

# Kopekan UTS Matkul Coding & Machine Learning Open Book

By Lathif Ramadhan (5231811022)

## Modul Hal 12: 1.4. Tugas: Apa yg kalian ketahui tentang film tersebut

### Konteks Sejarah dan Latar Belakang

Pada masa Perang Dunia II, Alan Turing bekerja di Bletchley Park, pusat pemecahan kode rahasia Inggris. Tugas utama Turing adalah memecahkan kode Enigma, mesin yang digunakan oleh Jerman untuk mengirim pesan rahasia. Enigma dianggap mustahil dipecahkan karena memiliki miliaran kombinasi kode yang berubah setiap hari. Tantangan ini memaksa Turing untuk berpikir di luar batas metode konvensional.

Di sinilah konsep awal AI mulai muncul. Turing menyadari bahwa manusia tidak mungkin memecahkan kode Enigma secara manual dalam waktu singkat. Oleh karena itu, ia mengusulkan pembuatan sebuah mesin yang dapat melakukan pekerjaan tersebut secara otomatis. Mesin ini, yang disebut "Bombe," adalah cikal bakal dari komputer modern dan menjadi langkah pertama menuju pengembangan AI.

### Konsep AI dalam Film

Meskipun istilah "artificial intelligence" belum digunakan pada masa itu, konsep yang diusulkan oleh Turing dalam film ini mencerminkan prinsip-prinsip dasar AI. Berikut adalah beberapa aspek AI yang dapat dilihat dalam *The Imitation Game*:

- **Otomatisasi dan Pemrosesan Informasi**
- **Pembelajaran Mesin (Machine Learning)**
- **Kecerdasan Buatan sebagai Alat Bantu Manusia**
- **Etika dan Dampak Sosial**

### Alan Turing dan Konsep Turing Test

Salah satu kontribusi terbesar Alan Turing dalam bidang AI adalah konsep "Turing Test," yang diperkenalkan dalam makalahnya pada tahun 1950 berjudul *Computing Machinery and Intelligence*. Dalam makalah ini, Turing mengajukan pertanyaan: "Bisakah mesin berpikir?" Untuk menjawabnya, ia merancang sebuah tes di mana seorang manusia berinteraksi dengan mesin dan manusia lain melalui antarmuka teks. Jika manusia tersebut tidak dapat membedakan mana yang mesin dan mana yang manusia, maka mesin tersebut dianggap memiliki kecerdasan.

Meskipun Turing Test tidak secara eksplisit disebutkan dalam film, konsep ini sangat relevan dengan apa yang digambarkan dalam *The Imitation Game*. Turing percaya bahwa mesin dapat meniru kecerdasan manusia, dan ini adalah prinsip dasar dari AI.

### Relevansi AI Modern dengan *The Imitation Game*

Film ini memberikan gambaran tentang bagaimana konsep AI mulai berkembang dari upaya untuk memecahkan masalah yang sangat spesifik (seperti memecahkan kode Enigma) menjadi teknologi yang dapat diterapkan dalam berbagai bidang. Saat ini, AI digunakan dalam berbagai aplikasi, mulai dari asisten virtual seperti Siri dan Alexa, hingga sistem rekomendasi di platform seperti Netflix dan Amazon.

Namun, film ini juga mengingatkan kita tentang tantangan dan risiko yang terkait dengan AI. Turing sendiri menghadapi diskriminasi dan penganiayaan karena orientasi seksualnya, yang akhirnya menyebabkan kematiannya yang tragis. Ini adalah pengingat bahwa kemajuan teknologi harus disertai dengan kesadaran akan dampak sosial dan etika.

### Definisi, rumus, contoh kasus, cara kerja, kelebihan/kekurangan, serta kapan menggunakan

#### 1. Logistic Regression

Logistic Regression adalah algoritma **supervised learning** untuk klasifikasi biner (2 kelas) atau multikelas (>2 kelas) yang memprediksi probabilitas suatu instance termasuk dalam kelas tertentu menggunakan fungsi logistik (sigmoid).

#### Rumus Matematis

- **Fungsi Logistik (Sigmoid):**

$$P(Y = 1) = \frac{1}{1 + e^{-(b_0 + b_1 X_1 + b_2 X_2 + \dots + b_n X_n)}}$$

- $P(Y = 1)$ : Probabilitas kelas 1.
- $b_0$ : Intercept.
- $b_1, b_2, \dots, b_n$ : Koefisien regresi.
- $X_1, X_2, \dots, X_n$ : Fitur input.

- **Log-Odds (Logit Function):**

$$\ln \left( \frac{P(Y = 1)}{1 - P(Y = 1)} \right) = b_0 + b_1 X_1 + \dots + b_n X_n$$

#### Cara kerja:

1. **Transformasi Linear:**
  - a. Algoritma memulai dengan menghitung kombinasi linear dari fitur input:  $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n$
  - b. Dimana  $\beta$  adalah koefisien yang akan dipelajari, dan  $X$  adalah variabel input
2. **Fungsi Sigmoid:**
  - a. Hasil transformasi linear ( $z$ ) kemudian dimasukkan ke fungsi sigmoid:  $\sigma(z) = 1/(1+e^{-z})$
  - b. Fungsi ini mengubah nilai  $z$  menjadi probabilitas antara 0 dan 1
3. **Estimasi Parameter:**
  - a. Menggunakan Maximum Likelihood Estimation (MLE) untuk menemukan nilai  $\beta$  yang memaksimalkan probabilitas data observasi
  - b. Dilakukan melalui optimasi (biasanya Gradient Descent)
4. **Klasifikasi:**
  - a. Jika probabilitas  $>$  threshold (biasanya 0.5), data diklasifikasikan ke kelas 1
  - b. Jika probabilitas  $\leq$  threshold, data diklasifikasikan ke kelas 0
5. **Evaluasi Model:**
  - a. Menggunakan metrik seperti accuracy, precision, recall, dan ROC-AUC

#### Contoh Kasus Nyata

- **Klasifikasi risiko kredit** (memprediksi apakah pelanggan akan gagal bayar atau tidak).
- **Diagnosis medis** (prediksi penyakit berdasarkan gejala).
- **Marketing** (prediksi pelanggan yang akan membeli produk).

#### Dataset Contoh:

Usia	Pendapatan	Pinjaman	Default (Target)
25	30,000	50,000	Ya
40	60,000	10,000	Tidak

#### Kapan Menggunakan?

- Ketika target **kategorikal (binary/multiclass)**.
- Membutuhkan interpretasi koefisien untuk analisis (misal: pengaruh pendapatan terhadap risiko gagal bayar).
- Data tidak memiliki asumsi linearitas kuat (karena menggunakan transformasi logistik).

#### Kelebihan

- Mudah diinterpretasi (koefisien menunjukkan pengaruh fitur).
- Efisien untuk dataset kecil-menengah.
- Tidak memerlukan normalisasi fitur.

#### Kekurangan

- Hanya cocok untuk hubungan **linear** antara fitur dan log-odds.
- Rentan **overfitting** jika fitur terlalu banyak.
- Tidak bekerja baik untuk **non-linear decision boundaries**.

#### 2. Generalized Linear Models (GLM)

GLM adalah **ekstensi** dari **Linear Regression** yang mendukung berbagai distribusi error (tidak hanya Gaussian) dan link function untuk memodelkan hubungan antara fitur dan target.

#### Rumus Matematis

$$g(E(Y)) = b_0 + b_1 X_1 + \dots + b_n X_n$$

- $g(\cdot)$ : **Link function** (misal: logit, log, identity).
- $E(Y)$ : Nilai ekspektasi target.

#### Cara kerja:

1. **Pemilihan Komponen:**

- a. Memilih distribusi keluarga (Gaussian, Binomial, Poisson, Gamma, dll) sesuai sifat variabel respon
  - b. Menentukan fungsi link yang sesuai (identity, log, logit, inverse, dll)
2. **Pembentukan Model:**
- a. Membangun hubungan:  $g(E(Y)) = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n$
  - b. Dimana  $g()$  adalah fungsi link,  $E(Y)$  adalah nilai harapan variabel respon
3. **Estimasi Parameter:**
- a. Menggunakan Iteratively Reweighted Least Squares (IRLS) untuk estimasi parameter
  - b. Proses iteratif yang menyesuaikan bobot observasi berdasarkan residual
4. **Diagnosis Model:**
- a. Mengecek asumsi melalui analisis residual
  - b. Menghitung deviance untuk mengevaluasi kualitas fit model
5. **Prediksi:**
- a. Mengaplikasikan inverse link function untuk mendapatkan prediksi dalam skala asli

#### Contoh Link Functions:

- **Logit:** Untuk klasifikasi (Logistic Regression).
- **Log:** Untuk regresi Poisson (count data).
- **Identity:** Untuk Linear Regression (continuous data).

#### Contoh Kasus Nyata

- **Insurance Claim Prediction** (menggunakan distribusi Gamma).
- **Count Data Analysis** (misal: jumlah kunjungan website menggunakan Poisson Regression).
- **Binary Classification** (Logistic Regression termasuk GLM).

#### Dataset Contoh:

Usia	Klaim_Asuransi (Count)
30	2
45	5

#### Kapan Menggunakan?

- Ketika target **tidak terdistribusi normal** (misal: count, binary, skewed data).
- Membutuhkan fleksibilitas dalam pemilihan distribusi error (Binomial, Poisson, Gamma).

#### Cara Kerja di RapidMiner

1. **Pilih Operator:**
  - a. Gunakan **"Generalized Linear Model"** di RapidMiner.
2. **Konfigurasi:**
  - a. Pilih **family distribution** (Binomial, Poisson, Gaussian, dll.).
  - b. Tentukan **link function**.
3. **Evaluasi:**
  - a. Untuk regresi: **RMSE, R-squared**.
  - b. Untuk klasifikasi: **Confusion Matrix**.

#### Kelebihan

- Fleksibel (support berbagai jenis data).
- Cocok untuk data non-normal.
- Interpretasi koefisien mirip Linear Regression.

#### Kekurangan

- Membutuhkan pemilihan **family distribution** yang tepat.
- Kurang cocok untuk **high-dimensional data**.

### 3. Naive Bayes

Naive Bayes adalah algoritma **supervised learning** berbasis **teorema Bayes** dengan asumsi *independensi antar-fitur* (naive). Cocok untuk klasifikasi teks, spam detection, dan kasus dengan fitur kategori.

#### Rumus Matematis

#### Teorema Bayes:

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)}$$

- $P(Y|X)$ : Probabilitas kelas  $Y$  diberikan fitur  $X$  (*posterior*).
- $P(X|Y)$ : Probabilitas fitur  $X$  pada kelas  $Y$  (*likelihood*).
- $P(Y)$ : Probabilitas kelas  $Y$  (*prior*).
- $P(X)$ : Probabilitas fitur  $X$  (*evidence*).

#### Asumsi Naif (Independensi):

$$P(X_1, X_2, \dots, X_n|Y) = \prod_{i=1}^n P(X_i|Y)$$

#### Cara kerja:

1. **Perhitungan Prior Probability:**
  - a. Menghitung probabilitas prior setiap kelas  $P(Y)$  berdasarkan frekuensi kelas dalam data training
2. **Perhitungan Likelihood:**
  - a. Untuk setiap fitur  $X_i$ , menghitung probabilitas kondisional  $P(X_i|Y)$
  - b. Untuk data kontinu: asumsi distribusi normal, hitung mean dan variance
  - c. Untuk data diskrit: hitung frekuensi relatif
3. **Asumsi Independensi:**
  - a. Mengasumsikan semua fitur independen diberikan kelas
  - b. Menghitung joint likelihood sebagai produk marginal:  $P(X|Y) = \prod P(X_i|Y)$
4. **Applikasi Teorema Bayes:**
  - a. Menghitung posterior probability:  $P(Y|X) = [P(X|Y)P(Y)]/P(X)$
  - b.  $P(X)$  dihitung sebagai evidence dengan hukum total probability
5. **Prediksi:**

- a. Memilih kelas dengan posterior probability tertinggi (Maximum A Posteriori)
- b. Menggunakan log probabilities untuk menghindari underflow numerik

#### Contoh Kasus Nyata

- **Spam Filtering** (klasifikasi email spam vs. bukan).
- **Sentiment Analysis** (analisis ulasan produk: positif/negatif).
- **Diagnosis Penyakit** berdasarkan gejala.

