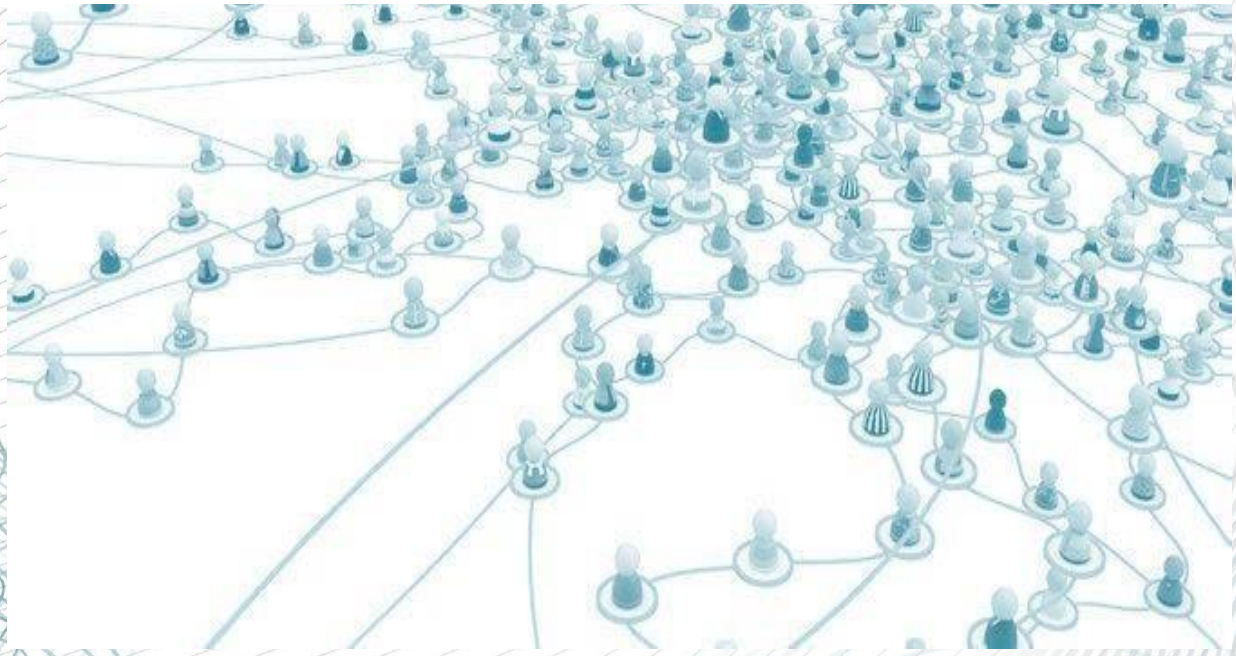
MODUL PRAKTIKUM

SD25-31001 DATA MINING



Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2025

MODUL 5

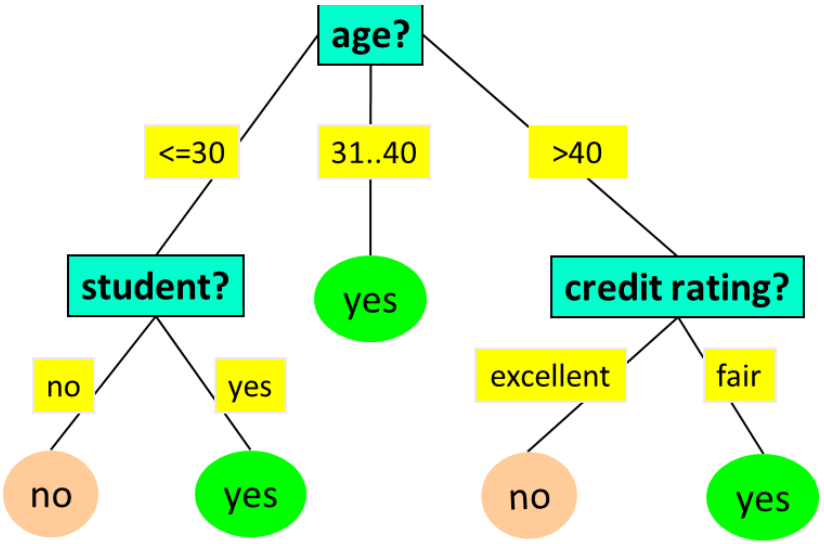
Supervised Learning

Decision Tree for Classification

A. Konsep Dasar

Klasifikasi merupakan salah satu teknik data mining yang dapat digunakan untuk menganalisis sekumpulan data dengan membangun suatu model. Klasifikasi juga dapat diartikan sebagai pemprosesan untuk menemukan sebuah model atau fungsi yang menjelaskan dan mencirikan konsep atau kelas data, untuk kepentingan tertentu. Hasil atau luaran dari penggunaan klasifikasi yaitu dapat dijadikan dasar dalam pengambilan keputusan bagi *data analysis*. Banyak pilihan model yang dapat digunakan untuk klasifikasi, dalam modul ini dua model yang akan digunakan yaitu algoritma *Decision tree*.

Decision Tree (pohon keputusan) adalah representasi grafis dari berbagai kemungkinan solusi terhadap suatu keputusan dan merupakan salah satu algoritma paling populer dalam pembelajaran terawasi (*supervised learning*). Algoritma ini mudah dipahami, mudah divisualisasikan, serta adaptif untuk berbagai jenis data. Decision Tree dapat digunakan untuk tugas klasifikasi maupun regresi, meskipun lebih sering diterapkan pada masalah klasifikasi. Algoritma ini bekerja dengan membagi dataset menjadi beberapa subset berdasarkan fitur tertentu hingga terbentuk struktur bercabang menyerupai pohon, di mana setiap cabang merepresentasikan suatu kondisi atau keputusan, dan setiap daun (*leaf node*) menunjukkan hasil akhir atau kelas prediksi seperti yang ditunjukkan pada Gambar 1.



Gambar 1. Model *Decision Tree*

