



PROGRAM STUDI  
TEKNIK INFORMATIKA – S1  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS DIAN NUSWANTORO

MATA KULIAH  
**DATA MINING**



< a href='https://www.freepik.com/vectors/technology'>Technology vector created by sentavio - www.freepik.com</a>

# DATA MINING

## “Klasifikasi Decision Tree”

**TIM PENGAMPU DOSEN DATA MINING**

2021

**MATERI PERKULIAHAN****Materi Pra UTS**

- #1 Pengantar Data Mining
- #2 Data utk Data Mining ( Jenis2 Data, Pengukuran Data, Nilai dan Atribut)
- #3 Preprosesing Data (Data Cleaning, Missing Value, Transformasi Data, koding python)
- #4 Metode Learning (Disiplin Data Mining, Supervised & Unsupervised, Klasifikasi,Prediksi, Estimasi, Klastering,dan Asosiasi)
- #5 Klasifikasi dengan Naive Bayes + Python
- #6 Klasifikasi dengan KNN + Python
- #7 Klasifikasi Decision Tree + Python

**Evaluasi Tengah Semester (UTS)****Materi Pasca UTS**

- #8 ANN & Deep Learning + Python
- #9 Klastering (Teknik Klaster, Metode Partisi, Metode Hirarkis)
- #10 Metode Partisi (K-Means Klastering + Python)
- #11 Metode Hirarkis (HAC + Python)
- #12 Regresi (Sederhana dan Multivariate) + Python
- #13 Asosiasi + Apriori / FP-Growth + Python
- #14 Validasi dan Pengujian Model

**Evaluasi Akhir Semester (UAS)**

## Introduction

- Algoritma C4.5 merupakan algoritma yang digunakan untuk membentuk pohon keputusan (*Decision Tree*).
- Pohon keputusan merupakan metode klasifikasi dan prediksi yang terkenal.
- Pohon keputusan berguna untuk mengekspolari data, menemukan hubungan tersembunyi antara sejumlah calon variabel input dengan sebuah variabel target.

## Varian Algoritma Decision Tree

- Banyak algoritma yang dapat dipakai dalam pembentukan pohon keputusan, antara lain : ID3, CART, dan C4.5 (Larose, 2005).
- Algoritma C4.5 merupakan pengembangan dari algoritma ID3 (Larose, 2005).
- Proses pada pohon keputusan adalah mengubah bentuk data (tabel) menjadi model pohon, mengubah model pohon menjadi rule, dan menyederhanakan rule (Basuki & Syarif, 2003).

## Contoh Data Bermain Tenis

| NO | OUTLOOK | TEMPERATURE | HUMIDITY | WINDY | PLAY |
|----|---------|-------------|----------|-------|------|
| 1  | Sunny   | Hot         | High     | FALSE | No   |
| 2  | Sunny   | Hot         | High     | TRUE  | No   |
| 3  | Cloudy  | Hot         | High     | FALSE | Yes  |
| 4  | Rainy   | Mild        | High     | FALSE | Yes  |
| 5  | Rainy   | Cool        | Normal   | FALSE | Yes  |
| 6  | Rainy   | Cool        | Normal   | TRUE  | Yes  |
| 7  | Cloudy  | Cool        | Normal   | TRUE  | Yes  |
| 8  | Sunny   | Mild        | High     | FALSE | No   |
| 9  | Sunny   | Cool        | Normal   | FALSE | Yes  |
| 10 | Rainy   | Mild        | Normal   | FALSE | Yes  |
| 11 | Sunny   | Mild        | Normal   | TRUE  | Yes  |
| 12 | Cloudy  | Mild        | High     | TRUE  | Yes  |
| 13 | Cloudy  | Hot         | Normal   | FALSE | Yes  |
| 14 | Rainy   | Mild        | High     | TRUE  | No   |

## Algoritma C4.5

Secara umum algoritma C4.5 untuk membangun pohon keputusan adalah sebagai berikut :

1. Pilih atribut sebagai akar.
2. Buat cabang untuk tiap-tiap nilai.
3. Bagi kasus dalam cabang.
4. Ulangi proses untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama.

## Konsep Entropy

- Entropy ( $S$ ) merupakan jumlah bit yang diperkirakan dibutuhkan untuk dapat mengekstrak suatu kelas (+ atau -) dari sejumlah data acak pada ruang sampel  $S$ .
- Entropy dapat dikatakan sebagai kebutuhan bit untuk menyatakan suatu kelas.
- Entropy digunakan untuk mengukur ketidakaslian  $S$ .

