Bab 5
Decision Trees

A. KOMPETENSI DASAR

- Memahami konsep *Decision Trees*.
- Memahami eksperimen klasifikasi menggunakan Decision Trees
- Memahami cara kerja decision tree melalui visualisasi data
- Mengetahui cara mengevaluasi AGNES

B. ALOKASI WAKTU

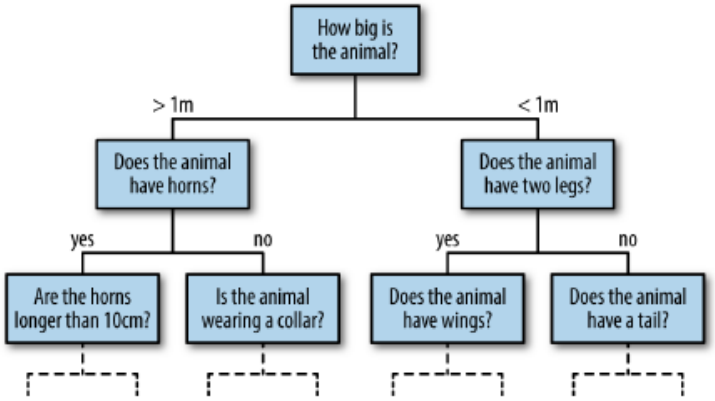
4 js (4x50 menit)

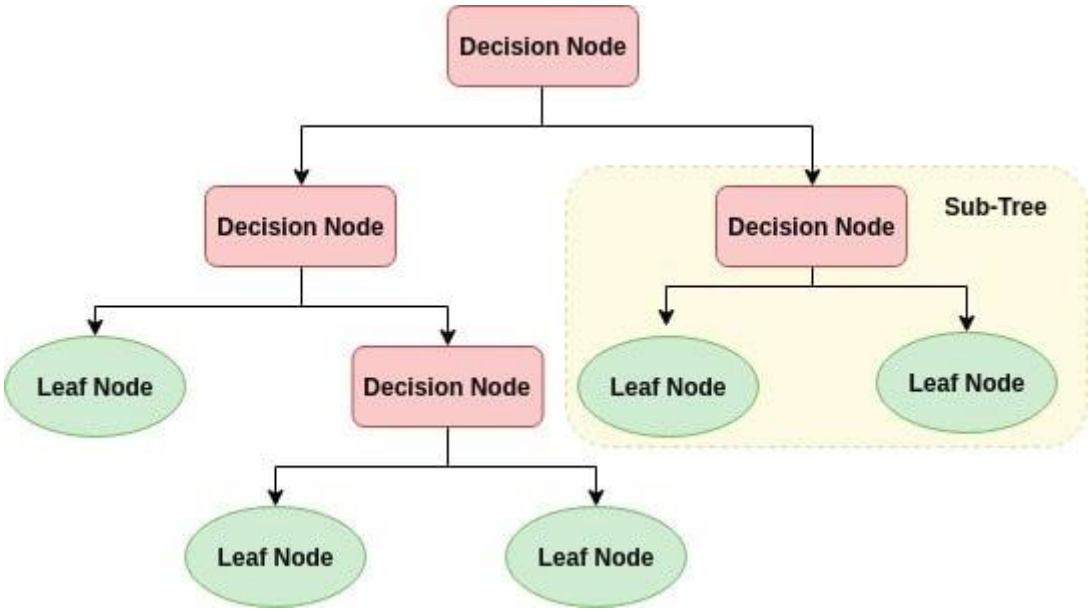
C. PETUNJUK

- Awali setiap aktivitas dengan do'a, semoga berkah dan mendapat kemudahan.
- Pahami Tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar.
- Kerjakan tugas-tugas dengan baik, sabar, dan jujur.
- Tanyakan kepada teman lalu asisten/dosen apabila ada hal-hal yang kurang jelas.

