Tubes 1A Pembelajaran Mesin

Eksplorasi Variasi Algoritma *Decision Tree Learning*

Anggota Kelompok

1. Ahmad Mutawalli - 13517026, K02
2. Harry Rahmadi Munly - 13517033, K03
3. Ardysatrio Fakhri Haroen - 13517062, K02
4. Dandi Agus Maulana - 13517077, K02

In [36]:
```
#Load utility libraries
from sklearn.datasets import load_iris
import pandas as pd
from sklearn.tree import export_text as et_dtl
from id3 import export_text as et_id3
from sklearn.preprocessing import LabelEncoder

#Load ML libraries
from id3 import Id3Estimator
from sklearn.tree import DecisionTreeClassifier
```

Membaca Data

In [25]:
```
#read iris data
data, target = load_iris(return_X_y=True)
iris_data = pd.DataFrame(data, columns=['sepal_length', 'sepal_width', 'petal_le
iris_data['label'] = pd.Series(target)

#read play-tennis data
tennis_data = pd.read_csv("data/play_tennis.csv")
tennis_data.drop('day', axis=1, inplace=True)
```

In [3]:
```
#overview tennis data
tennis_data.head()
```

Out[3]:

	outlook	temp	humidity	wind	play
0	Sunny	Hot	High	Weak	No
1	Sunny	Hot	High	Strong	No
2	Overcast	Hot	High	Weak	Yes
3	Rain	Mild	High	Weak	Yes
4	Rain	Cool	Normal	Weak	Yes

In [28]:
```
#Label-encode tennis data
encoder = LabelEncoder()
tennis_data = tennis_data.apply(encoder.fit_transform)
tennis_data.head()
```

Out[28]:

	outlook	temp	humidity	wind	play
0	2	1	0	1	0
1	2	1	0	0	0
2	0	1	0	1	1
3	1	2	0	1	1
4	1	0	1	1	1

In [5]:
```
#overview iris data
iris_data.head()
```

Out[5]:

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

Fitting Data dengan Algoritma *Machine Learning*

1. Iris Data

In [ ]:
```
X_iris = iris_data.drop('label', axis=1)
y_iris = iris_data['label']
```

- **Fitting with ID3**

In [23]:
```
id3 = Id3Estimator()
id3_fit = id3.fit(X_iris, y_iris)
tree_id3 = et_id3(id3_fit.tree_, feature_names=X_iris.columns.tolist())
print(tree_id3)
```

```
petal_length <=2.45: 0 (50)
petal_length >2.45
|   petal_width <=1.75
|   |   sepal_length <=7.10
|   |   |   sepal_width <=2.85: 1 (27/4)
|   |   |   sepal_width >2.85: 1 (22)
|   |   sepal_length >7.10: 2 (1)
|   petal_width >1.75
|   |   sepal_length <=5.95
|   |   |   sepal_width <=3.10: 2 (6)
|   |   |   sepal_width >3.10: 1 (1)
|   |   sepal_length >5.95: 2 (39)
```

- **Fitting with DTL**

In [38]:
```
dtl = DecisionTreeClassifier()
dtl_fit = dtl.fit(X_iris, y_iris)
tree_dtl = et_dtl(dtl_fit, feature_names=X_iris.columns.tolist())
print(tree_dtl)
```

```
|--- petal_length <= 2.45
|   |--- class: 0
|--- petal_length >  2.45
|   |--- petal_width <= 1.75
|   |   |--- petal_length <= 4.95
|   |   |   |--- petal_width <= 1.65
|   |   |   |   |--- class: 1
|   |   |   |--- petal_width >  1.65
|   |   |   |   |--- class: 2
|   |   |--- petal_length >  4.95
|   |   |   |--- petal_width <= 1.55
|   |   |   |   |--- class: 2
|   |   |   |--- petal_width >  1.55
|   |   |   |   |--- sepal_length <= 6.95
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- sepal_length >  6.95
|   |   |   |   |   |--- class: 2
|   |--- petal_width >  1.75
|   |   |--- petal_length <= 4.85
|   |   |   |--- sepal_length <= 5.95
|   |   |   |   |--- class: 1
|   |   |   |--- sepal_length >  5.95
|   |   |   |   |--- class: 2
|   |   |--- petal_length >  4.85
|   |   |   |--- class: 2
```

2. Tennis Data

In [31]:
```
X_tennis = tennis_data.drop('play', axis=1)
y_tennis = tennis_data['play']
```