#### Dataset Contoh:

Teks	Label (Target)
"Diskon 50% hari ini!"	Spam
"Meeting jam 10 pagi"	Bukan Spam

#### Kapan Menggunakan?

- Data dengan **fitur kategori** atau teks.
- Dataset besar dengan dimensi tinggi (e.g., NLP).
- Membutuhkan prediksi cepat dengan akurasi cukup.

#### Cara Kerja di RapidMiner

1. **Input Data:** Baca dataset (CSV/Excel).
2. **Preprocessing:**
  - a. Tokenisasi teks (jika menggunakan **Text Processing Extension**).
  - b. Transformasi fitur kategorikal ke numerik.
3. **Pilih Operator:**
  - a. Operators > Modeling > Classification > Naive Bayes.
4. **Evaluasi:**
  - a. Gunakan **Cross-Validation** dengan metrik *Accuracy, Precision, Recall*.

#### Kelebihan

- Cepat dan efisien untuk dataset besar.
- Handal untuk fitur independen (e.g., analisis teks).
- Tidak memerlukan tuning hyperparameter kompleks.

#### Kekurangan

- Gagal jika ada ketergantungan antar-fitur (*asumsi naif*).
- Kurang akurat untuk data numerik dengan distribusi kompleks.

### 4. Decision Tree

Decision Tree adalah algoritma **supervised learning** untuk klasifikasi/regresi yang membagi data secara rekursif berdasarkan nilai fitur hingga mencapai keputusan (daun/leaf).

#### Rumus Matematis

### Kriteria Pembagian (Split Criteria):

#### 1. Entropy & Information Gain (ID3/C4.5):

$$\text{Entropy}(S) = - \sum_{i=1}^c p_i \log_2 p_i$$

$$\text{Information Gain} = \text{Entropy}(S) - \sum_{v \in \text{Values}} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

#### 2. Gini Index (CART):

$$\text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2$$

### Cara kerja:

- Seleksi Fitur:
  - Menghitung impurity (Gini/Entropy) untuk setiap fitur
  - Memilih fitur dengan information gain tertinggi
- Pembagian Node:
  - Membagi data menjadi subset berdasarkan nilai fitur terpilih
  - Untuk kontinu: threshold optimal dipilih
  - Untuk kategorikal: setiap nilai kategori menjadi cabang
- Kriteria Berhenti:
  - Berhenti membagi jika:
    - Sebuah sampel di node termasuk kelas sama
    - Mencapai kedalaman maksimum
    - Jumlah sampel di node < threshold minimum
    - Tidak ada pembagian yang mengurangi impurity
- Pembentukan Leaf Node:
  - Node terminal menentukan prediksi
  - Klasifikasi: kelas mayoritas
  - Regresi: nilai rata-rata
- Pruning (Opsional):
  - Memangkas cabang yang tidak signifikan untuk hindari overfitting
  - Menggunakan error pada validation set

### Contoh Kasus Nyata

- Risk Assessment (pinjaman bank).
- Medical Diagnosis (klasifikasi penyakit).
- Customer Segmentation.

### Dataset Contoh:

Usia	Pendapatan	Pinjaman	Default (Target)
25	Rendah	Tinggi	Ya
40	Tinggi	Rendah	Tidak

### Kapan Menggunakan?

- Membutuhkan interpretasi model yang **transparan** (rules bisa divisualisasikan).
- Data memiliki **hubungan non-linear**.
- Fitur campuran (kategori & numerik).

### Cara Kerja di RapidMiner

- Pilih Operator:
  - Operators > Modeling > Classification > Decision Tree.
- Konfigurasi:
  - Pilih kriteria split (Gini, Information Gain).
  - Atur kedalaman maksimal (*max depth*).
- Visualisasi:
  - Gunakan **View Tree** untuk melihat struktur pohon.

### Kelebihan

- Mudah diinterpretasi (visualisasi pohon).
- Tidak memerlukan normalisasi data.
- Robust terhadap outlier.

### Kekurangan

- Rentan **overfitting** (perlu pruning/tuning depth).
- Kurang stabil (perubahan kecil data mengubah struktur pohon).

## 5. Neural Networks (Jaringan Saraf Tiruan)

Neural Networks adalah algoritma **non-linear** yang terinspirasi dari otak manusia, terdiri dari lapisan (*layers*) neuron untuk memodelkan pola kompleks.

### Rumus Matematis

#### Fungsi Aktivasi Neuron:

$$1. \text{Sigmoid: } \sigma(z) = \frac{1}{1+e^{-z}}$$

$$2. \text{ReLU: } f(z) = \max(0, z)$$

### Forward Propagation:

$$a^{(l)} = g(W^{(l)} \cdot a^{(l-1)} + b^{(l)})$$

- $a^{(l)}$ : Output layer ke- $l$ .
- $W^{(l)}$ : Weight matrix.
- $b^{(l)}$ : Bias.

### Cara kerja:

- Forward Propagation:

- Input data melewati serangkaian layer
- Setiap neuron menghitung:  $z = w \cdot x + b$
- Aplikasi fungsi aktivasi:  $a = \sigma(z)$

#### 2. Fungsi Aktivasi:

- Layer tersembunyi: ReLU, tanh, sigmoid
- Output layer: softmax (klasifikasi), linear (regresi)

#### 3. Perhitungan Loss:

- Menghitung error antara prediksi dan target
- Cross-entropy (klasifikasi), MSE (regresi)

#### 4. Backpropagation:

- Menghitung gradient loss terhadap setiap parameter
- Menggunakan chain rule dari kalkulus

#### 5. Optimasi:

- Update bobot dengan metode seperti SGD, Adam, RMSprop
- Learning rate mengontrol ukuran update

#### 6. Iterasi:

- Proses diulang untuk beberapa epoch
- Mini-batch processing untuk efisiensi

### Contoh Kasus Nyata

- Computer Vision (klasifikasi gambar).
- Natural Language Processing (terjemahan, chatbot).
- Prediksi Time Series (harga saham).

### Dataset Contoh:

Pixel1	Pixel2	...	Pixel784	Label (0-9)
0.12	0.45	...	0.88	7

### Kapan Menggunakan?

- Data sangat kompleks (gambar, teks, audio).
- Memiliki sumber daya komputasi besar (GPU/TPU).

### Cara Kerja di RapidMiner

#### 1. Pilih Operator:

- Operators > Modeling > Neural Networks > Deep Learning.

#### 2. Konfigurasi Arsitektur:

- Tambahkan lapisan (*Dense, Convolutional, LSTM*).
- Pilih **optimizer** (Adam, SGD) dan **loss function**.

#### 3. Training:

- Gunakan **GPU Acceleration** jika tersedia.

### Kelebihan

- Mampu memodelkan hubungan **non-linear kompleks**.
- State-of-the-art untuk data unstructured (gambar, teks).

### Kekurangan

- Butuh data besar** dan komputasi intensif.
- Black-box** (sulit diinterpretasi).

## 6. Linear Regression

Linear Regression adalah algoritma **supervised learning** untuk memodelkan hubungan linear antara variabel independen (fitur) dan variabel dependen (target kontinu). Digunakan untuk prediksi nilai numerik seperti harga, sales, dll.

### Rumus Matematis

#### Model Dasar:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n + \epsilon$$

- $Y$ : Variabel dependen (target).
- $\beta_0$ : Intercept.
- $\beta_1, \dots, \beta_n$ : Koefisien regresi.
- $X_1, \dots, X_n$ : Variabel independen (fitur).
- $\epsilon$ : Error term.

#### Optimasi (Ordinary Least Squares - OLS):

##### Minimalkan Sum of Squared Residuals (SSR):

$$SSR = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

### Cara kerja:

1. **Pembentukan Model:**
  - Membangun hubungan linear:  $\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$
  - Dimana  $\beta$  adalah parameter yang akan diestimasi
2. **Estimasi Parameter:**
  - Metode Ordinary Least Squares (OLS)
  - Meminimalkan sum of squared residuals:  $\sum (y_i - \hat{y}_i)^2$
  - Solusi analitik:  $\beta = (X^T X)^{-1} X^T y$
3. **Diagnosis Model:**
  - Mengecek linearitas, homoskedastisitas, normalitas residual
  - Menghitung  $R^2$  dan adjusted  $R^2$
4. **Prediksi:**
  - Menggunakan parameter terestimasi untuk prediksi baru
  - Interval kepercayaan dapat dihitung
5. **Regularisasi (Opsiional):**
  - Lasso (L1) dan Ridge (L2) untuk hindari overfitting
  - Menambahkan penalty term ke loss function

### Contoh Kasus Nyata

#### Aplikasi:

- Prediksi **harga rumah** berdasarkan luas, lokasi, kamar tidur.
- Forecast **penjualan produk** berdasarkan iklan dan musim.
- Analisis **pengaruh gizi** terhadap berat badan.

Luas (m <sup>2</sup> )	Kamar Tidur	Harga (Juta)
60	2	500
80	3	700

#### Kapan Menggunakan?

- Target adalah **variabel kontinu**.
- Hubungan antara fitur dan target **linear** (cek scatter plot).
- Membutuhkan interpretasi koefisien (misal: "Setiap penambahan 1m<sup>2</sup> meningkatkan harga RpX").

#### Cara Kerja di RapidMiner

1. **Input Data:** Baca dataset (CSV/Excel).
2. **Preprocessing:**
  - Handle missing values (Replace Missing Values).
  - Normalisasi jika fitur berbeda skala (Normalize).
3. **Pilih Operator:**
  - Operators > Modeling > Regression > Linear Regression.
4. **Evaluasi:**
  - RMSE (Root Mean Squared Error), R-squared.
  - Visualisasi residual plot.

#### Kelebihan

- Mudah diinterpretasi (koefisien jelas).
- Cepat dan efisien untuk dataset kecil.
- Dasar untuk banyak algoritma lanjut (e.g., Ridge/Lasso).

#### Kekurangan

- Sensitive terhadap outlier.**
- Gagal menangkap hubungan **non-linear**.
- Asumsi multicollinearity harus dipenuhi (fitur tidak berkorelasi tinggi).

## 7. K-Means Clustering

K-Means adalah algoritma **unsupervised learning** untuk mengelompokkan data ke dalam **k cluster** berbasis jarak Euclidean. Setiap cluster direpresentasikan oleh centroid (titik pusat).

### Rumus Matematis

#### Langkah Algoritma:

1. Pilih **k** centroid awal (acak atau heuristic).
2. Hitung jarak tiap titik ke centroid (**Euclidean Distance**):
$$d(x, c) = \sqrt{\sum_{i=1}^n (x_i - c_i)^2}$$
3. Kelompokkan titik ke cluster dengan centroid terdekat.
4. Update centroid sebagai rata-rata titik dalam cluster.
5. Ulangi hingga konvergensi (centroid stabil).