## Konsep Entropy [2]

- Untuk perhitungan nilai Entropy sbb :

$$\text{Entropy}(S) = \sum_{i=1}^n -pi * \log_2 pi$$

- Keterangan :
  - S : himpunan kasus.
  - A : fitur.
  - n : jumlah partisi S.
  - pi : proporsi dari  $S_i$  terhadap S

## Konsep Gain

- Gain ( $S, A$ ) merupakan perolehan informasi dari atribut  $A$  relative terhadap output data  $S$ .
- Perolehan informasi didapat dari output data atau variable dependent  $S$  yang dikelompokkan berdasarkan atribut  $A$ , dinotasikan dengan gain ( $S, A$ ).

## Konsep Gain [2]

- Untuk memilih atribut sebagai akar, didasarkan pada nilai gain tertinggi dari atribut-atribut yang ada.
- Untuk menghitung *gain* digunakan rumus :

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^n \frac{|S_i|}{|S|} * Entropy(S_i)$$

- Keterangan :
  - S : himpunan kasus
  - A : atribut
  - n : jumlah partisi atribut A
  - $|S_i|$  : jumlah kasus pada partisi ke-i
  - $|S|$  : jumlah kasus dalam S

## Langkah 1

- Menghitung jumlah kasus, jumlah kasus untuk keputusan **Yes**, jumlah kasus untuk keputusan **No**, dan Entropy dari semua kasus dan kasus yang dibagi berdasarkan atribut **OUTLOOK**, **TEMPERATURE**, **HUMIDITY**, dan **WINDY**.
- Setelah itu lakukan perhitungan Gain untuk setiap atribut.
- Hasil perhitungan ditunjukan di bawah ini.

# Perhitungan Node 1

|                 |        | jml kasus(S) | Tidak (S1) | Ya (S2) | Entropy | Gain      |
|-----------------|--------|--------------|------------|---------|---------|-----------|
| <b>total</b>    |        | 14           | 4          | 10      | 0.86312 |           |
| <b>outlook</b>  |        |              |            |         |         | 0.258521  |
|                 | cloudy | 4            | 0          | 4       | 0       |           |
|                 | rainy  | 5            | 1          | 4       | 0.72193 |           |
|                 | sunny  | 5            | 3          | 2       | 0.97095 |           |
| <b>temp</b>     |        |              |            |         |         | 0.1838509 |
|                 | col    | 4            | 0          | 4       | 0       |           |
|                 | hot    | 4            | 2          | 2       | 1       |           |
|                 | mild   | 6            | 2          | 4       | 0.9183  |           |
| <b>humidity</b> |        |              |            |         |         | 0.3705065 |
|                 | high   | 7            | 4          | 3       | 0.98523 |           |
|                 | normal | 7            | 0          | 7       | 0       |           |
| <b>windy</b>    |        |              |            |         |         | 0.0059777 |
|                 | FALSE  | 8            | 2          | 6       | 0.81128 |           |
|                 | TRUE   | 6            | 4          | 2       | 0.9183  |           |

## Cara Perhitungan Node 1

- Baris total kolom Entropy dihitung dengan persamaan:

*Entropy(Total)*

$$= \left( -\frac{4}{14} * \log_2 \left( \frac{4}{14} \right) \right) + \left( -\frac{10}{14} * \log_2 \left( \frac{10}{14} \right) \right)$$

