LAPORAN PRAKTIKUM
WORKSHOP KECERDASAN KOMPUTASIONAL
"Water Potability"

Oleh :

Muhammad Rifqi Aminuddin
NRP. 3123640039

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

1. dataset ← milk_training.csv

   - Kode

```
import pandas as pd
import numpy as np
dataset = pd.read_csv('../Dataset/water_potability.csv')
print("Dataset\n", dataset)
```

   - Keluaran

```
Dataset
             ph    Hardness        Solids  Chloramines     Sulfate  \
0           NaN  204.890456  20791.31898     7.300212  368.516441
1      3.716080  129.422921  18630.05786     6.635246         NaN
2      8.099124  224.236259  19909.54173     9.275884         NaN
3      8.316766  214.373394  22018.41744     8.059332  356.886136
4      9.092223  181.101509  17978.98634     6.546600  310.135738
...         ...         ...          ...          ...         ...
3271   4.668102  193.681736  47580.99160     7.166639  359.948574
3272   7.808856  193.553212  17329.80216     8.061362         NaN
3273   9.419510  175.762646  33155.57822     7.350233         NaN
3274   5.126763  230.603758  11983.86938     6.303357         NaN
3275   7.874671  195.102299  17404.17706     7.509306         NaN

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
0       564.308654       10.379783        86.990970   2.963135           0
1       592.885359       15.180013        56.329076   4.500656           0
2       418.606213       16.868637        66.420093   3.055934           0
3       363.266516       18.436525       100.341674   4.628771           0
4       398.410813       11.558279        31.997993   4.075075           0
...            ...             ...              ...        ...         ...
3271    526.424171       13.894419        66.687695   4.435821           1
3272    392.449580       19.903225              NaN   2.798243           1
3273    432.044783       11.039070        69.845400   3.298875           1
3274    402.883113       11.168946        77.488213   4.708658           1
3275    327.459761       16.140368        78.698446   2.309149           1

[3276 rows x 10 columns]
```

   - Analisa

     Pada kode di atas diberikan perintah untuk memanggil library yang diperlukan selama proses komputasional ini berlangsung, yaitu mulai dari pandas dan numpy yang keduanya memiliki kegunaan masing-masing dan disusul oleh inisialisasi library pada bagian berikutnya. Selain itu pada kode di atas, dibuatlah variabel dataset yang berisikan dari dataset dengan nama “water_potability.csv”. Dataset tersebut

dipanggil dan disimpan ke dalam variabel untuk digunakan kemudian pada tahapan berikutnya.

2. Lakukan pengisian missing value jika atribut yang anda pilih ada yang nilainya kosong

- Kode

```
dataset = dataset.fillna(dataset.groupby('Potability').transform('mean'))
print("\n\nDataset setelah pengisian missing value:\n")
dataset.head()
```

- Keluaran

```
Dataset setelah pengisian missing value:
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	7.085378	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.05786	6.635246	334.564290	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.54173	9.275884	334.564290	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771	0
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

- Analisa

Pada kode di atas dilakukan perintah untuk melengkapi beberapa kolom yang entitas dan atributnya berupa NaN atau biasa kita tau dengan data kosong. Pengisian tersebut bertujuan untuk memperoleh pendekatan yang paling dekat guna mengisi letak kolom yang kosong tersebut. Aturan pengisiannyapun juga tidak sembarangan, melainkan dengan berbagai perhitungan model matematika. Seperti pada contoh di atas menggunakan mean yang berarti mengisi angka-angkanya dengan mengambil nilai rata-rata.

3. Buatlah skenario tertentu pada pemilihan fitur sehingga error ratio dari klasifikasi test_data dapat sekecil mungkin

- Kode

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

data = dataset.loc[:, dataset.columns != 'Potability']
label = dataset.loc[:,['Potability']]

chi2_selector = SelectKBest(chi2, k=6)
chi2_selector.fit(data, label)

cols = chi2_selector.get_support(indices=True)
df_selected_features = dataset.iloc[:,cols]
df_selected_features.head()
```

- Keluaran

	Hardness	Solids	Sulfate	Conductivity	Organic_carbon	Trihalomethanes
0	204.890456	20791.31898	368.516441	564.308654	10.379783	86.990970
1	129.422921	18630.05786	334.564290	592.885359	15.180013	56.329076
2	224.236259	19909.54173	334.564290	418.606213	16.868637	66.420093
3	214.373394	22018.41744	356.886136	363.266516	18.436525	100.341674
4	181.101509	17978.98634	310.135738	398.410813	11.558279	31.997993

- Analisa

  Pada kode di atas diberikan perintah untuk mengambil kolom yang paling mencolok untuk kemudiah menjadi data latih dan data uji. Pemilihan ini berdasarkan kriteria seperti, apabila kolom tersebut tidak ada apakah akan berdampak besar. Seperti contoh pada praktikum di atas, pemilihan ke-enam kolom tersebut dilakukan oleh komputer tanpa melibatkan pengguna. Keenam kolom tersebut yang nantinya akan digunakan datanya pengganti database.

