JOBSHEET 14

Decision Tree
(Pohon Keputusan)

Tujuan

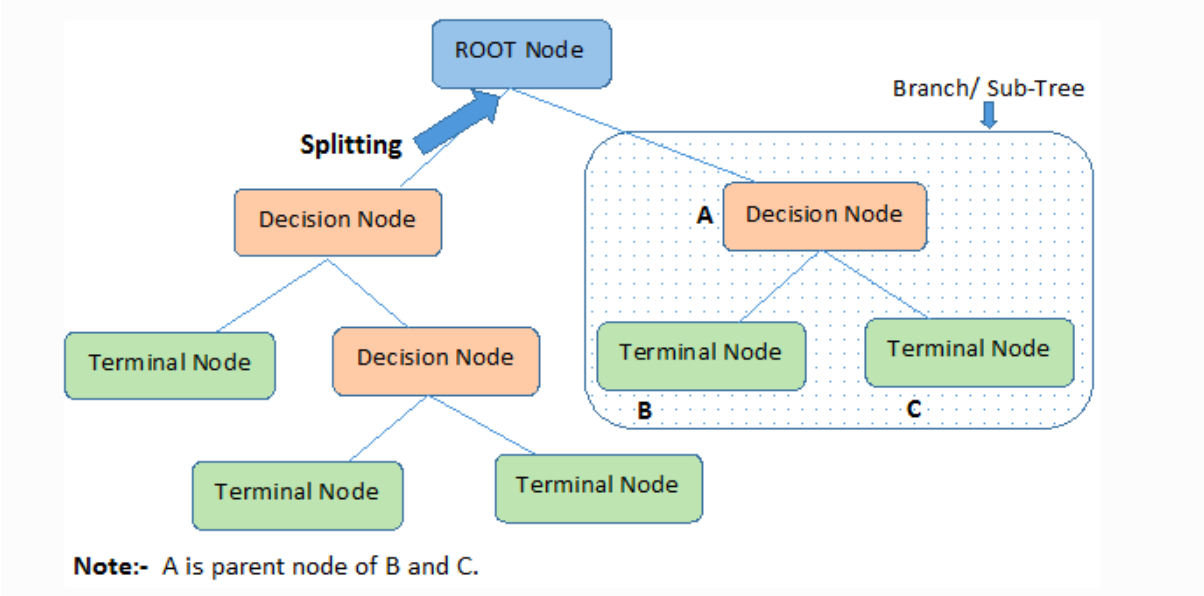
Mahasiswa diharapkan dapat:

- Mahasiswa memahami cara kerja decision tree.
- Mahasiswa dapat mengetahui dan mengenal contoh studi kasus yang dapat diselesaikan dengan metode decision tree
- Mahasiswa mampu menerapkan metode decision tree menggunakan bahasa Python

Ringkasan Materi

1. Definisi Decision Tree

Alat pendukung keputusan yang menggunakan model keputusan seperti pohon. Istilah pada decision tree adalah seperti berikut:



- **Root Node** : merepresentasikan seluruh populasi
- **Leaf Node/terminal node** : node yang tidak dapat dipisahkan lagi menjadi node yang lain

