# LAPORAN PRAKTIKUM

# KECERDASAN BUATAN

## Jobsheet 7: Decision Tree



| | |
|---|---|
| Mata Kuliah | : Kecerdasan Buatan |
| Dosen | : Dr. Ir. Kurnianingsih, S.T., M.T. |
| Disusun oleh | : |
| Nama | : Farid Fikri Khoirudin |
| NIM | : 4.33.20.0.10 |
| Kelas | : TI-2A |

**PROGRAM STUDI D4 TEKNOLOGI REKAYASA KOMPUTER**

**JURUSAN TEKNIK ELEKTRO**

**POLITEKNIK NEGERI SEMARANG**


**2021**

# Jobsheet 7

# Decision Tree

**Kompetensi Dasar**

- Mahasiswa dapat memahami konsep regresi dan klasifikasi
- Mahasiswa dapat menerapkan regresi dan klasisifikasi menggunakan decision tree

## 1. Regresi

Analisis regresi merupakan analisis yang bertujuan untuk mengetahui pengaruh suatu variabel terhadap variabel lain. Dalam analisis regresi, variabel yang mempengaruhi disebut Variabel Bebas (Independent Variable) dan variabel yang terpengaruh disebut Variabel Terikat (Dependent Variable). Jika dalam persamaan regresi hanya terdapat satu variabel bebas dan satu variabel terikat maka disebut persamaan regresi sederhana, sedangkan jika variabel bebas lebih dari satu disebut persamaan multiple regression.

Regresi adalah teknik statistik yang dapat digunakan untuk menggambarkan hubungan fungsional antara variabel bebas dan satu atau lebih variabel bebas. Regresi merupakan alat ukur yang digunakan untuk mengukur ada tidaknya korelasi antar variabel. Analisis regresi merupakan teknik statistik yang memiliki banyak pengguna dan memiliki manfaat yang cukup besar bagi pengambil keputusan.

Hampir semua peristiwa, baik peristiwa ekonomi maupun peristiwa sosial lainnya, saling berkaitan dan saling mempengaruhi. Peristiwa ini dapat dinyatakan sebagai perubahan nilai suatu variabel, misalnya variabel X dan Y. Masing-masing variabel ini disebut variabel independen dan variabel dependen.

Variabel bebas (independent) adalah variabel yang nilainya tidak tergantung pada variabel lain, biasanya disimbolkan dengan X. Variabel ini digunakan untuk memprediksi / menjelaskan nilai variabel lain. Variabel terikat adalah variabel yang nilainya bergantung pada variabel lain, biasanya disimbolkan dengan Y. Variabel ini merupakan variabel yang nilainya diramalkan atau dijelaskan.

### 1.1 Regresi Linear

Dalam statistik, regresi linier adalah pendekatan linier untuk memodelkan hubungan antara respons skalar (atau variabel dependen ) dan satu atau lebih variabel penjelas (atau variabel independen ). Kasus satu variabel penjelas disebut regresi linier sederhana. Untuk lebih dari satu variabel penjelas, prosesnya disebut regresi linier berganda. Istilah ini berbeda dari regresi linier multivariat, di mana beberapa variabel dependen berkorelasi diprediksi, bukan variabel skalar tunggal.

Dalam regresi linier, hubungan dimodelkan menggunakan fungsi prediktor linier yang parameter model tidak diketahui diperkirakan dari data. Model semacam itu disebut model linier. Paling umum, rata - rata kondisional dari respons yang diberikan nilai-nilai dari variabel penjelas (atau prediktor) diasumsikan sebagai fungsi affine dari nilai-nilai tersebut; lebih jarang, median kondisional atau kuantil lainnya digunakan. Seperti semua bentuk analisis regresi, regresi linier berfokus pada distribusi probabilitas bersyarat dari respons yang diberikan nilai-nilai prediktor, daripada pada distribusi probabilitas gabungan dari semua variabel ini, yang merupakan domain dari analisis multivariat.

Regresi linier adalah tipe pertama dari analisis regresi yang dipelajari dengan seksama, dan digunakan secara luas dalam aplikasi praktis. Ini karena model yang bergantung secara linier pada parameter yang tidak diketahui mereka lebih mudah dipasangkan daripada model yang tidak linier terkait dengan parameternya dan karena sifat statistik dari penduga yang dihasilkan lebih mudah untuk ditentukan.

Regresi linier memiliki banyak kegunaan praktis. Sebagian besar aplikasi termasuk dalam salah satu dari dua kategori luas berikut:

- Jika tujuannya adalah prediksi, peramalan, atau pengurangan kesalahan, [klarifikasi diperlukan] regresi linier dapat digunakan agar sesuai dengan model prediktif dengan set data yang diamati dari nilai-nilai respon dan variabel penjelas. Setelah mengembangkan model seperti itu, jika nilai-nilai tambahan dari variabel penjelas dikumpulkan tanpa nilai respons yang menyertainya, model yang sesuai dapat digunakan untuk membuat prediksi tanggapan.

- Jika tujuannya adalah untuk menjelaskan variasi dalam variabel respons yang dapat dikaitkan dengan variasi dalam variabel penjelas, analisis regresi linier dapat diterapkan untuk mengukur kekuatan hubungan antara respons dan variabel penjelas, dan khususnya untuk menentukan apakah beberapa variabel penjelas mungkin tidak memiliki hubungan linier dengan respons sama sekali, atau untuk mengidentifikasi himpunan bagian dari variabel penjelas yang mungkin berisi informasi yang berlebihan tentang respons tersebut.

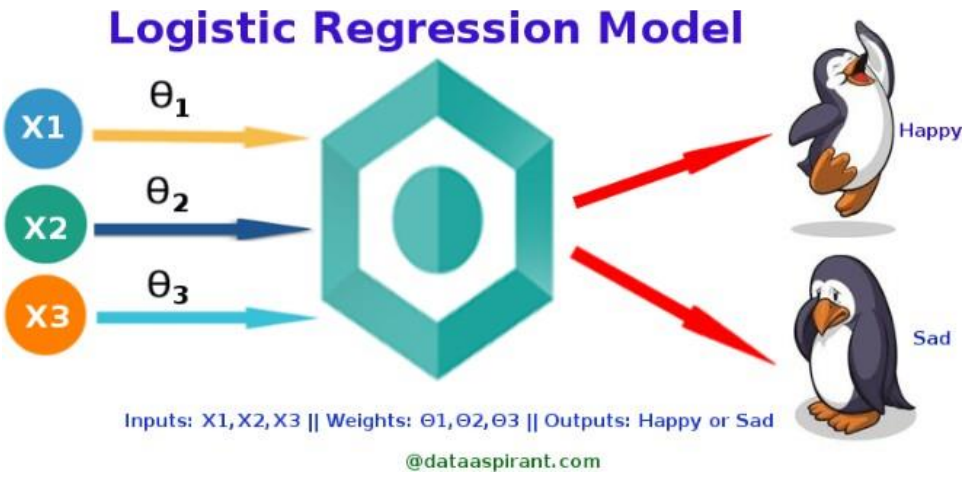
Model regresi linier sering dipasang menggunakan pendekatan kuadrat terkecil, tetapi mereka juga dapat dipasang dengan cara lain, seperti dengan meminimalkan "kurangnya kecocokan" dalam beberapa norma lain (seperti dengan regresi deviasi absolut terkecil), atau

dengan meminimalkan hukuman versi dari fungsi biaya kuadrat terkecil seperti pada regresi ridge (penalti L2 -norm) dan laso (penalti L1 -norm). Sebaliknya, pendekatan kuadrat terkecil dapat digunakan agar sesuai dengan model yang bukan model linier. Jadi, meskipun istilah "kuadrat terkecil" dan "model linier" sangat terkait, mereka tidak identik.

