**Pokok Bahasan IV**

**Decision Tree with Python**

**Kode Pokok Bahasan**: TIK.RPL03.001.003.01

**Deskripsi Pokok Bahasan**:

Membahas bagaimana pembuatan Decision Tree pada dataset yang diberikan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Elemen Kompetensi | Indikator Kinerja | Jml Jam | Hal |
| 1 | Menerapkan Decision Tree pada data Weather Nominal dataset 1. | 1.1 Membuat Decision Tree menggunakan Library Scikit-Learn menggunakan data Weather Nominal | 1 | 12 |
| 2 | Menerapkan prediksi dan class pada Decision Tree data Weather Nominal dataset 2. | 1.1 Membuat Decision Tree menggunakan Library Scikit-Learn menggunakan data Weather Nominal  1.2 Membuat prediksi dari Decision Tree  1.3 Membuat class dari prediksi | 2 | 15 |

**TUGAS PENDAHULUAN**

Hal yang harus dilakukan dan acuan yang harus dibaca sebelum praktikum :

Menginstal Anaconda Python pada PC masing-masing praktikan.

2. Menginstal Jupyter Notebook pada Anaconda Python pada PC masing-masing praktikan.

**DAFTAR PERTANYAAN**

1. Apa itu decision tree?
2. Apa kegunaan Decision tree?
3. Adakah perbedaan dalam membuat decision tree antara R dan Python, berikan alasannya!

1. Decision tree merupakan suatu struktur yang digunakan untuk membantu proses pengambilan keputusan. Disebut sebagai “tree” karena struktur ini menyerupai sebuah pohon lengkap dengan akar, batang, dan percabangannya.

2. Manfaat utama dari penggunaan decision tree adalah kemampuannya untuk mem-break down proses pengambilan keputusan yang kompleks menjadi lebih simple, sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan.

3. Perbedaan yang sangat terlihat dari kedua bahasa ini adalah R digunakan untuk analisis statistik, sedangkan Python lebih general. Python adalah bahasa multipurpose, sama seperti C++ dan Java. Penggunaannya pun cenderung lebih mudah dipelajari, tak seperti R yang lebih kompleks.

**TEORI SINGKAT**

Decision tree adalah salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia. Decision tree adalah model prediksi menggunakan struktur pohon atau struktur berhirarki.

Konsep dari pohon keputusan adalah mengubah data menjadi decision tree dan aturan-aturan keputusan. Manfaat utama dari penggunaan decision tree adalah kemampuannya untuk mem-break down proses pengambilan keputusan yang kompleks menjadi lebih simple, sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan.

Salah satu algoritma dari decision tree adalah CART (Classification and Regression Tree). Dimana metode ini merupakan gabungan dari dua jenis pohon, yaitu classification tree dan juga regression tree. Untuk memudahkan, berikut ilustrasi dari keduanya.

**LAB SETUP**

Hal yang harus disiapkan dan dilakukan oleh praktikan untuk menjalankan praktikum modul ini.

1. Menginstall library yang dibutuhkan untuk mengerjakan modul.

2. Menjalankan R Studio.

**ELEMEN KOMPETENSI I**

**Deskripsi:**

Menerapkan Decision Tree pada data Weather Nominal dataset 1.

**Kompetensi Dasar**:

Membuat Decision Tree menggunakan Library Scikit-Learn menggunakan data Weather Nominal.

**Latihan 1.1.1**

**Penjelasan Singkat :**

Pada latihan ini anda akan diminta untuk membangun decision tree menggunakan library yang disediakan oleh Python.

**Langkah-Langkah Praktikum:**

1. Buat file baru pada Jupyter Notebook atau Google Colab <https://colab.research.google.com/>
2. Letakkan file data yang digunakan pada direktori yang sama. Jika menggunakan Google Colab, upload data terlebih dahulu session storage.

Klik upload to session storage seperti pada gambar di bawah.

Maka akan muncul seperti di bawah ini, pilihlah data prak4.csv pada penyimpanan anda, kemudian klik open.