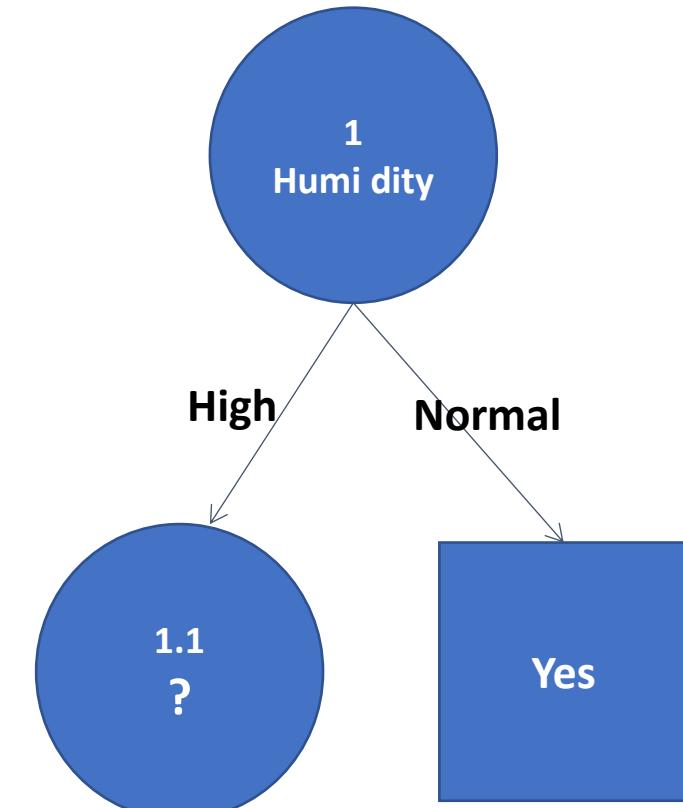
- *Entropy(Total) = 0,863120569*

## Cara Perhitungan Node 1 [2]

- Nilai gain pada baris **OUTLOOK** dihitung :
- $Gain(Total, Outlook) = Entropy(Total) - \sum_{i=1}^n \frac{|Outlook_i|}{|Total|} * Entropy(Outlook_i)$
- $Gain(Total, Outlook) = 0.863120569 - \left( \left( \frac{4}{14} * 0 \right) + \left( \frac{5}{14} * 0.723 \right) + \left( \frac{5}{14} * 0.97 \right) \right)$
- $Gain(Total, Outlook) = 0.23$

## Cara Perhitungan Node 1[3]

- Dari hasil diketahui bahwa atribut dengan gain tertinggi adalah **HUMIDITY** yaitu sebesar 0.37. Sehingga **HUMIDITY** dapat menjadi node akar.
- Ada dua nilai atibut dari **HUMIDITY**, yaitu **HIGH** dan **NORMAL**.
- Nilai atribut **NORMAL** sudah mengklasifikasikan kasus menjadi 1, yaitu keputusannya Yes, sehingga tidak perlu dilakukan perhitungan lebih lanjut.
- Tetapi untuk nilai **HIGH** masih perlu dilakukan perhitungan lagi.



## Langkah 2

- Menghitung jumlah kasus, jumlah kasus untuk keputusan **Yes**, jumlah kasus untuk keputusan **No**.
- Entropy dari semua kasus dan kasus yang dibagi berdasarkan atribut **OUTLOOK**, **TEMPERATURE** dan **WINDY**, yang dapat menjadi node akar dari nilai atribut **HIGH**.
- Setelah itu lakukan perhitungan Gain, untuk tiap-tiap atribut.

