Materi 6: Decision Tree

Pokok Bahasan

Mengenal dan mengimplementasikan Decision Tree

Tujuan

Pada akhir pertemuan ini, diharapkan mahasiswa dapat memahami apa itu Decision Tree dan bagaimana cara mengimplementasikan dengan menggunakan Python.

Penjelasan Singkat

Decision Tree

Decision tree merupakan struktur pohon yang terdiri dari node-node yang merepresentasikan keputusan dan cabang-cabang yang merepresentasikan konsekuensi dari keputusan tersebut. Setiap node dalam decision tree merepresentasikan variabel dalam dataset yang mempengaruhi keputusan dan konsekuensi tersebut.

Decision tree menjadi salah satu model machine learning yang paling populer dan paling sering digunakan dalam problem klasifikasi dan regresi. Digunakan untuk memecah data menjadi subset-subset yang semakin kecil dan homogen hingga didapatkan suatu hasil atau keputusan.

Jenis-jenis decision tree meliputi Classification Tree, Regression Tree, dan Multi-output Tree.

1. Classification tree adalah decision tree yang digunakan untuk memecahkan masalah klasifikasi. Dalam classification tree, variabel target atau dependen merupakan variabel kategorikal. Setiap cabang pada pohon decision tree merepresentasikan suatu keputusan yang dapat menghasilkan prediksi kelas atau label pada data yang diberikan.
2. Regression tree adalah decision tree yang digunakan untuk memecahkan masalah regresi. Dalam regression tree, variabel target atau dependen merupakan variabel kontinu. Setiap cabang pada pohon decision tree merepresentasikan suatu keputusan yang dapat menghasilkan prediksi nilai kontinu pada data yang diberikan.
3. Multi-output tree adalah decision tree yang digunakan untuk memecahkan masalah yang melibatkan lebih dari satu variabel target atau dependen. Multi-output tree dapat digunakan dalam problem klasifikasi maupun regresi, dan seringkali digunakan dalam masalah yang kompleks dan heterogen.

Proses ini dapat dibagi menjadi beberapa tahap. Tahapan tersebut diantaranya meliputi:

1. Pengumpulan dan persiapan data. Pada tahap ini, data yang diperlukan untuk membuat model decision tree dikumpulkan dan dipersiapkan. Data yang diperlukan terdiri dari variabel target atau dependen yang akan diprediksi, dan variabel prediktor atau independen yang digunakan sebagai acuan dalam membuat keputusan. Data yang diberikan dapat berupa data numerik atau kategorikal.
2. Selanjutnya, pembentukan model dengan menggunakan decision tree. pada tahap ini, decision tree dibentuk dari data yang telah dikumpulkan. Proses ini dilakukan dengan menggunakan algoritma decision tree, beberapa algoritma yang dapat digunakan ID3, C4.5, CART, atau CHAID.
3. Setelah decision tree terbentuk, dilakukan pruning atau pemangkasan pada cabang-cabang yang tidak signifikan atau tidak memberikan kontribusi pada prediksi. Proses pruning bertujuan untuk menghindari overfitting atau kelebihan fitting pada model decision tree.
4. Tahapan selanjutnya, model akan dilakukan evaluasi dengan menggunakan data yang belum dipakai selama pembuatan decision tree (data validasi atau testing). Evaluasi dilakukan dengan menghitung akurasi, presisi, recall, F1 score, dan metrik lainnya.
5. Setelah decision tree terbukti memiliki performa yang baik, model decision tree dapat digunakan untuk memprediksi nilai target atau dependen pada data baru.

Kelebihan decision tree yaitu:

- Mudah dipahami
- Cocok untuk data non-linier
- Tidak memerlukan normalisasi data
- Mampu menangani variabel kategorikal
- Dapat digunakan untuk klasifikasi atau regresi
- Mampu menentukan variabel yang paling informatif

Sedangkan kekurangan decision tree yaitu:

- Cenderung overfitting
- Tidak stabil terhadap perubahan data
- Tidak mampu menangani data kontinu
- Sensitif terhadap noise
- Memerlukan tuning parameter

Modul 1: Membuat model dengan Decision Tree

Pengumpulan dan persiapan data

Data yang digunakan dalam membuat model dengan Decision Tree disini yakni data yang diambil dari data Klasifikasi Bunga, selengkapnya informasi data dapat dilihat pada tautan ini: https://en.wikipedia.org/wiki/Iris_flower_data_set