4. Lakukan validation Model dengan metode Hold-out

- Kode

```
# Validation Model Metode Hold-out
from sklearn.model_selection import train_test_split
train_data, test_data, train_label, test_label = train_test_split(
                                                  df_selected_features,
                                                  label, test_size = 0.30,
                                                  random_state=100)

train_data.head()
```

- Keluaran

	Hardness	Solids	Sulfate	Conductivity	Organic_carbon	Trihalomethanes
2020	212.705332	25626.48105	338.663535	330.253375	11.052838	64.472001
106	213.541932	22539.71161	334.041186	463.398847	11.623019	69.615850
1849	235.944436	20862.45587	378.758715	418.697530	13.608075	66.303555
834	203.537833	23603.06542	324.084334	288.912457	18.064957	85.916666
141	167.386127	20944.62081	334.564290	566.339294	11.318807	66.303555

- Analisa

  Pada data di atas dilakukannlah Validation Model yang mana pada studi kasus ini menggunakan metode validasi Hold Out. Metode ini berjalan dengan rumusan pembagian antara data_training, dengan data_set yang mana biasanya ialah 70-30 yang mana rumus ini berarti dari dataset akan diubahnya menjadi 70% atau 30% sesuai dengan parameter yang dikirimkan

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

- Kode

```
from sklearn.preprocessing import MinMaxScaler

sc = MinMaxScaler(feature_range=(0,1))

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

- Keluaran

```
Hasil normalisasi train_data:
[[0.55767381 0.41548342 0.5955832  0.25391227 0.27920894 0.51706123]
[0.56102515 0.36480275 0.58245267 0.51673053 0.30303786 0.55879224]
[0.65076735 0.33726444 0.70948004 0.42849379 0.38599705 0.53192026]
...
[0.17451738 0.4603304  0.81201961 0.53382563 0.58376188 0.6133811 ]
[0.55198948 0.43724443 0.57826497 0.86137927 0.39998634 0.55699581]
[0.53502211 0.24476912 0.54654164 0.63485548 0.34736671 0.69306007]]
```

- Analisa

  Pada kode di atas diberikan perintah untuk melakukan normalisasi terhadap train_data. Perintah Normalisasi itu sendiri diperlukan untuk mengurangi bolong antara data teratas dengan data terbawah. Tentunya selain itu, proses ini juga dapat menghemat penggunaan baris maupun lainnya. Hal ini juga dapat membuat semua data hanya memiliki nilai jika tidak 0 ya pasti 1 ataupun nilai sesuai degengan skala yang telah dikenali oleh model kNN

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

- Kode

```
# Normalisasi test_data dengan min-max(0-1)
test_dataNorm = sc.fit_transform(test_data.values)
print(" Hasil normalisasi test_data: \n", test_dataNorm)
```

- Keluaran

```
Hasil normalisasi test_data:
[[0.58182505 0.33994593 0.76794359 0.34138939 0.35169766 0.70217778]
[0.48871945 0.40470372 0.43879334 0.83252477 0.27671731 0.52381243]
[0.79874614 0.58786063 0.53501012 0.42912618 0.46692973 0.19329234]
...
[0.         0.35877378 0.68734452 0.55741411 0.39843643 0.50182154]
[0.56736472 0.32322877 0.52751296 0.25526579 0.56195963 0.61871685]
[0.51874442 0.55063958 0.52751296 0.22024586 0.26324128 0.3418015 ]]
```

- Analisa

  Pada kode di atas diberikan perintah untuk melakukan normalisasi terhadap test_data. Perintah Normalisasi itu sendiri diperlukan untuk mengurangi bolong antara data teratas dengan data terbawah. Tentunya selain itu, proses ini juga dapat menghemat penggunaan baris maupun

lainnya. Hal ini juga dapat membuat semua data hanya memiliki nilai jika tidak 0 ya pasti 1 ataupun nilai sesuai degengan skala yang telah dikenali oleh model kNN

7. Algoritma klasifikasi yang dipakai adalah k-NN, Naïve Bayes, Decision Tree, tunjukkan akurasinya

- Kode

```
from sklearn.metrics import accuracy_score

# A. Klasifikasi KNN (k=3)
from sklearn.neighbors import KNeighborsClassifier

kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')
kNN.fit(train_dataNorm,train_label)
pred_knn = kNN.predict(test_dataNorm)

acc_knn=round(accuracy_score(pred_knn, test_label)*100,2)

print("A. Accuracy k-NN", acc_knn, "%")

# B. Klasifikasi Naive Bayes
from sklearn.naive_bayes import GaussianNB

classifier = GaussianNB()
classifier.fit(train_dataNorm, train_label)
pred_bayes = classifier.predict(test_dataNorm)

acc_bayes=round(accuracy_score(pred_bayes, test_label)*100,2)

print("B. Accuracy Bayesian", acc_bayes, "%")

# C.Klasifikasi Decision Tree
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()
dtc.fit(train_dataNorm, train_label)
pred_tree = dtc.predict(test_dataNorm)

acc_tree=round(accuracy_score(pred_tree, test_label)*100,2)

print("C. Accuracy Decision Tree", acc_tree, "%")
```

- Keluaran

```
A. Accuracy k-NN 56.77 %
B. Accuracy Bayesian 59.21 %
C. Accuracy Decision Tree 50.56 %
```

- Analisa

  Pada kode di atas diberikan perintah untuk menerapkan berbagai metode klasifikasi yaitu mulai dari k-NN, Bayesian, dan hingga sampai pada Decision Tree. Pada tahap ini selain dicek, semuanya juga dilakukan klasifikasi beserta prediknya sehingga dengan ini kita tahu keakurasian dan error rasio dari skenario yang telah dibuat sehingga dari hasil keakurasian tersebut dapat menjadikan kita tahu langkah apa yang sebaiknya perlu diambil. Seperti pada percobaan di atas, akurasi paling unggu dimiliki oleh Bayesian dengan 59,21%, lalu yang ke dua diduduki oleh klasifikasi k-NN dengan 56,77% dan posisi terakhir diisi oleh klasifikasi Decission Tree dengan 50,56%. Hasil tersebut tentunya bisa jadi berbeda-beda bergantung pada data yang dimiliki dan kekompleksan data yang dimiliki