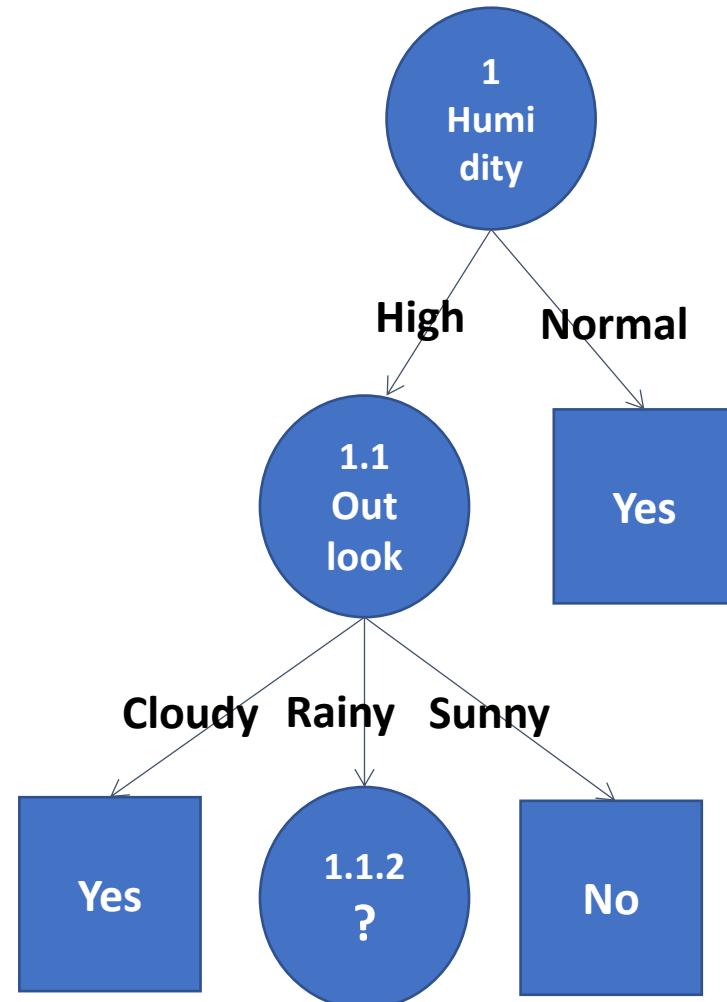
## Perhitungan Node 1.1

|               |        | jml kasus(S) | Tidak (S1) | Ya (S2) | Entropy    | Gain       |
|---------------|--------|--------------|------------|---------|------------|------------|
| Humidity High |        | 7            | 4          | 3       | 0.98522814 |            |
| outlook       | cloudy | 2            | 0          | 2       | 0          | 0.69951385 |
|               | rainy  | 2            | 1          | 1       | 1          |            |
|               | sunny  | 3            | 3          | 0       | 0          |            |
| temp          |        |              |            |         |            | 0.02024421 |
|               | col    | 0            | 0          | 0       | 0          |            |
|               | hot    | 3            | 2          | 1       | 0.91829583 |            |
| windy         | mild   | 4            | 2          | 2       | 1          | 0.02024421 |
|               | FALSE  | 4            | 2          | 2       | 1          |            |
|               | TRUE   | 3            | 2          | 1       | 0.91829583 |            |

## Cara Perhitungan Node 1.1

- Atribut dengan Gain tertinggi adalah **OUTLOOK**, yaitu sebesar **0.6995**.
- Sehingga **OUTLOOK** dapat menjadi node cabang dari nilai atribut **HIGH**.
- Ada tiga nilai dari atribut **OUTLOOK** yaitu **CLOUDY**, **RAINY** dan **SUNNY**.
  - **CLOUDY** => klasifikasi kasus 1 (Yes)
  - **SUNNY** => klasifikasi kasus 1 (No)
  - **RAINY** => masih perlu perhitungan lagi.

## Cara Perhitungan Node 1.1 [2]



## Langkah 3

- Menghitung jumlah kasus, jumlah kasus untuk keputusan **Yes**, jumlah kasus untuk keputusan **No**.
- Entropy dari semua kasus dan kasus yang dibagi berdasarkan atribut **TEMPERATURE** dan **WINDY**, yang dapat menjadi node cabang dari nilai atribut **RAINY**.
- Setelah itu lakukan perhitungan Gain, untuk tiap-tiap atribut.