Terdapat 150 data yang diambil pada data ini diantaranya terbagi atas beberapa jenis:

1. Iris setosa
2. Iris versicolor
3. Iris virginica

Untuk dapat melihat dataset tersebut anda dapat mencoba dengan menggunakan langkah-langkah berikut ini:

1. Buatlah satu buah project baru dan lakukan load library berikut ini:

```
# %%
# Loading library
from sklearn import datasets
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Kemudian lakukan load data sehingga data tersebut diubah formatnya kedalam dataframe seperti berikut ini. Disini anda akan meihat sebaran data dari setiap data set yang berjumlah 150 data dan dihitung secara statistik mulai dari mean, standar deviasi, min, max, dan sebagainya.

```
# %%
# Load data as a dataframe
def sklearn_to_df(sklearn_dataset):
    df = pd.DataFrame(sklearn_dataset.data, columns=sklearn_dataset.feature_names)
    df['target'] = pd.Series(sklearn_dataset.target)
    return df

iris = sklearn_to_df(datasets.load_iris())
iris.rename(columns={'target':'species'},inplace=True)

iris.describe().T
```

	count	mean	std	min	25%	50%	75%	max
sepal length (cm)	150.0	5.843333	0.828066	4.3	5.1	5.80	6.4	7.9
sepal width (cm)	150.0	3.057333	0.435866	2.0	2.8	3.00	3.3	4.4
petal length (cm)	150.0	3.758000	1.765298	1.0	1.6	4.35	5.1	6.9
petal width (cm)	150.0	1.199333	0.762238	0.1	0.3	1.30	1.8	2.5
species	150.0	1.000000	0.819232	0.0	0.0	1.00	2.0	2.0

3. Selanjutnya untuk memastikan data tersebut sesuai dan anda ingin melihat data tersebut kita dapat menuliskan kode berikut untuk menampilkan dataset.

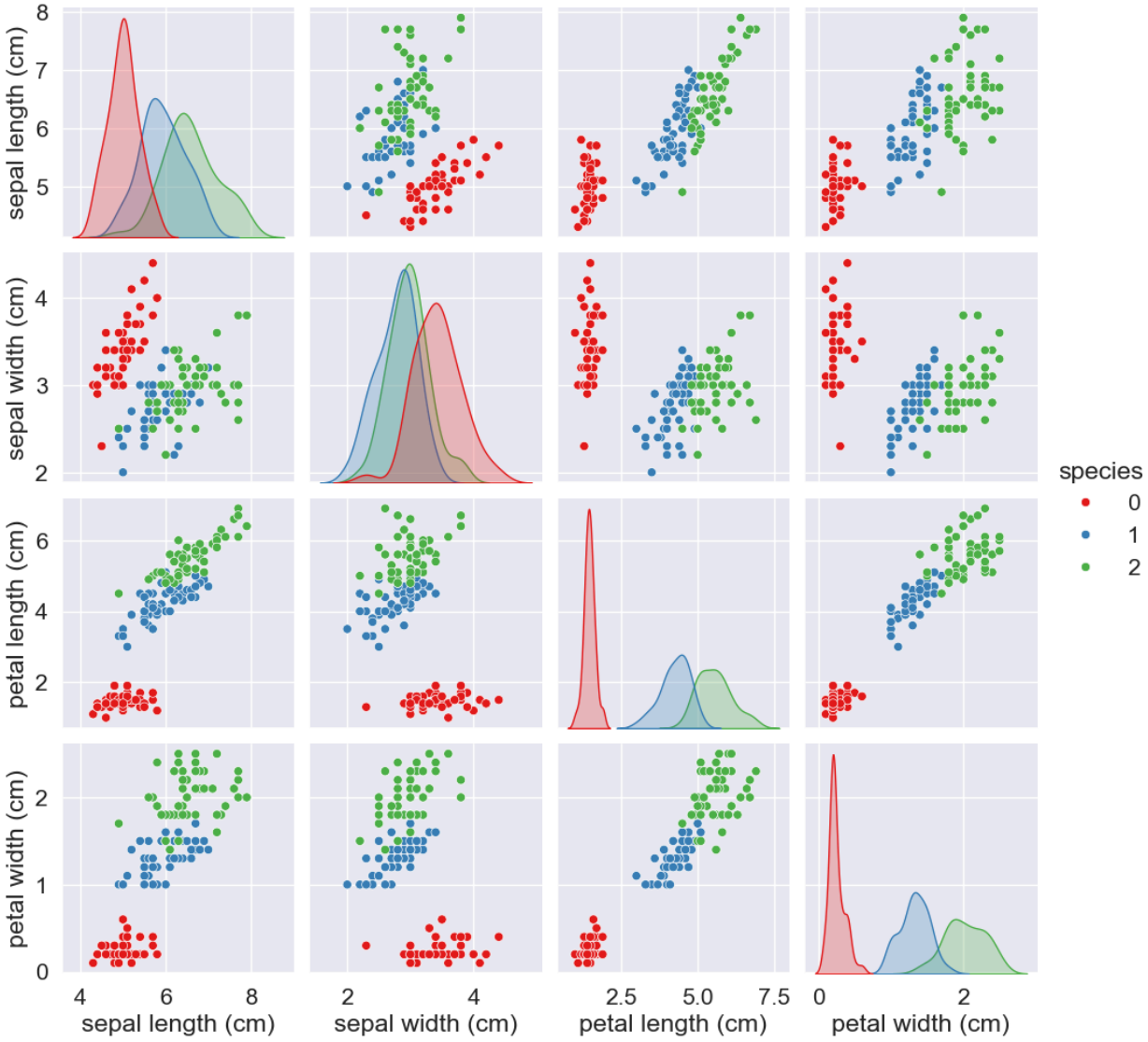
```
# %%
# Show the data
iris.head(10)
print(iris)
```

