**Mata Kuliah Coding & Machine Learning**

**Laporan Tugas Pertemuan 4**

**Dosen Pengampu: Sri Wulandari, S.Kom., M.Cs.**



**Disusun oleh:**

**Lathif Ramadhan (5231811022)**

**PROGRAM STUDI SAINS DATA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS TEKNOLOGI YOGYAKARTA**

**YOGYAKARTA**

**2024**

# Daftar Isi

[Daftar Isi ii](#_Toc192356809)

[1. Berikan penjelasan perbedaan dan contoh implementasi dari Regresi Linear, Logistic Regression dan GLM serta Naïve Bayes dan Decision Tree! 1](#_Toc192356810)

[A. Regresi Linear 1](#_Toc192356811)

[Rumus Regresi Linear 1](#_Toc192356812)

[Contoh Implementasi 1](#_Toc192356813)

[Kapan Menggunakan Regresi Linear? 4](#_Toc192356814)

[B. Logistic Regression 5](#_Toc192356815)

[Rumus Logistic Regression 5](#_Toc192356816)

[Contoh Implementasi 5](#_Toc192356817)

[Kapan Menggunakan Logistic Regression? 7](#_Toc192356818)

[C. Generalized Linear Model (GLM) 8](#_Toc192356819)

[Contoh Implementasi 8](#_Toc192356820)

[Kapan Menggunakan GLM? 9](#_Toc192356821)

[Tabel Perbedaan Algoritma Regresi Linear, Logistic Regression dan GLM 10](#_Toc192356822)

[D. Naïve Bayes 11](#_Toc192356823)

[Cara kerja Naïve Bayes: 11](#_Toc192356824)

[Kelebihan Naïve Bayes: 11](#_Toc192356825)

[Kekurangan Naïve Bayes: 11](#_Toc192356826)

[E. Decision Tree 12](#_Toc192356827)

[Cara kerja Decision Tree: 12](#_Toc192356828)

[Kelebihan Decision Tree: 12](#_Toc192356829)

[Kekurangan Decision Tree: 12](#_Toc192356830)

[Tabel Perbedaan Algoritma Naïve Bayes dan Decision Tree 12](#_Toc192356831)

[2. Perhatikan parameter-parameter yang dimiliki oleh algoritma-algoritma pembelajaran terbimbing yang telah dijelaskan pada bab 2 ini, berikan penjelasan makna dari parameter-parameter tersebut! 14](#_Toc192356832)

[A. Regresi Linear 14](#_Toc192356833)

[1. Feature Selection 14](#_Toc192356834)

[2. Alpha 14](#_Toc192356835)

[3. Max Iterations 14](#_Toc192356836)

[4. Forward Alpha 14](#_Toc192356837)

[5. Backward Alpha 15](#_Toc192356838)

[6. Eliminate Colinear Features 15](#_Toc192356839)

[7. Min Tolerance 15](#_Toc192356840)

[8. Use Bias 15](#_Toc192356841)

[9. Ridge 15](#_Toc192356842)

[B. Logistic Regression 16](#_Toc192356843)

[1. Kernel Type 16](#_Toc192356844)

[2. Kernel Gamma 16](#_Toc192356845)

[3. Kernel Sigma1, Sigma2, Sigma3 16](#_Toc192356846)

[4. Kernel Shift 16](#_Toc192356847)

[5. Kernel Degree 17](#_Toc192356848)

[6. Kernel A dan B 17](#_Toc192356849)

[7. C (Complexity Constant) 17](#_Toc192356850)

[8. Start Population Type 17](#_Toc192356851)

[9. Max Generations 17](#_Toc192356852)

[10. Generations Without Improval 17](#_Toc192356853)

[11. Population Size 17](#_Toc192356854)

[12. Tournament Fraction 18](#_Toc192356855)

[13. Keep Best 18](#_Toc192356856)

[14. Mutation Type 18](#_Toc192356857)

[15. Selection Type 18](#_Toc192356858)

[16. Crossover Prob 18](#_Toc192356859)

[17. Use Local Random Seed 18](#_Toc192356860)

[18. Local Random Seed 18](#_Toc192356861)

[19. Show Convergence Plot 18](#_Toc192356862)

[C. Generalized Linear Models (GLM) 19](#_Toc192356863)

[1. Family 19](#_Toc192356864)

[2. Solver 19](#_Toc192356865)

[3. Link 19](#_Toc192356866)

[4. Reproducible 20](#_Toc192356867)

[5. Maximum Number of Threads 20](#_Toc192356868)

[6. Specify Beta Constraints 20](#_Toc192356869)

[7. Use Regularization 20](#_Toc192356870)

[8. Lambda 20](#_Toc192356871)

[9. Lambda Search 20](#_Toc192356872)

[10. Number of Lambdas 21](#_Toc192356873)

[11. Lambda Min Ratio 21](#_Toc192356874)

[12. Early Stopping 21](#_Toc192356875)

[13. Stopping Rounds 21](#_Toc192356876)

[14. Stopping Tolerance 21](#_Toc192356877)

[15. Alpha 21](#_Toc192356878)

[16. Standardize 21](#_Toc192356879)

[17. Non-Negative Coefficients 21](#_Toc192356880)

[18. Compute P-Values 22](#_Toc192356881)

[19. Remove Collinear Columns 22](#_Toc192356882)

[20. Add Intercept 22](#_Toc192356883)

[21. Missing Values Handling 22](#_Toc192356884)

[22. Max Iterations 22](#_Toc192356885)

[23. Beta Constraints 22](#_Toc192356886)

[24. Max Runtime Seconds 22](#_Toc192356887)

[25. Expert Parameters 22](#_Toc192356888)

[D. Naïve Bayes 23](#_Toc192356889)

[1. Laplace Correction 23](#_Toc192356890)

[E. Decision Tre 23](#_Toc192356891)

[1. Criterion 23](#_Toc192356892)

[2. Maximal Depth 24](#_Toc192356893)

[3. Apply Pruning 24](#_Toc192356894)

[4. Confidence 24](#_Toc192356895)