D. DASAR TEORI Decision Trees

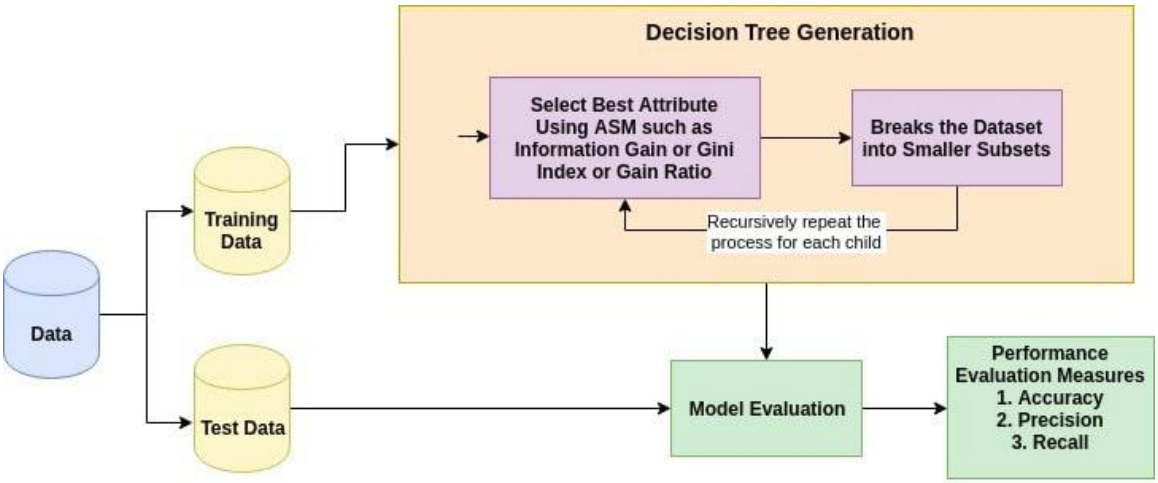
Decision Trees adalah metode *supervised learning* berbentuk non-parametrik yang digunakan untuk klasifikasi dan regresi. Tujuannya adalah untuk membuat model yang memprediksi nilai variabel target dengan mempelajari aturan keputusan sederhana yang disimpulkan dari fitur data. Teknik ini merupakan cara yang sangat intuitif untuk mengklasifikasikan atau melabeli objek: cukup dengan mengajukan serangkaian pertanyaan yang dirancang untuk membobol klasifikasi. Misalnya, jika ingin membuat Decision Tree untuk mengklasifikasikan hewan yang Anda temui saat mendaki, Anda bisa membuat yang ditunjukkan pada Gambar di bawah ini.





Cara kerja algoritma Decision Tree:

1. Seleksi attribute terbaik menggunakan Attribute Selection Measures (ASM) untuk memisah hasil.
2. Jadikan attribute tersebut sebagai decision node dan pecahkan dataset menjadi subset kecil.
3. Mulai lakukan pembangunan pohon secara rekursif di setiap child sampai salah satu kondisi ini ditemukan:
   a. Semua tupel memiliki nilai atribut yang sama.
   b. Tidak ada lagi atribut yang tersisa.
   c. Tidak ada contoh lagi



Attribute Selection Measure (ASM0 adalah heuristik untuk memilih kriteria pemisahan yang membagi data menjadi cara terbaik. Ia juga dikenal sebagai aturan pemisahan karena membantu kita untuk menentukan breakpoints untuk tuple pada node yang diberikan. ASM memberikan peringkat untuk setiap fitur (atau atribut) dengan menjelaskan dataset yang diberikan. Atribut skor terbaik akan dipilih sebagai atribut pemisahan. Dalam kasus atribut bernilai kontinu, titik perpecahan untuk cabang juga

perlu ditentukan. Ukuran seleksi yang paling populer adalah Information Gain, Gain Ratio, dan Gini Index. Langkah rumus untuk perhitungan Gini Index seperti ini:

$$\text{Gini(D)}=1-\sum_{i=1}^{m}\text{Pi}^2$$

Untuk contoh perhitungannya bisa dilihat pada halaman web ini:
http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/

Berikut dari kelebihan menggunakan Decision Tree:

- Mudah dimengerti dan diinterpretasikan. Pohon bisa divisualisasikan.
- Membutuhkan sedikit persiapan data. Teknik lain sering membutuhkan normalisasi data, variabel dummy perlu dibuat dan nilai-nilai kosong harus dihapus. Namun perlu dicatat bahwa modul ini tidak mendukung nilai yang hilang.
- Biaya menggunakan pohon (yaitu, memprediksi data) adalah logaritmik dalam jumlah titik data yang digunakan untuk melatih pohon.
- Mampu menangani data numerik dan kategorikal. Teknik lain biasanya khusus dalam menganalisis dataset yang hanya memiliki satu jenis variabel. Lihat algoritma untuk informasi lebih lanjut.
- Mampu menangani masalah multi-output.
- Menggunakan model kotak putih. Jika situasi tertentu dapat diamati dalam model, penjelasan untuk kondisi tersebut mudah dijelaskan oleh logika boolean. Sebaliknya, dalam model kotak hitam (mis., Dalam jaringan saraf tiruan), hasilnya mungkin lebih sulit untuk ditafsirkan.
- Kemungkinan untuk memvalidasi model menggunakan tes statistik. Itu memungkinkan untuk memperhitungkan keandalan model.
- Berkinerja baik bahkan jika asumsinya agak dilanggar oleh model sebenarnya dari mana data dihasilkan.

Namun kerugian dari Decision Tree meliputi:

- Decision Tree dapat membuat pohon terlalu rumit yang tidak menggeneralisasikan data dengan baik. Ini disebut overfitting. Mekanisme seperti pemangkasan (saat ini tidak didukung), pengaturan jumlah sampel minimum yang diperlukan pada simpul daun atau pengaturan kedalaman maksimum pohon diperlukan untuk menghindari masalah ini.
- Decision Tree dapat menjadi tidak stabil karena variasi kecil dalam data mungkin menghasilkan pohon yang sama sekali berbeda yang dihasilkan. Masalah ini dikurangi dengan menggunakan Decision Tree dalam sebuah ensemble.
- Masalah belajar Decision Tree yang optimal dikenal sebagai NP-lengkap di bawah beberapa aspek optimalitas dan bahkan untuk konsep sederhana. Akibatnya, algoritma pembelajaran Decision Tree praktis didasarkan pada algoritma heuristik seperti algoritma serakah di mana keputusan yang optimal secara lokal dibuat di setiap node. Algoritme semacam itu tidak dapat menjamin untuk mengembalikan Decision Tree yang optimal secara global. Ini dapat dikurangi dengan

melatih banyak pohon dalam pembelajar ensembel, di mana fitur dan sampel secara acak dijadikan sampel dengan penggantian.

- Ada konsep yang sulit dipelajari karena Decision Tree tidak mengekspresikannya dengan mudah, seperti masalah XOR, paritas atau multiplekser.
- Pembelajar Decision Tree membuat pohon bias jika beberapa kelas mendominasi. Oleh karena itu disarankan untuk menyeimbangkan dataset sebelum disesuaikan dengan Decision Tree.

E. LATIHAN (jawaban dapat ditulis pada halaman baru)

1. Persiapkan data berupa array 2 dimensi menggunakan library make_blobs:

```
X, y = make_blobs(n_samples=300, centers=4,
random_state=0, cluster_std=1.0)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='rainbow');
```

Decision tree sederhana yang akan dibangun dengan menggunakan data di atas ini secara iteratif akan membagi data di sepanjang satu atau lainnya sesuai dengan beberapa kriteria kuantitatif, dan pada setiap tingkat menetapkan label wilayah baru berdasarkan suara terbanyak dari poin di dalamnya.

2. Selanjutnya, terapkan decision tree visualisasi hasilnya:

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier().fit(X, y)
```

```
def visualize_classifier(model, X, y, ax=None, cmap='rainbow'):
    ax = ax or plt.gca()
    # Plot the training points
    ax.scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=cmap,
    clim=(y.min(), y.max()), zorder=3)
    ax.axis('tight')
    ax.axis('off')
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()
    # fit the estimator
    model.fit(X, y)
    xx, yy = np.meshgrid(np.linspace(*xlim, num=200),
    np.linspace(*ylim, num=200))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    # Create a color plot with the results
    n_classes = len(np.unique(y))
    contour = ax.contourf(xx, yy, Z, alpha=0.3,
    levels=np.arange(n_classes + 1) - 0.5,
    cmap=cmap, clim=(y.min(), y.max()),
    zorder=1)
    ax.set(xlim=xlim, ylim=ylim)
```

F. Latihan (Menggunakan dataset lain)

1. Download dataset di kaggle: https://www.kaggle.com/uciml/pima-indians-diabetes-database

2. Setup python anda untuk menerapkan machine learning:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import os

def load_datasets(filename, dataset_path="datasets/diabetes"):
    csv_path = os.path.join(dataset_path, filename)
    return pd.read_csv(csv_path, skiprows=1, header=None, names=col_names)
```