Dalam algoritma *Decision Tree* terdapat beberapa ukuran penting yaitu *entropi*, *information gain*, dan *gini index*. Ketiga ukuran ini digunakan untuk menilai seberapa baik suatu atribut dapat memisahkan data ke dalam kelompok yang lebih homogen. Dengan menghitung nilai-nilai tersebut, algoritma dapat memilih atribut terbaik pada setiap percabangan sehingga pohon keputusan yang terbentuk menjadi lebih akurat dan efisien dalam melakukan klasifikasi.

1) Entropy

Entropy adalah ukuran ketidakpastian atau tingkat "kemurnian" suatu himpunan data. Nilai *entropy* semakin kecil, distribusi semakin homogen, node semakin murni (pure)

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

*Info(D) = Entropy (D)

2) Information Gain

Information Gain digunakan untuk memilih atribut terbaik yang akan dijadikan node pembagi pada pohon keputusan. Gain(A) menggambarkan seberapa besar entropy berkurang akibat atribut A. Semakin besar, semakin bagus.

$$Gain(A) = Info(D) - Info_A(D)$$

3) Gini Index

Gini Index merupakan alternatif dari entropi untuk mengukur impurity (ketidakhomogenan) data.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Kelebihan utama *Decision Tree* adalah kemudahan interpretasi dan visualisasi. Struktur model yang menyerupai alur logika "jika–maka" (*if–then rules*) membuatnya mudah dipahami, bahkan oleh pengguna non-teknis. Selain itu, Decision Tree juga dapat menangani data numerik dan kategorikal, serta tidak membutuhkan banyak pra-pemrosesan data.

B. Tujuan Praktikum

I. Tujuan Instruksional Umum

Praktikum bertujuan untuk menerapkan teori *Decision Tree* untuk klasifikasi pada penambangan data.

II. Tujuan Instruksional Khusus

1. Mahasiswa mampu menguasai konsep dasar *Decision Tree.*
2. Mahasiswa mampu menganalisis studi kasus menggunakan metode *Decision Tree*.
3. Mahasiswa mampu menampilkan visualisasi struktur pohon keputusan dan menganalisis hasil klasifikasi.