[5. Apply Prepruning 24](#_Toc192356896)

[6. Minimal Gain 24](#_Toc192356897)

[7. Minimal Leaf Size 24](#_Toc192356898)

[8. Minimal Size for Split 25](#_Toc192356899)

[9. Number of Prepruning Alternatives 25](#_Toc192356900)

[F. Neural Nets 25](#_Toc192356901)

[1. Hidden Layers (Lapisan Tersembunyi) 25](#_Toc192356902)

[2. Training Cycles (Siklus Pelatihan) 26](#_Toc192356903)

[3. Learning Rate (Laju Pembelajaran) 26](#_Toc192356904)

[4. Momentum 26](#_Toc192356905)

[5. Decay (Pelarutan Learning Rate) 26](#_Toc192356906)

[6. Shuffle (Mengacak Data) 26](#_Toc192356907)

[7. Normalize (Normalisasi Data) 26](#_Toc192356908)

[8. Error Epsilon 27](#_Toc192356909)

[9. Use Local Random Seed (Gunakan Seed Acak Lokal) 27](#_Toc192356910)

[10. Local Random Seed (Seed Acak Lokal) 27](#_Toc192356911)

[3. Implementasi Model Algoritma Naïve Bayes dan Decision Tree di Rapidminer dari Video Pembelajaran 28](#_Toc192356912)

[A. Naïve Bayes 28](#_Toc192356913)

[1. Retrieve dan Melihat Isi Dataset Deals 28](#_Toc192356914)

[2. Implementasi Model Naïve Bayes di Rapidminer 29](#_Toc192356915)

[3. Penjelasan Implementasi Naive Bayes di RapidMiner 29](#_Toc192356916)

[4. Hasil Klasifikasi Model Naïve Bayes 31](#_Toc192356917)

[5. Simple Distribution model Naïve Bayes 33](#_Toc192356918)

[6. Hasil Akurasi Model Naïve Bayes 34](#_Toc192356919)

[7. Implementasi Model Naïve Bayes pada Dataset Iris 36](#_Toc192356920)

[B. Decision Tree 39](#_Toc192356921)

[1. Retrieve dan Melihat Isi Dataset Deals 39](#_Toc192356922)

[2. Implementasi Model Decision Tree di Rapidminer 40](#_Toc192356923)

[3. Penjelasan Implementasi Decision Tree di RapidMiner 40](#_Toc192356924)

[4. Gambar Model Decision Tree 41](#_Toc192356925)

[5. Hasil Klasifikasi Model Decision Tree 42](#_Toc192356926)

[6. Hasil Akurasi Model Decision Tree 44](#_Toc192356927)

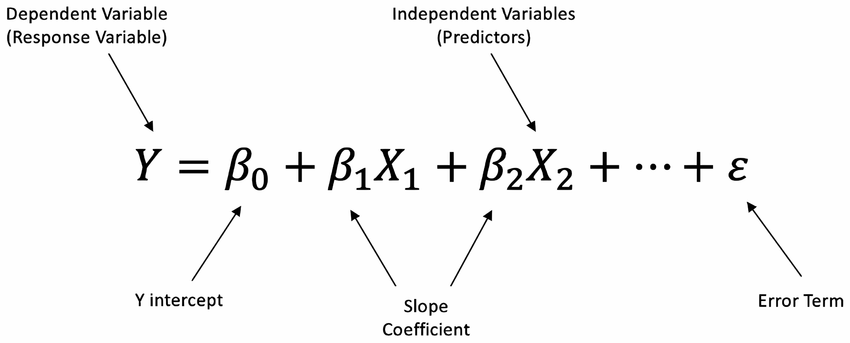
[Referensi 47](#_Toc192356928)

# Berikan penjelasan perbedaan dan contoh implementasi dari Regresi Linear, Logistic Regression dan GLM serta Naïve Bayes dan Decision Tree!

## A. Regresi Linear

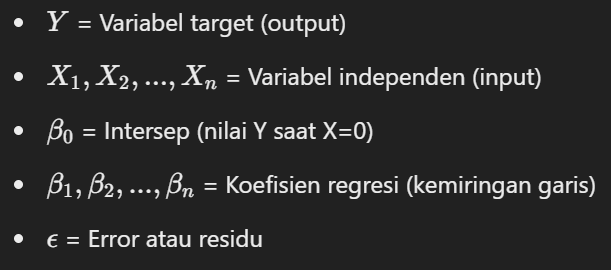
**Regresi linear** adalah metode yang digunakan untuk memprediksi nilai **kontinu** berdasarkan hubungan linear antara variabel independen dan variabel dependen. Model ini bekerja dengan mencari garis lurus terbaik yang meminimalkan selisih antara nilai prediksi dan nilai aktual.

### Rumus Regresi Linear



Gambar 1.1. Rumus Regresi Linear

Dimana:



Gambar 1.2. Keterangan/Variabel Rumus Regresi Linear

### Contoh Implementasi

Misalkan kita ingin memprediksi **harga rumah** berdasarkan luas tanah dan jumlah kamar. Kita bisa menggunakan regresi linear untuk menemukan hubungan antara luas rumah dan harganya.

**Contoh kode Python:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Data (Luas rumah dalam meter persegi dan harga dalam juta rupiah)

X = np.array([30, 50, 70, 90, 110]).reshape(-1, 1)

y = np.array([300, 500, 700, 900, 1100])

# Membuat model regresi linear

model = LinearRegression()

model.fit(X, y)

# Prediksi harga rumah

X\_test = np.array([60]).reshape(-1, 1)

y\_pred = model.predict(X\_test)

print(f"Prediksi harga rumah untuk luas 60m²: {y\_pred[0]:.2f} juta")

# Visualisasi

plt.scatter(X, y, color='blue', label='Data aktual')

plt.plot(X, model.predict(X), color='red', label='Garis regresi')