```
✓ # Show the data ···

     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                  5.1               3.5                1.4               0.2
1                  4.9               3.0                1.4               0.2
2                  4.7               3.2                1.3               0.2
3                  4.6               3.1                1.5               0.2
4                  5.0               3.6                1.4               0.2
..                 ...               ...                ...               ...
145                6.7               3.0                5.2               2.3
146                6.3               2.5                5.0               1.9
147                6.5               3.0                5.2               2.0
148                6.2               3.4                5.4               2.3
149                5.9               3.0                5.1               1.8

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

[150 rows x 5 columns]
```

Melakukan pembagian data (Training dan Testing)

1. Sebelum melakukan pembagian data, kita dapat menampilkan grafik dari data tersebut menggunakan perintah berikut ini:

   ```
   # %%
   # Visualisasi data dengan Grafik pada data Iris
   sns.pairplot(iris,hue='species',palette='Set1')
   ```



2. Seperti pada pertemuan sebelumnya kita dapat melakukan pembagian data dengan tujuan menggunakan data training yang dipergunakan untuk membuat model. Pembagian data yang digunakan yaitu 70% data training dan 30% data testing. Adapun tahapannya adalah dengan menambahkan kode program berikut:

```
# %%
# Split training and testing data
from sklearn.model_selection import train_test_split
x = iris.drop('species', axis = 1)
y = iris['species']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 10)
```

3. Anda dapat menambahkan perintah print(len(x_train)) untuk melihat jumlah dari data training yang digunakan.

Membuat Report Hasil Klasifikasi

Adapun report hasil klasifikasi dapat ditunjukkan dengan melakukan pengukuran pada nilai Precision, Recall, F1 Score yang dapat ditunjukkan dengan menuliskan kode program berikut ini.

```
# %%
# Classification report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Berikut hasil dari classification report yang dihasilkan:

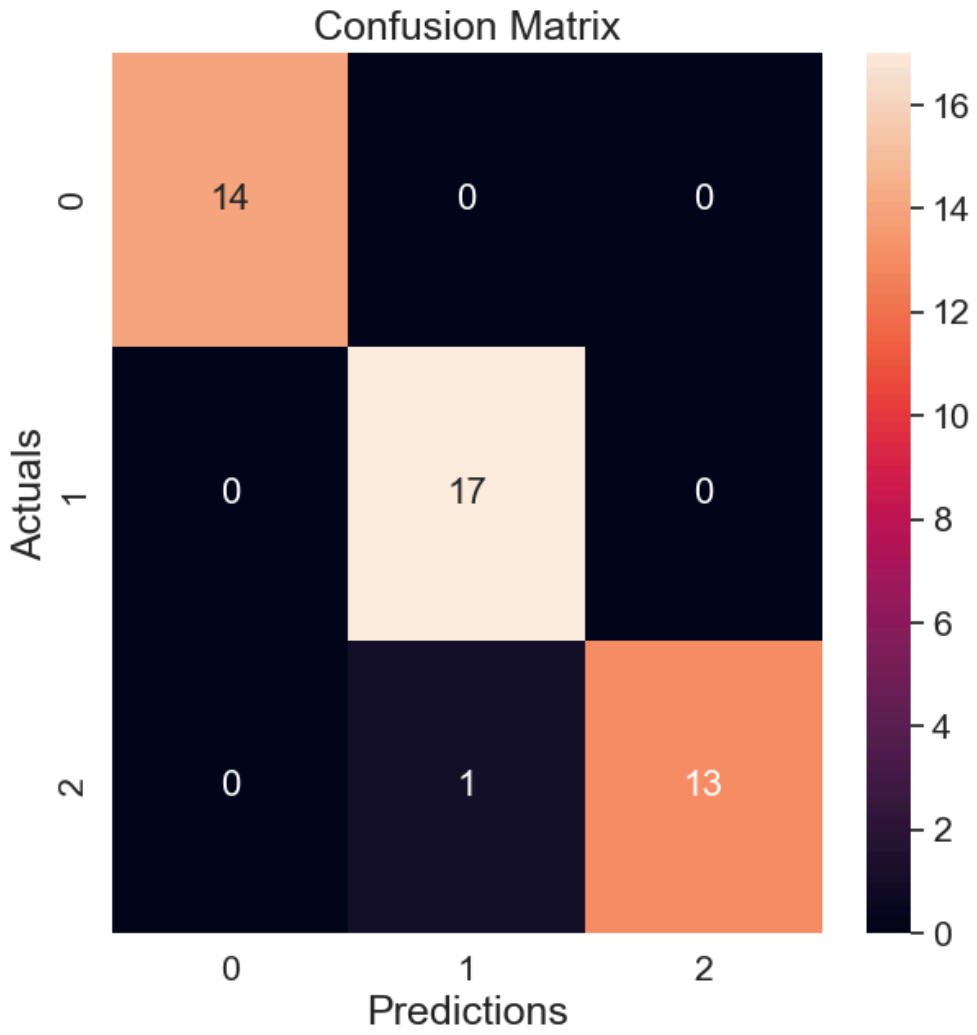
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.94      1.00      0.97        17
           2       1.00      0.93      0.96        14

    accuracy                           0.98        45
   macro avg       0.98      0.98      0.98        45
weighted avg       0.98      0.98      0.98        45
```

Melakukan Evaluasi

Selain itu untuk melakukan evaluasi terhadap model, kita dapat memperlihatkan confusion matrix yang dihasilkan dengan melakukan visualisasi dan tambahkan kode berikut ini.

```
#%%
# Confusion matrix
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

```
fig, ax = plt.subplots(figsize=(7,7))

sns.set(font_scale=1.4) # for label size
sns.heatmap(cm, ax=ax,annot=True, annot_kws={"size": 16}) # font size

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Confusion Matrix', fontsize=18)
plt.show()
```