C. Dataset dan Bahasa Pemrograman Python

Dataset yang digunakan pada praktikum kali ini yaitu dataset bawaan (*built-in*) yang tersedia di *Seaborn* dan dapat langsung digunakan tanpa perlu mengunduh data eksternal dengan rincian sebagai berikut:

Tabel 1. informasi dataset

	Dataset 1 : Iris	**Dataset 2 : Penguins**
Tujuan	Klasifikasi jenis bunga iris	Klasifikasi jenis penguin
Jumlah kelas (target)	3 spesies: *setosa*, *versicolor*, *virginica*	3 spesies: *Adelie*, *Chinstrap*, *Gentoo*
Jumlah fitur	4 fitur numerik	6 fitur utama (5 numerik + 1 kategorik)
Fitur	*sepal_length, sepal_width, petal_length, petal_width*	*bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g, sex, island*
Target kolom	species	species
Jenis data	Semua numerik kecuali target	Campuran numerik dan kategorik
Jumlah sampel	150 data	333 data (rata-rata, bisa bervariasi tergantung versi)

Berikut ini daftar *library* beserta fungsinya yang digunakan dalam praktikum untuk membantu proses pengolahan data, pelatihan model, evaluasi, dan visualisasi hasil.

Tabel 2. daftar library dan deskripsi

Kode	**Deskripsi**
import pandas as pd	Mengimpor library untuk manipulasi data berbentuk tabel (*DataFrame*).

import numpy as np	Mengimpor library untuk operasi numerik dan array.
import matplotlib.pyplot as plt	Mengimpor library untuk membuat grafik dan visualisasi data.
import seaborn as sns	Mengimpor library untuk visualisasi data statistik.
%matplotlib inline	Menampilkan grafik.
from sklearn.preprocessing import LabelEncoder	Mengimpor fungsi untuk mengubah data kategorik menjadi numerik.
from sklearn.model_selection import train_test_split	Mengimpor fungsi untuk membagi data menjadi train dan test set.
from sklearn.tree import DecisionTreeClassifier	Mengimpor model Decision Tree untuk klasifikasi.
from sklearn.metrics import classification_report, confusion_matrix	Mengimpor metrik evaluasi untuk mengukur performa model.
from sklearn.tree import plot_tree	Mengimpor fungsi untuk menampilkan visualisasi struktur pohon keputusan.

D. Implementasi Klasifikasi

Dalam modul praktikum ini akan digunakan data ***iris*** untuk mengklasifikasikan jenis bunga iris. Berikut adalah langkah-langkah yang dapat dilakukan:

1. *Import libraries*

   Beberapa library Python digunakan untuk membangun model Decision Tree, seperti dijelaskan pada Tabel 2.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

#for encoding
from sklearn.preprocessing import LabelEncoder

#for train test splitting
from sklearn.model_selection import train_test_split

#for decision tree object
from sklearn.tree import DecisionTreeClassifier

#for checking testing results
from sklearn.metrics import classification_report, confusion_matrix

#for visualizing tree
from sklearn.tree import plot_tree
```

2. *Import data*

Lakukan import dataset bawaan (built-in) yang tersedia di Seaborn.

```
# to know all the available dataset
sns.get_dataset_names()
```

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'dowjones',
 'exercise',
 'flights',
 'fmri',
 'geyser',
 'glue',
 'healthexp',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'seaice',
 'taxis',
 'tips',
 'titanic']
```

Pilih dan tampilkan data iris

```
#reading the data
df = sns.load_dataset('iris')
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Simpan data dengan nama file Iris.csv