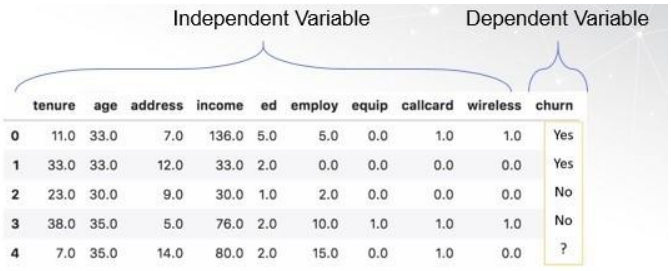
## 1.2 Regresi Logistik

- Logistic Regression merupakan salah satu teknik machine learning untuk melakukan klasifikasi record dari dataset.

- Logistic Regression atau regresi logistik adalah sebuah pendekatan untuk membuat model prediksi seperti halnya regresi linear atau yang biasa disebut dengan istilah Ordinary Least Squares (OLS) regression. Perbedaannya adalah pada regresi logistik, peneliti memprediksi variabel terikat yang berskala dikotomi. Skala dikotomi yang dimaksud adalah skala data nominal dengan dua kategori, misalnya: Ya dan Tidak, Baik dan Buruk atau Tinggi dan Rendah.




Variabel yang ada pada logistic regression:



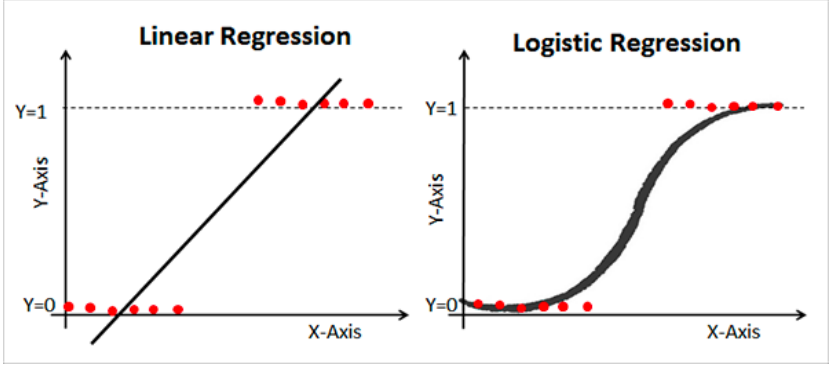

| | tenure | age | address | income | ed | employ | equip | callcard | wireless | churn |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 11.0 | 33.0 | 7.0 | 136.0 | 5.0 | 5.0 | 0.0 | 1.0 | 1.0 | Yes |
| 1 | 33.0 | 33.0 | 12.0 | 33.0 | 2.0 | 0.0 | 0.0 | 0.0 | 0.0 | Yes |
| 2 | 23.0 | 30.0 | 9.0 | 30.0 | 1.0 | 2.0 | 0.0 | 0.0 | 0.0 | No |
| 3 | 38.0 | 35.0 | 5.0 | 76.0 | 2.0 | 10.0 | 1.0 | 1.0 | 1.0 | No |
| 4 | 7.0 | 35.0 | 14.0 | 80.0 | 2.0 | 15.0 | 0.0 | 1.0 | 0.0 | ? |

- Indepentent Variable = Variable / Fitur yang merupakan input dan akan dipakai untuk memprediksi sebuah output, churn.

- Dependent Variable = Nilainya bergantung pada nilai-nilai input
- Pelanggan akan berhenti atau tidak bergantung dari data pelanggan tsb.

**Perbedaan antara Linear Regression dengan Logistic Regression:**

| Linear Regression | Logistic Regression |
|---|---|
| • Melakukan Prediksi<br>• Prediksi nilai kontinyu dari sebuah variable, seperti:<br>➢ Harga rumah berdasarkan ciri<br>➢ Tekanan darah berdasarkan symptom<br>➢ Konsumsi bensin berdasarkan kondisi mobil | • Melakukan Klasifikasi<br>• Klasifikasi nilai biner, seperti:<br>Kelompok A atau B<br>Sukses atau tidak sukses<br>Tetap berlangganan atau tidak. |

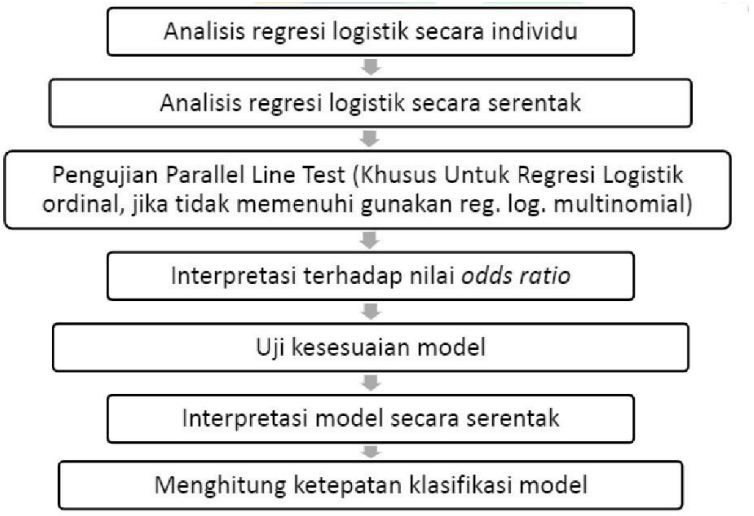



Asumsi yang harus dipenuhi dalam Regresi Logistik antara lain:

1. Regresi logistik tidak membutuhkan hubungan linier antara variabel bebas dengan variabel terikat.
2. Regresi logistik dapat menyeleksi hubungan karena menggunakan pendekatan non linier log transformasi untuk memprediksi odds ratio. Odd dalam regresi logistik sering dinyatakan sebagai probabilitas. Misal Odd sebuah perusahaan dapat bangkrut atau berhasil atau odd seorang anak dapat lulus atau tidak pada Ujian Nasional.
3. Variabel bebas tidak memerlukan asumsi multivariate normality

4. Asumsi homokedastis tidak diperlukan
5. Variabel bebas tidak perlu dirubah ke bentuk metric (interval atau skala ratio)
6. Pengamatan dilakukan secara independen (misalnya, dengan teknik random sampling)
7. Logistic Regression mewajibkan seluruh data dalam **bentuk numerik**
8. Jika berkategori (Pria/Wanita, Ya/Tidak) harus diubah dalam bentuk angka.

**Langkah analisis regresi logistik**




**Kapan kita gunakan Logistic Regression?**

- Jika data berupa binary, seperti:
  - Kelompok A atau B
  - Lulus atau Tidak
  - Berlangganan atau Tidak
- Jika kita membutuhkan pengelompokkan dalam bentuk probabilitas
- Data bersifat “linearly separable”
- Linearly Separable
- Dapat dipisahkan secara linear

- Jika data 2D, dipisahkan garis
- Jika data 3D, dipisahkan plane
- Jika data >3D, dipisahkan hyper-plane.
- Secara teori, Logistic Regression sebenarnya juga dapat digunakan untuk data yang bersifat "non-linearly separable"

## 2. Decision Tree

Pohon keputusan dalam aturan keputusan (decision rule) merupakan metodologi data mining yang banyak diterapkan sebagai solusi untuk regresi dan klasifikasi. Decision tree merupakan suatu metode regresi dan klasifikasi yang menggunakan struktur pohon, dimana setiap node merepresentasikan atribut dan cabangnya merepresentasikan nilai dari atribut, sedangkan daunnya digunakan untuk merepresentasikan kelas. Node teratas dari decision tree ini disebut dengan root.