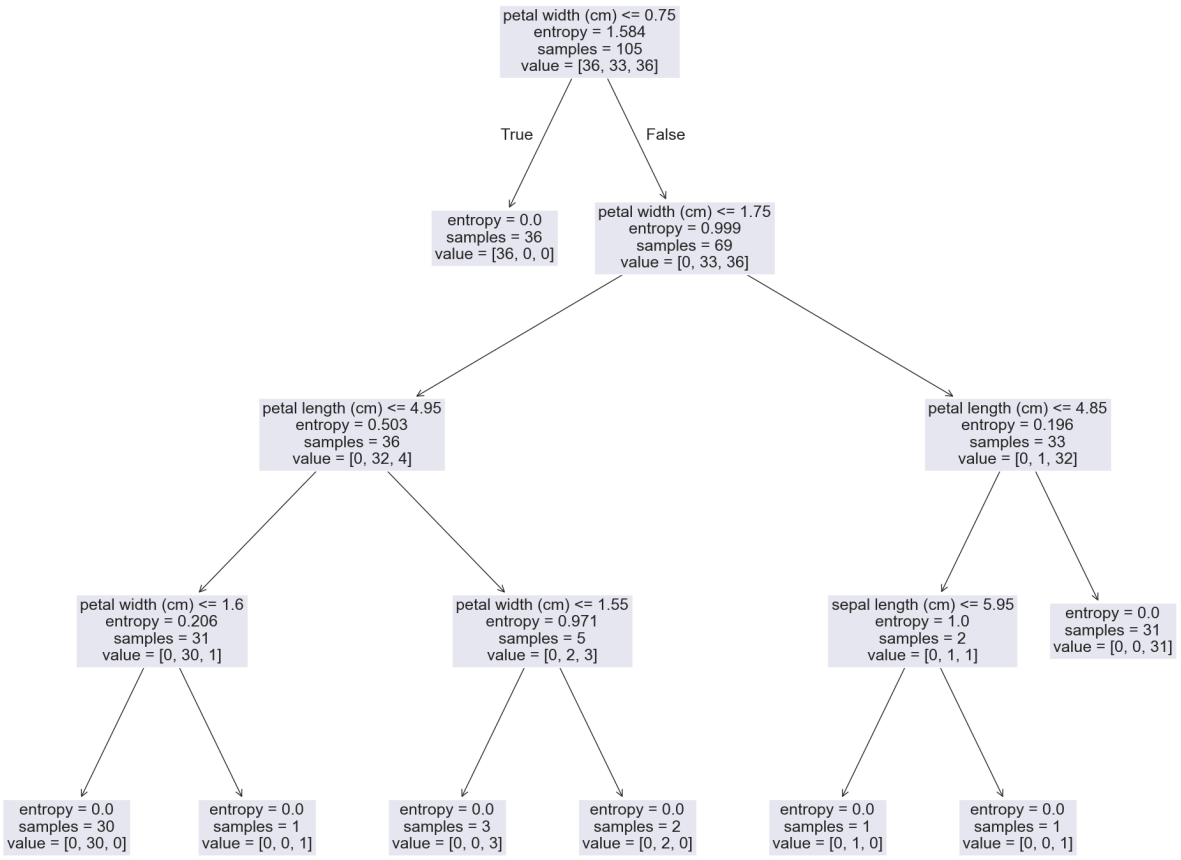


Hasil tersebut menunjukkan bahwa model sudah cukup baik dalam melakukan prediksi, namun terdapat kondisi dimana model tidak dapat mengenali iris dengan baik yang ditunjukkan bahwa model mengenali kelas (2): irisi virginica padahal seharusnya model diklasifikasi sebagai (1): iris versicolor.

Menampilkan visualisasi Tree (Decision Tree)

Bagian ini untuk menunjukkan bagaimana tree/ pohon dalam melakukan klasifikasi, anda dapat menggunakan kode program ini.

```
#%%
features = ['sepal length (cm)','sepal width (cm)', 'petal length (cm)', 'petal width (cm)' ]

#%%
# Visualize tree
from sklearn import tree
fig, ax = plt.subplots(figsize=(25,20))
tree.plot_tree(model, feature_names=features)
plt.show()
```



Ujicoba data diluar Data Testing

Kemudian silahkan uji data berikut dan apa keluaran output dari model yang dihasilkan dan tambahkan kode berikut:

```
# %%
# Example of creating a single Iris data point as a dictionary
iris_test_data = {
    'sepal length (cm)': 5.1,
    'sepal width (cm)': 3.5,
    'petal length (cm)': 1.4,
    'petal width (cm)': 0.1
}

# Ensure the order of features matches the training data
feature_order = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
prediction_input_df = pd.DataFrame([iris_test_data])
prediction = model.predict(prediction_input_df[feature_order]) # Ensure correct column order
print(prediction)
```

Latihan

Cobalah cari dataset lain dan lakukan hal yang sama seperti yang dilakukan pada praktikum ini. Jika sudah kumpulkan kode programnya dalam repository Github masing-masing

Referensi

https://dqlab.id/apa-itu-decision-tree-di-machine-learning-model
https://www.jeffthurk.com/ml_lecture05_decisiontrees_finished