```
df.to_csv('Iris.csv', index=False)
```

3. *Data Analysis / Preprocessing*

Exploratory Data Analysis (EDA) adalah proses menganalisis dan merangkum karakteristik utama kumpulan data, dengan tujuan memperoleh insight tentang

struktur, korelasi, dan pola yang mendasari data. EDA membantu mengidentifikasi fitur penting, anomali, dan tren dalam data yang dapat menginformasikan analisis dan pemodelan lebih lanjut.

EDA biasanya melibatkan beberapa langkah penting, termasuk:

- **Data cleaning and preparation** melibatkan penghapusan nilai yang hilang atau salah, transformasi variabel, dan penanganan outlier.
- **Analisis statistik** melibatkan penerapan metode matematika dan statistik pada data untuk mengidentifikasi fitur dan hubungan penting
- **Data visualization** adalah proses membuat grafik, bagan, dan representasi visual lainnya dari data untuk membantu mengidentifikasi pola, hubungan, dan anomali.

Pra-pemrosesan bertujuan untuk menyiapkan data sedemikian rupa sehingga memungkinkan analisis dan pemodelan yang efektif serta menghilangkan bias atau kesalahan apa pun yang dapat memengaruhi hasil. Anda bisa mengikuti langkah-langkah *data preprocessing* yang ada pada **modul 1, 2, dan 3 data mining**.

Dengan dataset iris, selesaikan langkah-langkah berikut dan tampilkan output pada setiap step:

a) Informasi dataset

Cek dimensi data, dataset ini memiliki 150 record, 5 kolom.

```
df.shape
```

```
(150, 5)
```

```
#getting information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

dengan empat kolom pertama bertipe float dan kolom terakhir bertipe objek str dan tidak ada nilai NAN seperti yang terbentuk setelah perintah berikut.

	0
sepal_length	False
sepal_width	False
petal_length	False
petal_width	False
species	False