- **Fitting with ID3**

In [34]:
```
id3 = Id3Estimator()
id3_fit = id3.fit(X_tennis, y_tennis)
tree_id3 = et_id3(id3_fit.tree_, feature_names=X_tennis.columns.tolist())
print(tree_id3)
```

```
outlook <=0.50: 1 (4)
outlook >0.50
|   humidity <=0.50
|   |   temp <=1.50: 0 (2)
|   |   temp >1.50
|   |   |   wind <=0.50: 0 (1)
|   |   |   wind >0.50: 0 (1/1)
|   humidity >0.50
|   |   wind <=0.50
|   |   |   temp <=1.00: 0 (1)
|   |   |   temp >1.00: 1 (1)
|   |   wind >0.50: 1 (3)
```

- **Fitting with DTL**

In [39]:
```
dtl = DecisionTreeClassifier()
dtl_fit = dtl.fit(X_tennis, y_tennis)
tree_dtl = et_dtl(dtl_fit, feature_names=X_tennis.columns.tolist())
print(tree_dtl)
```

```
|--- outlook <= 0.50
|   |--- class: 1
|--- outlook >  0.50
|   |--- humidity <= 0.50
|   |   |--- outlook <= 1.50
|   |   |   |--- wind <= 0.50
|   |   |   |   |--- class: 0
|   |   |   |--- wind >  0.50
|   |   |   |   |--- class: 1
|   |   |--- outlook >  1.50
|   |   |   |--- class: 0
|   |--- humidity >  0.50
|   |   |--- wind <= 0.50
|   |   |   |--- outlook <= 1.50
|   |   |   |   |--- class: 0
|   |   |   |--- outlook >  1.50
|   |   |   |   |--- class: 1
|   |   |--- wind >  0.50
|   |   |   |--- class: 1
```

Menjawab Pertanyaan

Note: DTL Sklearn menggunakan algoritma CART (Classification and Regression Tree)

a. Penentuan atribut terbaik

- DTL (CART)

Metode untuk menentukan atribut terbaik pada CART biasa disebut sebagai *splitting criterion*

splitting criterion umumnya dan defaultnya menggunakan Gini index sebagai metrik, yang rumusnya dinyatakan sebagai berikut:

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

dimana i: value-value pada label

Untuk tiap atribut, pada setiap iterasi, dihitung Gini index untuk masing-masing. Atribut terbaik adalah yang Gini indexnya paling tinggi

- ID3 (Buku)

Menggunakan Entropy dan Information Gain

$$Entropy = \Sigma_{i=1}^{C} (-p_i) * log_2(p_i)$$

$$Gain(S, A) = Entropy(S) - \Sigma_{v \epsilon Values(A)} (\frac{|S_v|}{|S|}) Entropy(S_v)$$

Untuk setiap atribut dilakukan perhitungan gain. Atribut terbaik adalah yang memiliki nilai Gain terbesar

- ID3Estimator (Library)

Sama dengan ID3 pada buku

b. Penanganan label dari cabang setiap nilai atribut

- DTL (CART)

- Jika pada suatu node, *stopping criterion* yang terdefinisi belum tercapai, maka akan dibuat subtree dibawah cabang tersebut menggunakan atribut dengan gini index tertinggi pada subset data disaat itu.
- Jika stopping criterion telah tercapai, maka akan dibuat daun yang menyatakan hasil prediksi. Diantara stopping criterion yang biasa digunakan:

```
    - Pada suatu node, nilai target variabel sudah identik seluruhnya
    - Pada suatu node, nilai variabel yang digunakan untuk splitting s
ama seluruhnya
```

- ID3 (Buku)

- Jika pada cabang nilai atribut, label sudah bisa terprediksi dengan sempurna (Nilai entropi subset data adalah 0 atau 1), maka akan dibuat daun pada *tree* yang menyatakan hasil prediksi variabel target
- Jika pada cabang nilai atribut, label belum bisa terprediksi dengan sempurna (Nilai entropi diantara 1 atau 0), maka akan dibuat *sub-tree* dibawah cabang yang bersangkutan, dengan cara memilih information gain terbesar dari tiap atribut yang ada pada subset data (dilakukan iterasi atau splitting lebih lanjut)

- ID3Estimator (Library)

Sama dengan yang ada pada buku

c. Penentuan label jika examples kosong di cabang tersebut

- DTL (CART)