### Contoh Kasus Nyata

- **Segmentasi pelanggan** berdasarkan pembelian/demografi.
- **Image Compression** (pengelompokan warna dominan).
- **Anomaly detection** (data jauh dari centroid).

#### Dataset Contoh:

Pendapatan (Juta)	Pengeluaran (Juta)
5	3
20	15

#### Cara kerja:

1. **Inisialisasi:**
  - Memilih **k** centroid awal secara acak atau heuristic
2. **Assignment Step:**
  - Menghitung jarak setiap titik ke semua centroid
  - menggunakan Euclidean distance:  $d(x, c) = \sqrt{\sum (x_i - c_i)^2}$
  - Mengassign titik ke cluster dengan centroid terdekat
3. **Update Step:**
  - Menghitung ulang centroid sebagai mean semua titik dalam cluster
  - Centroid baru =  $(1/n) \sum x_i$  untuk semua  $x$  dalam cluster
4. **Konvergensi:**
  - Iterasi berlanjut hingga:
    - Centroid stabil (perubahan < threshold)
    - Mencapai maksimum iterasi
    - Tidak ada perubahan assignment
5. **Evaluasi:**
  - Menghitung within-cluster sum of squares (WCSS)
  - Elbow method untuk menentukan **k** optimal
6. **Output:**
  - Label cluster untuk setiap titik data
  - Posisi centroid final
  - Jarak setiap titik ke centroidnya

#### Kapan Menggunakan?

- Tidak ada label/target (**unsupervised**).
- Inisial eksplorasi pola data.
- Cluster berbentuk **spherical** dan ukuran seragam.

#### Cara Kerja di RapidMiner

1. **Pilih Operator:**
  - Operators > Modeling > Clustering > K-Means.
2. **Konfigurasi:**
  - Tentukan **jumlah cluster (k)** (gunakan Elbow Method).
  - Pilih **distance measure** (Euclidean/Manhattan).
3. **Evaluasi:**

- a. **Within-Cluster Sum of Squares (WCSS).**
- b. Visualisasi scatter plot dengan warna cluster.

#### Kelebihan

- Cepat dan scalable untuk dataset besar.
- Mudah diimplementasikan.
- Efektif untuk data terpisah jelas.

#### Kekurangan

- Harus tentukan **k** manual (Elbow Method bisa subjektif).
- Sensitif terhadap **outlier** dan inisialisasi centroid.
- Gagal untuk cluster **non-spherical** atau ukuran tidak seimbang.

## 8. Random Forest

#### Pengertian

Random Forest adalah algoritma **supervised learning** berbasis *ensemble* yang menggabungkan banyak *Decision Tree* (pohon keputusan) untuk meningkatkan akurasi dan stabilitas prediksi. Dengan teknik **bagging** (*Bootstrap Aggregating*) dan **feature randomness**, Random Forest mengurangi risiko *overfitting* yang umum terjadi pada *Decision Tree* tunggal. Cocok untuk **klasifikasi** dan **regresi**.

#### Rumus Matematis

##### 1. Bootstrap Aggregating (Bagging):

- Setiap *Decision Tree* dilatih pada subset data acak (*sampling with replacement*).
- Untuk dataset berukuran  $NN$ , subset diambil dengan ukuran sama ( $NN$ ), tetapi beberapa sampel mungkin terpilih berulang.

##### 2. Feature Randomness:

- Pada setiap *split*, hanya subset fitur ( $m$ ) yang dipertimbangkan:
  - Klasifikasi:  $m = \sqrt{\text{total fitur}}$
  - Regresi:  $m = \frac{\text{total fitur}}{3}$

##### 3. Prediksi Akhir:

- **Klasifikasi:** Voting mayoritas dari semua pohon.
- $$\hat{Y} = \text{mode}(\hat{Y}_1, \hat{Y}_2, \dots, \hat{Y}_k)$$
- **Regresi:** Rata-rata prediksi semua pohon.
- $$\hat{Y} = \frac{1}{k} \sum_{i=1}^k \hat{Y}_i$$
- $k$ : Jumlah pohon dalam *forest* (biasanya 100–500).

##### 4. Kriteria Split:

#### • Gini Index (klasifikasi):

$$\text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2$$

#### • Variance Reduction (regresi):

$$\text{Variance}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y})^2$$

#### Contoh Kasus Nyata

#### Aplikasi:

1. **Prediksi Risiko Kredit**
  - a. Mengklasifikasikan peminjam berisiko tinggi/rendah berdasarkan riwayat keuangan.
2. **Deteksi Penyakit**
  - a. Mengidentifikasi pasien diabetes berdasarkan gejala dan biomarker.
3. **Segmentasi Pelanggan**
  - a. Mengelompokkan pelanggan berdasarkan perilaku pembelian.

#### Dataset Contoh:

Usia	Pendapatan (Juta)	Pinjaman (Juta)	Riwayat Kredit	Default (Target)
25	5	50	Buruk	Ya
40	15	10	Baik	Tidak

#### Kapan Menggunakan?

##### ☒ Data Kompleks dengan Banyak Fitur

- Efektif untuk dataset high-dimensional (e.g., genetika, teks).
- **Membutuhkan Akurasi Tinggi**
- Lebih stabil dan akurat dibanding *Decision Tree* tunggal.
- **Robust terhadap Noise dan Outlier**
- Tidak memerlukan normalisasi data.
- **Feature Importance**
- Dapat mengidentifikasi fitur paling berpengaruh.

#### Cara Kerja

1. **Bootstrap Sampling:**
  - a. Ambil  $NN$  sampel acak dengan penggantian dari dataset.
2. **Pembangunan Pohon:**
  - a. Untuk setiap subset data:
    - i. Pilih subset fitur acak ( $m$ ) untuk setiap *split*.
    - ii. Bangun *Decision Tree* tanpa **pruning** (biarkan tumbuh maksimal).
3. **Aggregasi Prediksi:**
  - a. Gabungkan hasil prediksi semua pohon:

- i. **Klasifikasi:** Voting mayoritas.
- ii. **Regresi:** Rata-rata nilai prediksi.

#### 4. Evaluasi Model:

- a. Hitung **out-of-bag error (OOB)** sebagai estimasi akurasi.

#### Implementasi di RapidMiner

#### Langkah-Langkah:

1. **Baca Data:**
  - a. Gunakan operator **Read CSV** atau **Excel**.
2. **Preprocessing:**
  - a. Handle missing values: **Replace Missing Values**.
  - b. Konversi tipe data jika perlu: **Nominal to Numerical**.
3. **Pilih Operator:**
  - a. Operators > Modeling > Classification > Random Forest (klasifikasi).
  - b. Operators > Modeling > Regression > Random Forest (regresi).
4. **Konfigurasi Parameter:**
  - a. **Number of Trees:** 100–500 (default 100).
  - b. **Max Depth:** Kosongkan untuk biarkan pohon tumbuh penuh.
  - c. **Criterion:** Gini (klasifikasi), Variance (regresi).
5. **Training & Evaluasi:**
  - a. Gunakan **Cross-Validation** atau **Split Validation**.
  - b. Metrik evaluasi:
    - i. Klasifikasi: **Accuracy, Precision, Recall, AUC-ROC**.
    - ii. Regresi: **RMSE, R<sup>2</sup>**.
6. **Ekstrak Feature Importance:**
  - a. Extract Model → **Attribute Weights** untuk melihat kontribusi fitur.

#### Contoh Workflow:

[Read CSV] → [Replace Missing Values] → [Random Forest] → [Apply Model] → [Performance]

#### Kelebihan

- Akurasi Tinggi:** Kombinasi banyak pohon mengurangi varians.
- Robust:** Tahan terhadap overfitting dan noise data.
- Feature Importance:** Identifikasi fitur paling relevan.
- Handles Non-Linearity:** Cocok untuk hubungan kompleks antar-fitur.

#### Kekurangan

- Komputasi Intensif:** Semakin banyak pohon, semakin lambat.
- Black-Box:** Sulit diinterpretasi dibanding *Decision Tree* tunggal.
- Memory-Intensive:** Membutuhkan penyimpanan besar untuk ratusan pohon.

#### Perbandingan dengan Algoritma Lain

Algoritma	Akurasi	Interpretabilitas	Kecepatan	Handling Non-Linear Data
Random Forest	Tinggi	Sedang	Lambat	✓ Excellent
Decision Tree	Sedang	Tinggi	Cepat	✓ Good
Logistic Reg.	Rendah	Tinggi	Sangat Cepat	✗ Poor

## Kesimpulan

Random Forest adalah **algoritma powerful** untuk tugas klasifikasi dan regresi yang memadukan kekuatan banyak Decision Tree. Meskipun kompleks, algoritma ini menawarkan akurasi tinggi dan stabilitas yang sulit dikalahkan oleh model tunggal. Di RapidMiner, implementasinya mudah dengan dukungan **feature importance** dan parameter tuning yang fleksibel.

Gunakan Random Forest ketika:

- Anda membutuhkan akurasi tinggi tanpa khawatir tentang overfitting.
- Dataset memiliki banyak fitur atau hubungan non-linear.
- Interpretasi model tidak menjadi prioritas utama.

Untuk kasus yang membutuhkan interpretasi mendalam (e.g., regulasi ketat), pertimbangkan **Decision Tree** atau **Logistic Regression**.

---

**Perhatikan parameter-parameter yang dimiliki oleh algoritma-algoritma pembelajaran terbimbing yang telah dijelaskan pada bab 2 ini, berikan penjelasan makna dari parameter-parameter tersebut!**

## A. Regresi Linear

Penjelasan Parameter GLM di RapidMiner:

### 1. Feature Selection

- Ini adalah parameter tingkat ahli yang menentukan metode seleksi fitur yang akan digunakan selama regresi. Ada beberapa opsi yang tersedia:
  - **none**: Tidak menggunakan seleksi fitur.
  - **M5 prime**: Menggunakan metode M5 prime untuk seleksi fitur.
  - **greedy**: Menggunakan metode greedy untuk seleksi fitur.

- **T-Test**: Menggunakan uji statistik T-Test untuk seleksi fitur.
- **Iterative T-Test**: Menggunakan T-Test secara iteratif untuk seleksi fitur.
- Seleksi fitur membantu memilih fitur yang paling relevan untuk model, sehingga meningkatkan performa dan mengurangi kompleksitas.

### 2. Alpha

- Parameter ini hanya tersedia jika feature selection diatur ke T-Test. Ini menentukan nilai alpha (tingkat signifikansi) yang digunakan dalam uji T-Test.
- Nilai alpha yang lebih kecil akan membuat seleksi fitur lebih ketat, hanya memilih fitur yang sangat signifikan.

### 3. Max Iterations

- Parameter ini hanya tersedia jika feature selection diatur ke iterative T-Test. Ini menentukan jumlah maksimum iterasi yang dilakukan dalam proses seleksi fitur iteratif.
- Iterasi yang lebih banyak memungkinkan algoritma untuk mengevaluasi lebih banyak kombinasi fitur, tetapi juga membutuhkan waktu lebih lama.

### 4. Forward Alpha

- Parameter ini hanya tersedia jika feature selection diatur ke iterative T-Test. Ini menentukan nilai alpha yang digunakan saat menambahkan fitur ke model (forward selection).
- Mengontrol seberapa ketat algoritma dalam menambahkan fitur baru selama iterasi.

### 5. Backward Alpha

- Parameter ini hanya tersedia jika feature selection diatur ke iterative T-Test. Ini menentukan nilai alpha yang digunakan saat menghapus fitur dari model (backward selection).
- Mengontrol seberapa ketat algoritma dalam menghapus fitur yang tidak signifikan selama iterasi.