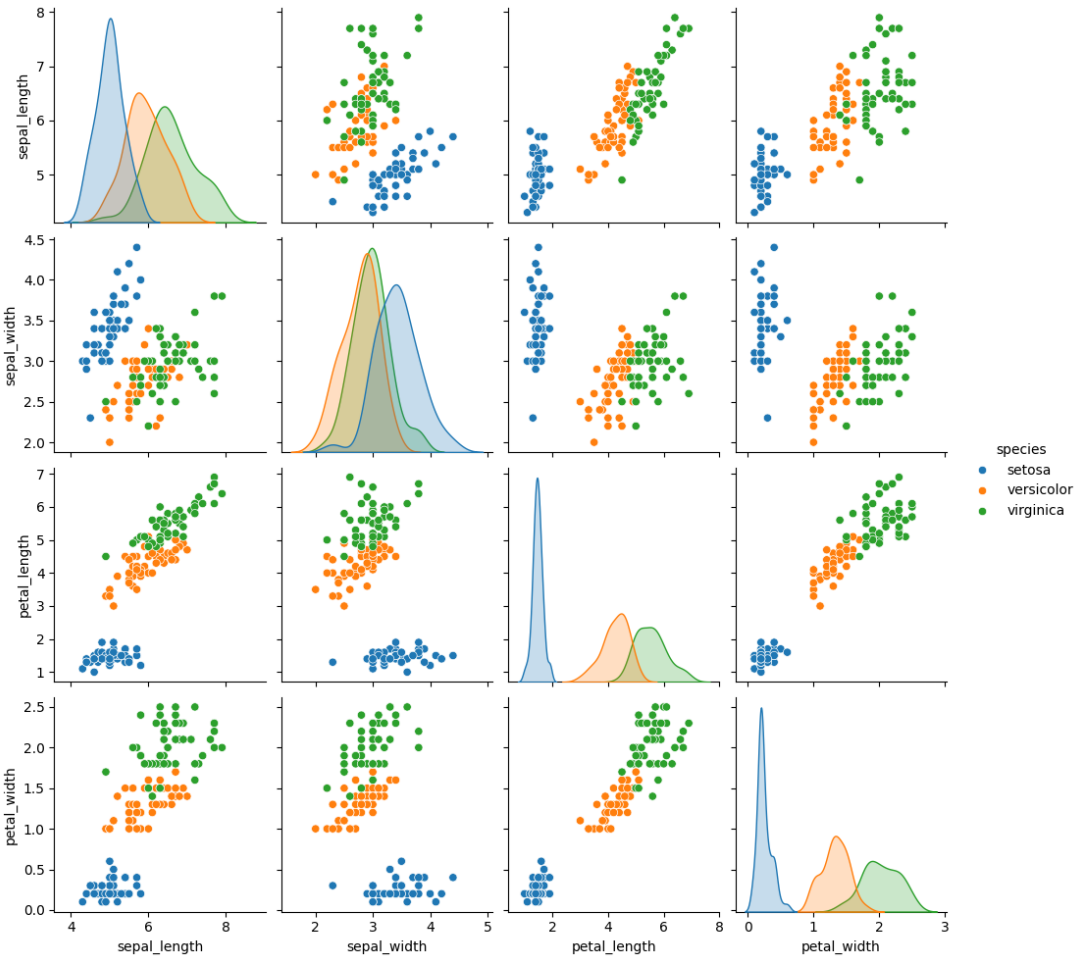
dtype: bool

b) Deskripsi data

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

c) Distribusi data

```
# let's plot pair plot to visualise the attributes all at once
sns.pairplot(data=df, hue = 'species')
plt.savefig("pne.png")
```



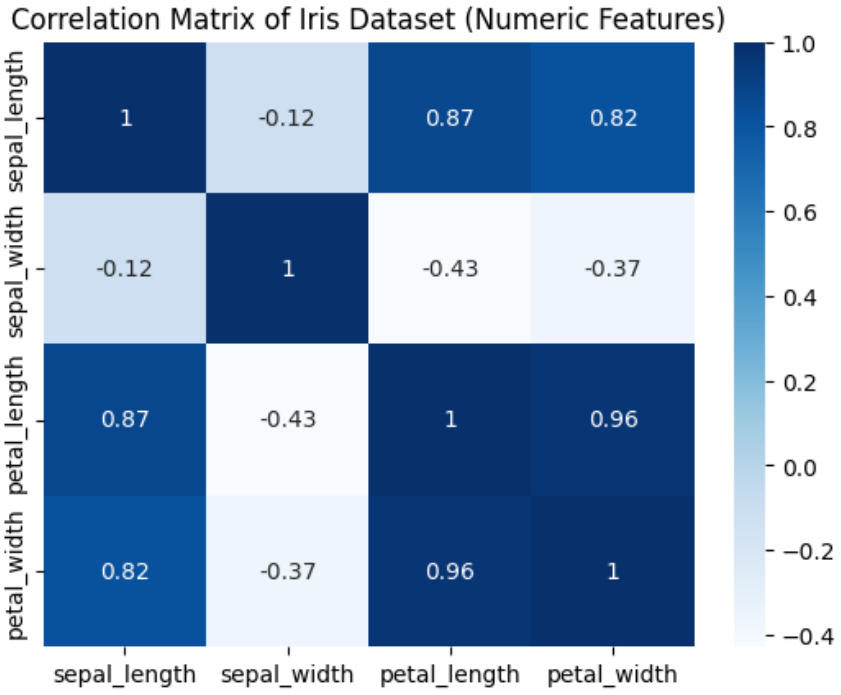
Dapat kita lihat**, Setosa** selalu membentuk klaster yang berbeda.

d) Correlation matrix

Correlation matrix adalah tabel yang merangkum hubungan antara beberapa variabel dalam suatu kumpulan data. Ini menunjukkan koefisien korelasi antara setiap pasangan variabel, dan juga menunjukkan kekuatan dan arah hubungan antar variabel. Hal ini berguna untuk mengidentifikasi variabel yang berkorelasi tinggi dan memilih subset variabel untuk analisis lebih lanjut.

```
# correlation matrix
# Exclude non-numeric columns before calculating correlation
numeric_df = df.select_dtypes(include=np.number)
sns.heatmap(numeric_df.corr(), annot=True, cmap='Blues')
plt.title('Correlation Matrix of Iris Dataset (Numeric Features)')
plt.savefig("one.png")
```

Output *Correlation matrix:*



Dari matriks korelasi terlihat bahwa:

- **Petal length** memiliki hubungan yang sangat kuat dengan **petal width**.
- **Sepal length** tidak memiliki hubungan dengan **sepal width**.

e) Label encoding

Label encoding adalah teknik pra-pemrosesan dalam analisis data di mana data kategorikal diubah menjadi nilai numerik, agar kompatibel dengan operasi dan model matematika. Ikuti langkah-langkah berikut:

Memisahkan variabel target (y) dan fitur (X) sebagai berikut:

```
target = df['species']
df1 = df.copy()
df1 = df1.drop('species', axis =1)
df1.shape
```

```
(150, 4)
```

