LAPORAN PRAKTIKUM
WORKSHOP KECERDASAN KOMPUTASIONAL
"Decission Tree"

Oleh :

Muhammad Rifqi Aminuddin
NRP. 3123640039

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

1. train_dataset ← milk_training.csv

- Kode

```
# Import library yang dibutuhkan
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from graphviz import Source
from IPython.display import Image
import matplotlib.pyplot as plt
import pandas as pd

train_dataset = pd.read_csv('../Dataset/milk_training.csv')
print(train_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    6.6          37      1     1    1          1     255    high
3    6.8          45      0     1    1          1     255    high
4    6.6          45      0     1    1          1     250    high
..   ...         ...    ...   ...  ...        ...     ...     ...
737  6.7          45      1     1    1          0     245  medium
738  6.5          38      1     0    1          0     255  medium
739  6.7          41      1     0    0          0     247  medium
740  6.8          41      0     0    0          0     255  medium
741  6.8          38      0     0    0          0     255  medium

[742 rows x 8 columns]
```

- Analisa

  Pada kode di atas diberikan perintah untuk memanggil library yang digunakan yaitu mulai dari import DecisionTreeClassifier, graphviz, hingga pandas. Semua library tersebut digunakan dalam proses berikut dari memanggil file, menampilkan file, hingga melakukan komputasi dengan berbagai metode. Selain itu pada kode di atas juga diberikan perintah untuk memanggil dataset training dengan nama "milk_training.csv". Dataset yang dipanggil tersebut untuk kemudian diinisialisasi dengan nama train_dataset. Penggunaan dataset tersebut dan pemberian namanya dilakukan untuk mempermudah apabila membutuhkan datanya pada perintah selanjutnya.

2. train_data ← ambil train_dataset kolom fitur (pH, Temprature, Taste, Odor, Fat, Turbidity, Colour).

   - Kode

```
train_data = train_dataset.loc[:, train_dataset.columns != 'Grade']
print(train_data)
```

   - Keluaran

```
      pH  Temprature  Taste  Odor  Fat  Turbidity  Colour
0    6.6          35      1     0    1          0     254
1    6.6          36      0     1    0          1     253
2    6.6          37      1     1    1          1     255
3    6.8          45      0     1    1          1     255
4    6.6          45      0     1    1          1     250
..   ...         ...    ...   ...  ...        ...     ...
737  6.7          45      1     1    1          0     245
738  6.5          38      1     0    1          0     255
739  6.7          41      1     0    0          0     247
740  6.8          41      0     0    0          0     255
741  6.8          38      0     0    0          0     255

[742 rows x 7 columns]
```

   - Analisa

     Kode di atas merupakan sintaks yang digunakan untuk memisahkan antara data dengan label pada dataset training. Langkah ini dilakukan dengan cara mengambil seluruh kolom yang tidak bernama “Grade”. Sintaks loc digunakan untuk mengatasi pemisahan antara data dengan label ini tersebut. Pemisahan dilakukan agar data yang terlah dipisahkan nantinya dapat dimasukkan ke dalam variabel train_data untuk kemudian digunakan dalam train model, yaitu melatih model dengan memasukkan beberapa data latih untuk kemudian dikenali dan dipelajari polanya. Pada praktikum ini digunakanlah metode Decision Tree yang mana ia akan menghitung tiap satu data dan menghitungnya dengan rumus komputasi.

3. test_dataset ← milk_testing.csv

   - Kode

```
test_dataset = pd.read_csv('../Dataset/milk_testing.csv')
print(test_dataset)
```

- Keluaran

```
      pH  Temprature  Taste  Odor  Fat  Turbidity  Colour   Grade
0    6.8          45      1     1    1          0     245    high
1    6.6          37      1     1    1          1     255    high
2    6.7          38      1     0    1          0     255    high
3    6.8          45      0     1    1          1     255    high
4    6.6          37      1     1    1          1     255    high
..   ...         ...    ...   ...  ...        ...     ...     ...
312  6.5          36      0     0    0          0     247  medium
313  6.6          38      0     0    0          0     255  medium
314  6.5          37      0     0    0          0     255  medium
315  6.5          40      1     0    0          0     250  medium
316  6.7          45      1     1    0          0     247  medium

[317 rows x 8 columns]
```

- Analisa

  Pada kode di atas diberikan perintah untuk memanggil dataset testing dengan nama "milk_testing.csv". Dataset yang dipanggil tersebut untuk kemudian diinisialisasi dengan nama test_dataset. Penggunaan dataset tersebut dan pemberian namanya dilakukan untuk mempermudah apabila membutuhkan datanya pada perintah selanjutnya seperti untuk diinisialisasi atau dipecah untuk kemudian disimpan dalam berbagai variabel.