### 6. Eliminate Colinear Features

- Parameter ini menentukan apakah algoritma akan mencoba menghapus fitur yang memiliki kolinearitas tinggi (korelasi tinggi) selama regresi.
- Menghapus fitur yang berkorelasi tinggi dapat meningkatkan stabilitas model dan mengurangi overfitting.

### 7. Min Tolerance

- Parameter ini hanya tersedia jika eliminate colinear features diatur ke true. Ini menentukan toleransi minimum untuk menghapus fitur yang berkolinear.
- Nilai toleransi yang lebih kecil akan membuat algoritma lebih ketat dalam menghapus fitur yang berkorelasi.

### 8. Use Bias

- Parameter ini menentukan apakah model akan menghitung nilai intercept atau tidak.
- Jika diaktifkan, model akan lebih fleksibel dalam menyesuaikan data. Jika tidak, garis regresi akan dipaksa melalui origin (0,0).

### 9. Ridge

- Parameter ini menentukan nilai parameter ridge yang digunakan dalam ridge regression.
- Ridge regression membantu mengurangi overfitting dengan menambahkan penalti pada koefisien model. Nilai ridge yang lebih besar akan meningkatkan efek regularisasi.

## B. Logistic Regression

Penjelasan Parameter GLM di RapidMiner:

### 1. Kernel Type

- Parameter ini memilih jenis fungsi kernel yang akan digunakan. Opsi yang tersedia:
  - **dot**: Kernel dot didefinisikan sebagai  $k(x,y) = x \cdot y$  (produk dalam dari  $xx$  dan  $yy$ ).
  - **radial**: Kernel radial didefinisikan sebagai  $\exp(-g\|x-y\|^2)$ , di mana  $gg$  adalah  $\gamma$ .
  - **polynomial**: Kernel polinomial didefinisikan sebagai  $k(x,y) = (x \cdot y + 1)^d$ , di mana  $dd$  adalah derajat polinomial.
  - **sigmoid**: Kernel sigmoid didefinisikan sebagai  $\tanh(a \cdot x + b)$ , di mana  $aa$  adalah  $\alpha$  dan  $bb$  adalah konstanta intercept.
  - **anova**: Kernel anova didefinisikan sebagai  $(\sum \exp(-g(x-y)))^d / (\sum \exp(-g(x-y)))$ , di mana  $gg$  adalah  $\gamma$  dan  $dd$  adalah derajat.
  - **epachnenikov**: Kernel Epanechnikov didefinisikan sebagai  $34(1-u^2)43(1-u^2)$  untuk  $uu$  antara -1 dan 1.
  - **gaussian combination**: Kernel kombinasi Gaussian.
  - **multiquadric**: Kernel multiquadric didefinisikan sebagai  $\|x-y\|^2 + c_2\|x-y\|^2 + c_2$ .
- Pemilihan kernel memengaruhi bagaimana data dipetakan ke ruang fitur yang lebih tinggi, yang dapat meningkatkan performa model.

### 2. Kernel Gamma

- Parameter  $\gamma$  untuk kernel radial atau anova.
- $\gamma$  mengontrol seberapa jauh pengaruh satu titik data terhadap titik lainnya. Nilai  $\gamma$  yang besar membuat model lebih sensitif terhadap data.

### 3. Kernel Sigma1, Sigma2, Sigma3

- Parameter sigma untuk kernel epachnenikov, gaussian combination, atau multiquadric.
- Parameter ini mengontrol bentuk dan skala dari kernel.

#### 4. Kernel Shift

- Parameter shift untuk kernel multiquadric.
- Menggeser kernel untuk menyesuaikan distribusi data.

#### 5. Kernel Degree

- Parameter derajat untuk kernel polynomial, anova, atau epachnenikov.
- Derajat yang lebih tinggi memungkinkan model untuk menangkap hubungan yang lebih kompleks dalam data.

#### 6. Kernel A dan B

- Parameter alpha ( $a$ ) dan intercept ( $b$ ) untuk kernel sigmoid.
- Parameter ini mengontrol bentuk dari fungsi sigmoid.

#### 7. C (Complexity Constant)

- Konstanta kompleksitas yang mengatur toleransi untuk misklasifikasi. Nilai C yang lebih tinggi memungkinkan batas yang lebih "lunak", sedangkan nilai C yang lebih rendah menciptakan batas yang lebih "keras".
- Nilai C yang terlalu besar dapat menyebabkan overfitting, sedangkan nilai yang terlalu kecil dapat menyebabkan underfitting.

#### 8. Start Population Type

- Menentukan jenis inisialisasi populasi awal.
- Inisialisasi populasi yang baik dapat mempercepat konvergensi algoritma.

#### 9. Max Generations

- Menentukan jumlah generasi maksimum sebelum algoritma dihentikan.
- Mengontrol berapa lama algoritma akan berjalan.

#### 10. Generations Without Improval

- Menentukan kriteria berhenti awal, yaitu berhenti setelah n generasi tanpa peningkatan performa.
- Membantu menghentikan algoritma jika tidak ada peningkatan lebih lanjut.

#### 11. Population Size

- Menentukan ukuran populasi, yaitu jumlah individu per generasi. Jika diatur ke -1, semua contoh akan dipilih.
- Ukuran populasi yang lebih besar dapat meningkatkan diversitas solusi, tetapi juga membutuhkan lebih banyak komputasi.

#### 12. Tournament Fraction

- Menentukan fraksi populasi saat ini yang akan digunakan sebagai anggota turnamen.
- Mengontrol seleksi individu untuk reproduksi.

#### 13. Keep Best

- Menentukan apakah individu terbaik harus bertahan ke generasi berikutnya (elitist selection).
- Memastikan solusi terbaik tidak hilang selama evolusi.

#### 14. Mutation Type

- Menentukan jenis operator mutasi.
- Mutasi membantu menjaga diversitas populasi.

#### 15. Selection Type

- Menentukan skema seleksi untuk algoritma evolusioner.
- Skema seleksi memengaruhi bagaimana individu dipilih untuk reproduksi.

#### 16. Crossover Prob

- Menentukan probabilitas individu untuk dipilih untuk crossover.
- Mengontrol seberapa sering crossover terjadi.

#### 17. Use Local Random Seed

- Menentukan apakah seed acak lokal akan digunakan untuk randomisasi.
- Menggunakan seed yang sama akan menghasilkan randomisasi yang sama, yang berguna untuk reproduktibilitas.

#### 18. Local Random Seed

- Menentukan seed acak lokal. Hanya tersedia jika use local random seed diatur ke true.
- Memastikan hasil yang konsisten.

#### 19. Show Convergence Plot

- Menentukan apakah plot konvergensi akan ditampilkan.
- Membantu memantau performa algoritma selama pelatihan.

### C. Generalized Linear Models (GLM)

Penjelasan Parameter GLM di RapidMiner:

#### 1. Family

- Parameter ini menentukan distribusi keluarga eksponensial yang digunakan. Opsi yang tersedia:
  - **AUTO**: Pemilihan otomatis. Menggunakan multinomial untuk label polinomial, binomial untuk label binomial, dan gaussian untuk label numerik.
  - **gaussian**: Data harus numerik (real atau integer).
  - **binomial**: Data harus binomial atau polinomial dengan 2 level/kelas.
  - **multinomial**: Data harus polinomial dengan lebih dari dua level/kelas.
  - **poisson**: Data harus numerik dan non-negatif (integer).
  - **gamma**: Data harus numerik, kontinu, dan positif (real atau integer).
  - **tweedie**: Data harus numerik, kontinu (real), dan non-negatif.
- Pemilihan family yang tepat sangat penting untuk memastikan model sesuai dengan jenis data yang digunakan.

#### 2. Solver

- Memilih solver yang digunakan untuk optimasi. Opsi yang tersedia:
  - **AUTO**: Pemilihan otomatis.
  - **IRLSM**: Cepat untuk masalah dengan jumlah prediktor kecil dan untuk pencarian lambda dengan penalti L1.
  - **L\_BFGS**: Lebih baik untuk dataset dengan banyak kolom.
  - **COORDINATE\_DESCENT** (eksperimental): IRLSM dengan pembaruan kovarians.
  - **COORDINATE\_DESCENT\_NAIVE** (eksperimental): IRLSM dengan pembaruan naif.
- Pemilihan solver memengaruhi kecepatan dan efisiensi model.

#### 3. Link

- Fungsi link menghubungkan prediktor linear dengan fungsi distribusi. Opsi yang tersedia:
  - **family\_default**: Menggunakan link kanonik untuk family yang dipilih.
  - **identity**: Untuk family gaussian, poisson, dan gamma.
  - **log**: Untuk family gaussian, poisson, dan gamma.
  - **inverse**: Untuk family gaussian dan gamma.
- Fungsi link yang tepat memastikan hubungan yang sesuai antara prediktor dan distribusi.

#### 4. Reproducible

- Membuat pembangunan model dapat direproduksi. Jika diaktifkan, parameter **maximum\_number\_of\_threads** mengontrol tingkat paralelisme.
- Memastikan hasil yang konsisten saat model dibangun ulang.

## 5. Maximum Number of Threads

- Mengontrol tingkat paralelisme dalam pembangunan model.
- Menentukan seberapa banyak sumber daya komputasi yang digunakan.

## 6. Specify Beta Constraints

- Jika diaktifkan, batasan beta untuk atribut reguler dapat diberikan.
- Membatasi nilai koefisien untuk memastikan interpretasi yang lebih baik.

## 7. Use Regularization

- Mengaktifkan regularisasi. Jika diaktifkan, parameter lambda, alpha, dan pencarian lambda dapat ditentukan.
- Regularisasi membantu mengurangi overfitting.

## 8. Lambda

- Parameter lambda mengontrol jumlah regularisasi yang diterapkan. Jika lambda adalah 0.0, tidak ada regularisasi yang diterapkan.
- Nilai lambda yang lebih besar meningkatkan efek regularisasi.

## 9. Lambda Search

- Jika diaktifkan, pencarian lambda akan dilakukan mulai dari lambda maksimum.
- Membantu menemukan nilai lambda yang optimal.

## 10. Number of Lambdas

- Jumlah nilai lambda yang akan dicari jika lambda search diaktifkan.
- Menentukan seberapa banyak nilai lambda yang akan dievaluasi.

## 11. Lambda Min Ratio

- Nilai lambda minimum sebagai fraksi dari lambda.max.
- Mengontrol rentang nilai lambda yang akan dicari.

## 12. Early Stopping

- Menghentikan pencarian lambda lebih awal berdasarkan parameter stopping rounds dan stopping tolerance.

- Menghemat waktu komputasi jika model sudah konvergen.

## 13. Stopping Rounds

- Jumlah iterasi tanpa peningkatan sebelum pencarian dihentikan.
- Mengontrol kapan pencarian dihentikan.

## 14. Stopping Tolerance

- Toleransi relatif untuk kriteria penghentian berbasis metrik.
- Menentukan seberapa kecil peningkatan yang dianggap signifikan.

## 15. Alpha

- Parameter alpha mengontrol distribusi antara penalti L1 (Lasso) dan L2 (Ridge regression). Nilai 1.0 untuk alpha menghasilkan Lasso, sedangkan nilai 0.0 menghasilkan Ridge regression.
- Mengontrol jenis regularisasi yang diterapkan.

## 16. Standardize

- Menstandarisasi kolom numerik untuk memiliki mean nol dan varian satu.
- Memastikan semua fitur memiliki skala yang sama.

## 17. Non-Negative Coefficients

- Membatasi koefisien (bukan intercept) untuk menjadi non-negatif.
- Berguna untuk interpretasi model yang lebih baik.