Sebaiknya jangan menghapus atau menambahkan kolom baru ke dataset asli. Buat salinannya, lalu modifikasi sehingga jika hasilnya tidak sesuai harapan, kita memiliki data asli untuk memulai lagi dengan pendekatan yang berbeda.

```
# Defining the attributes
X = df1
```

Sekarang kita lihat variabel target kita.

```
target
```

	species
0	setosa
1	setosa
2	setosa
3	setosa
4	setosa
...	...
145	virginica
146	virginica
147	virginica
148	virginica
149	virginica

150 rows × 1 columns

dtype: object

Target memiliki variabel kategoris yang tersimpan di dalamnya, selanjutnya lakukan *label encoding* dalam nilai numerik agar berfungsi.

```
#label encoding
le = LabelEncoder()
target = le.fit_transform(target)
target
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Kita mendapatkan label seperti di atas, setosa:0, versicolor:1, virginica:2

Simpan target.

```
y = target
```

4. *Split data*

Membagi dataset menjadi data pelatihan dan pengujian.

```
# Splitting the data - 80:20 ratio
X_train, X_test, y_train, y_test = train_test_split(X , y, test_size = 0.2, random_state = 42)

print("Training split input- ", X_train.shape)
print("Testing split input- ", X_test.shape)
```

```
Training split input-  (120, 4)
Testing split input-  (30, 4)
```

5. *Train model*

Train model/pelatihan model melibatkan penggunaan kumpulan data pelatihan untuk memperkirakan parameter model.

```
# Defining the decision tree algorithm

dtree=DecisionTreeClassifier()
dtree.fit(X_train,y_train)

print('Decision Tree Classifer Created')
```

```
Decision Tree Classifer Created
```

pada kode di atas, kita membuat sebuah objek dari kelas ***DecisionTreeClassifier*** dan menyimpan alamatnya ke dalam variabel ***dtree***, sehingga objek tersebut dapat diakses menggunakan ***dtree***. Selanjutnya, kita melatih pohon keputusan tersebut menggunakan **X_train** dan **y_train**. Terakhir, kita mencetak pernyataan ***Decision Tree Classifier Created*** setelah pohon keputusan berhasil dibuat.

6. *Predict result / Score model*

Kita memperoleh akurasi sebesar 100% pada dataset pengujian yang berjumlah 30 data.

```
# Predicting the values of test data
y_pred = dtree.predict(X_test)
print("Classification report - \n", classification_report(y_test,y_pred))
```

```
Classification report -
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

7. Evaluasi model

plot confusion matrix seperti berikut.

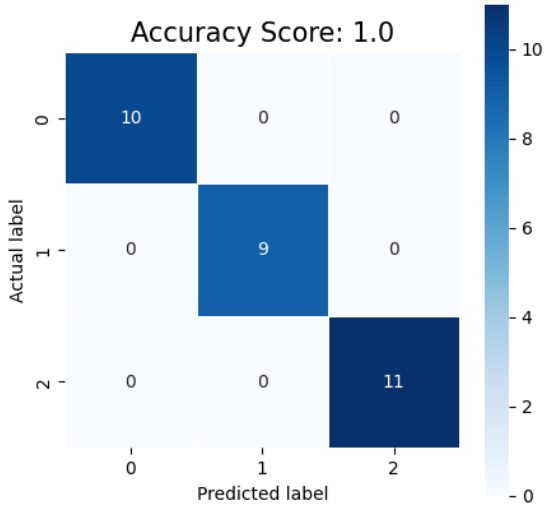
```
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(5,5))

sns.heatmap(data=cm,linewidths=.5, annot=True,square = True,  cmap = 'Blues')

plt.ylabel('Actual label')
plt.xlabel('Predicted label')

all_sample_title = 'Accuracy Score: {0}'.format(dtree.score(X_test, y_test))
plt.title(all_sample_title, size = 15)

plt.savefig("one.png")
```