Breiman et al. (1984) menyatakan bahwa metode ini merupakan metode yang sangat populer untuk digunakan karena hasil dari model yang terbentuk mudah untuk dipahami. Dinamakan pohon keputusan karena aturan yang terbentuk mirip dengan bentuk pohon. Pohon terbentuk dari proses pemilahan rekursif biner pada suatu gugus data sehingga nilai variabel respon pada setiap gugus data hasil pemilahan akan lebih homogen. Pada pohon keputusan terdapat tiga jenis node, antara lain:

1) Akar

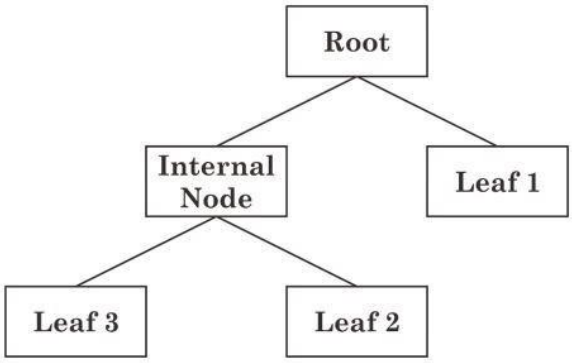
   Merupakan node teratas, pada node ini tidak ada input dan dapat tidak mempunyai output atau dapat mempunyai output lebih dari satu.

2) Internal node

   Merupakan node percabangan, pada node ini hanya terdapat satu input dan mempunyai output minimal dua.

3) Daun

   Merupakan node percabangan, pada node ini hanya terdapat satu input dan mempunyai output minimal dua.




Konsep dari pohon keputusan adalah mengubah data menjadi pohon keputusan dan aturan-aturan keputusan. Pohon keputusan merupakan himpunan aturan if – then, dimana setiap path dalam pohon dihubungkan dengan sebuah aturan dimana premis terdiri atas sekumpulan node yang ditemui dan kesimpulan dari aturan terdiri atas kelas yang dihubungkan dengan daun dari path. Pembentukan pohon keputusan terdiri dari beberapa tahap:

1. Konstruksi pohon diawali dengan pembentukan akar (terletak paling atas). Kemudian data dibagi berdasarkan atribut-atribut yang cocok untuk dijadikan daun.

2. Pemangkasan pohon (tree pruning) yaitu mengidentifikasikan dan membuang cabang yang tidak diperlukan pada pohon yang telah terbentuk. Hal ini dikarenakan pohon keputusan yang dikontruksi dapat berukuran besar, maka dapat disederhanakan dengan melakukan pemangkasan berdasarkan nilai kepercayaan (confident level). Pemangkasan pohon dilakukan selain untuk pengurangan ukuran pohon juga bertujuan untuk mengurangi tingkat kesalahan prediksi pada kasus baru dari hasil pemecahan yang dilakukan dengan divide and conquer. Pruning ada dua pendekatan yaitu:

   a) Pre-pruning yaitu menghentikan pembangunan suatu subtree lebih awal (dengan memutuskan untuk tidak lebih jauh mempartisi data training). Saat seketika berhenti, maka node berubah menjadi leaf (node akhir). Node akhir ini menjadi kelas yang paling sering muncul di antara subset sampel.

   b) Post-pruning yaitu menyederhanakan tree dengan cara membuang beberapa cabang subtree setelah tree selesai dibangun. Node yang jarang dipotong akan menjadi leaf (node akhir) dengan kelas yang paling sering muncul.

3. Pembentukan aturan keputusan yaitu membuat aturan keputusan dari pohon yang telah dibentuk. Aturan tersebut dapat dalam bentuk if – then diturunkan dari pohon keputusan dengan melakukan penelusuran dari akar sampai ke daun. Untuk setiap simpul dan percabangannya akan diberikan di if, sedangkan nilai pada daun akan ditulis di then. Setelah semua aturan dibuat maka aturan dapat disederhanakan atau digabung.

Decision tree adalah suatu model klasifikasi yang paling populer karena mudah diinterpretasikan oleh manusia. Banyak algoritma yang dapat digunakan dalam pembentukan pohon keputusan seperti ID3, C4.5, CART, dan GUIDE. Algoritma decision tree banyak digunakan dalam proses data mining karena memiliki beberapa kelebihan:

1. Mudah mengintegrasikan dengan sistem basis data.
2. Memiliki ketelitian yang baik.
3. Dapat menemukan gabungan tak terduga dari suatu data.
4. Daerah pengambilan keputusan yang sebelumnya kompleks dan sangat global dapat diubah menjadi lebih sederhana dan spesifik.
5. Dapat melakukan eliminasi untuk perhitungan-perhitungan yang tidak diperlukan. Karena ketika menggunakan metode ini maka sampel hanya diuji berdasarkan kriteria atau kelas tertentu.
6. Fleksibel untuk memilih fitur dari internal node yang berbeda, fitur yang terpilih akan membedakan suatu kriteria dibandingkan kriteria yang lain dalam node yang sama.

Kekurangan pohon keputusan adalah.

1. Terjadi overlap terutama ketika kelas-kelas dan kriteria yang digunakan jumlahnya sangat banyak. Hal tersebut juga dapat menyebabkan meningkatnya waktu pengambilan keputusan dan jumlah memori yang diperlukan.
2. Pengakumulasian jumlah error dari setiap tingkat dalam sebuah pohon keputusan yang besar.
3. Kesulitan dalam mendesain pohon keputusan yang optimal.
4. Hasil kualitas keputusan yang didapatkan dari metode pohon keputusan sangat bergantung pada bagaimana pohon tersebut didesain.

## 2.1 POHON REGRESI

Pohon regresi adalah salah satu metode yang menggunakan kaidah pohon keputusan (decision tree) yang dibentuk melalui suatu algoritma penyekatan secara rekursif. Metode ini menganalisa suatu gugus data dengan cara menyekatnya menjadi beberapa anak gugus (simpul) secara bertahap.

Komalasari (2007) menyatakan bahwa keabsahan penggunaan analisis regresi parametrik sangat bergantung pada banyak asumsi, sehingga untuk mendapatkan dugaan persamaan regresi yang memenuhi semua asumsi menjadi sangat sulit. Masalah ini dapat diatasi oleh metode pohon

regresi yang tidak memerlukan asumsi. Pohon regresi merupakan salah satu metode eksplorasi nonparametrik yang dapat digunakan untuk melihat hubungan antara variabel respon kontinu dengan variabel-variabel bebas yang berukuran besar dan kompleks. Kekompleksan tersebut dapat berupa dimensi yang besar atau jenis variabelnya campuran, misalnya kontinu dan kategorik, baik nominal maupun ordinal.

Sama halnya dengan metode regresi biasa, pohon regresi juga menjelaskan bagaimana hubungan antara variabel respon dan variabel-variabel bebasnya. Perbedaannya adalah bahwa pada metode pohon regresi, pengaruh variabel bebas serta pendugaan responnya dilakukan pada kelompok-kelompok pengamatan yang ditentukan berdasarkan variabel-variabel bebas, sehingga interpretasi hasil dari metode ini lebih mudah dilakukan. Hal ini karena identifikasi pengaruh dari variabel bebas dari pohon regresi dilakukan dalam masing-masing subgrup data bukan dalam keseluruhan data seperti halnya regresi biasa. Disamping itu pohon regresi dapat mengatasi masalah adanya pencilan. Perhitungan statistik yang dilakukan dalam metode pohon regresi juga tidak rumit sehingga menjadi kelebihan lainnnya dari metode ini.

2.2 Random Forest

Dalam machine learning sering kita mendengar tentang metode Random Forest yang digunakan untuk menyelesaikan permasalahan. Metode Random Forest merupakan salah satu metode dalam Decision Tree. Decision Tree atau pohon pengambil keputusan adalah sebuah diagram alir yang berbentuk seperti pohon yang memiliki sebuah root node yang digunakan untuk mengumpulkan data, Sebuah inner node yang berada pada root node yang berisi tentang pertanyaan tentang data dan sebuah leaf node yang digunakan untuk memecahkan masalah serta membuat keputusan. Decision tree mengklasifikasikan suatu sampel data yang belum diketahui kelasnya kedalam kelas – kelas yang ada. Penggunaan decision tree agar dapat menghindari overfitting pada sebuah set data saat mencapai akurasi yang maksimum.