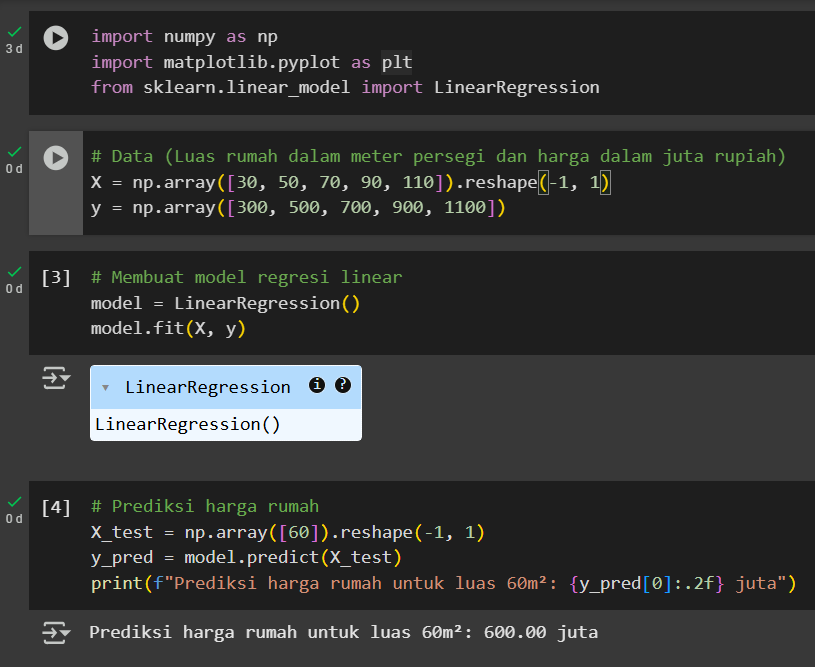
plt.xlabel('Luas Rumah (m²)')

plt.ylabel('Harga Rumah (Juta Rupiah)')

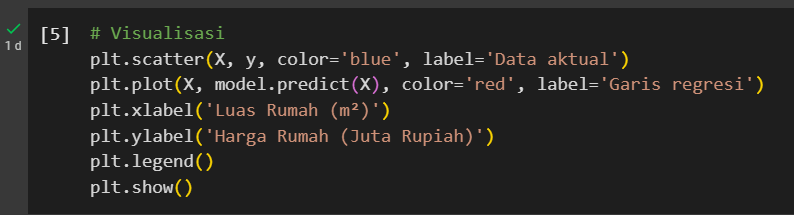
plt.legend()

plt.show()

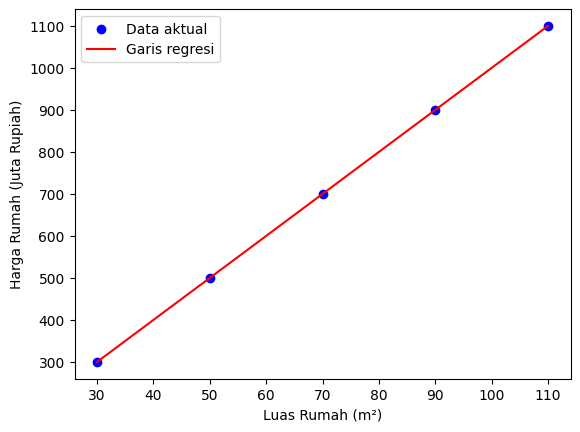
**Berikut implementasinya di Google Colab:**



Gambar 1.3. Implementasi Regresi Linear Menggunakan Python



Gambar 1.4. Implementasi Regresi Linear Menggunakan Python



Gambar 1.4. Visualisasi Hasil Implementasi Regresi Linear Menggunakan Python

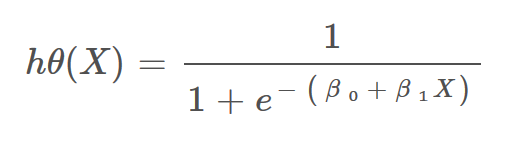
### Kapan Menggunakan Regresi Linear?

* Jika variabel target berupa **angka kontinu** (contoh: harga, gaji, suhu).
* Jika hubungan antara variabel independen dan target **bersifat linear**.

## B. Logistic Regression

**Logistic Regression** bukan digunakan untuk prediksi angka kontinu, melainkan untuk **klasifikasi biner** (ya/tidak, positif/negatif, sukses/gagal). Metode ini digunakan untuk menghitung probabilitas suatu kejadian terjadi atau tidak terjadi.

### Rumus Logistic Regression



Gambar 1.5. Rumus Logistic Regression

Fungsi yang digunakan dalam logistic regression adalah **sigmoid function**, yang mengubah nilai regresi linear menjadi nilai probabilitas antara 0 dan 1.

### Contoh Implementasi

Misalnya, kita ingin memprediksi apakah seseorang akan diterima di universitas berdasarkan nilai ujian mereka.

**Contoh kode Python:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

# Data: Nilai ujian dan status diterima (1) atau tidak diterima (0)

X = np.array([50, 55, 60, 65, 70, 75, 80, 85, 90, 95]).reshape(-1, 1)

y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

# Membuat model logistic regression

model = LogisticRegression()

model.fit(X, y)

# Prediksi peluang diterima untuk nilai 72

X\_test = np.array([72]).reshape(-1, 1)

y\_pred\_prob = model.predict\_proba(X\_test)

print(f"Peluang diterima dengan nilai 72: {y\_pred\_prob[0][1]:.2f}")

# Visualisasi

plt.scatter(X, y, color='blue', label='Data aktual')

plt.plot(X, model.predict\_proba(X)[:, 1], color='red', label='Kurva Sigmoid')