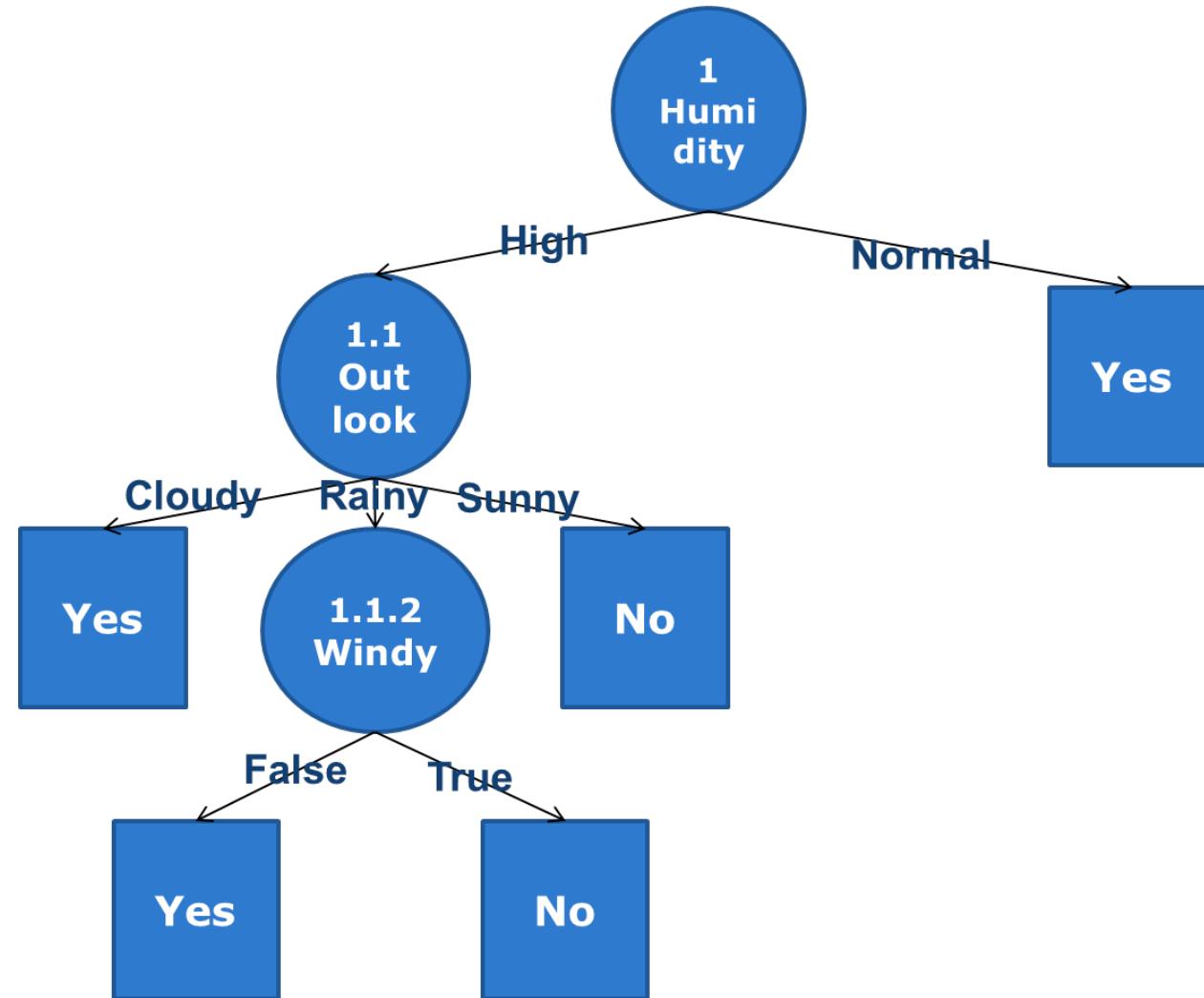
## Perhitungan Node 1.1.2

| Node 1.1.2                            |       | jml kasus(S) | Tidak (S1) | Ya (S2) | Entropy | Gain |
|---------------------------------------|-------|--------------|------------|---------|---------|------|
| Humidity High<br>and Outlook<br>Rainy |       | 2            | 1          | 1       | 1       |      |
| temp                                  |       |              |            |         |         | 0    |
|                                       | cool  | 0            | 0          | 0       | 0       |      |
|                                       | hot   | 0            | 0          | 0       | 0       |      |
|                                       | mild  | 2            | 1          | 1       | 1       |      |
| windy                                 |       |              |            |         |         | 1    |
|                                       | FALSE | 1            | 0          | 1       | 0       |      |
|                                       | TRUE  | 1            | 1          | 0       | 0       |      |

## Cara Perhitungan Node 1.1.2

- Atribut dengan Gain tertinggi adalah WINDY, yaitu sebesar 1.
- Sehingga WINDY dapat menjadi node cabang dari nilai atribut RAINY.
- Ada dua nilai dari atribut WINDY, yaitu FALSE dan TRUE.
  - Nilai atribut FALSE sudah mengklasifikasikan kasus menjadi 1 (**Yes**).
  - Nilai atribut TRUE sudah mengklasifikasikan kasus menjadi 1 (**No**).
  - Sehingga tidak perlu dilakukan perhitungan lagi.

## Cara Perhitungan Node 1.1.2 [2]



# Implementasi Python

- Mengimpor library yang diperlukan:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
import matplotlib.pyplot as plt
```

## Implementasi Python [2]

- Membaca data input.
- Sebagai ilustrasi , dataset yang digunakan di dalam program ini adalah dataset bunga IRIS yang bersumber pada data yang dipublikasikan oleh Fisher (Fisher, 1950) yang tersedia di website UCI Machine Learning Repository.

```
iris = datasets.load_iris()  
features = iris['data']  
target = iris['target']
```

## Implementasi Python [3]

- Membuat objek model Decision Tree.