## 18. Compute P-Values

- Menghitung nilai p. Hanya bekerja dengan solver IRLSM dan tanpa regularisasi.
- Memberikan informasi statistik tentang signifikansi koefisien.

## 19. Remove Collinear Columns

- Menghapus kolom yang bergantung linear jika ada.
- Meningkatkan stabilitas model.

## 20. Add Intercept

- Menambahkan konstanta ke model.
- Memungkinkan model untuk lebih fleksibel.

## 21. Missing Values Handling

- Menangani nilai yang hilang. Opsi yang tersedia:
  - Melewatan nilai yang hilang.
- Mengganti nilai yang hilang dengan nilai rata-rata.
- Memastikan data yang digunakan lengkap.

## 22. Max Iterations

- Jumlah maksimum iterasi untuk pelatihan model.
- Mengontrol berapa lama algoritma akan berjalan.

## 23. Beta Constraints

- Batasan untuk nilai beta. Setiap baris berisi nama atribut, kategori, batas bawah, batas atas, dan nilai beta yang diberikan.
- Membatasi nilai koefisien untuk interpretasi yang lebih baik.

## 24. Max Runtime Seconds

- Waktu maksimum yang diizinkan untuk pelatihan model dalam detik.
- Mengontrol durasi pelatihan.

## 25. Expert Parameters

- Parameter ini untuk penyesuaian lebih lanjut. Biasanya nilai default sudah cukup baik.
- Untuk penyesuaian lebih lanjut jika diperlukan.

## D. Naïve Bayes

Penjelasan Parameter GLM di RapidMiner:

### 1. Laplace Correction

- Kesederhanaan Naive Bayes memiliki kelemahan: jika dalam data pelatihan, suatu nilai atribut tertentu tidak pernah muncul dalam konteks kelas tertentu, maka probabilitas bersyarat  $(P(\text{Atribut}|\text{Kelas})P(\text{Atribut}|\text{Kelas}))$  akan diatur menjadi nol. Ketika nilai nol ini dikalikan dengan probabilitas lain, hasilnya juga akan menjadi nol, dan prediksi yang dihasilkan akan menyesatkan. Laplace Correction adalah trik sederhana untuk menghindari masalah ini dengan menambahkan 1 ke setiap hitungan kemunculan, sehingga tidak ada probabilitas yang bernilai nol.
- Laplace Correction memastikan bahwa tidak ada probabilitas yang bernilai nol, sehingga prediksi model tetap valid dan akurat.

## E. Decision Tree

Penjelasan Parameter GLM di RapidMiner:

### 1. Criterion

- Memilih kriteria yang digunakan untuk memilih atribut yang akan digunakan untuk pemisahan (splitting). Kriteria yang tersedia:
  - **information gain**: Entropi dari semua atribut dihitung, dan atribut dengan entropi terkecil dipilih untuk pemisahan. Metode ini cenderung memilih atribut dengan banyak nilai.
  - **gain ratio**: Variasi dari information gain yang menyesuaikan gain untuk setiap atribut agar memperhitungkan luas dan keseragaman nilai atribut.
  - **gini index**: Mengukur ketidaksetaraan antara distribusi karakteristik label. Pemisahan pada atribut yang dipilih akan mengurangi rata-rata indeks gini dari subset yang dihasilkan.
  - **accuracy**: Atribut dipilih untuk pemisahan yang memaksimalkan akurasi dari seluruh pohon.
  - Atribut dipilih untuk pemisahan yang meminimalkan jarak kuadrat antara rata-rata nilai dalam node dengan nilai sebenarnya.
- Kriteria ini menentukan bagaimana algoritma memilih atribut terbaik untuk memisahkan data.

## 2. Maximal Depth

- Kedalaman pohon bervariasi tergantung pada ukuran dan karakteristik ExampleSet. Parameter ini digunakan untuk membatasi kedalaman pohon keputusan. Jika nilainya diatur ke '1', tidak ada batasan kedalaman pohon. Jika diatur ke '1', pohon dengan satu node (akar) akan dihasilkan.
- Membatasi kedalaman pohon membantu mencegah overfitting.

## 3. Apply Pruning

- Model pohon keputusan dapat dipangkas (pruning) setelah pembuatan. Jika diaktifkan, beberapa cabang akan diganti dengan daun sesuai dengan parameter confidence.
- Pruning membantu menyederhanakan pohon dan mencegah overfitting.

## 4. Confidence

- Parameter ini menentukan tingkat kepercayaan yang digunakan untuk perhitungan kesalahan pesimis saat pruning.
- Nilai confidence yang lebih tinggi akan menghasilkan pemangkasan yang lebih agresif.

## 5. Apply Prepruning

- Parameter ini menentukan apakah kriteria penghentian tambahan selain maximal depth akan digunakan selama pembuatan model pohon keputusan. Jika diaktifkan, parameter minimal gain, minimal leaf size, minimal size for split, dan number of prepruning alternatives akan digunakan sebagai kriteria penghentian.

- Prepruning membantu menghentikan pembuatan pohon lebih awal jika kriteria tertentu terpenuhi.

## 6. Minimal Gain

- Gain dari sebuah node dihitung sebelum pemisahan. Node akan dipisah jika gain-nya lebih besar dari minimal gain. Nilai minimal gain yang lebih tinggi menghasilkan lebih sedikit pemisahan dan pohon yang lebih kecil.
- Mengontrol seberapa signifikan pemisahan harus dilakukan.

## 7. Minimal Leaf Size

- Ukuran daun adalah jumlah contoh (examples) dalam subset-nya. Pohon dibuat sedemikian rupa sehingga setiap daun memiliki setidaknya jumlah contoh sesuai dengan minimal leaf size.
- Memastikan bahwa setiap daun memiliki cukup data untuk generalisasi yang baik.

## 8. Minimal Size for Split

- Ukuran node adalah jumlah contoh dalam subset-nya. Hanya node yang ukurannya lebih besar atau sama dengan minimal size for split yang akan dipisah.
- Mencegah pemisahan pada node yang terlalu kecil.

## 9. Number of Prepruning Alternatives

- Ketika pemisahan dicegah oleh prepruning pada suatu node, parameter ini akan menyesuaikan jumlah node alternatif yang diuji untuk pemisahan. Ini terjadi karena prepruning berjalan paralel dengan proses pembuatan pohon.
- Mencoba node alternatif jika pemisahan pada node tertentu tidak meningkatkan kekuatan diskriminatif pohon.

## F. Neural Nets

Penjelasan Parameter GLM di RapidMiner:

## 1. Hidden Layers (Lapisan Tersembunyi)

Parameter ini menjelaskan nama dan ukuran semua lapisan tersembunyi dalam jaringan saraf. Pengguna dapat menentukan struktur jaringan saraf menggunakan parameter ini.

- Setiap entri dalam daftar menunjukkan satu lapisan tersembunyi baru.
- Setiap entri membutuhkan nama dan ukuran lapisan tersembunyi.
- Nama lapisan dapat dipilih secara bebas, hanya digunakan untuk menampilkan model.

- Jumlah node sebenarnya akan lebih satu dari ukuran yang ditentukan, karena ada node tambahan yang ditambahkan ke setiap lapisan. Node ini tidak terhubung ke lapisan sebelumnya.
- Jika ukuran lapisan tersembunyi diatur ke **-1**, ukuran lapisan akan dihitung berdasarkan jumlah atribut dalam dataset. Dalam hal ini, ukuran lapisan akan menjadi:  $(\text{Jumlah atribut} + \text{jumlah kelas}) / 2 + 1$ .
- Jika pengguna tidak menentukan lapisan tersembunyi, maka secara default akan dibuat satu lapisan tersembunyi dengan fungsi aktivasi **sigmoid** dan ukuran yang dihitung dengan rumus di atas.
- Jika hanya ada satu lapisan tanpa node, maka node input akan langsung terhubung ke node output, dan tidak ada lapisan tersembunyi yang digunakan.

## 2. Training Cycles (Siklus Pelatihan)

- Parameter ini menentukan jumlah siklus pelatihan yang digunakan dalam pelatihan jaringan saraf.
- Dalam metode **back-propagation**, nilai output dibandingkan dengan jawaban yang benar untuk menghitung nilai dari fungsi error tertentu.
- Error ini kemudian dikembalikan ke jaringan, dan algoritma menggunakan informasi tersebut untuk menyesuaikan bobot setiap koneksi guna mengurangi error sedikit demi sedikit.
- Proses ini diulang sebanyak  $n$  kali, di mana  $n$  ditentukan oleh parameter ini.

## 3. Learning Rate (Laju Pembelajaran)

- Parameter ini menentukan seberapa besar perubahan bobot pada setiap langkah pembelajaran.
- Nilainya tidak boleh 0, karena itu akan membuat jaringan saraf tidak belajar.

## 4. Momentum

- Momentum menambahkan sebagian kecil dari pembaruan bobot sebelumnya ke pembaruan bobot saat ini.
- Ini membantu mencegah algoritma terjebak dalam maksimum lokal dan membuat proses optimasi lebih stabil.

## 5. Decay (Pelarutan Learning Rate)

- Ini adalah parameter untuk pengguna tingkat lanjut.
- Jika diaktifkan, laju pembelajaran akan berkurang secara bertahap selama proses pembelajaran.

## 6. Shuffle (Mengacak Data)

- Ini juga merupakan parameter untuk pengguna tingkat lanjut.
- Jika diaktifkan, data input akan diacak sebelum pelatihan dimulai.

- Meskipun meningkatkan penggunaan memori, tetapi disarankan jika data sebelumnya sudah dalam urutan tertentu untuk menghindari bias pelatihan.

## 7. Normalize (Normalisasi Data)

- Neural Net di RapidMiner menggunakan fungsi aktivasi sigmoid, sehingga rentang nilai atribut sebaiknya berada dalam kisaran -1 hingga +1.
- Jika parameter ini diaktifkan, normalisasi akan dilakukan sebelum pelatihan.
- Meskipun meningkatkan waktu komputasi, tetapi normalisasi sangat disarankan dalam kebanyakan kasus untuk meningkatkan akurasi model.

## 8. Error Epsilon

- Proses optimasi akan dihentikan jika error pelatihan turun di bawah nilai epsilon yang ditentukan dalam parameter ini.

## 9. Use Local Random Seed (Gunakan Seed Acak Lokal)

- Jika diaktifkan, algoritma akan menggunakan seed acak lokal untuk randomisasi.
- Ini berguna untuk mereproduksi hasil yang sama setiap kali model dijalankan.

## 10. Local Random Seed (Seed Acak Lokal)

- Parameter ini menentukan nilai seed acak lokal.
- Hanya akan tersedia jika parameter Use Local Random Seed diaktifkan.

### Penjelasan Implementasi Naive Bayes di RapidMiner

#### a. Retrieve Deals

- Langkah pertama adalah mengambil dataset Deals yang akan digunakan untuk melatih model.
- Dataset ini berisi informasi tentang transaksi pelanggan, seperti age, gender, dan apakah mereka menjadi pelanggan di masa depan atau tidak.
- Dataset ini akan digunakan untuk melatih model Naive Bayes agar bisa memprediksi kelas (yakni, "No" atau "Yes") berdasarkan fitur-fitur yang ada.

#### b. Naive Bayes

- Di sini, model Naive Bayes dibangun menggunakan dataset yang telah diambil. Naive Bayes adalah algoritma klasifikasi yang menggunakan probabilitas bersyarat untuk memprediksi kelas.
- **Cara Kerja:**

- Algoritma menghitung probabilitas setiap kelas berdasarkan fitur-fitur yang ada.
- Disini terdapat fitur "Age" dan "Gender", algoritma akan menghitung seberapa besar kemungkinan seseorang membeli produk berdasarkan usia dan jenis kelaminnya.