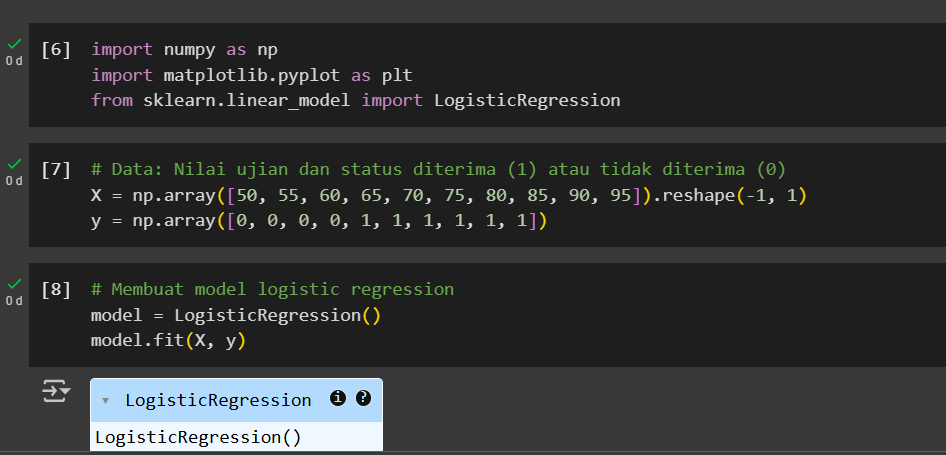
plt.xlabel('Nilai Ujian')

plt.ylabel('Probabilitas Diterima')

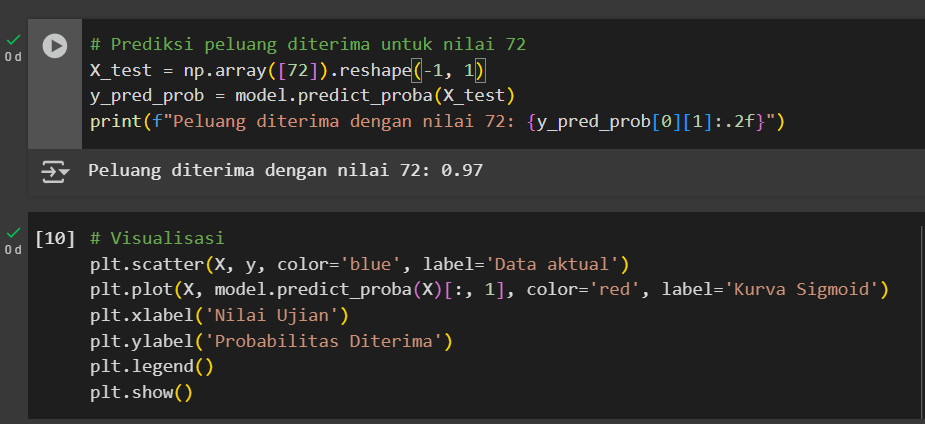
plt.legend()

plt.show()

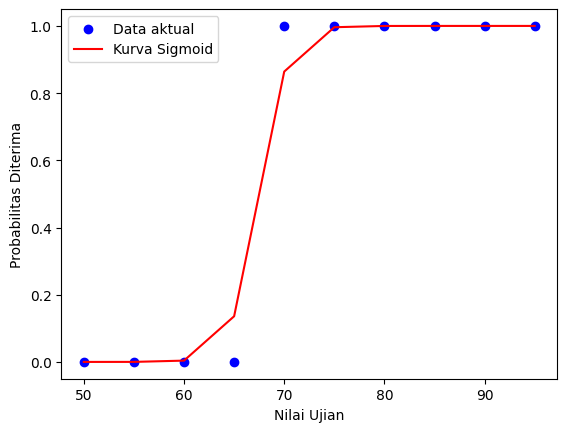
**Berikut implementasinya di Google Colab:**



Gambar 1.6. Implementasi Logistic Regression Menggunakan Python



Gambar 1.7. Implementasi Logistic Regression Menggunakan Python

\

Gambar 1.8. Hasil Visualisasi Implementasi Logistic Regression Menggunakan Python

### Kapan Menggunakan Logistic Regression?

* Jika variabel target berupa **kategori (0 atau 1, spam atau bukan spam, lulus atau tidak lulus)**.
* Jika ingin menghitung **probabilitas** suatu kejadian.

## C. Generalized Linear Model (GLM)

**GLM (Generalized Linear Model)** adalah model statistik yang lebih fleksibel dibandingkan regresi linear dan logistic regression. GLM dapat menangani berbagai jenis distribusi data, seperti **normal, binomial, poisson, gamma**, dan lainnya.

GLM memiliki tiga komponen utama:

1. **Distribusi eksponensial** (Normal, Poisson, Binomial, dll).
2. **Fungsi link** (Linear, Logit, Log, dll).
3. **Model linear umum** yang menghubungkan variabel independen dengan variabel dependen.

### Contoh Implementasi

Jika kita ingin memprediksi jumlah pelanggan yang datang ke toko dalam sehari (data **count**, cocok untuk distribusi Poisson).

**Contoh kode Python (GLM dengan Poisson Regression):**

import statsmodels.api as sm

import numpy as np

# Data jumlah iklan dan pelanggan datang

X = np.array([1, 2, 3, 4, 5]) # Jumlah iklan

y = np.array([3, 7, 9, 14, 18]) # Jumlah pelanggan

# Membuat model GLM dengan distribusi Poisson

X = sm.add\_constant(X) # Tambahkan konstanta untuk intersep

model = sm.GLM(y, X, family=sm.families.Poisson()).fit()

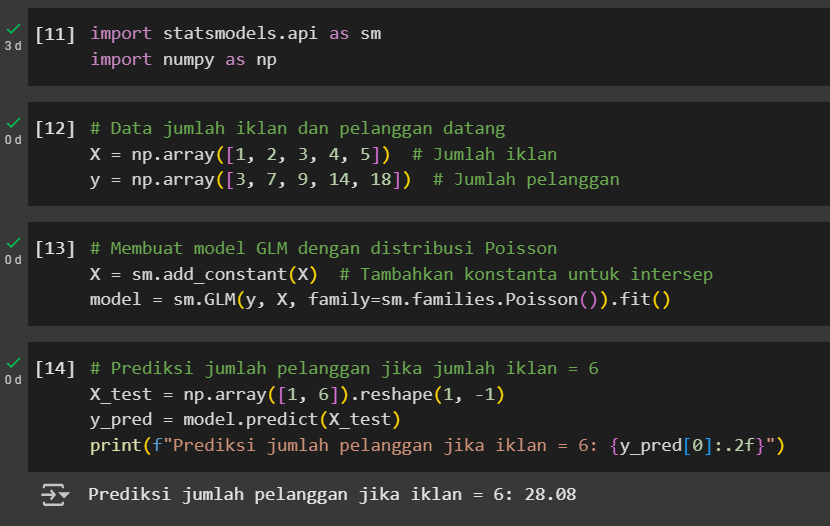
# Prediksi jumlah pelanggan jika jumlah iklan = 6