```
decisiontree = DecisionTreeClassifier(random_state=0,  
                                      max_depth=None, min_samples_split=2,  
                                      min_samples_leaf=1, min_weight_fraction_leaf=0,  
                                      max_leaf_nodes=None, min_impurity_decrease=0)
```

## Implementasi Python [4]

- Mentraining model Decision Tree.

```
model = decisiontree.fit(features, target)
```

## Implementasi Python [5]

- Membuat prediksi.
- Pada tahap ini dilakukan pengambilan sampel observasi dan membuat prediksi.
- Sampel yang diberikan adalah data dimensi kelopak.
- Fungsi predict() digunakan untuk memeriksa kelas yang dimilikinya, sedangkan predict\_proba digunakan untuk memeriksa probabilitas kelas dari prediksi tersebut.

```
observation = [[5, 4, 3, 2]]  
model.predict(observation)  
model.predict_proba(observation)
```

## Implementasi Python [6]

- Membuat grafik visualisasi Decision Tree.

```
import pydotplus
from sklearn import tree
dot_data = tree.export_graphviz(decisiontree, out_file=None,
|    feature_names=iris['feature_names'], class_names=iris['target_names'])
from IPython.display import Image
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
graph.write_png("D:\iris.png")
```

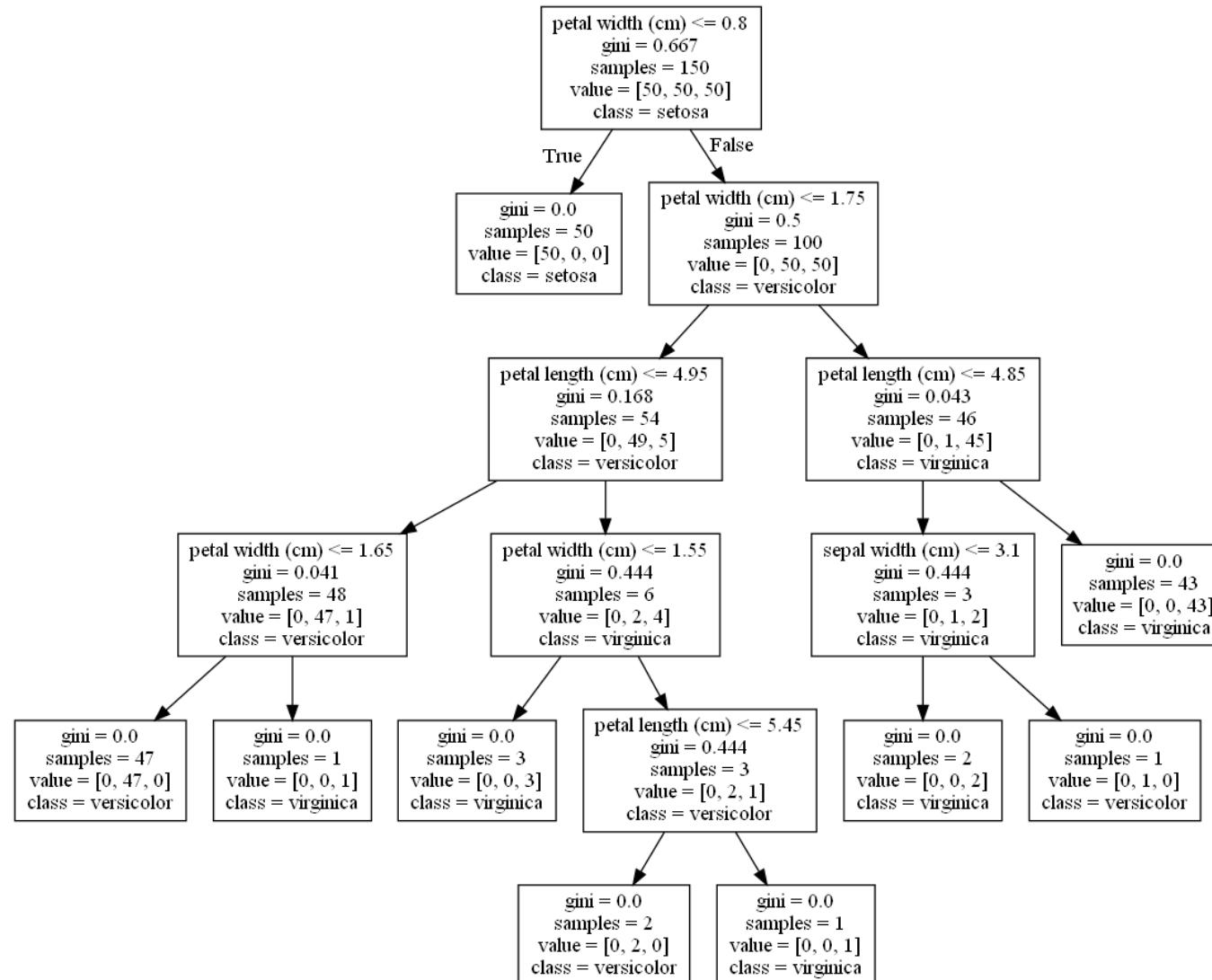
## Implementasi Python [7]

- Ket:
  - ***max\_depth*** => parameter bagi kedalaman maksimum dari decision tree. Jika ***max\_depth = None***, maka decision tree akan dibuat sampai seluruh data termasuk kedalam tree. Sedangkan jika ***max\_depth=k*** (*k* sebuah bilangan bulat) maka kedalaman tree dibatasi sampai ke dalam *k*.
  - ***min\_sample\_split*** => parameter bagi jumlah minimum sampel dalam setiap node sebelum node dipecah (displit). Jika ***min\_samples\_split = k*** (*k* sebuah bilangan bulat) maka jumlah minimum sampel dalam node tersebut adalah *k*. Sedangkan jika ***min\_samples\_split = r*** (*r* sebuah bilangan pecahan) maka jumlah minimum sampel dalam node tersebut adalah *r* persen dari keseluruhan data input.

## Implementasi Python [8]

- Ket:
  - ***max\_samples\_leaf*** => parameter bagi jumlah minimum data yang dibutuhkan dalam sebuah leaf. Aturan pemberian nilai bagi ***min\_samples\_leaf*** sama dengan ***min\_samples\_split***.
  - ***max\_leaf\_nodes*** => parameter bagi jumlah maksimum leaf.
  - ***min\_impurity\_split*** => parameter bagi jumlah minimum penurunan impurity sebelum dilakukan split.

# Implementasi Python [9]



- Hasil Visualisasi Pohon Keputusan dari Kode Python sebelumnya seperti terlihat pada gambar disamping.

# Implementasi Python dengan Dataset CSV