Random forest adalah kombinasi dari masing – masing tree yang baik kemudian dikombinasikan ke dalam satu model. Random Forest bergantung pada sebuah nilai vector random dengan distribusi yang sama pada semua pohon yang masing masing decision tree memiliki kedalaman yang maksimal. Random forest adalah classifier yang terdiri dari classifier yang berbentuk pohon {h(x, θ k ), k = 1, . . .} dimana θk adalah random vector yang diditribusikan secara independen dan masing masing tree pada sebuah unit kan memilih class yang paling popular pada input x. Berikut ini karakteristik akurasi pada random forest.

1. **Memusatkan random forest**

   Terdapart classifier h1(x), h2(x), . . . , hk (x) dan dengan training set dari distribusi random vector Y,X, Berikut fungsi yang terbentuk:

$$mg(\mathbf{X}, Y) = av_k I\left(h_k(\mathbf{X}) = Y\right) - \max_{j \neq Y} av_k I\left(h_k(\mathbf{X}) = j\right).$$

Fungsi error yang digunakan:

$$PE^* = P_{\mathbf{X},Y}(mg(\mathbf{X}, Y) < 0)$$

Hasil dari penggabungan fungsi:

$$P_{\mathbf{X},Y}(P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j) < 0).$$

Pada hasil tersebut menjelaskan mengapa random forest tidak overfit saat tree di tambahkan, tetapi menghasilkan nilai yang terbatas pada error.

2. **Kekuatan dan Korelasi**

Fungsi yang dihasilkan adalah: Pada fungsi tersebut kekuatan tidak bergantung pada forest.

$$PE^* \leq \sum_i \text{var}(P_\Theta(h(\mathbf{X}, \Theta) = Y) - P_\Theta(h(\mathbf{X}, \Theta) = j))s_j^2$$

Pada fungsi tersebut kekuatan tidak bergantung pada forest.

3. **Random Forest menggunakan seleksi input yang random**

Bagging digunakan untuk pemilihan fitur secara random. Masing – masing training set diambil dengan penggantinya dari training set asli. Kemudian sebuah tree di tanam pada sebuah training set menggunakan seleksi fitur random. Ada dua alasan penggunaan bagging yaitu yang pertama penggunaan bagging untuk meningkatkan akurasi ketika fitur random digunakan. Yang kedua bagging digunakan untuk memberikan perkiraan dari kesalahan generalisasi (PE*) dari gabungan tree, untuk memperkirakan kekuatan dan korelasi. Random Forest yang paling sederhana dengan fitur random dibentuk dengan seleksi secara random, pada masing – masing node, sebuah grup kecil dari input variable yang terbagi. Membentuk tree menggunakan metodologi CART ke ukuran yang maksimum.

4. **Random Forest menggunakan kombinasi input yang linear**

Misalkan terdapat beberapa input, M, F mengambil fraksi pada M yang akan memimpin dalam meningkatkan kekuatan tetapi pada korelasi yang tinggi. Pendekatan yang lain terbentuk dengan mendefinisikan lebih banyak fitur dengan mengambil kombinasi random linear dari sejumlah variable input. Fitur tersebut variabel L yaitu jumlah dari variable yang dikombinasikan. Variabel L secara random diseleksi dan ditambahkan bersama dengan koefisien yang memiliki nomor random [-1,1]. Kombinasi linear F dihasilkan. Prosedur ini di sebut Forest-RC.

2.3 Algoritme C4.5

Algoritme ini digunakan untuk membuat pohon keputusan. Algoritme ini memiliki kelebihan mudah dimengerti, fleksibel, dan dapat dvisualisasikan dalam bentuk gambar pohon keputusan. Ada pun tahapan dalam pembentukan pohon keputusan sebagai berikut:

Hitung jumlah data untuk setiap klasifikasi.

1. Tentukan atribut akar pohon. Untuk menentukan atribut akar didasarkan pada nilai Gain tertinggi dari atribut-atribut yang ada.

$$Gain(S,A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i)$$

Dimana:

- S: simpul acuan
- A: atribut
- n: jumlah partisi atribut A
- |Si|: jumlah kasus pada partisi ke-i
- |S|: jumlah kasus dalam S

Sedangkan nilai Entropy dapat diperoleh dari:

$$Entropy(S) = \sum_{i=1}^{n} -pi * \log_2 pi$$

Dimana :

- pi: proporsi dari Si terhadap S
- Log2 pi = log pi / log 2

2. Buat cabang untuk setiap nilai atribut.

3. Bagi kasus untuk setiap cabang.

4. Ulangi proses 3-4 untuk setiap cabang sampai semua kasus pada cabang memiliki kelas yang sama

**PRAKTIKUM**

Python 3.8

1. Buka software Anaconda Navigator, kemudian buka Jupyter Notebook anda.

2. Kemudian ketik program seperti berikut

## REGRESSION

### Simple linear Regression

Program untuk HRD mengelompokkan karyawan yang mendapat gaji tambahan karena lembur. Bila di bawah garis kurva maka karyawan tersebut digolongkan sebagai karyawan yang suka terlambat sesuai dengan masa kerjanya

```
# Import library yang dibutuhkan

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# masukkan dataset

dataset = pd.read_csv('Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

# mengacak data dari dataset untuk dijadikan data training dan data testing

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 1/3, random_state = 0)

# metode yang digunakan untuk klasifikasi dataset

regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Predicting the Test set results

y_pred = regressor.predict(X_test)

# Visualising the Training set results

plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
```

```
# Visualising the Test set results

plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```

Hasil praktikum:

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression

        dataset = pd.read_csv('Salary_Data.csv')
        X = dataset.iloc[:, :-1].values
        y = dataset.iloc[:, 1].values

        X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size = 1/3, random_state = 0)

        regressor = LinearRegression()
        regressor.fit(X_train, y_train)

        y_pred = regressor.predict(X_test)

        plt.scatter(X_train, y_train, color = 'red')
        plt.plot(X_train, regressor.predict(X_train), color = 'blue')
        plt.scatter(X_test, y_test, color = 'red')
        plt.plot(X_train, regressor.predict(X_train), color = 'blue')
        plt.title('Salary vs Experience (Test set)')
        plt.xlabel('Years of Experience')
        plt.ylabel('Salary')
        plt.show()
```


**Simple Logistic Regression**

Sales marketing suatu perusahaan otomotif ingin memasang iklan menjual SUV agar iklannya tepat sasaran menarik minat pembeli maka Sales menggunakan data pembeli SUV kemudian di kelompokkan menggunakan Simple Logistic Regression

#Logistic Regression

# Import library yang dibutuhkan

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression as LR
```

# masukkan dataset

```
dataset = pd.read csv('Social Network Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
```

# mengacak data dari dataset untuk dijadikan data training dan data testing

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state = 0)
```

# menormalisasi dataset

```
sc_X = StandardScaler()
X train = sc X.fit transform(X train)
X_test = sc X.transform(X_test)
```

#Fitting Logistic Regression into the training set

```
classifier = LR(random_state=0)
classifier.fit(X_train, y_train)
```

# Predicting the test results

```
y_pred = classifier.predict(X_test)
```

# Making the confusion matrix

```
cm = confusion_matrix(y_test, y_pred)
```

# Visualising the Training test results

```
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min()-1, stop=X_set[:,
0].max()+1, step= 0.01),
                     np.arange(start=X_set[:, 1].min()-1, stop=X_set[:,
1].max()+1, step= 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75,cmap= ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j, 0], X_set[y_set==j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)
plt.title('Logistic Regression (Training Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

# Visualising the Testing test results

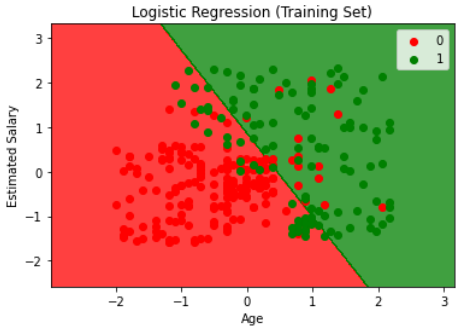
```
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min()-1, stop=X_set[:,
0].max()+1, step= 0.01),
                     np.arange(start=X_set[:, 1].min()-1, stop=X_set[:,
1].max()+1, step= 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75,cmap= ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j, 0], X_set[y_set==j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)
plt.title('Logistic Regression (Testing Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