X\_test = np.array([1, 6]).reshape(1, -1)

y\_pred = model.predict(X\_test)

print(f"Prediksi jumlah pelanggan jika iklan = 6: {y\_pred[0]:.2f}")

**Berikut implementasinya di Google Colab:**



Gambar 1.9. Implementasi Generalized Linear Model (GLM) Menggunakan Python

### Kapan Menggunakan GLM?

* Jika data **tidak berdistribusi normal** (contoh: data count atau proporsi).
* Jika ingin memilih fungsi link yang lebih fleksibel.

## Tabel Perbedaan Algoritma Regresi Linear, Logistic Regression dan GLM

Berikut tabel perbedaan Regresi Linear, Logistic Regression, dan Generalized Linear Model (GLM)

|  |  |  |  |
| --- | --- | --- | --- |
| **Aspek** | **Regresi Linear** | **Logistic Regression** | **Generalized Linear Model (GLM)** |
| **Jenis Model** | Regresi | Klasifikasi | Regresi Generalisasi |
| **Tujuan** | Memprediksi nilai kontinu berdasarkan hubungan linear. | Memprediksi kategori biner berdasarkan probabilitas. | Memodelkan hubungan variabel dengan distribusi eksponensial. |
| **Jenis Data Target (Y)** | Variabel Kontinu (misalnya: harga, suhu, pendapatan). | Variabel Kategorikal Biner (misalnya: lulus/gagal, sehat/sakit). | Variabel Kontinu atau Diskrit (misalnya: jumlah pelanggan, kecelakaan). |
| **Fungsi Aktivasi** | Identitas (Linear). | Sigmoid (Logit Function). | Bergantung pada distribusi (log, identity, logit, dll). |
| **Distribusi Data** | Normal (Gaussian) | Binomial | Normal, Binomial, Poisson, Gamma, dll. |
| **Persamaan Dasar** | Y = β0 + β1X1 + β2X2 + ... + βnXn + ε | P(Y=1) = 1 / (1 + e^-(β0 + β1X1 + ... + βnXn)) | g(E(Y)) = β0 + β1X1 + ... + βnXn, dengan g sebagai fungsi link. |
| **Hasil Output** | Nilai numerik kontinu. | Probabilitas antara 0 dan 1, diklasifikasikan ke 0 atau 1. | Nilai numerik kontinu, count, atau proporsi. |
| **Metode Estimasi** | Ordinary Least Squares (OLS). | Maximum Likelihood Estimation (MLE). | Maximum Likelihood Estimation (MLE) untuk berbagai distribusi. |
| **Kapan Digunakan?** | Jika target kontinu dan hubungan linear. | Jika ingin klasifikasi biner dan menghitung probabilitas. | Jika data tidak berdistribusi normal. |
| **Contoh Kasus** | Prediksi harga rumah berdasarkan luas tanah. | Prediksi apakah email adalah spam atau bukan. | Prediksi jumlah pelanggan ke toko berdasarkan iklan. |
| **Kelebihan** | Mudah dipahami dan diimplementasikan. | Dapat digunakan untuk klasifikasi probabilistik. | Fleksibel untuk berbagai distribusi data. |
| **Kekurangan** | Tidak bisa digunakan jika hubungan tidak linear. | Hanya menangani klasifikasi biner, tidak multi-kelas. | Lebih kompleks dan butuh pemahaman distribusi. |

D. Naïve Bayes  
Naïve Bayes adalah algoritma yang bekerja berdasarkan teorema Bayes, yaitu menghitung peluang suatu kejadian berdasarkan informasi sebelumnya. Sederhananya, algoritma ini mencoba mencari tahu seberapa besar kemungkinan sebuah data termasuk ke dalam suatu kategori berdasarkan probabilitas fitur-fiturnya.

Asumsi utama Naïve Bayes adalah bahwa setiap fitur dalam data dianggap independen satu sama lain, meskipun dalam kenyataannya hal ini jarang terjadi.

### Cara kerja Naïve Bayes:

1. Menghitung probabilitas setiap kelas berdasarkan data pelatihan.
2. Menghitung probabilitas setiap fitur dalam kelas tertentu.
3. Menggunakan teorema Bayes untuk menentukan kemungkinan terbesar suatu data baru masuk ke kelas tertentu.

### Kelebihan Naïve Bayes:

* Sangat cepat, bahkan pada dataset besar.
* Cocok untuk **teks klasifikasi**, seperti analisis sentimen dan filtering spam.
* Bekerja dengan baik meskipun data memiliki banyak fitur.

### Kekurangan Naïve Bayes:

* Asumsi independensi antar fitur sering kali tidak realistis.
* Kurang fleksibel untuk menangkap pola yang kompleks.

Contoh penerapan Naïve Bayes yang umum adalah **klasifikasi email sebagai spam atau bukan spam** berdasarkan kata-kata yang muncul dalam email tersebut.

## E. Decision Tree

Decision Tree bekerja dengan membangun pohon keputusan, di mana setiap node mewakili suatu pertanyaan atau aturan, dan setiap cabang mewakili hasil dari aturan tersebut.

### Cara kerja Decision Tree:

1. Memilih fitur yang paling berpengaruh terhadap klasifikasi sebagai root node.
2. Membagi dataset berdasarkan nilai dari fitur tersebut, membentuk cabang dalam pohon.
3. Mengulangi proses ini hingga semua data sudah terklasifikasi dengan baik atau memenuhi kriteria tertentu (misalnya kedalaman maksimum pohon).