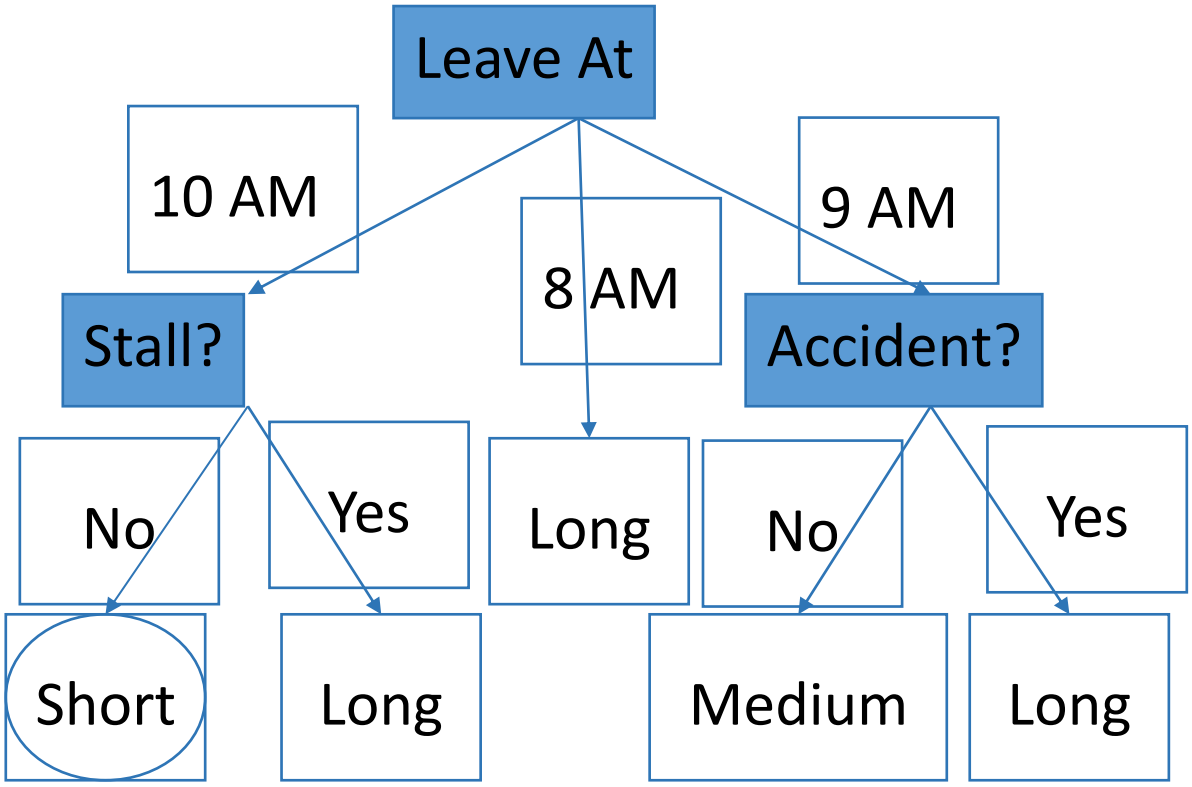
- **Parent/Child node** : root node adalah parent node. Dan semua cabangnya adalah child node
- **Splitting** : Memisah root node/sub node menjadi bagian yang berbeda
- **Branch/Subtree** : Dibentuk dengan splitting tree/node
- **Pruning** : menghapus branch dari tree (kebalikan dari splitting)

Proses pembentukan decision tree menggunakan beberapa algoritma ID3, C4.5, CHART, CHAID, MARS. Pada Jobsheet ini akan forkus menggunakan ID3 untuk pembentukan decision tree. Langkah algoritma decision tree adalah seperti berikut:

1. Tree dimulai dengan sebuah simpul yang merepresentasikan sampel data pelatihan yaitu dengan membuat node
2. Pada setiap iterasi pada atribut yang tidak digunakan hitung entropi dan information gain
3. Memilih atribut yang memiliki entropi terkecil atau information gain terbesar
4. kemudian sample data displit berdasarkan atribut terpilih
5. Algoritma terus berulang untuk setiap subset, memproses atribut yang belum dipilih pada langkah sebelumnya

2. Contoh Decision Tree

Representasi Pengetahuan secara logika untuk tree di bawah ini adalah sebagai berikut:



Dari tree di atas kita dapat membuat keputusan seperti berikut:

- Jika kita berangkat (leave) jam 10 pagi dan tidak ada mobil yang mogok (stall) di jalan, berapa waktu perjalanan kita? Short

PRAKTIKUM

1) Pada percobaan yang menerapkan decision tree dengan bahasa python ini, mula-mula kita mengimport library yang digunakan:

```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log
```