- **Output:** Model yang sudah dilatih (disebut mod) dan data pelatihan (disebut tra).

#### c. Performance

- Setelah model dilatih, langkah selanjutnya adalah mengevaluasi performanya. Ini dilakukan dengan menggunakan dataset yang sama atau dataset validasi.
- **Cara Kerja:**
  - Model akan memprediksi kelas untuk setiap contoh dalam dataset.
  - Hasil prediksi ini kemudian dibandingkan dengan kelas sebenarnya (label) untuk menghitung akurasi, presisi, recall, atau metrik lainnya.
- **Output:** Laporan performa (disebut per) yang menunjukkan seberapa baik model bekerja.

#### d. Apply Model

- Setelah model dievaluasi, langkah berikutnya adalah menerapkan model ini ke dataset baru yang belum diketahui labelnya (unlabeled data).
- **Cara Kerja:**
  - Model akan menggunakan fitur-fitur dalam dataset baru untuk memprediksi kelas.
  - Misalnya, jika dataset baru berisi informasi tentang pelanggan baru, model akan memprediksi apakah mereka akan menjadi pelanggan atau tidak.
- **Output:** Prediksi kelas untuk dataset baru (disebut unl).

#### e. Retrieve Deals-Testset

- Langkah ini mirip dengan langkah pertama, tetapi dataset yang diambil adalah dataset uji (test set). Dataset ini digunakan untuk menguji model yang sudah dilatih.
- Tujuan: Memastikan bahwa model dapat bekerja dengan baik pada data yang belum pernah dilihat sebelumnya.

#### Alur Kerja Secara Keseluruhan

- Ambil Data Pelatihan: Dataset diambil untuk melatih model.
- Latih Model: Model Naive Bayes dibangun menggunakan dataset pelatihan.
- Evaluasi Performa: Model diuji menggunakan dataset yang sama atau dataset validasi untuk melihat seberapa baik performanya.
- Terapkan Model: Model yang sudah dilatih digunakan untuk memprediksi kelas pada dataset baru.
- Ambil Data Uji: Dataset uji diambil untuk menguji model pada data yang belum pernah dilihat sebelumnya.

### **Langkah 1: Persiapan Data**

#### 1. Download Dataset:

Kita download udlu datasetnya dari Kaggle dengan link:

<https://www.kaggle.com/datasets/quantbruce/real-estate-price-prediction>

#### 2. Buka RapidMiner:

Jalankan aplikasi RapidMiner di laptop/komputer.

Buat proses baru dengan mengklik **File > New Process**.

### **Langkah 2: Import Data ke RapidMiner**

#### • Tambahkan Operator Read CSV

- Cari operator **Read CSV** di panel operator (bisa ketik di search bar).
- Tarik operator tersebut ke area proses.

#### • Konfigurasi Operator Read CSV

- Klik operator **Read CSV**.
- Di panel kanan, klik **Import Configuration Wizard**.
- Pilih file CSV yang sudah diunduh.
- Pastikan format file sudah benar (misalnya, delimiter koma, encoding UTF-8).
- Klik **Finish** untuk mengimpor data.

#### • Cek Data

- Jalankan proses dengan mengklik tombol **Run**.
- Pastikan data sudah terimpor dengan benar. kita bisa melihat pratinjau data di panel **Results**.

### **Langkah 3: Preprocessing Data**

#### 1. Hapus Kolom yang Tidak Dibutuhkan

- **Kolom No** tidak diperlukan karena hanya nomor urut. Maka dari itu kita ambilkan operator **Select Attributes**.
- Konfigurasi operator untuk memilih kolom yang relevan (semua kolom kecuali **No**).

**Jelaskan langkah-langkah saudara dalam mengestimasi harga rumah tersebut mulai dari pengambilan data sampai terbangunnya model.**

**Parameters**

**Select Attributes**

type: include attributes

attribute filter type: a subset

select subset: [Select Attributes...](#)

also apply to special attributes (id, label..)

**Select Attributes: select subset**

Select Attributes: select subset  
Click to select the attribute subset.

Attributes

Selected Attributes

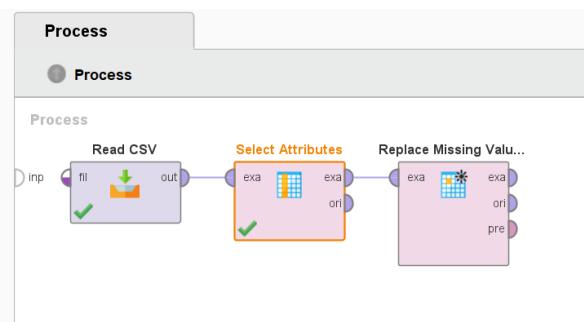
# No

# X1 transaction date  
# X2 house age  
# X3 distance to the nearest MRT station  
# X4 number of convenience stores  
# X5 latitude  
# X6 longitude  
# Y house price of unit area

[Apply](#) [Cancel](#)

## 2. Cek Missing Values

- Tambahkan operator **Replace Missing Values** jika ada data yang hilang.



- Konfigurasi operator untuk mengisi missing values dengan nilai rata-rata (mean) atau median.

**Parameters**

**Replace Missing Values**

attribute filter type: all

invert selection

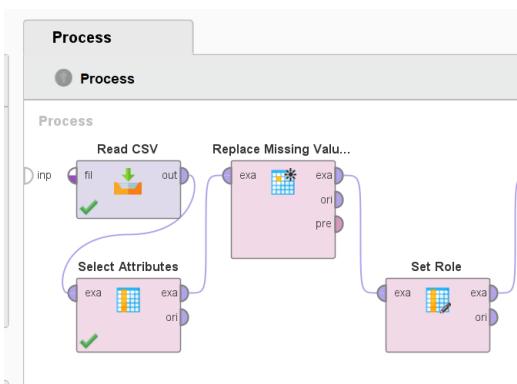
include special attributes

default: average

columns: [Edit List \(0\)](#)

## 3. Set Role untuk Kolom Label

- Tambahkan operator **Set Role**.



- Klik operator **Set Role**.
- Di panel kanan, pilih kolom **Y house price of unit area** sebagai **label** (target variabel).

**Edit Parameter List: set roles**

This parameter defines new attribute roles.

attribute name	target role
Y house price of unit area	label

- Caranya:

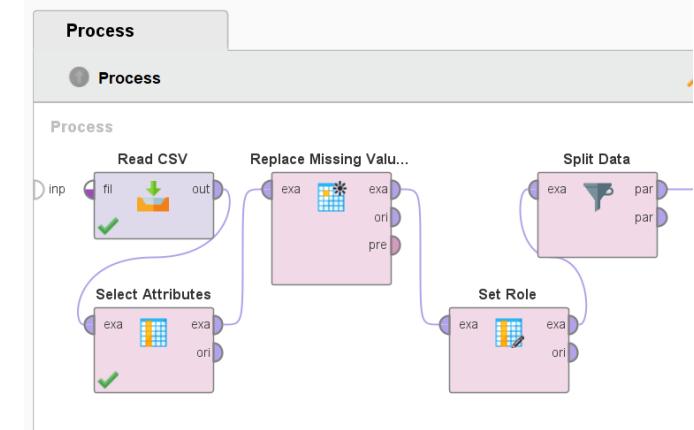
- Klik **Edit List** di bagian **attribute filter**.

- Pilih kolom **Y house price of unit area**.
- Set **target role** menjadi **label**.
- Klik **OK**.

## Langkah 4: Bagi Data menjadi Training dan Testing

### 1. Tambahkan Operator Split Data

- Operator ini digunakan untuk membagi dataset menjadi data training dan data testing.



### 2. Konfigurasi Operator Split Data

- Set rasio pembagian (misalnya, 0.7 untuk training dan 0.3 untuk testing).

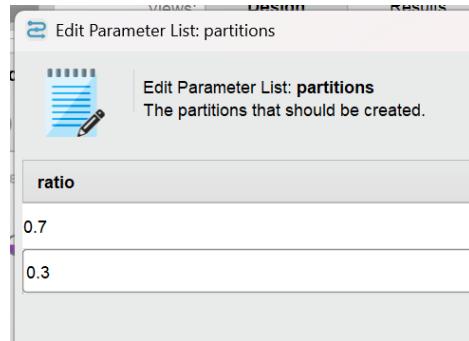
**Parameters**

**Split Data**

partitions: [Edit Enumeration \(0\)](#)

sampling type: automatic

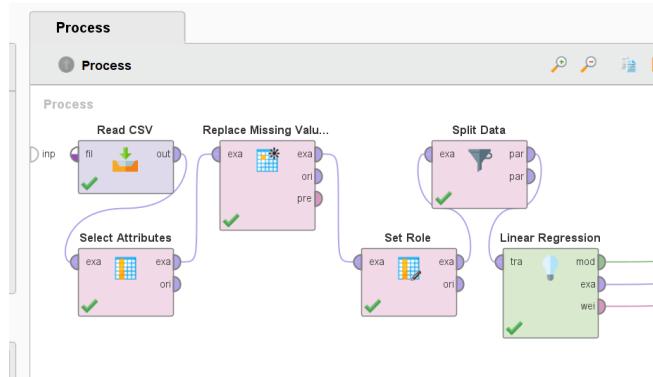
use local random seed



#### Langkah 5: Bangun Model Regresi

##### 1. Pilih Algoritma Regresi

- Untuk kasus ini, kita bisa menggunakan **Linear Regression**.
- Cari operator **Linear Regression** di panel operator.
- Tarik operator tersebut ke area proses.



##### 1. Hubungkan Operator

- Hubungkan output **Split Data (training)** ke input **Linear Regression**.
- Output **Linear Regression** akan menghasilkan model.

attribute	weight
X1 transaction date	4.982
X2 house age	-0.299
X3 distance to the nearest MRT station	-0.004
X4 number of convenience stores	1.245
X5 latitude	208.932
X6 longitude	0

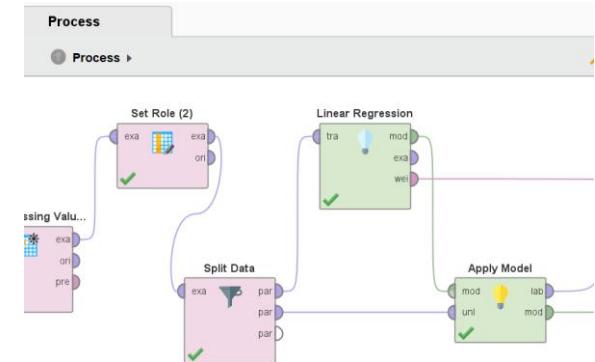
Row No.	Y house price of unit area	X1 transaction date	X2 house age	X3 distance to the nearest MRT stati...	X4 number of conveni...	X5 latitude	X6 longitude
1	37.900	2012.917	32	84.879	10	24.983	121.540
2	42.200	2012.917	19.800	306.095	9	24.980	121.540
3	47.300	2013.583	13.300	561.985	5	24.987	121.544
4	32.100	2012.667	7.100	2179.030	3	24.983	121.513
5	40.300	2012.667	34.800	623.473	7	24.979	121.536
6	46.700	2013.417	20.300	287.603	6	24.980	121.542
7	18.800	2013.500	31.700	5512.038	1	24.951	121.485
8	22.100	2013.417	17.900	1783.180	3	24.967	121.515
9	41.400	2013.083	34.800	405.213	1	24.973	121.534
10	58.100	2013.333	6.300	90.496	9	24.974	121.543
11	39.300	2012.917	13	492.231	5	24.965	121.537
12	23.800	2012.667	20.400	2469.645	4	24.961	121.510
13	50.500	2013.583	35.700	579.208	2	24.982	121.546

Attribute	Coefficient	Std. Error	Std. Coefficient	Tolerance
X1 transaction date	4.982	1.902	0.102	1.000
X2 house age	-0.299	0.049	-0.243	1.000
X3 distance to the ne...	-0.004	0.001	-0.382	0.500
X4 number of conveni...	1.245	0.240	0.263	0.664
X5 latitude	208.932	58.633	0.184	0.658
(Intercept)	-15204.872	4008.404	?	?