Jika sudah makan akan muncul pada side bar seperti gambar berikut.

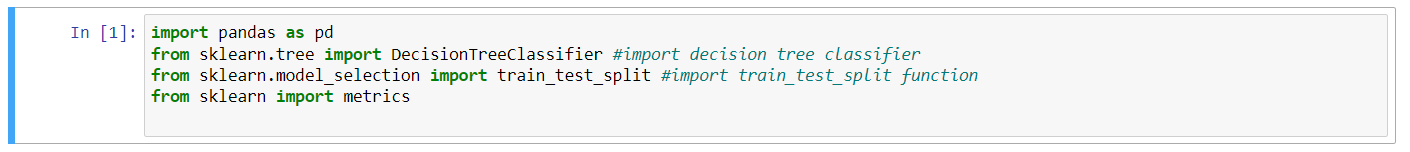
1. Import library python yang akan digunakan.

import pandas as pd

from sklearn.tree import DecisionTreeClassifier #import decision tree classifier

from sklearn.model\_selection import train\_test\_split #import train\_test\_split function

from sklearn import metrics



1. Membaca data prak4.csv dengan metode read\_csv

data\_namapraktikan = pd.read\_csv(“prak4.csv”)

data\_namapraktikan.head()

Graphical user interface, text, application

Description automatically generated

1. Mengecek data secara umum

data\_namapraktikan.info()

Text, table

Description automatically generated with medium confidence

1. Pada saat harus bekerja dengan kolom atau atribut bertipe kolom, maka data harus diubah dalam format numerik. Pada library sklearn.preprocessing tersedia dua metode untuk mengubah tipe data kategorikal menjadi data numerik yaitu label encoding dan one hot encoding.
2. Import library sklearn untuk melakukan label encoding dan onehot encoding. Kemudian melakukan datapreprocessing menggunakan label encoding dan one hot encoding.

from numpy import array

from numpy import argmax

from sklearn.preprocessing import LabelEncoder

from sklearn.preprocessing  import OneHotEncoder

print(data\_namapraktikan.columns)

fiturs = ['cuaca', 'suhu', 'kelembaban', 'berangin']

labels\_encoder = LabelEncoder()

jobs\_encoder = OneHotEncoder(sparse=False)

data\_namapraktikan1 = data\_namapraktikan.copy(deep=True)

for fitur in fiturs:

    print(f'{fitur}:{data\_namapraktikan1[fitur].unique()}')

    values = array(data\_namapraktikan1[fitur])

    integer\_encoded = labels\_encoder.fit\_transform(values)

    transformed = jobs\_encoder.fit\_transform(integer\_encoded.reshape(len(integer\_encoded), 1))

    ohe\_df = pd.DataFrame(transformed,columns=jobs\_encoder.get\_feature\_names())

    data\_namapraktikan1 = pd.concat([data\_namapraktikan1, ohe\_df], axis=1).drop([fitur], axis=1)

data\_namapraktikan1.head()

**A picture containing calendar

Description automatically generated**

1. Mengubah label pada kolom bermain dengan label encoder.

integer\_encoded = labels\_encoder.fit\_transform(data\_namapraktikan1['bermain'])

main = pd.DataFrame(integer\_encoded,columns=['label\_bermain',])

data\_namapraktikan1 = pd.concat([data\_namapraktikan1, main], axis=1).drop(['bermain'], axis=1)

data\_namapraktikan1.head()

**Text

Description automatically generated**

1. Menyiapkan fitur untuk prediksi dan label klasifikasi.

Data untuk prediksi yaitu data pada dataset kecuali kolom “bermain”. Data untuk label klasifikasi yaitu data pada kolom “bermain”.

y = data\_namapraktikan1['label\_bermain']

x = data\_namapraktikan1.drop(columns=['label\_bermain'])

print(f'variabel prediktor:{x.columns}')

print(f'label klasifikasi:{y.name}')