Hasil praktikum:

In [6]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression as LR

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state = 0)

sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

classifier = LR(random_state=0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

cm = confusion_matrix(y_test, y_pred)

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min()-1, stop=X_set[:, 0].max()+1, step= 0.01),
np.arange(start=X_set[:, 1].min()-1, stop=X_set[:, 1].max()+1, step= 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha=0.75,cmap= ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set==j, 0], X_set[y_set==j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)

plt.title('Logistic Regression (Training Set)
```

```
            c = ListedColormap(('red', 'green'))(i), label=j)

plt.title('Logistic Regression (Training Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```




## DECISION TREE DAN RANDOM FOREST

Komparasi Performa Klasifikasi dari Decision Tree, Random Forest dan Naïve Bayes menggunakan data sintetik

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
```

```
h = .02

names = ["Decision Tree", "Random Forest",
         "Naive Bayes",]

classifiers = [
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    GaussianNB()]

X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,
random_state=1, n_clusters_per_class=1)

rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)
linearly_separable = (X, y)

datasets = [make_moons(noise=0.3, random_state=0),
            make_circles(noise=0.2, factor=0.5, random_state=1),
            linearly_separable
            ]

figure = plt.figure(figsize=(27, 9))

i = 1
```

# iterate over datasets

```
for ds_cnt, ds in enumerate(datasets):
    X, y = ds
    X = StandardScaler().fit_transform(X)
    X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=.4, random_state=42)

    x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
    y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                         np.arange(y_min, y_max, h))
```

# just plot the dataset first

```
cm = plt.cm.RdBu
```

```
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
if ds_cnt == 0:
    ax.set_title("Input data")
```

# Plot the training points

```
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,
           edgecolors='k')
```

```
    # Plot the testing points
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test,
cmap=cm_bright, alpha=0.6, edgecolors='k')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
i += 1

    # iterate over classifiers
for name, clf in zip(names, classifiers):
    ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)

    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max]x[y_min, y_max].
if hasattr(clf, "decision_function"):
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
else:
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
```

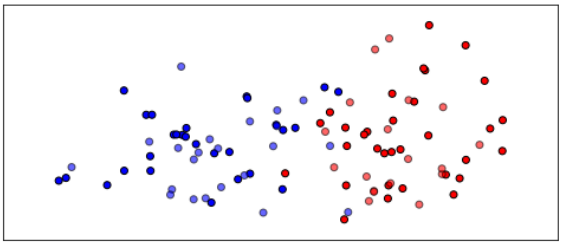
Hasil praktikum:

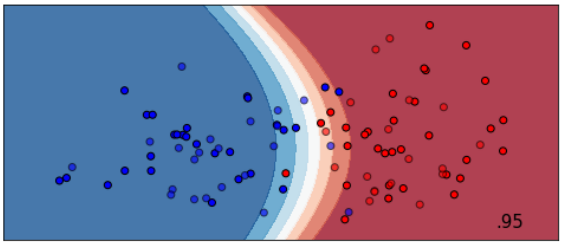
In [9]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_moons, make_circles, make_classification
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB

h = .02
names = ["Decision Tree", "Random Forest", "Naive Bayes",]
classifiers = [
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    GaussianNB()]

X, y = make_classification(n_features=2, n_redundant=0, n_informative=2,
                           random_state=1, n_clusters_per_class=1)
rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)
linearly_separable = (X, y)

datasets = [make_moons(noise=0.3, random_state=0),
            make_circles(noise=0.2, factor=0.5, random_state=1),
            linearly_separable]

figure = plt.figure(figsize=(27, 9))
i = 1

for ds_cnt, ds in enumerate(datasets):
    X, y = ds
    X = StandardScaler().fit_transform(X)
    X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=.4, random_state=42)

    x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
    y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
```

```
    x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
    y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

cm = plt.cm.RdBu
cm_bright = ListedColormap(['#FF0000', '#0000FF'])
ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
if ds_cnt == 0:
    ax.set_title("Input data")

ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm_bright,
           edgecolors='k')

ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test,
cmap=cm_bright, alpha=0.6, edgecolors='k')
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
i += 1

for name, clf in zip(names, classifiers):
    ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)

if hasattr(clf, "decision_function"):
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
else:
    Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]

Z = Z.reshape(xx.shape)
ax.contourf(xx, yy, Z, cmap=cm, alpha=.8)
ax.scatter(X_train[:, 0], X_train[:, 1],
c=y_train, cmap=cm_bright, edgecolors='k')

ax.scatter(X_test[:, 0], X_test[:, 1],
c=y test, cmap=cm bright, edgecolors='k', alpha=0.6)
```

```
ax.scatter(X_test[:, 0], X_test[:, 1],
c=y_test, cmap=cm_bright, edgecolors='k', alpha=0.6)

ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
if ds_cnt == 0:
    ax.set_title(name)
ax.text(xx.max() - .3, yy.min() + .3, ('%.2f' % score).lstrip('0'), size=15, horizontalalignment='right')
i += 1

plt.tight_layout()
plt.show()
```

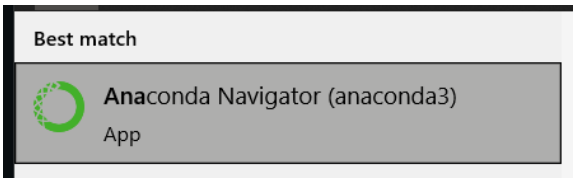
```
<ipython-input-9-2be3411e3e00>:59: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes cur
rently reuses the earlier instance.  In a future version, a new instance will always be created and returned.  Meanwhile, this
warning can be suppressed, and the future behavior ensured, by passing a unique label to each axes instance.
  ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
<ipython-input-9-2be3411e3e00>:59: MatplotlibDeprecationWarning: Adding an axes using the same arguments as a previous axes cur
rently reuses the earlier instance.  In a future version, a new instance will always be created and returned.  Meanwhile, this
warning can be suppressed, and the future behavior ensured, by passing a unique label to each axes instance.
  ax = plt.subplot(len(datasets), len(classifiers) + 1, i)
```



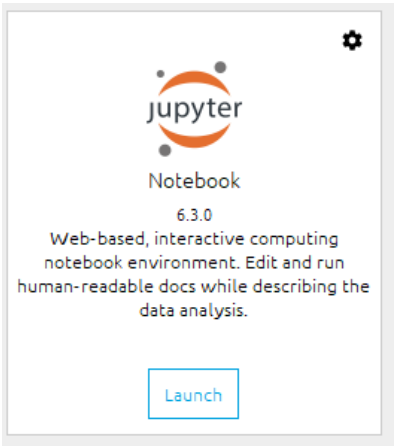



## DECISION TREE CLASSIFICATION

1. Sebelum memulai praktikum, persiapkan software Anaconda 3




2. Kemudian buka Jupyter Notebook




3. Buat file Python baru, kemudian ketikkan kode program decision tree. Pertama, imporlibrary yang kita perlukan.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap
```

4. Setelah itu, impor dataset yang akan digunakan. Variabel X menampung data yang akandiolah dan variabel y menampung data target. Variabel X dan y didefinisikan menggunakan metode slicing.

```
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

5. Bagi data menjadi training data set dan testing data set

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25, random_state = 0)
```

6. Lakukan feature scaling, mendefinisikan variabel sc untuk melakukan feature scaling.