### Kelebihan Decision Tree:

* Mudah dipahami karena menyerupai cara berpikir manusia dalam mengambil keputusan.
* Dapat menangani fitur numerik maupun kategori tanpa perlu normalisasi.
* Bisa digunakan baik untuk klasifikasi maupun regresi.

### Kekurangan Decision Tree:

* Rentan terhadap overfitting, terutama jika pohon terlalu dalam.
* Sensitif terhadap data yang bervariasi, sehingga membutuhkan pruning untuk menghindari model yang terlalu kompleks.

Contoh penerapan Decision Tree yang umum adalah memutuskan apakah seorang nasabah berhak mendapatkan pinjaman berdasarkan riwayat kredit, penghasilan, dan faktor lainnya.

## Tabel Perbedaan Algoritma Naïve Bayes dan Decision Tree

Berikut adalah tabel perbedaan utama antara algoritma Naïve Bayes dan Decision Tree dalam konteks klasifikasi data. Tabel ini dapat membantu dalam memahami kapan dan bagaimana menggunakan masing-masing algoritma berdasarkan karakteristik data dan kebutuhan analisis.

|  |  |  |
| --- | --- | --- |
| **Aspek** | **Naïve Bayes** | **Decision Tree** |
| **Tipe Algoritma** | Berbasis probabilitas (statistik) | Berbasis aturan (pohon keputusan) |
| **Cara Kerja** | Menghitung probabilitas suatu kelas berdasarkan fitur | Membagi dataset menjadi cabang berdasarkan aturan keputusan |
| **Kecepatan** | Sangat cepat, bahkan untuk dataset besar | Bisa lambat jika pohon sangat dalam |
| **Akurasi** | Bisa kurang akurat jika asumsi independensi tidak terpenuhi | Cenderung lebih akurat, terutama untuk data kompleks |
| **Kemampuan Menangani Overfitting** | Tidak rentan terhadap overfitting | Bisa mengalami overfitting jika pohon terlalu dalam |
| **Interpretabilitas** | Kurang intuitif karena berbasis probabilitas | Mudah dipahami karena menyerupai pengambilan keputusan manusia |
| **Jenis Data yang Cocok** | Data dengan banyak fitur independen, seperti teks (email spam, analisis sentimen) | Data dengan hubungan kompleks antar fitur, seperti keputusan kredit atau diagnosis penyakit |
| **Skalabilitas** | Sangat baik, bisa menangani dataset besar | Bisa menjadi lambat untuk dataset besar dengan banyak fitur |

# 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⋅y*k*(*x*,*y*)=*x*⋅*y* (produk dalam dari x*x* dan y*y*).
  + **radial**: Kernel radial didefinisikan sebagai exp⁡(−g∥x−y∥2)exp(−*g*∥*x*−*y*∥2), di mana g*g* adalah gamma.
  + **polynomial**: Kernel polinomial didefinisikan sebagai k(x,y)=(x⋅y+1)d*k*(*x*,*y*)=(*x*⋅*y*+1)*d*, di mana d*d* adalah derajat polinomial.
  + **sigmoid**: Kernel sigmoid didefinisikan sebagai tanh⁡(a⋅x⋅y+b)tanh(*a*⋅*x*⋅*y*+*b*), di mana a*a* adalah alpha dan b*b* adalah konstanta intercept.
  + **anova**: Kernel anova didefinisikan sebagai (∑exp⁡(−g(x−y)))d(∑exp(−*g*(*x*−*y*)))*d*, di mana g*g* adalah gamma dan d*d* adalah derajat.
  + **epachnenikov**: Kernel Epanechnikov didefinisikan sebagai 34(1−u2)43​(1−*u*2) untuk u*u* antara -1 dan 1.
  + **gaussian combination**:  
    Kernel kombinasi Gaussian.
  + **multiquadric**: Kernel multiquadric didefinisikan sebagai ∥x−y∥2+c2∥*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:
  + Melewatkan 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(Atribut∣Kelas)*P*(Atribut∣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 Tre

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:  
  *(Jumlah atribut + 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.

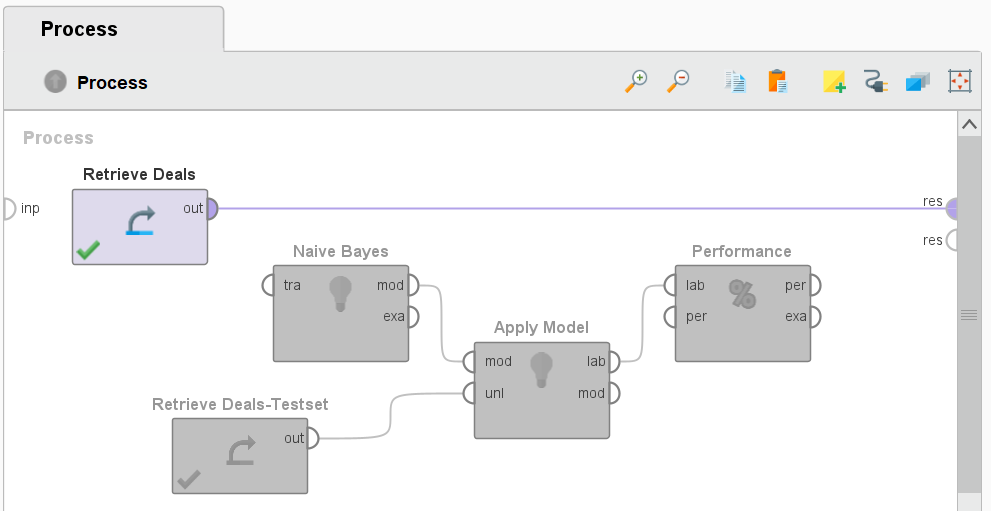
### 10. Local Random Seed (Seed Acak Lokal)