**Graphical user interface, text, application

Description automatically generated**

1. Membagi dataset menjadi data training dan data testing.

Data training = X\_train dan y\_train

Data testing = x\_test dan y\_test

Data training : data testing = 70 : 30 sehingga test\_size = 0.3

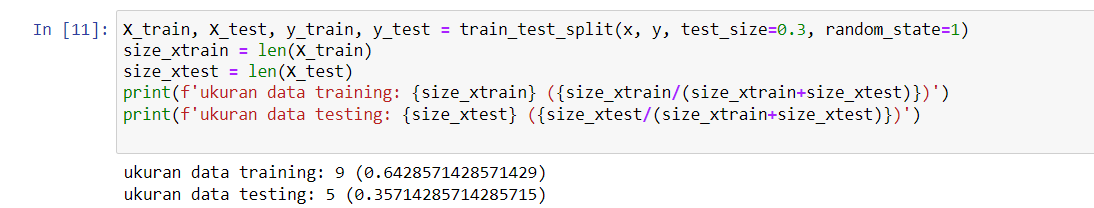
X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3, random\_state=1)

size\_xtrain = len(X\_train)

size\_xtest = len(X\_test)

print(f'ukuran data training: {size\_xtrain} ({size\_xtrain/(size\_xtrain+size\_xtest)})')

print(f'ukuran data testing: {size\_xtest} ({size\_xtest/(size\_xtrain+size\_xtest)})')



1. Melakukan split training dan testing model dengan membuat klasifier.

clf\_namapraktikan = DecisionTreeClassifier()

# proses training model

clf\_namapraktikan = clf\_namapraktikan.fit(X\_train,y\_train)

#proses testing model

y\_pred = clf\_namapraktikan.predict(X\_test)

Text

Description automatically generated

1. Membuat confusion matrix.

from sklearn.metrics import confusion\_matrix

tn, fp, fn, tp = confusion\_matrix(y\_test, y\_pred).ravel()

print(f'confusion matrix')

print(f'{tp} | {fp}')

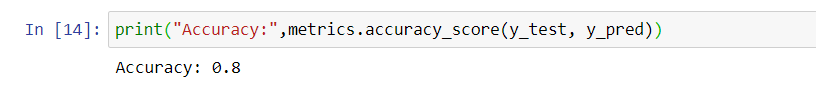
print(f'{fn} | {tn}')

Graphical user interface, text, application

Description automatically generated

1. Melihat keakuratan model

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))



1. Menampilkan decision tree

import numpy as np

from matplotlib import pyplot as plt

from sklearn import tree

n\_nodes = clf\_namapraktikan.tree\_.node\_count

children\_left = clf\_namapraktikan.tree\_.children\_left

children\_right = clf\_namapraktikan.tree\_.children\_right

feature = clf\_namapraktikan.tree\_.feature

threshold = clf\_namapraktikan.tree\_.threshold

node\_depth = np.zeros(shape=n\_nodes, dtype=np.int64)

is\_leaves = np.zeros(shape=n\_nodes, dtype=bool)

stack = [(0, 0)]  # start with the root node id (0) and its depth (0)

while len(stack) > 0:

    # `pop` ensures each node is only visited once

    node\_id, depth = stack.pop()

    node\_depth[node\_id] = depth

    # If the left and right child of a node is not the same we have a split

    # node

    is\_split\_node = children\_left[node\_id] != children\_right[node\_id]    # If a split node, append left and right children and depth to `stack`

    # so we can loop through them

    if is\_split\_node:

        stack.append((children\_left[node\_id], depth + 1))

        stack.append((children\_right[node\_id], depth + 1))

    else:

        is\_leaves[node\_id] = True

print("The binary tree structure has {n} nodes and has "

      "the following tree structure:\n".format(n=n\_nodes))

for i in range(n\_nodes):

    if is\_leaves[i]:

        print("{space}node={node} is a leaf node.".format(

            space=node\_depth[i] \* "\t", node=i))

    else:

        print("{space}node={node} is a split node: "

              "go to node {left} if X[:, {feature}] <= {threshold} "