```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

7. Membuat decision tree menggunakan training data set. Definisikan variabel classifier untuk proses DTC. Pilih parameter ‘entropy’ agar data hasil pembagiannya bersifat homogen.

```
classifier = DecisionTreeClassifier(criterion = 'entropy',
random_state = 0)
classifier.fit(X_train, y_train)
```

8. Memprediksi hasil dari test data set.

```
y_pred = classifier.predict(X_test)
```

9. Membuat confusion matrix

```
cm = confusion_matrix(y_test, y_pred)
```

10. Visualisasi model decision tree menggunakan training data set

```
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Usia')
plt.ylabel('Estimasi Gaji')
plt.legend()
plt.show()
```

11. Visualisasi model decision tree menggunakan test data set

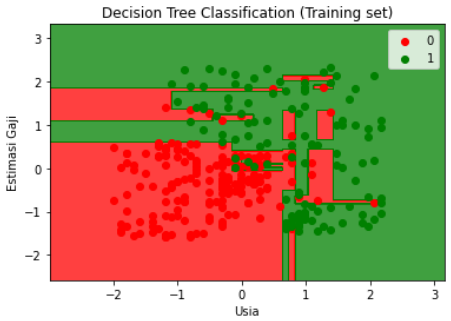
```
X_set, y_set = X_test, y_test
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop
= X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop
= X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.75, cmap = ListedColormap(('red',
'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Usia')
plt.ylabel('Estimasi Gaji')
plt.legend()
plt.show()
```

Hasil praktikum:

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
from matplotlib.colors import ListedColormap

dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.25, random_state = 0)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)

X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T)
             .reshape(X1.shape), alpha = 0.75, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)
plt.title('Decision Tree Classification (Training set)')
```

```
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label=j)
plt.title('Decision Tree Classification (Training set)')
plt.xlabel('Usia')
plt.ylabel('Estimasi Gaji')
plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in
case its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2-D array with a single row if yo
u intend to specify the same RGB or RGBA value for all points.
```




## DECISION TREE REGRESSION

1. Import library yang diperlukan

```
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
```

2. Membuat dataset random

```
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))
```

3. Fitting model regression

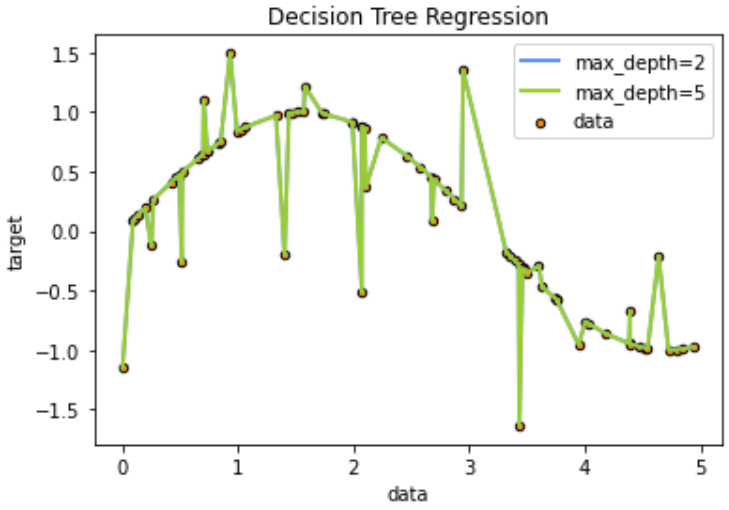
```
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X, y)
regr_2.fit(X, y)
```

4. Visualisasi hasil model

```
plt.figure()
plt.scatter(X, y, s=20, edgecolor="black", c="darkorange", label="data")
plt.plot(X_test, y_1, color="cornflowerblue", label="max_depth=2", linewidth=2)
plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=5", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```

Hasil praktikum:

In [2]:
```
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
X_test, y_1, y_2 = (X, y , y)
y[::5] += 3 * (0.5 - rng.rand(16))

regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X, y)
regr_2.fit(X, y)

plt.figure()
plt.scatter(X, y, s=20, edgecolor="black", c="darkorange", label="data")
plt.plot(X_test, y_1, color="cornflowerblue", label="max_depth=2", linewidth=2)
plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=5", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```




**Tugas**

1. Buatlah decision tree untuk klasifikasi keuntungan perusahaan menggunakan tabel di bawah ini.

| Usia Aset | Nilai Saing | Jenis | Keuntungan |
|---|---|---|---|
| Lama | Ada | Software | Menurun |
| Lama | Tidak Ada | Software | Menurun |
| Lama | Tidak Ada | Hardware | Menurun |
| Tengah | Ada | Software | Menurun |
| Tengah | Ada | Hardware | Menurun |
| Tengah | Tidak Ada | Hardware | Meningkat |
| Tengah | Tidak Ada | Software | Meningkat |
| Baru | Ada | Software | Meningkat |
| Baru | Tidak Ada | Hardware | Meningkat |
| Baru | Tidak Ada | Software | Meningkat |

*Jawaban:*

- Source code:

```
# TUGAS NO.1
# KEUNTUNGAN ASET PERUSAHAAN


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("aset.csv", sep=';')
dataset

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

dataset['usia_aset'] = enc.fit_transform(dataset['usia_aset'].values)
dataset['nilai_saing'] = enc.fit_transform(dataset['nilai_saing'].val
ues)
dataset['jenis'] = enc.fit_transform(dataset['jenis'].values)
dataset['keuntungan'] = enc.fit_transform(dataset['keuntungan'].value
s)

dataset

atr_dataset = dataset.drop(columns="keuntungan")
```

```
atr_dataset

cls_dataset = dataset['keuntungan']
cls_dataset

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classif
ication_report
from sklearn.tree import DecisionTreeClassifier

xtrain, xtest, ytrain, ytest = train_test_split(atr_dataset, cls_data
set, test_size=0.2, random_state=1)
tree_dataset = DecisionTreeClassifier(random_state=1)
tree_dataset.fit(xtrain, ytrain)

y_pred = tree_dataset.predict(xtest)
cm = confusion_matrix(ytest, y_pred)
print("Confusion Matrix")
print(cm)
akurasi = classification_report(ytest, y_pred)
print("Tingkat Akurasi ")
print("Akurasi:", akurasi)
akurasi = accuracy_score(ytest, y_pred)
print("Tingkat Akurasi: %d persen" %(akurasi*100))

from sklearn.tree import export_graphviz
export_graphviz(tree_dataset, out_file="aset.dot", class_names=["A","
B","C","D"],
                feature_names=atr_dataset.columns, impurity=False, fi
lled=True)

import graphviz

with open("aset.dot") as fig:
    dot_graph = fig.read()
    graph = graphviz.Source(dot_graph)

graph.view()
```

- Hasil output:

Tugas1_Job_7.ipynb

File Edit Lihat Sisipkan Runtime Fitur Bantuan Semua perubahan disimpan

Komentar Bagikan

+ Kode + Teks

RAM Disk Mengedit

```
# TUGAS NO.1
# KEUNTUNGAN ASET PERUSAHAAN


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("aset.csv", sep=';')
dataset

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

dataset['usia_aset'] = enc.fit_transform(dataset['usia_aset'].values)
dataset['nilai_saing'] = enc.fit_transform(dataset['nilai_saing'].values)
dataset['jenis'] = enc.fit_transform(dataset['jenis'].values)
dataset['keuntungan'] = enc.fit_transform(dataset['keuntungan'].values)

dataset

atr_dataset = dataset.drop(columns="keuntungan")
atr_dataset

cls_dataset = dataset['keuntungan']
cls_dataset

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
```

0 d selesai pada 18.58

Tugas1_Job_7.ipynb

File Edit Lihat Sisipkan Runtime Fitur Bantuan Semua perubahan disimpan

Komentar Bagikan

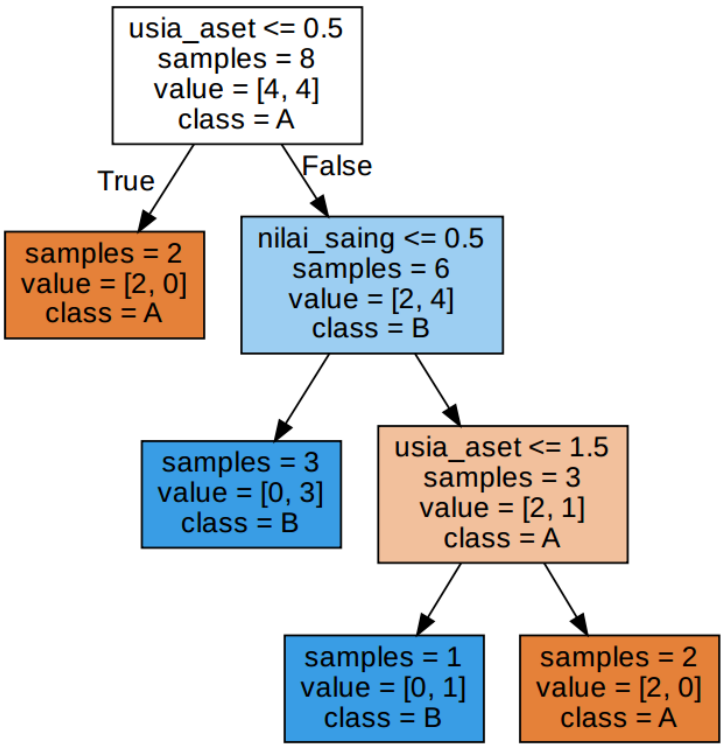
+ Kode + Teks

RAM Disk Mengedit