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

# Implementasi Model Algoritma Naïve Bayes dan Decision Tree di Rapidminer dari Video Pembelajaran

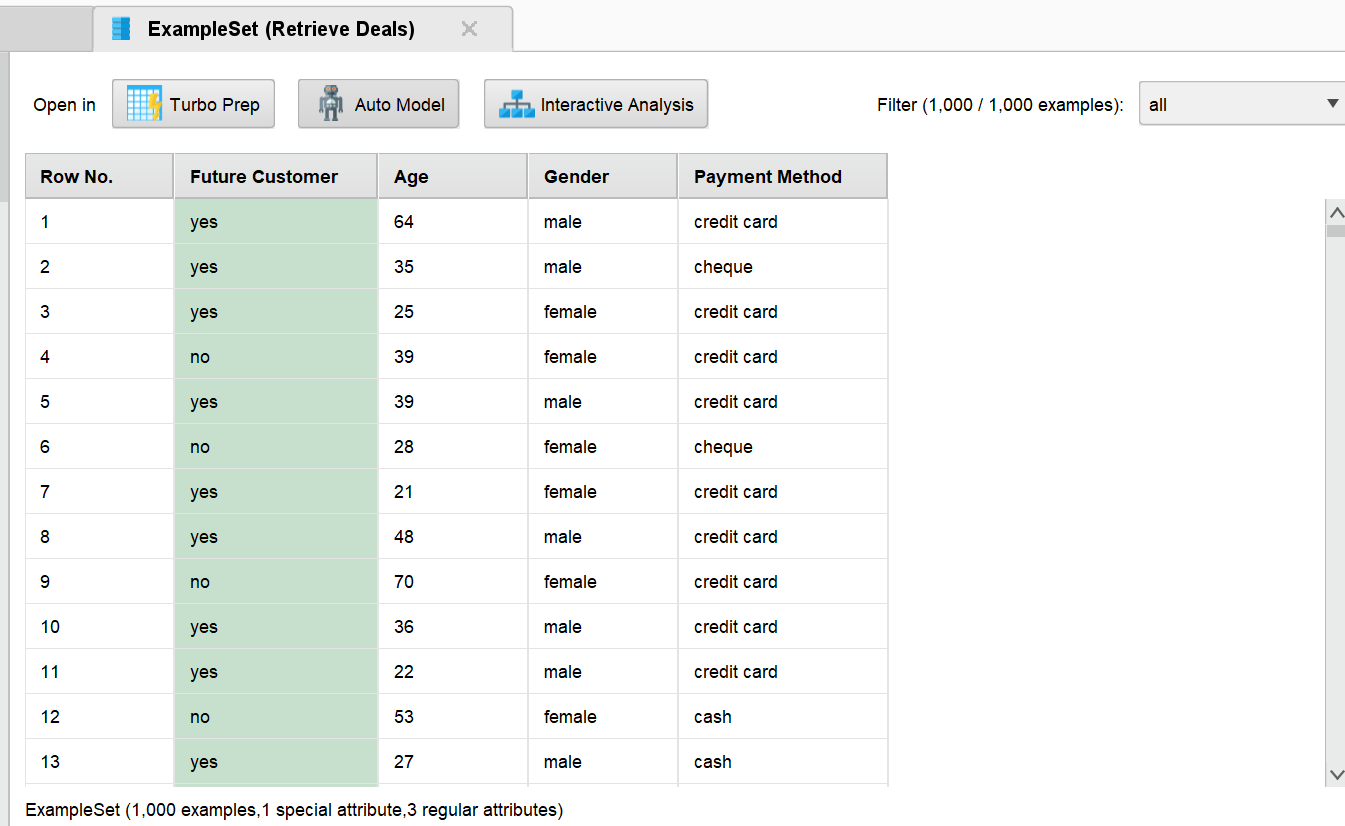
## A. Naïve Bayes

### 1. Retrieve dan Melihat Isi Dataset Deals



Gambar 3.1. Retrieve dataset Deals dari Sample Data di Rapidminer

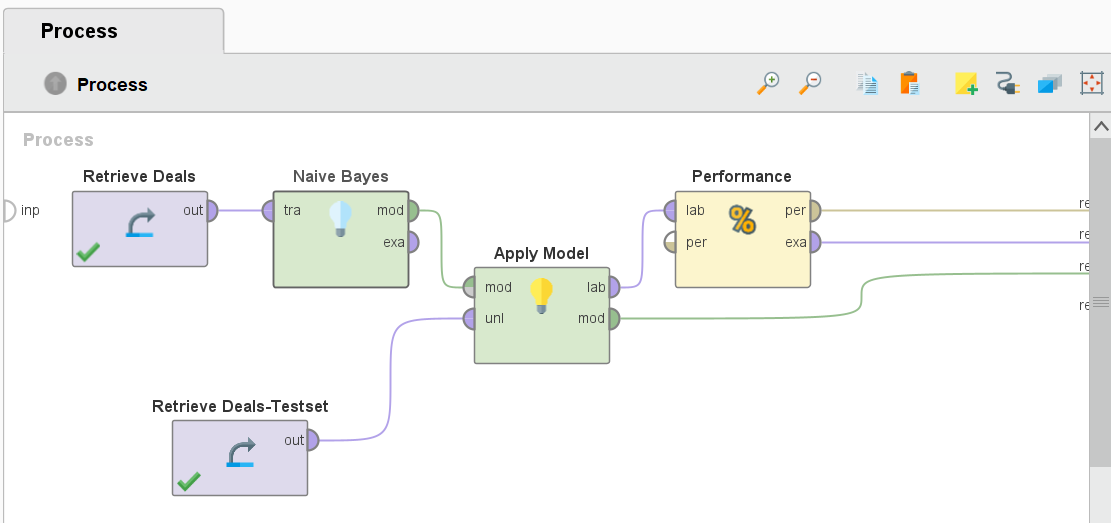
Disini kita menggunakan dataset Deals yang berasal dari folder sample data di Rapidminer ke dalam process. Pada gambar diatas, kita akan mencoba untuk melihat dan memahami isi dataset Deals terlebih dahulu sebelum memulai proses prediksi atau implementasi model algoritma Naïve Bayes di Rapidminer ini.



