = sc.fit_transform(xtrainLoo)
normalTestLoo = (np.array(xtestLoo)[:,:] - mindata) / (maxdata - mindata)
print("\n Hasil normalisasi test_data LOO: \n", normalTestLoo)
```

- 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

  Langkah di atas merupakan langkah dalam melakukan normalisasi terhadap data_training dan data_testing untuk masing-masing metode validasi. Tahapan normalisasi ini dirasa perlu dilakukan karena untuk menghindari bias pada data yang akan diolah, sehingga hasil keluaran yang diperoleh dapat maksimal dan tidak terganggu dengan adanya data bias tersebut. Selain itu normalisasi juga digunakan untuk menghilangkan jarak antara data maksimal dengan data minimal yang mana terpaut cukup jauh. Oleh karenanya pada tahapan normalisasi ini data yang tersedia diolah menjadi dalam rentang 0 hingga 1.

5. Bandingkan nilai akurasi klasifikasi dengan Naïve Bayes pada salah satu metode validasi jika data training & data test dilakukan normalisasi & tidak dinormalisasi

- Kode

```
# A. Menghitung akurasi Hold-out tanpa normalisasi
acHold = accuracy_score(ytestHold,ypredHold)
print("Akurasi Hold-Method tanpa normalisasi", acHold*100, " %")

# B. Menghitung akurasi Hold-out dengan normalisasi
classifierHoldNorm = GaussianNB()
classifierHoldNorm.fit(normalTrainHold, ytrainHold)
ypredHoldNorm = classifierHoldNorm.predict(normalTestHold)
acHoldNorm = accuracy_score(ytestHold,ypredHoldNorm)
print("Akurasi Hold-Method dengan normalisasi", acHoldNorm*100, " %")
```

- Keluaran

```
Akurasi Hold-Method tanpa normalisasi 91.82389937106919  %
Akurasi Hold-Method dengan normalisasi 91.82389937106919  %
```

- Analisa

  Langkah di atas digunakan untuk mendeteksi hasil akurasi antara klasifikasi Naïve Bayes dengan normalisasi dan tanpa normalisasi. Dari percobaan di atas, diperoleh hasil yang mana tingkat akurasi klasifikasi Naïve Bayes dan validasi Hold-out dengan normalisasi maupun normalisasi ternyata ialah sama, yaitu di angka 91,823%. Hal tersebut biasa terjadi apabila data yang digunakan sudah stabil ataupun optimal yaitu terhindar dari bias maupun residu residu data lainnya yang membuat perhitungan data dari model kurang optimal dan berakibat penurunan tingkat akurasi dalam mendeteksi data_test.