#### Langkah 6: Training Model

##### 1. Tambahkan Operator Apply Model:

- Operator ini digunakan untuk menerapkan model ke data training.
- Hubungkan output **Linear Regression (model)** ke input **Apply Model**.
- Hubungkan output **Split Data (training)** ke input **Apply Model**.



##### 2. Jalankan Proses:

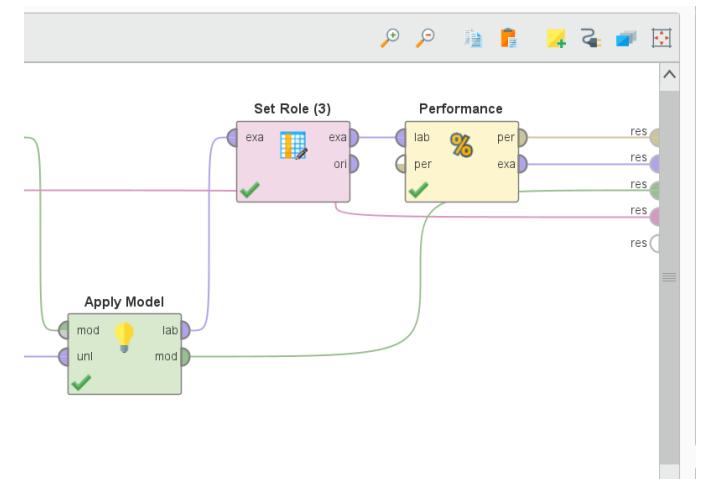
- Klik tombol **Run** untuk melatih model.
- Model akan dipelajari dari data training.

#### Langkah 7: Evaluasi Model

Tujuan: Mengevaluasi performa model menggunakan data testing.

##### 1. Tambahkan Operator Performance (Regression):

- Cari operator **Performance (Regression)** di panel operator.
- Tarik operator tersebut ke area proses.

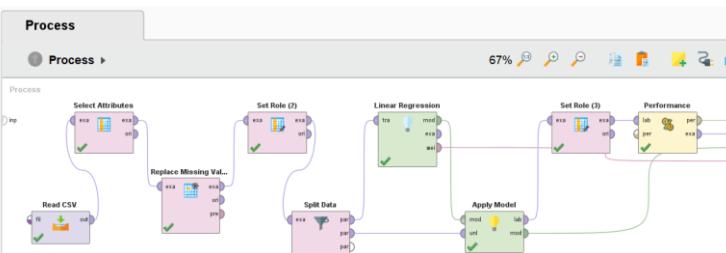


##### 2. Hubungkan Operator:

- Hubungkan output labelled dari **Apply Model** ke input **Performance**.
- Hubungkan output testing dari **Split Data** ke input **Performance**.

### 3. Jalankan Proses:

- Klik tombol Run untuk mengevaluasi model.
- Hasil evaluasi akan muncul di panel Results. Perhatikan metrik seperti R-squared, Mean Absolute Error (MAE), dan Root Mean Squared Error (RMSE).



Berikut adalah penjelasan langkah-langkah yang saya lakukan untuk mengklasifikasikan jamur menggunakan model Decision Tree:

#### 1. Retrieve Mushrooms (Pengambilan Data)

- Operator: 'Retrieve' atau 'Read CSV'
- Tujuan: Mengimpor dataset "Mushroom Classification" ke dalam RapidMiner.
- Detail: Dataset ini berisi 8124 sampel jamur dengan 23 atribut, termasuk kolom target 'class' yang menunjukkan apakah jamur tersebut beracun ('p') atau dapat dimakan ('e').

#### 2. Set Role (Menentukan Peran Kolom)

- Operator: 'Set Role'
- Tujuan: Menentukan kolom mana yang akan digunakan sebagai label (target) untuk klasifikasi.
- Detail: Anda menetapkan kolom 'class' sebagai label (target) yang akan diprediksi oleh model. Ini dilakukan dengan mengatur peran kolom 'class' sebagai 'label'.

#### 3. Split Data (Pembagian Data)

- Operator: 'Split Data'
- Tujuan: Membagi dataset menjadi dua bagian: data training dan data testing.
- Detail: Biasanya, dataset dibagi dengan rasio 70:30 atau 80:20. Data training digunakan untuk melatih model, sedangkan data testing digunakan untuk menguji performa model.

#### 4. Decision Tree (Membangun Model)

- Operator: Decision Tree
- Tujuan: Membangun model klasifikasi menggunakan algoritma Decision Tree.
- Detail: Model ini akan mempelajari pola dari data training dengan membangun struktur pohon keputusan berdasarkan atribut-atribut yang diberikan. Setiap cabang pohon mewakili keputusan berdasarkan nilai atribut tertentu, dan daun pohon mewakili prediksi apakah jamur tersebut beracun

atau dapat dimakan. Decision Tree memilih atribut terbaik untuk memisahkan data pada setiap langkah, sehingga membentuk aturan-aturan yang mudah dipahami untuk klasifikasi.

#### 4. Naive Bayes (Membangun Model)

- Operator: 'Naive Bayes'
- Tujuan: Membangun model klasifikasi menggunakan algoritma Naive Bayes.
- Detail: Model ini akan mempelajari pola dari data training berdasarkan atribut-atribut yang diberikan untuk memprediksi apakah suatu jamur beracun atau dapat dimakan.

#### 4. Decision Tree (Membangun Model)

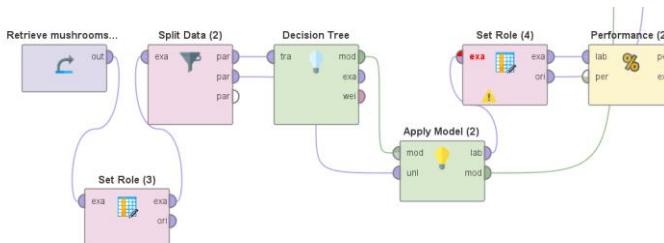
- Operator: Decision Tree
- Tujuan: Membangun model klasifikasi menggunakan algoritma Decision Tree.
- Detail: Model ini akan mempelajari pola dari data training dengan membangun struktur pohon keputusan berdasarkan atribut-atribut yang diberikan. Setiap cabang pohon mewakili keputusan berdasarkan nilai atribut tertentu, dan daun pohon mewakili prediksi apakah jamur tersebut beracun atau dapat dimakan. Decision Tree memilih atribut terbaik untuk memisahkan data pada setiap langkah, sehingga membentuk aturan-aturan yang mudah dipahami untuk klasifikasi.

#### 5. Apply Model (Menerapkan Model)

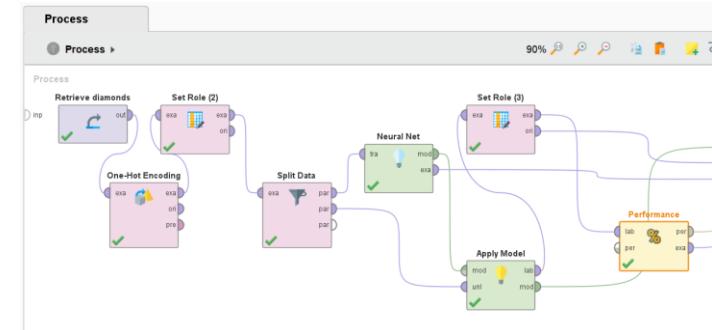
- Operator: 'Apply Model'
- Tujuan: Menerapkan model yang telah dibangun (Naive Bayes) pada data testing.
- Detail: Model akan memprediksi label ('class') untuk setiap sampel dalam data testing berdasarkan pola yang telah dipelajari selama proses training.

#### 6. Performance (Evaluasi Model)

- Operator: 'Performance'
- Tujuan: Mengevaluasi performa model dengan membandingkan prediksi model dengan label sebenarnya dari data testing.
- Detail: Beberapa metrik evaluasi yang umum digunakan adalah akurasi, precision, recall, dan F1-score. Metrik ini membantu Anda memahami seberapa baik model Anda dalam mengklasifikasikan jamur.



#### Ringkasan Workflow



4. Ambil Data: Mengimpor dataset Diamonds.

5. Handle Missing Values: Menangani data yang hilang.

6. One-Hot Encoding: Mengubah data kategorikal menjadi numerik.

7. Operator: One-Hot Encoding

- a. Tujuan: Mengubah kolom kategorikal (cut, color, clarity) menjadi numerik agar bisa diproses oleh model Neural Network.

- b. Penjelasan: Kolom kategorikal seperti cut (Fair, Good, Very Good, Premium, Ideal) diubah menjadi kolom numerik biner (0 atau 1). Misalnya, kolom cut akan dipecah menjadi beberapa kolom baru seperti cut\_Fair, cut\_Good, cut\_Very Good, dst.

- c. Output: Dataset dengan semua fitur dalam bentuk numerik.

8. Set Role: Menetapkan kolom target dan fitur.

9. Split Data: Membagi data menjadi training dan testing.

10. Neural Net: Membangun dan melatih model Neural Network.

11. Apply Model: Menerapkan model pada data testing.

12. Performance: Mengevaluasi performa model dengan RMSE.

#### c. Tuliskan rekomendasi saudara setelah model saudara dapatkan.

Setelah melakukan analisis dan evaluasi terhadap model Neural Network yang telah saya buat, berikut adalah beberapa rekomendasi yang dapat saya pertimbangkan untuk meningkatkan performa model dan memastikan bahwa model tersebut dapat digunakan secara efektif dalam memprediksi harga berlian:

##### 1. Handle Outlier:

- **Apa Itu Outlier?** Outlier adalah data yang sangat berbeda dari data lainnya, seperti harga berlian yang sangat tinggi atau sangat rendah.
- **Mengapa Perlu Dihandle?** Outlier dapat memengaruhi performa model, terutama pada prediksi harga yang ekstrem (misalnya, prediksi negatif atau prediksi yang terlalu tinggi).
- **Cara Handle Outlier:**
  - Identifikasi outlier menggunakan visualisasi (seperti boxplot) atau metode statistik (seperti Z-score atau IQR).
  - Pertimbangkan untuk menghapus outlier atau mengubah nilainya dengan metode tertentu (misalnya, mengganti dengan nilai median).

##### 2. Feature Engineering:

- **Apa Itu Feature Engineering?** Feature engineering adalah proses menciptakan fitur baru atau memodifikasi fitur yang sudah ada untuk meningkatkan performa model.
- **Mengapa Perlu Dilakukan?** Fitur yang lebih informatif dapat membantu model memahami pola dalam data dengan lebih baik.
- **Contoh Feature Engineering:**
  - Tambahkan fitur baru seperti rasio dimensi (x/y, z/depth) atau interaksi antar fitur (misalnya, carat \* cut).
  - Lakukan transformasi pada fitur numerik, seperti logaritmik atau normalisasi, untuk membuat distribusi data lebih baik.