2) Setelah itu kita Mendefinisikan dataset yang akan digunakan:

```
outlook = 'overcast,overcast,overcast,overcast,rainy,rainy,rainy,rainy,rainy,sunny,sunny,sunny,sunny,sunny'.split(',')
temp = 'hot,cool,mild,hot,mild,cool,cool,mild,mild,hot,hot,mild,cool,mild'.split(',')
humidity = 'high,normal,high,normal,high,normal,normal,normal,high,high,high,high,normal,normal'.split(',')
windy = 'FALSE,TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,FALSE,FALSE,TRUE'.split(',')
play = 'yes,yes,yes,yes,yes,yes,no,yes,no,no,no,no,yes,yes'.split(',')
```

3) Membuat dataframe

```
dataset ={'outlook':outlook,'temp':temp,'humidity':humidity,'windy':windy,'play':play}
df = pd.DataFrame(dataset,columns=['outlook','temp','humidity','windy','play'])
```

	outlook	temp	humidity	windy	play
0	overcast	hot	high	FALSE	yes
1	overcast	cool	normal	TRUE	yes
2	overcast	mild	high	TRUE	yes
3	overcast	hot	normal	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	rainy	mild	normal	FALSE	yes
8	rainy	mild	high	TRUE	no
9	sunny	hot	high	FALSE	no
10	sunny	hot	high	TRUE	no
11	sunny	mild	high	FALSE	no
12	sunny	cool	normal	FALSE	yes
13	sunny	mild	normal	TRUE	yes

4) Setelah itu mencari entropi suatu kejadian

```
##1. claculate entropy o the whole dataset

entropy_node = 0  #Initialize Entropy
values = df.play.unique()  #Unique objects - 'Yes', 'No'
for value in values:
    fraction = df.play.value_counts()[value]/len(df.play)
    entropy_node += -fraction*np.log2(fraction)
entropy_node
```

```
0.9402859586706311
```

5) Setelah itu mencari entropi untuk setiap atribut

Menggunakan rumus:

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

```
def ent(df,attribute):
    target_variables = df.play.unique()  #This gives all 'Yes' and 'No'
    variables = df[attribute].unique()    #This gives different features in that attribute (like 'Sweet')


    entropy_attribute = 0
    for variable in variables:
        entropy_each_feature = 0
        for target_variable in target_variables:
            num = len(df[attribute][df[attribute]==variable][df.play ==target_variable]) #numerator
            den = len(df[attribute][df[attribute]==variable])  #denominator
            fraction = num/(den+eps)  #pi
            entropy_each_feature += -fraction*log(fraction+eps) #This calculates entropy for one feature like 'Sweet'
        fraction2 = den/len(df)
        entropy_attribute += -fraction2*entropy_each_feature   #Sums up all the entropy ETaste

    return(abs(entropy_attribute))
```

Menyimpan entropi untuk setiap atribut

```
a_entropy = {k:ent(df,k) for k in df.keys()[:-1]}
a_entropy
```

```
{'outlook': 0.6935361388961914,
 'temp': 0.9110633930116756,
 'humidity': 0.7884504573082889,
 'windy': 0.892158928262361}
```

6) Menghitung information gain untuk setiap atribut

Menggunakan rumus :

$$Information\ Gain\ =\ Entropy(before) - \sum_{j=1}^{K} Entropy(j,\ after)$$

```
def ig(e_dataset,e_attr):
    return(e_dataset-e_attr)
```

Menyimpan information gain untuk setiap atribut di dict

```
#entropy_node = entropy of dataset
#a_entropy[k] = entropy of k(th) attr
IG = {k:ig(entropy_node,a_entropy[k]) for k in a_entropy}
IG
```

```
{'outlook': 0.24674981977443977,
 'temp': 0.029222565658955535,
 'humidity': 0.15183550136234225,
 'windy': 0.048127030408270155}
```