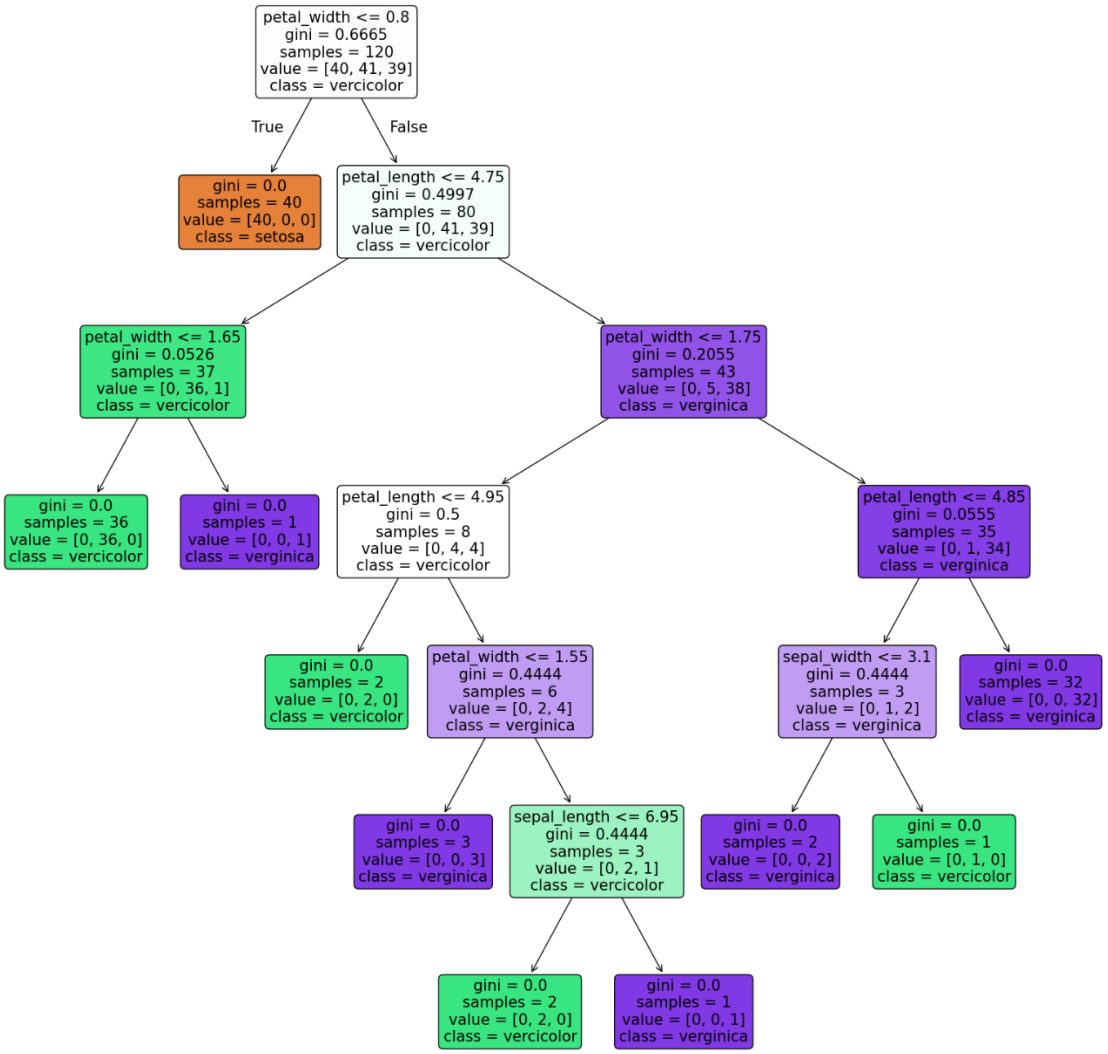
8. *Visualization.*

menampilkan plot dari pohon yang telah dibuat menggunakan perintah berikut.

```
# Visualising the graph without the use of graphviz

plt.figure(figsize = (20,20))
dec_tree = plot_tree(decision_tree=dtree, feature_names = df1.columns,
                     class_names =["setosa", "vercicolor", "verginica"] , filled = True , precision = 4, rounded = True)

plt.savefig("two.png")
```



Kita dapat melihat bagaimana pohon terbagi, nilai GI pada setiap node, jumlah data dalam node tersebut, serta labelnya.