```
#import numpy, pandas dan scikit-learn
import numpy as np
import pandas as pd
from sklearn import tree

#membaca dataset dari file ke pandas DataFrame
irisDataset = pd.read_csv('D:\klasifikasi_dataset_iris.csv',
                           delimiter=',', header=0)
#mengubah kelas (kolom "Species") dari string ke unique-integer
irisDataset["Species"] = pd.factorize(irisDataset.Species)[0]
#menghapus kolom "Id"
irisDataset = irisDataset.drop(labels="Id", axis=1)

#mengubah dataframe ke array numpy
#irisDataset = irisDataset.as_matrix()
irisDataset = irisDataset.to_numpy()
```

- Import library yang diperlukan seperti numpy, pandas dan sklearn.
- Membaca dataser dari file csv ke pandas dataframe.
- Mengubah dataframe ke array numpy.

## Implementasi Python dengan Dataset CSV [2]

| A  | B             | C            | D             | E            | F               |
|----|---------------|--------------|---------------|--------------|-----------------|
| Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species         |
| 1  | 7             | 03.02        | 04.07         | 01.04        | Iris-versicolor |
| 2  | 06.04         | 03.02        | 04.05         | 01.05        | Iris-versicolor |
| 3  | 06.09         | 03.01        | 04.09         | 01.05        | Iris-versicolor |
| 4  | 05.05         | 02.03        | 4             | 01.03        | Iris-versicolor |
| 5  | 06.05         | 02.08        | 04.06         | 01.05        | Iris-versicolor |
| 6  | 05.07         | 02.08        | 04.05         | 01.03        | Iris-versicolor |
| 7  | 06.03         | 03.03        | 04.07         | 01.06        | Iris-versicolor |
| 8  | 04.09         | 02.04        | 03.03         | 1            | Iris-versicolor |
| 9  | 06.06         | 02.09        | 04.06         | 01.03        | Iris-versicolor |
| 10 | 05.02         | 02.07        | 03.09         | 01.04        | Iris-versicolor |
| 11 | 5             | 2            | 03.05         | 1            | Iris-versicolor |
| 12 | 05.09         | 3            | 04.02         | 01.05        | Iris-versicolor |
| 13 | 6             | 02.02        | 4             | 1            | Iris-versicolor |
| 14 | 06.01         | 02.09        | 04.07         | 01.04        | Iris-versicolor |
| 15 | 05.06         | 02.09        | 03.06         | 01.03        | Iris-versicolor |
| 16 | 06.07         | 03.01        | 04.04         | 01.04        | Iris-versicolor |
| 17 | 05.06         | 3            | 04.05         | 01.05        | Iris-versicolor |
| 18 | 05.08         | 02.07        | 04.01         | 1            | Iris-versicolor |
| 19 | 06.02         | 02.02        | 04.05         | 01.05        | Iris-versicolor |
| 20 | 05.06         | 02.05        | 03.09         | 01.01        | Iris-versicolor |

- Sampel dataset bunga IRIS seperti gambar disamping.

# Implementasi Python dengan Dataset CSV [3]

- Selanjutnya membagi dataset, dimana 40 baris data untuk training dan 20 baris data untuk testing.
  - Kemudian memecah dataset ke input dan label.

## Implementasi Python dengan Dataset CSV [4]