```
cls_dataset = dataset['keuntungan']
cls_dataset

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier

xtrain, xtest, ytrain, ytest = train_test_split(atr_dataset, cls_dataset, test_size=0.2, random_state=1)
tree_dataset = DecisionTreeClassifier(random_state=1)
tree_dataset.fit(xtrain, ytrain)

y_pred = tree_dataset.predict(xtest)
cm = confusion_matrix(ytest, y_pred)
print("Confusion Matrix")
print(cm)
akurasi = classification_report(ytest, y_pred)
print("Tingkat Akurasi ")
print("Akurasi:", akurasi)
akurasi = accuracy_score(ytest, y_pred)
print("Tingkat Akurasi: %d persen" %(akurasi*100))

from sklearn.tree import export_graphviz
export_graphviz(tree_dataset, out_file="aset.dot", class_names=["A","B","C","D"],
                feature_names=atr_dataset.columns, impurity=False, filled=True)

import graphviz

with open("aset.dot") as fig:
    dot_graph = fig.read()
    graph = graphviz.Source(dot_graph)

graph.view()
```

0 d selesai pada 18.58

```
Confusion Matrix
[[1 0]
 [0 1]]
Tingkat Akurasi
Akurasi:                precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

Tingkat Akurasi: 100 persen
'Source.gv.pdf'
```




2. Buatlah decision tree berdasaarkan tabel di bawah ini untuk menentukan risiko kredit dari calon nasabah, dengan kategori pendapatan 0k hingga 50k, 51k hingga 75k, dan lebih dari 75k.

| Nasabah | Tabungan | Aset | Pendapatan | Risiko Kredit |
|---|---|---|---|---|
| 1 | Sedang | Tinggi | 75k | Baik |
| 2 | Rendah | Rendah | 50k | Buruk |
| 3 | Tinggi | Sedang | 25k | Buruk |
| 4 | Sedang | Sedang | 50k | Baik |
| 5 | Rendah | Sedang | 100k | Baik |
| 6 | Tinggi | Tinggi | 25k | Baik |
| 7 | Rendah | Rendah | 25k | Buruk |
| 8 | Sedang | Sedang | 75k | Baik |

*Jawaban:*

- Source code:

```
# TUGAS NO.2
# RESIKO KREDIT


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("kredit.csv", sep=';')
dataset

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

dataset['nasabah'] = enc.fit_transform(dataset['nasabah'].values)
dataset['tabungan'] = enc.fit_transform(dataset['tabungan'].values)
dataset['aset'] = enc.fit_transform(dataset['aset'].values)
dataset['pendapatan'] = enc.fit_transform(dataset['pendapatan'].value
s)
dataset['resiko_kredit'] = enc.fit_transform(dataset['resiko_kredit']
.values)

dataset

atr_dataset = dataset.drop(columns="pendapatan")
```

```
atr_dataset

cls_dataset = dataset['pendapatan']
cls_dataset

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classif
ication_report
from sklearn.tree import DecisionTreeClassifier

xtrain, xtest, ytrain, ytest = train_test_split(atr_dataset, cls_data
set, test_size=0.2, random_state=1)
tree_dataset = DecisionTreeClassifier(random_state=1)
tree_dataset.fit(xtrain, ytrain)

y_pred = tree_dataset.predict(xtest)
cm = confusion_matrix(ytest, y_pred)
print("Confusion Matrix")
print(cm)
akurasi = classification_report(ytest, y_pred)
print("Tingkat Akurasi ")
print("Akurasi:", akurasi)
akurasi = accuracy_score(ytest, y_pred)
print("Tingkat Akurasi: %d persen" %(akurasi*100))

from sklearn.tree import export_graphviz
export_graphviz(tree_dataset, out_file="kredit.dot", class_names=["A"
,"B","C","D"],
                feature_names=atr_dataset.columns, impurity=False, fi
lled=True)

import graphviz

with open("kredit.dot") as fig:
    dot_graph = fig.read()
    graph = graphviz.Source(dot_graph)

graph.view()
```

- Hasil output:

Tugas2_Job7.ipynb

File Edit Lihat Sisipkan Runtime Fitur Bantuan Semua perubahan disimpan

Komentar Bagikan

+ Kode + Teks

RAM Disk Mengedit

```
# TUGAS NO.2
# RESIKO KREDIT


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv("kredit.csv", sep=';')
dataset

from sklearn.preprocessing import LabelEncoder

enc = LabelEncoder()

dataset['nasabah'] = enc.fit_transform(dataset['nasabah'].values)
dataset['tabungan'] = enc.fit_transform(dataset['tabungan'].values)
dataset['aset'] = enc.fit_transform(dataset['aset'].values)
dataset['pendapatan'] = enc.fit_transform(dataset['pendapatan'].values)
dataset['resiko_kredit'] = enc.fit_transform(dataset['resiko_kredit'].values)

dataset

atr_dataset = dataset.drop(columns="pendapatan")
atr_dataset

cls_dataset = dataset['pendapatan']
cls_dataset

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier
```

1 d selesai pada 19.16

Tugas2_Job7.ipynb

File Edit Lihat Sisipkan Runtime Fitur Bantuan Semua perubahan disimpan

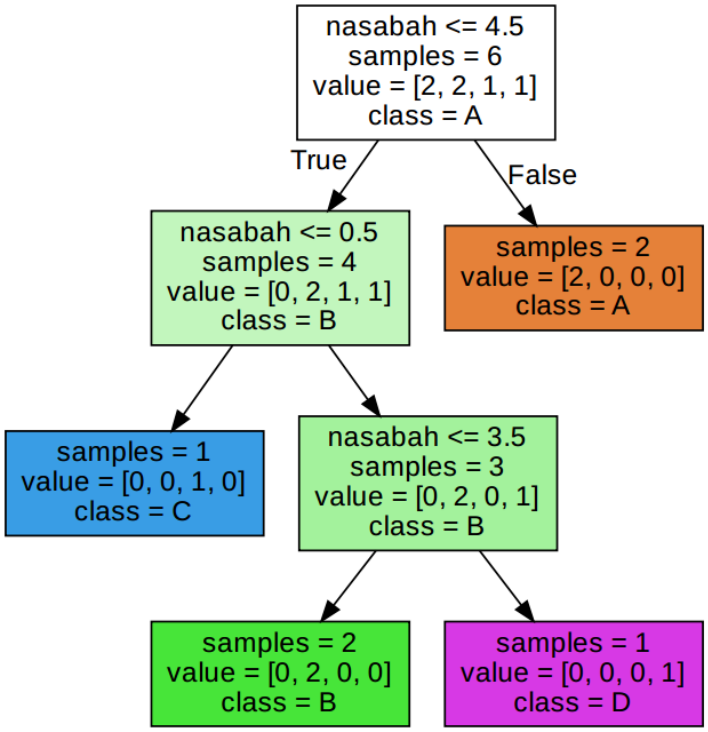
Komentar Bagikan

+ Kode + Teks

RAM Disk Mengedit