7) adalah code yang sudah complete

```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log

outlook = 'overcast,overcast,overcast,overcast,rainy,rainy,rainy,rainy,rainy,sunny,sunny,sunny,sunny,sunny'.split(',')
temp = 'hot,cool,mild,hot,mild,cool,cool,mild,mild,hot,hot,mild,cool,mild'.split(',')
humidity = 'high,normal,high,normal,high,normal,normal,normal,high,high,high,high,normal,normal'.split(',')
windy = 'FALSE,TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,FALSE,FALSE,TRUE'.split(',')
play = 'yes,yes,yes,yes,yes,yes,no,yes,no,no,no,no,yes,yes'.split(',')

dataset ={'outlook':outlook,'temp':temp,'humidity':humidity,'windy':windy,'play':play}
df = pd.DataFrame(dataset,columns=['outlook','temp','humidity','windy','play'])

def find_entropy(df):
    Class = df.keys()[-1]   #To make the code generic, changing target variable class name
    entropy = 0
    values = df[Class].unique()
    for value in values:
        fraction = df[Class].value_counts()[value]/len(df[Class])
        entropy += -fraction*np.log2(fraction)
    return entropy
```

```
def find_entropy_attribute(df,attribute):
  Class = df.keys()[-1]   #To make the code generic, changing target variable class name
  target_variables = df[Class].unique()  #This gives all 'Yes' and 'No'
  variables = df[attribute].unique()    #This gives different features in that attribute (like 'Hot','Cold' in Temperature)
  entropy2 = 0
  for variable in variables:
      entropy = 0
      for target_variable in target_variables:
          num = len(df[attribute][df[attribute]==variable][df[Class] ==target_variable])
          den = len(df[attribute][df[attribute]==variable])
          fraction = num/(den+eps)
          entropy += -fraction*log(fraction+eps)
      fraction2 = den/len(df)
      entropy2 += -fraction2*entropy
  return abs(entropy2)

def find_winner(df):
    Entropy_att = []
    IG = []
    for key in df.keys()[:-1]:
#         Entropy_att.append(find_entropy_attribute(df,key))
        IG.append(find_entropy(df)-find_entropy_attribute(df,key))
    return df.keys()[:-1][np.argmax(IG)]

def get_subtable(df, node,value):
  return df[df[node] == value].reset_index(drop=True)
```

```
def buildTree(df,tree=None):
    Class = df.keys()[-1]   #To make the code generic, changing target variable class name
    #Get attribute with maximum information gain
    node = find_winner(df)
    #Get distinct value of that attribute e.g Salary is node and Low,Med and High are values
    attValue = np.unique(df[node])
    #Create an empty dictionary to create tree
    if tree is None:
        tree={}
    |   tree[node] = {}

   #We make loop to construct a tree by calling this function recursively.
    #In this we check if the subset is pure and stops if it is pure.

    for value in attValue:

        subtable = get_subtable(df,node,value)
        clValue,counts = np.unique(subtable[Class],return_counts=True)

        if len(counts)==1:#Checking purity of subset
            tree[node][value] = clValue[0]
        else:
            tree[node][value] = buildTree(subtable) #Calling the function recursively

    return tree
```

8) Cara untuk menggunakan code tersebut adalah

Pertama lakukan instalasi pprintpp

```
pip install pprintpp
```

```
Collecting pprintpp
  Downloading pprintpp-0.4.0-py2.py3-none-any.whl (16 kB)
Installing collected packages: pprintpp
Successfully installed pprintpp-0.4.0
Note: you may need to restart the kernel to use updated packages.
```

Laku gunakan code berikut ini

```
t  = buildTree(df)
import pprint
pprint.pprint(t)
```

```
{'outlook': {'overcast': 'yes',
             'rainy': {'windy': {'FALSE': 'yes', 'TRUE': 'no'}},
             'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}}}
```

9) Hasil akhir decision tree adalah seperti berikut