4. test_data ← ambil test_dataset kolom fitur (pH, Temprature, Taste, Odor, Fat, Turbidity, Colour).

- Kode

```
test_data = test_dataset.loc[:, test_dataset.columns != 'Grade']
print(test_data)
```

- Keluaran

```
      pH  Temprature  Taste  Odor  Fat  Turbidity  Colour
0    6.8          45      1     1    1          0     245
1    6.6          37      1     1    1          1     255
2    6.7          38      1     0    1          0     255
3    6.8          45      0     1    1          1     255
4    6.6          37      1     1    1          1     255
..   ...         ...    ...   ...  ...        ...     ...
312  6.5          36      0     0    0          0     247
313  6.6          38      0     0    0          0     255
314  6.5          37      0     0    0          0     255
```

```
315  6.5          40      1     0    0          0     250
316  6.7          45      1     1    0          0     247

[317 rows x 7 columns]
```

- Analisa

  Pada kode di atas diterapkan sintaks yang digunakan untuk memisahkan antara data dengan label pada dataset testing. Langkah ini dilakukan dengan cara mengambil seluruh kolom yang tidak bernama “Grade”. Sintaks loc digunakan untuk mengatasi pemisahan antara data dengan label ini tersebut. Pemisahan dilakukan agar data yang terlah dipisahkan nantinya dapat dimasukkan ke dalam variabel test_data untuk kemudian digunakan dalam train model, yaitu melatih model dengan memasukkan beberapa data latih untuk kemudian dikenali dan dipelajari polanya yang pada praktikum ini menggunakan metode Decision Tree yang akan menghitung tiap satu data dan menghitungnya dengan rumus komputasi.

5. train_label ← ambil train_data kolom kelas (Grade)

- Kode

```
train_label = train_dataset.loc[:,['Grade']]
print(train_label)
```

- Keluaran

```
      Grade
0      high
1      high
2      high
3      high
4      high
..      ...
737  medium
738  medium
739  medium
740  medium
741  medium

[742 rows x 1 columns]
```

- Analisa

  Pada kode di atas diberikan sintaks untuk memisahkan antara label pada train_dataset yang tersedia. Pemisahan ini dilakukan dengan menggunakan modul loc yang hanya akan mengambil kolom “Grade” dari dataset tersebut untuk kemudian disimpan ke dalam variabel train_label. Pemisahan diperlukan dalam melakukan training atau pelatihan dataset training, sehingga model dapat membedakan mana yang label dan mana yang data kemudian mempelajri polanya untuk memberikan model yang optimal.

6. test_label ← ambil test_data kolom kelas (Grade)
   - Kode

```
test_label = test_dataset.loc[:,['Grade']]
print(test_label)
```

   - Keluaran

```
      Grade
0      high
1      high
2      high
3      high
4      high
..      ...
312  medium
313  medium
314  medium
315  medium
316  medium

[317 rows x 1 columns]
```

   - Analisa

     Pada kode di atas diberikan sintaks untuk memisahkan antara label pada test_dataset yang tersedia. Pemisahan ini dilakukan dengan menggunakan modul loc yang hanya akan mengambil kolom “Grade” dari dataset tersebut untuk kemudian disimpan ke dalam variabel test_label. Pemisahan diperlukan dalam melakukan testing atau prediksi pada model dengan dataset testing, sehingga kita dapat menentukan

akurasi dari model tersebut dalam melakukan prediksi sebuah data yang masuk..

7. Lakukan klasifikasi test_data terhadap train_data dengan Decision Tree, dan berapakah error rationya?

    - Kode

```
dtc = DecisionTreeClassifier()
dtcFit = dtc.fit(train_data, train_label)
class_result = dtc.predict(test_data)
acr = dtc.score(train_data, train_label)
err = round((1-acr)*100, 2)

print("Acuration =",round(acr*100, 2),"%")
print("Error ration =",err,"%")
print("Class =\n",class_result)
```

    - Keluaran

```
Acuration = 100.0 %
Error ration = 0.0 %
Class =
 ['high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high'
 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'high' 'medium'
 'high' 'high' 'high' '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']
```