```
#mendefinisikan decision tree classifier
model = tree.DecisionTreeClassifier()
#mentraining model
model = model.fit(inputTraining, labelTraining)
#memprediski input data testing
hasilPrediksi = model.predict(inputTesting)
print("label sebenarnya ", labelTesting)
print("hasil prediksi: ", hasilPrediksi)
#menghitung akurasi
prediksiBenar = (hasilPrediksi == labelTesting).sum()
prediksiSalah = (hasilPrediksi != labelTesting).sum()
print("prediksi benar: ", prediksiBenar, " data")
print("prediksi salah: ", prediksiSalah, " data")
print("akurasi: ", prediksiBenar/(prediksiBenar+prediksiSalah)
     * 100, "%")
```

- Mendefinisikan model decision tree classifier.
- Mentraining model.
- Memprediksi input data testing.
- Kemudian mengevaluasi model dengan menghitung akurasinya.

## Implementasi Python dengan Dataset CSV [5]

```
label sebenarnya  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]  
hasil prediksi:  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]  
prediksi benar:  20  data  
prediksi salah:  0  data  
akurasi:  100.0 %
```

- Tampilan hasil eksekusi program python.
- Terlihat untuk label sebenarnya dan hasil prediksi, juga terdapat prediksi benar dan salah, serta hasil akurasinya.

## Latihan Soal (Kuis)

- Hitung Entropy dan Gain serta tentukan pohon keputusan yang terbentuk dari contoh kasus keputusan bermain tenis dibawah ini :

| OUTLOOK | TEMPERATURE | HUMIDITY | WINDY | PLAY       |
|---------|-------------|----------|-------|------------|
| Sunny   | Hot         | High     | No    | Don't Play |
| Sunny   | Hot         | High     | Yes   | Don't Play |
| Cloudy  | Hot         | High     | No    | Play       |
| Rainy   | Mild        | High     | No    | Play       |
| Rainy   | Cool        | Normal   | No    | Play       |
| Rainy   | Cool        | Normal   | Yes   | Play       |
| Cloudy  | Cool        | Normal   | Yes   | Play       |
| Sunny   | Mild        | High     | No    | Don't Play |
| Sunny   | Cool        | Normal   | No    | Play       |
| Rainy   | Mild        | Normal   | No    | Play       |
| Sunny   | Mild        | Normal   | Yes   | Play       |
| Cloudy  | Mild        | High     | Yes   | Play       |
| Cloudy  | Hot         | Normal   | No    | Play       |
| Rainy   | Mild        | High     | Yes   | Don't Play |

## Referensi

1. Kusrini, Taufiq Emha, Algoritma Data Mining, *Penerbit Andi*, 2009.
2. Ian H. Witten, Frank Eibe, Mark A. Hall, Data mining: Practical Machine Learning Tools and Techniques 4th Edition, *Elsevier*, 2017.
3. Budi Santosa, Ardian Umam, Data Mining dan Big Data Analytics, Penebar Media Pustaka, 2018.
4. Yaya Heryadi, Teguh Wahyono, Machine Learning: Konsep dan Implementasi, Penerbit Gava Media, 2020.
5. Sumber gambar: [www.freepik.com](http://www.freepik.com).



# THANKS

ANY QUESTIONS?