### 3. Tuning Hyperparameter:

- **Apa Itu Hyperparameter?** Hyperparameter adalah parameter yang diatur sebelum melatih model, seperti jumlah lapisan tersembunyi, jumlah neuron, atau learning rate pada Neural Network.
- **Mengapa Perlu Dilakukan?** Hyperparameter yang optimal dapat meningkatkan akurasi model dan mengurangi overfitting.
- **Cara Tuning Hyperparameter:**
  - Gunakan metode seperti **Grid Search** atau **Random Search** untuk mencari kombinasi hyperparameter terbaik.
  - Contoh hyperparameter yang bisa di-tuning:
    - Jumlah lapisan tersembunyi (hidden layers).
    - Jumlah neuron di setiap lapisan.
    - Learning rate.
    - Fungsi aktivasi (misalnya, ReLU, sigmoid).

### 4. Coba Model Lain:

- **Mengapa Perlu Mencoba Model Lain?** Neural Network adalah model yang kompleks dan mungkin tidak selalu menjadi pilihan terbaik untuk dataset tertentu. Mencoba model lain dapat membantu saya menemukan model yang lebih cocok.
- **Model Alternatif yang Bisa Dicoba:**
  - **Random Forest Regression:** Model ini kuat terhadap outlier dan mudah diinterpretasi.
  - **Gradient Boosting Regression (XGBoost, LightGBM):** Model ini sering memberikan performa yang sangat baik pada dataset tabular.
  - **Support Vector Regression (SVR):** Cocok untuk dataset dengan jumlah fitur yang tidak terlalu besar.
- **Cara Mencoba Model Lain:** Bandingkan performa model-model tersebut dengan Neural Network menggunakan metrik yang sama (misalnya, RMSE).

### 5. Validasi Silang (Cross-Validation):

- **Apa Itu Cross-Validation?** Cross-validation adalah teknik untuk mengevaluasi performa model dengan membagi data menjadi beberapa subset dan melatih model pada subset yang berbeda.
- **Mengapa Perlu Dilakukan?** Cross-validation membantu memastikan bahwa performa model konsisten dan tidak overfitting pada data training.
- **Cara Melakukan Cross-Validation:**
  - Gunakan operator **Cross-Validation** di RapidMiner.
  - Bagi data menjadi 5 atau 10 subset (fold) dan evaluasi performa model pada setiap fold.

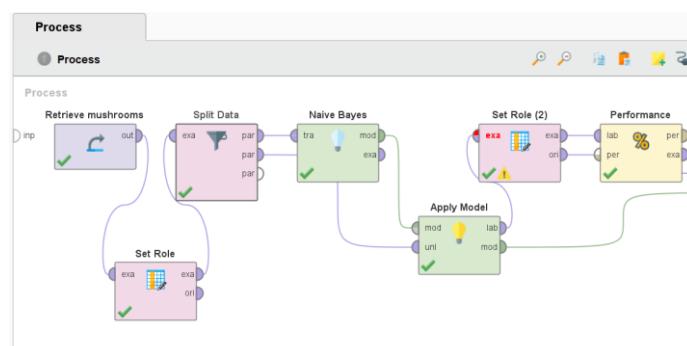
### 6. Analisis Residual:

- **Apa Itu Residual?** Residual adalah selisih antara harga prediksi dan harga aktual.
- **Mengapa Perlu Dilakukan?** Analisis residual dapat membantu saya memahami di mana model melakukan kesalahan dan mengidentifikasi pola kesalahan yang sistematis.
- **Cara Melakukan Analisis Residual:**
  - Plot residual terhadap harga aktual atau fitur-fitur tertentu.
  - Identifikasi pola tertentu (misalnya, model cenderung salah prediksi pada harga yang sangat tinggi atau sangat rendah).

### 7. Deployment dan Monitoring:

- **Apa Itu Deployment?** Deployment adalah proses mengimplementasikan model ke dalam sistem atau aplikasi yang dapat digunakan oleh pengguna.
- **Mengapa Perlu Dilakukan?** Setelah model dianggap cukup baik, model dapat digunakan untuk memprediksi harga berlian secara real-time.
- **Cara Deployment:**
  - Ekspor model yang sudah dilatih dari RapidMiner.
  - Integrasikan model dengan aplikasi atau sistem yang ada (misalnya, menggunakan API).
- **Monitoring:**
  - Pantau performa model secara berkala setelah deployment.
  - Jika performa menurun (misalnya, karena perubahan pola data), lakukan retraining model.

### Berikan penjelasan model yang saudara dapatkan.



Berikut penjelasan tentang model yang saya buat:

1. **Ambil Data:** Saya mengambil dataset jamur yang berisi informasi tentang berbagai jenis jamur dan apakah mereka berasam atau bisa dimakan.
2. **Bagi Data:** Dataset dibagi menjadi dua bagian: satu untuk melatih model (training data) dan satu lagi untuk menguji model (testing data).
3. **Set Role:** Saya menentukan kolom "class" sebagai target yang ingin diprediksi, yaitu apakah jamur berasam atau tidak.
4. **Naive Bayes:** Saya menggunakan algoritma Naive Bayes untuk melatih model. Algoritma ini belajar dari data training untuk memprediksi kategori jamur berdasarkan fitur-fiturnya.
5. **Decision Tree:** Model dibangun menggunakan algoritma Decision Tree. Algoritma ini membuat pohon keputusan berdasarkan atribut-

atribut dalam data untuk memprediksi kategori jamur. Setiap cabang pohon mewakili keputusan berdasarkan nilai atribut tertentu.

6. **Apply Model:** Model yang sudah dilatih kemudian digunakan untuk memprediksi kategori jamur pada data testing.
7. **Evaluasi:** Saya mengevaluasi performa model dengan membandingkan prediksi model dengan kategori sebenarnya dari data testing. Ini membantu saya melihat seberapa akurat model saya.

Jadi, model saya adalah sebuah sistem yang belajar dari data jamur untuk memprediksi apakah suatu jamur berasam atau bisa dimakan, dan saya menguji seberapa baik sistem ini bekerja.

## Neural Net Jamur

### 1. Retrieve Mushrooms (Pengambilan Data)

- **Operator:** Retrieve atau Read CSV
- **Tujuan:** Mengimpor dataset jamur ke dalam RapidMiner. Dataset ini berisi informasi tentang berbagai jenis jamur, termasuk atribut-atribut seperti bentuk topi, warna, bau, dan lainnya, serta kolom target (class) yang menunjukkan apakah jamur tersebut berasam (p) atau dapat dimakan (e).
- **Detail:** Dataset yang digunakan adalah "Mushroom Classification" dari Kaggle, yang terdiri dari 8124 sampel dan 23 atribut.

### 2. One-Hot Encoding (Transformasi Data)

- **Operator:** One-Hot Encoding
- **Tujuan:** Mengubah data kategorikal (seperti warna, bentuk, atau bau) menjadi format numerik yang bisa diproses oleh model machine learning.
- **Detail:** Karena algoritma Decision Tree (atau model lainnya) membutuhkan input numerik, atribut kategorikal seperti cap-shape, cap-color, atau odor diubah menjadi kolom-kolom biner (0 atau 1) untuk setiap nilai yang mungkin. Misalnya, jika cap-color memiliki nilai red, blue, dan green, maka akan dibuat tiga kolom baru: cap-color\_red, cap-color\_blue, dan cap-color\_green, dengan nilai 1 atau 0 tergantung pada nilai aslinya.

### 3. Set Role (Menentukan Peran Kolom)

- **Operator:** Set Role
- **Tujuan:** Menentukan kolom mana yang akan digunakan sebagai label (target) untuk klasifikasi.
- **Detail:** Kolom class ditetapkan sebagai label yang akan diprediksi oleh model. Ini dilakukan dengan mengatur peran kolom class sebagai label. Kolom lainnya tetap sebagai atribut (features) yang digunakan untuk memprediksi label.

### 4. Split Data (Pembagian Data)

- **Operator:** Split Data
- **Tujuan:** Membagi dataset menjadi dua bagian: data training dan data testing.
- **Detail:** Dataset dibagi dengan rasio tertentu, misalnya 70:30 atau 80:20. Data training digunakan untuk melatih model, sedangkan data testing digunakan untuk menguji performa model. Misalnya, jika dataset memiliki 1000 sampel, 700 sampel digunakan untuk training dan 300 sampel untuk testing.

### 5. Decision Tree (Membangun Model)

- **Operator:** Decision Tree

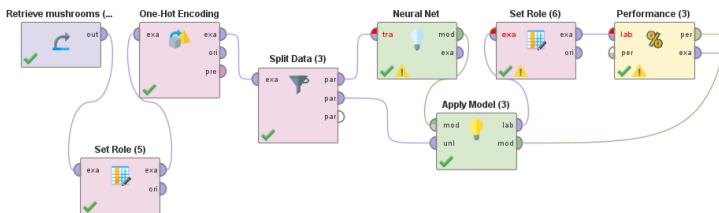
- **Tujuan:** Membangun model klasifikasi menggunakan algoritma Decision Tree.
- **Detail:** Model ini mempelajari pola dari data training dengan membangun struktur pohon keputusan. Setiap cabang pohon mewakili keputusan berdasarkan nilai atribut tertentu (misalnya, "apakah bau jamur menyengat?"), dan daun pohon mewakili prediksi (beracun atau dapat dimakan). Decision Tree memilih atribut terbaik untuk memisahkan data pada setiap langkah, sehingga membentuk aturan-aturan yang mudah dipahami.
- 6. **Performance:** Performa model diukur dengan membandingkan prediksi model dengan hasil sebenarnya dari data testing. Metrik seperti akurasi, precision, recall, dan F1-score digunakan untuk mengevaluasi seberapa baik model ini bekerja.

## 6. Apply Model (Menerapkan Model)

- **Operator:** Apply Model
- **Tujuan:** Menerapkan model yang telah dibangun (Decision Tree) pada data testing.
- **Detail:** Model menggunakan aturan-aturan yang telah dipelajari selama training untuk memprediksi label (class) pada data testing. Hasilnya adalah prediksi apakah setiap jamur dalam data testing beracun atau dapat dimakan.

## 7. Performance (Evaluasi Model)

- **Operator:** Performance
- **Tujuan:** Mengevaluasi performa model dengan membandingkan prediksi model dengan label sebenarnya dari data testing.
- **Detail:** Beberapa metrik evaluasi yang digunakan adalah:
  - o **Akurasi:** Seberapa sering model benar dalam memprediksi.
  - o **Precision:** Seberapa akurat prediksi positif (misalnya, prediksi "beracun" yang benar).
  - o **Recall:** Seberapa banyak kasus positif yang berhasil diidentifikasi.
  - o **F1-Score:** Gabungan dari precision dan recall untuk mengukur keseimbangan keduanya. **Berikan penjelasan model yang saudara dapatkan.**



Berikut penjelasan singkat tentang model yang dibuat:

1. **Ambil Data:** Dataset skor musik diambil untuk dianalisis. Dataset ini berisi informasi tentang berbagai atribut musik yang relevan.
2. **One-Hot Encoding:** Data kategorikal diubah menjadi format numerik menggunakan teknik one-hot encoding. Ini dilakukan karena model machine learning membutuhkan input numerik. Misalnya, jika ada kategori seperti genre musik, setiap genre akan diubah menjadi kolom terpisah dengan nilai 0 atau 1.
3. **Set Role:** Kolom target ditetapkan sebagai label yang ingin diprediksi. Ini membantu model memahami mana kolom yang menjadi fokus prediksi.
4. **Split Data:** Dataset dibagi menjadi dua bagian, yaitu data training untuk melatih model dan data testing untuk menguji model.
5. **Apply Model:** Model yang sudah dilatih digunakan untuk memprediksi hasil pada data testing. Ini dilakukan untuk melihat seberapa baik model bekerja pada data yang belum pernah dilihat sebelumnya.