```
cls_dataset = dataset['pendapatan']
cls_dataset

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.tree import DecisionTreeClassifier

xtrain, xtest, ytrain, ytest = train_test_split(atr_dataset, cls_dataset, test_size=0.2, random_state=1)
tree_dataset = DecisionTreeClassifier(random_state=1)
tree_dataset.fit(xtrain, ytrain)

y_pred = tree_dataset.predict(xtest)
cm = confusion_matrix(ytest, y_pred)
print("Confusion Matrix")
print(cm)
akurasi = classification_report(ytest, y_pred)
print("Tingkat Akurasi ")
print("Akurasi:", akurasi)
akurasi = accuracy_score(ytest, y_pred)
print("Tingkat Akurasi: %d persen" %(akurasi*100))

from sklearn.tree import export_graphviz
export_graphviz(tree_dataset, out_file="kredit.dot", class_names=["A","B","C","D"],
                feature_names=atr_dataset.columns, impurity=False, filled=True)

import graphviz

with open("kredit.dot") as fig:
    dot_graph = fig.read()
    graph = graphviz.Source(dot_graph)

graph.view()
```

1 d selesai pada 19.16



```
Confusion Matrix
[[0 1 0]
 [0 0 0]
 [1 0 0]]
Tingkat Akurasi
Akurasi:               precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       0.0
           2       0.00      0.00      0.00       1.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0

Tingkat Akurasi: 0 persen
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 i
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: UndefinedMetricWarning: Recall and F-score are ill-defined and being set to 0.0 i
  _warn_prf(average, modifier, msg_start, len(result))
'Source.gv.pdf'
```

# Latihan soal

## Description of Task (1)

You are making your weekend plans and find out that your parents might come to town. You'd like to have plans in place, but there are a few unknown factors that will determine what you can, and can't do. Time for a decision tree.

First, you draw your decision box. This is the box that includes the event that starts your decision tree. In this case it is your parents coming to town. Out of that box, you have a branch for each possible outcome. In this example, it's easy: yes or no, either your parents come or they don't.

Your parents love the movies, so if they come to town, you will go to the cinema. Since the goal of the decision tree is to decide your weekend plans, you have an answer.

199

## Description of Task (2)

But, what about if your parents don't come to town? We can go back up to the 'no branch' from the decision box and finish our decision tree.

If your parents don't come to town, you need to decide what you are going to do. As you think of options, you realize the weather is an important factor. So, weather becomes youe next box. Since it's spring time, you know it will be rainy, sunny or windy. Those three possibilities become your branches.

If it's sunny or rainy, you know what you will do-play tennis or stay in, respectively. But what if it's windy? If it's windy, you want to get out of the house, but you probably won't be able to play tennis. You could either go for movie or shopping. What will determine whether you go for shopping or movie. Money will determine so it will become your branch. If you have enough money or you are rich go for shopping but if not, go for a movie.

200

# Description of Task (3)

Reading Decision node to each end node:

If the parents are visiting, then go to the cinema

Or

If the parents are not visiting and it is sunny, then play tennis

Or

If the parents are not visiting and it is windy and you're rich, then go for shopping

Or

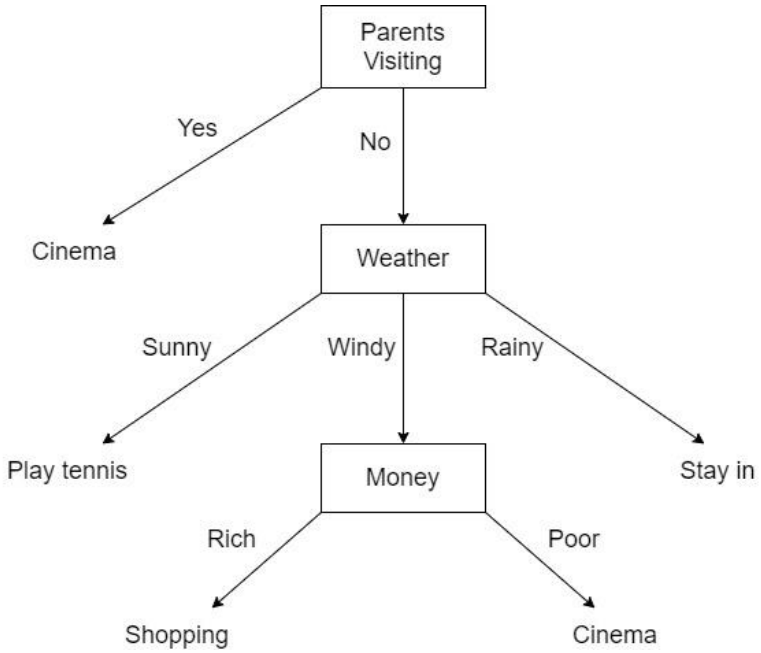
If the parents are not visiting and it is windy and you're poor, then go to cinema

Or

If the parents are not visiting and it is rainy, then stay in

201

*Jawaban:*




| Weekend | Weather | Parents | Money | Decision |
|---|---|---|---|---|
| W1 | Sunny | Yes | Rich | Cinema |
| W2 | Sunny | No | Rich | Tennis |
| W3 | Windy | Yes | Rich | Cinema |
| W4 | Rainy | Yes | Poor | Cinema |
| W5 | Rainy | No | Rich | Stay in |

| | | | | |
|---|---|---|---|---|
| W6 | Rainy | Yes | Poor | Cinema |
| W7 | Windy | No | Poor | Cinema |
| W8 | Windy | No | Rich | Shopping |
| W9 | Windy | Yes | Rich | Cinema |
| W10 | Sunny | No | Rich | Tennis |

**Entropy**

= - Pcinema log2(Pcinema)-Ptennis log2(Ptennis)-Pshopping log2(Pshopping)-Pstay_in log2(Pstay_in)

= - (6/10) * log2(6/10) -(2/10) * log2(2/10) - (1/10) * log2(1/10) - (1/10) * log2(1/10)

= - (6/10) * - 0.737 - (2/10) * -2.322 - (1/10) * -3.322 - (1/10) * -3.322

= 0.4422 + 0.4644 + 0.3322 + 0.3322

= 1.571

**Weather**

= 1.571 - (|Ssunny|/10)*Entropy(Ssunny) - (|Swindy|/10)*Entropy(Swindy) (|Srainy|/10) *Entropy(Srainy)

= 1.571 - (0.3)*Entropy(Ssunny) - (0.4)*Entropy(Swindy) - (0.3)*Entropy(Srainy)

= 1.571 - (0.3)*(0.918) - (0.4)*(0.81125) - (0.3)*(0.918)

= 0.70

**Parents**

= 1.571 - (|S yes|/10)*Entropy(S yes) - (|S no|/10)*Entropy(S no)

= 1.571 - (0.5) * 0 - (0.5) * 1.922 = 1.571 - 0.961

= 0.61

**Money**

= 1.571 - (|S rich|/10)*Entropy(S rich) - (|S poor|/10)*Entropy(S poor)

= 1.571 - (0.7) * (1.842) - (0.3) * 0 = 1.571 - 1.2894

= 0.2816

Pilih beberapa bagian tabel untuk diuji

| Weekend | Weather | Parents | Money | Decision |
|---|---|---|---|---|
| W1 | Sunny | Yes | Rich | Cinema |
| W2 | Sunny | No | Rich | Tennis |
| W10 | Sunny | No | Rich | Tennis |

**Sunny, Parents**

= 0.918 - (|S yes|/|S|)*Entropy(S yes) - (|S no|/|S|)*Entropy(S no)

= 0.918 - (1/3)*0 - (2/3)*0

= 0.918

**Sunny, Money**

= 0.918 - (|S rich|/|S|)*Entropy(S rich) - (|S poor|/|S|)*Entropy(S poor)

= 0.918 - (3/3)*0.918 - (0/3)*0

= 0.918 - 0.918

= 0