              "else to node {right}.".format(

                  space=node\_depth[i] \* "\t",

                  node=i,

                  left=children\_left[i],

                  feature=feature[i],

                  threshold=threshold[i],

                  right=children\_right[i]))

1. Menampilkan plot decision tree

tree.plot\_tree(clf\_namapraktikan)

plt.show()

Text

Description automatically generated

Text

Description automatically generated

Qr code

Description automatically generated

1. Melakukan klasifikasi berdasarkan entropi

clf1 = DecisionTreeClassifier(criterion="entropy")

# Train Decision Tree Classifer

clf1.fit(X\_train,y\_train)

#Predict the response for test dataset

y\_pred = clf1.predict(X\_test)

tn, fp, fn, tp = confusion\_matrix(y\_test, y\_pred).ravel()

print(f'confusion matrix')

print(f'{tp} | {fp}')

print(f'{fn} | {tn}')

#menampilkan akurasi model

print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

Graphical user interface, text, application

Description automatically generated

**Interpretasi output decision tree :**

|  |
| --- |
| Interpretasi nya adalah jika nilai x lebih dari 0 maka akan dipisahkan, seterusnya akan begitu sampai nilai x tidak lebih dari 0, nilai gini index juga ketika sudah mulai dibagi, maka nilainya akan terus berkurang sampai terbagi rata. |

**ELEMEN KOMPETENSI II**

**Deskripsi:**

Menerapkan prediksi menggunakan Decision Tree pada data Weather Nominal

dataset 2.

**Kompetensi Dasar**:

Membuat prediksi pada Tree menggunakan data Weather Nominal dataset 2.

**Latihan 1.2.1**

**Penjelasan Singkat :**

Pada latihan ini anda akan diminta untuk membangun decision tree dan melakukan prediksi menggunakan library yang disediakan oleh python.

**Langkah-Langkah Praktikum:**

**1.** Disediakan data sebagai berikut :

Link dataset : <https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset>

**2.** Buatlah decision tree untuk dataset di atas menggunakan langkah-langkah pada elemen kompetensi 1. Tampilkan langkah-langkah pembuatannya beserta hasil decision tree.

A picture containing table

Description automatically generated

Table

Description automatically generated with medium confidence

Graphical user interface, text, application

Description automatically generated

Calendar

Description automatically generated with low confidence

Graphical user interface, application

Description automatically generated with medium confidence

Graphical user interface, application, Word

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Scatter chart

Description automatically generated

Arrow

Description automatically generated with low confidence

Graphical user interface, text, application

Description automatically generated

3. Berikan penjelasan tentang output yang muncul!

**Interpretasi output decision tree :**

|  |
| --- |
| Interpretasi nya adalah jika nilai x lebih dari 0 maka akan dipisahkan, seterusnya akan begitu sampai nilai x tidak lebih dari 0, nilai gini index juga ketika sudah mulai dibagi, maka nilainya akan terus berkurang sampai terbagi rata. |

**CEK LIST**

|  |  |  |  |
| --- | --- | --- | --- |
| Elemen Kompetensi | No Latihan | Penyelesaian | |
| Selesai | Tidak selesai |
| 1 | 1.1.1 |  |  |
| 2 | 1.2.1 |  |  |

**FORM UMPAN BALIK**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Elemen Kompetensi** | **Tingkat Kesulitan** | | | **Tingkat Ketertarikan** | | | **Waktu Penyelesaian dalam menit** |
| Memahami decision tree |  |  |  |  |  |  |  |
|  |  | Sangat Mudah |  |  | Tidak Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Mudah |  |  | Cukup Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | * Biasa |  |  | Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sulit |  |  | * Sangat Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sangat Sulit |  |  |  |  |
|  |  |  |  |  |  |  |
| Mengimplementasikan decision tree. |  |  |  |  |  |  |  |
|  |  | Sangat Mudah |  |  | Tidak Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Mudah |  |  | Cukup Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | * Biasa |  |  | Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sulit |  |  | * Sangat Tertarik |  |
|  |  |  |  |  |  |  |
|  |  | Sangat Sulit |  |  |  |  |
|  |  |  |  |  |  |  |