Jika example pada suatu cabang node kosong, nilai untuk label akan diambil dari sub-tree (bisa jadi daun) dari *most common class*(kategori yang paling umum) yang ada pada node tsb (dari cabang lain)

- ID3 (Buku)

Jika example pada suatu cabang node kosong, akan mengembalikan pohon dengan node tunggal yang nilai dari target variable-nya adalah *most common value* dari nilai target variable yang ada pada dataset awal

- ID3Estimator (Library)

sama dengan yang ada pada buku

d. Penanganan atribut kontinu

- DTL (CART)

Penanganan untuk atribut kontinu sama dengan penanganan untuk atribut kategorikal yang ordinal, yakni dilakukan algoritma sbb:

- Dilakukan sorting dari nilai yang paling kecil hingga nilai yang paling besar.
- Iterasi tiap nilai, dimulai dari yang paling besar hingga yang paling kecil, untuk menemukan nilai (dari nilai yang ada) yang bisa memaksimalkan *splitting criterion* yang terdefinisi
- Setelah ditemukan nilai *split* yang optimal, maka splitting akan dilakukan dengan membagi data menjadi "kurang dari samadengan" nilai tersebut, dan "lebih dari" nilai tersebut

- ID3 (Buku)

Penanganan atribut kontinu adalah dengan membuat partisi-partisi diskrit (range) dari nilai kontinu yang ada. Secara spesifik, untuk sebuah atribut A yang nilainya kontinu dapat dibuat sebuah atribut A' yang nilainya boolean, yakni bernilai benar jika untuk sebuah *threshold* c, A < c, dan bernilai salah jika sebaliknya. Cara untuk memilih *threshold* c adalah sbb:

- Dilakukan sorting pada atribut yang bersangkutan
- Memasukkan setiap nilai yang bersebelahan namun mempunyai nilai target variabel yang berbeda kedalam himpunan kandidat *threshold*
- Untuk tiap nilai yang ada pada himpunan kandidat threshold, dihitung information gain yang didapat jika dilakukan splitting dengan *threshold* tersebut
- Nilai *threshold* dipilih dari kandidat *threshold* yang menghasilkan information gain paling tinggi

- ID3Estimator (Library)

Sama seperti yang ada pada buku

e. Penanganan atribut dengan missing values

- DTL

Modul DecisionTreeClassifier yang diberikan tidak dapat menangani kasus apabila menerima dataset yang memiliki missing values. Jadi untuk menggunakan DecisionTreeClassifier ini harus menggunakan dataset yang lengkap atau menangani masalah missing value ini dengan library lain dari scikit (imputation) atau dengan memisahkan data missing values dari dataset yang akan digunakan.

- ID3

Berdasarkan buku Tom Mitchel, ID3 dapat menangani missing attribut values dengan menggunakan dua cara, yaitu :

1. Most common value : Mengganti missing values dengan nilai most common suatu atribut pada node n.
2. Probability : Melakukan perhitungan probabilitas dari setiap values sebuah atribut. fraksional dari hasil probabilitas yang didapat akan digunakan untuk melakukan perhitungan information gain.

- ID3Estimator (Library)

Terdapat dua cara penanganan untuk mengatasi masalah missing attribute value : Cara pertama yaitu exclude missing value dari perhitungan membuat cabang. Cara kedua yaitu include missing value sebagai sebuah nilai berbeda dari atribut tersebut.

Dari kedua cara akan dibandingkan hasil information gainnya, kemudian dipilih berdasarkan information gain yang lebih besar.

f. Pruning dan parameter confidence

- DTL (CART)

metode pruning menggunakan minimal cost complexity, simpul terlemah akan dipruning terlebih dahulu.

mencegah overfitting dan mengontrol ukuran pohon digunakan parameter min_samples_leaf, max_depth, cost complexity parameter (ccp_alpha) semakin tinggi ccp_alpha tingkat impurities semakin tinggi

- ID3 (Buku)

Metode reduced-error pruning menggunakan training data dan validation data. Parameter confidencenya adalah minimum description length (MDL)

tidak ada parameter yang *customizable* pada algoritma yang tertulis pada buku

- ID3Estimator (Library)

metode reduced-error pruning, operasi pruning akan dilakukan jika nilai error dari decision tree awal melebihi nilai error dari versi pruned decision tree

parameter yang mengatur ukuran pohon adalah max_depth, min_entropy_decrease, prune, gain_ratio, is_repeating, dan min_samples_split