- Analisa

  Pada kode di atas dilakukanlah perintah untuk melakukan klasifikasi dengan menggunakan metode Decision Tree yang mana ia akan melatih model dengan memasukkan data dari train_data yang kemudian akan dipelajari pola apa yang muncul apabila dimasukkan data dan akan menghasilkan label seperti pada train_label. Apabila pelatihan data telah selesai dan dirasa cukup optimal, maka tahap selanjutnya ialah melakukan pengujian model dengan menggunakan data yang berasal dari test_data. Pengujian ini memproses data tersebut dan memproduksi atau memprediksi hasil keluaran maupun label yang ia hasilkan. Label yang dihasilkan tersebut kemudian akan dicocokkan dengan label asli milik variabel test_label yang sebelumnya telah dipisahkan dari datanya. Pencocokan ini digunakan untuk mencari akurasi dan error rasio pada model yang digunakan, sehingga apabila error rasio yang ditimbulkan masih tinggi maka ia masih memerlukan training atau pelatihan kembali dengan data yang beragam pula. Kegiatan pelatihan, pengujian, dan perhitungan akurasi ini diperlukan untuk menciptakan model yang optimal, cepat tetapi memiliki tingkat akurasi yang tinggi, seperti pada hasil keluaran di atas menghasilkan tingkat akurasi 100% dengan error rasio yang 0% atau dapat dikatakan model yang telah dibangun dirasa telah optimal dan dapat digunakan di lapangan untuk diuji dengan data yang lebih beragam.

8. Tampilkan hirarki dari Decision Tree

- Kode

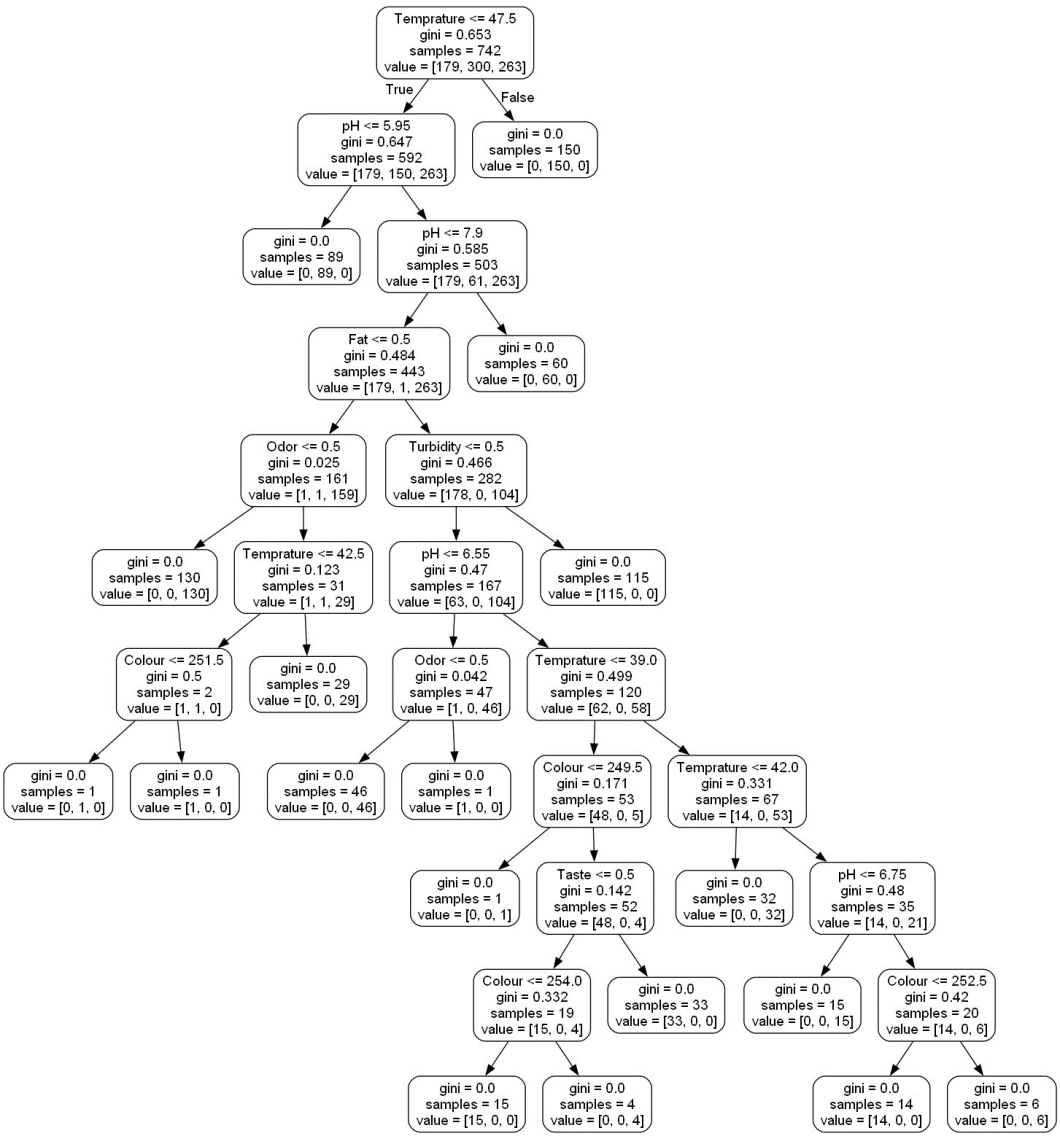
```
# Membuat data DOT untuk pohon keputusan
dot_data = export_graphviz(dtc, out_file=None,
                           feature_names=train_data.columns, rounded=True)
```

```
# Membuat objek graph dari data DOT
graph = Source(dot_data)

# Menyimpan grafik dalam format PNG (opsional)
graph.render("decision_tree", format="png")

# Menampilkan grafik dalam jendela pop-up
# graph
Image(filename="decision_tree.png", width=400, height=400)
```