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
data = load_datasets("diabetes.csv")
data.head()
```

3. Seleksi fitur (pisahkan data attribute dengan data label)

```
#Feature Selection
#split dataset in features and target variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

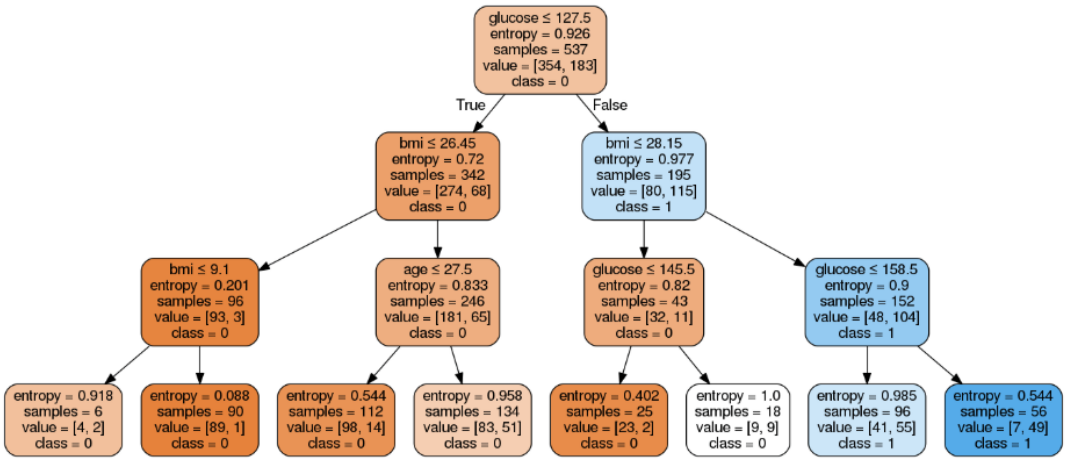
4. Bangun model Decision Trees

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

5. Print hasil akurasi menggunakan accuracy_score

G. TUGAS

1. Buatlah 2 modifikasi model Decision Tree pada Latihan F (Subbab F) untuk mengasilkan nilai akurasi lebih dari sebelumnya
2. Carilah 2 datasets dan terapkan model Decision Tree
3. Improvisasi hasil evaluasi kedua dataset yang anda dapatkan
4. Buatlah kesimpulan dan cara perhitungan sederhana split strategy (Information Gain, Gini Ratio atau Gini Index) yang anda pakai di nomor 3


glucose ≤ 127.5
entropy = 0.926
samples = 537
value = [354, 183]
class = 0
True
False
bmi ≤ 26.45
entropy = 0.72
samples = 342
value = [274, 68]
class = 0
bmi ≤ 28.15
entropy = 0.977
samples = 195
value = [80, 115]
class = 1
bmi ≤ 9.1
entropy = 0.201
samples = 96
value = [93, 3]
class = 0
age ≤ 27.5
entropy = 0.833
samples = 246
value = [181, 65]
class = 0
glucose ≤ 145.5
entropy = 0.82
samples = 43
value = [32, 11]
class = 0
glucose ≤ 158.5
entropy = 0.9
samples = 152
value = [48, 104]
class = 1
entropy = 0.918
samples = 6
value = [4, 2]
class = 0
entropy = 0.088
samples = 90
value = [89, 1]
class = 0
entropy = 0.544
samples = 112
value = [98, 14]
class = 0
entropy = 0.958
samples = 134
value = [83, 51]
class = 0
entropy = 0.402
samples = 25
value = [23, 2]
class = 0
entropy = 1.0
samples = 18
value = [9, 9]
class = 0
entropy = 0.985
samples = 96
value = [41, 55]
class = 1
entropy = 0.544
samples = 56
value = [7, 49]
class = 1