- Keluaran



- Analisa

  Pada kode di atas diberikan perintah untuk membuat pohon keputusan, yan gmana pohon tersebut berdasarkan hasil dari pembelajaran model yang telah di bangun sebelumnya. Pembuatan pohon keputusan juga merupakan salah satu cara untuk memahami metode Decision Tree tersebut melalui visualisasi atau gambaran visual. Keputusan yang dibuat juga bersifat True atau False, hal ini karena keputusan yang dibuat berupa data boolean yang hanya ada 1 ataupun 0. Panjang ataupun lebarnya pohon keputusan yang dibuat bergantung pada data yang diolah seperti pada praktikum di atas ialah menggunakan data dari train_data. Format gambar pohon keputusan dapat berubah bergantung pada parameter yang diberikan untuk menyimpan gambar pohon keputusan menjadi jenis file tertentu.

9. Bandingkan dengan hasil dari k-NN dan Bayesian

   - Kode

```
# Normalisasi train_data test_data dengan min-max(0-1)
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range=(0,1))
train_dataNorm = sc.fit_transform(train_data)
test_dataNorm = sc.fit_transform(test_data)
# Metode Decission Tree
print("A. Acuration Decission Tree   :",round(acr*100, 2),"%")
print("   Error ration Decission Tree:",err,"%")

# Metode k-NN
from sklearn.neighbors import KNeighborsClassifier
kNN=KNeighborsClassifier(n_neighbors=3, weights='distance')
kNN.fit(train_dataNorm,train_label)
predKNN = kNN.predict(test_dataNorm)
acrKNN = accuracy_score(test_label, predKNN)
print("B. Akurasi KNN                :", round(acrKNN*100,2), "%")
print("   Error rasio KNN            :", round((1-acrKNN)*100,2), "%")

# Metode Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
classifier = GaussianNB()
classifier.fit(train_data, train_label)
predBayes = classifier.predict(test_data)
acrBayes = accuracy_score(test_label,predBayes)
print("C. Akurasi Naive Bayes        :", round(acrBayes*100,2), "%")
print("   Error rasio Naive Bayes    :", round((1-acrBayes)*100,2), "%")
```

- Keluaran

```
A. Acuration Decission Tree   : 100.0 %
   Error ration Decission Tree: 0.0 %
B. Akurasi KNN                : 99.37 %
   Error rasio KNN            : 0.63 %
C. Akurasi Naive Bayes        : 95.58 %
   Error rasio Naive Bayes    : 4.42 %
```

- Analisa

  Pada kode di atas dilakukanlah komparasi untuk membandingkan tingkat keberhasilan atau tingkat akurasi dan error rasio dari model Decision Tree untuk dibandingkan dengan metode Naïve Bayes dan k-NN yang telah dipelajari sebelumnya. Pengujian tersebut dengan menggunakan data yang sama ialah train_data, train_label, test_data, dan test_label. Dari pengujian di atas diketahui model yang paling optimal dan paling akurat ialah dimiliki oleh model dari metode Decision Tree dengan akurasi 100% dan error rasio 0%, kemudian disusul model dari metode k-NN dengan akurasi 99.37% dan error rasio 0.63%, dan yang terakhir ialah model dari metode Naïve Bayes dengan akurasi 95.58% dan error rsio 4.42%. Berdasarkan ketiga model tersebut, ketiganya memiliki tingkat akurasi yang berbeda, hal ini dikarenakan ketiga metode tersebut memiliki algoritma masing-masing dalam melakukan pelatihan maupun pengujian sehingga tak dapat dipungkiri ketiganya memiliki hasil yang berbeda pula. Namun tingkat akurasi tersebut bisa jadi akan berbeda apabila dilatih dan diuji dengan data yang berbeda pula.