Gambar 3.2. Isi dataset Deals, terdapat 100 baris dan 4 kolom (*Future Customer*, *Age*, *Gender*, *Payment Method*)

Nah, bisa kita lihat isi dataset Deals di bagian Result, dalam dataset ini terdapat 1000 baris dan 4 kolom, yang dimana kolom-kolomnya meliputi *Future Customer*, *Age*, *Gender*, *Payment Method*. Lalu untuk target feature-nya adalah kolom Feature Customer.

### 2. Implementasi Model Naïve Bayes di Rapidminer



Gambar 3.3. Implementasi Model Naïve Bayes

### 3. 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).

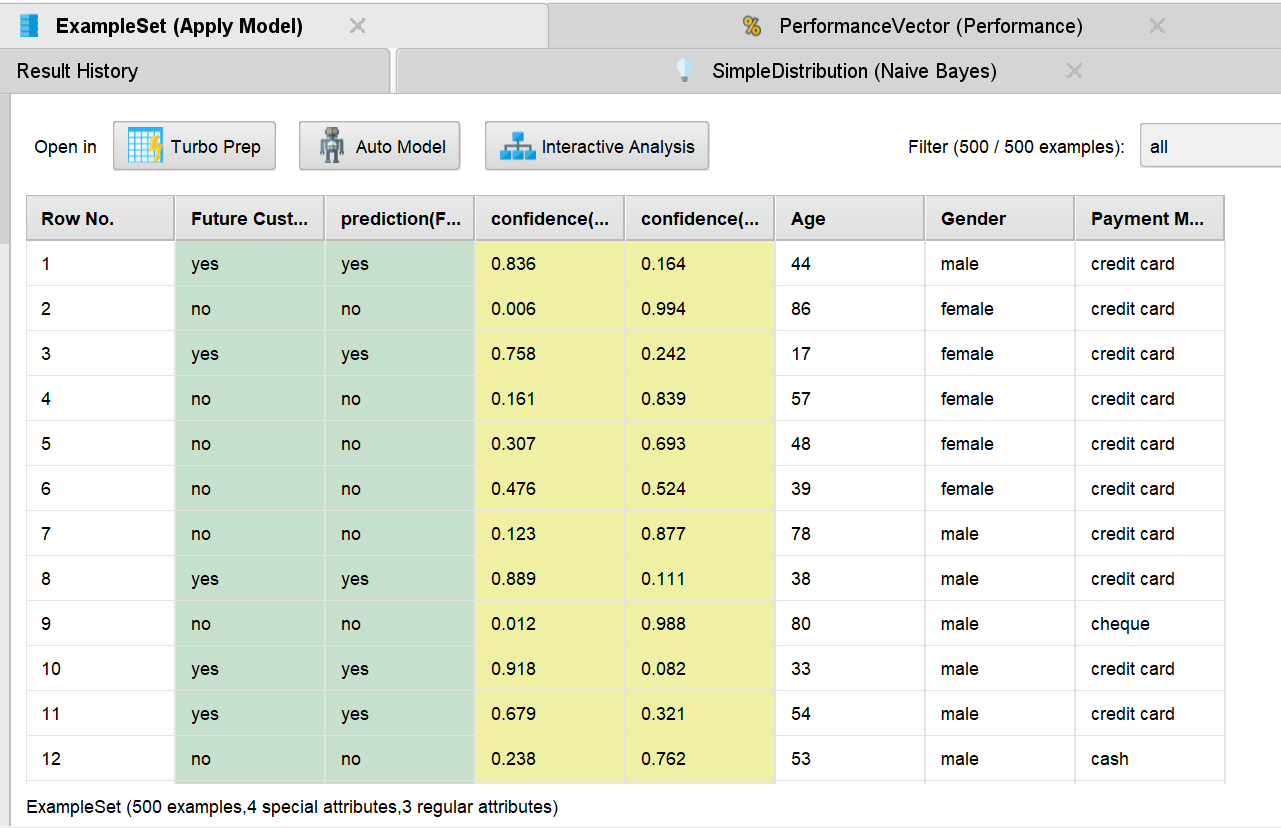
#### e. Retrieve Deals-Testset

* 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.

### 4. Hasil Klasifikasi Model Naïve Bayes



Gambar 3.4. Hasil klasifikasi model Naïve Bayes

**Penjelasan Hasil Klasifikasi Model Naive Bayes**

**1. Apa yang Terjadi?**

Setelah model Naive Bayes dilatih, model tersebut digunakan untuk memprediksi kelas pada dataset baru. Hasil prediksi ini ditampilkan dalam bentuk tabel yang berisi beberapa kolom penting. Mari kita bahas kolom-kolom tersebut satu per satu.

**2. Kolom-Kolom dalam Tabel**

1. **Row No.**: Nomor baris data. Ini hanya penanda untuk setiap contoh (data point) dalam dataset.
2. **Future Cust... (Future Customer)**: Ini adalah label sebenarnya (ground truth) dari data. Misalnya, "yes" berarti pelanggan tersebut benar-benar membeli produk, dan "no" berarti tidak.
3. **prediction(F...) (Prediction)**: Ini adalah hasil prediksi dari model Naive Bayes. Model mencoba menebak apakah pelanggan akan membeli produk ("yes") atau tidak ("no").
4. **confidence(...) (Confidence)**: Ini adalah tingkat kepercayaan model terhadap prediksinya. Nilainya antara 0 dan 1. Semakin mendekati 1, semakin yakin model dengan prediksinya.
   * **confidence(yes)**: Kepercayaan model bahwa pelanggan akan membeli produk.
   * **confidence(no)**: Kepercayaan model bahwa pelanggan tidak akan membeli produk.
5. **Age**: Usia pelanggan. Ini adalah salah satu fitur yang digunakan model untuk membuat prediksi.
6. **Gender**: Jenis kelamin pelanggan. Fitur lain yang digunakan model.
7. **Payment M... (Payment Method)**: Metode pembayaran yang digunakan pelanggan. Fitur lain yang digunakan model.

**3. Contoh Baris dalam Tabel**

Mari kita lihat beberapa baris dalam tabel untuk memahami hasilnya:

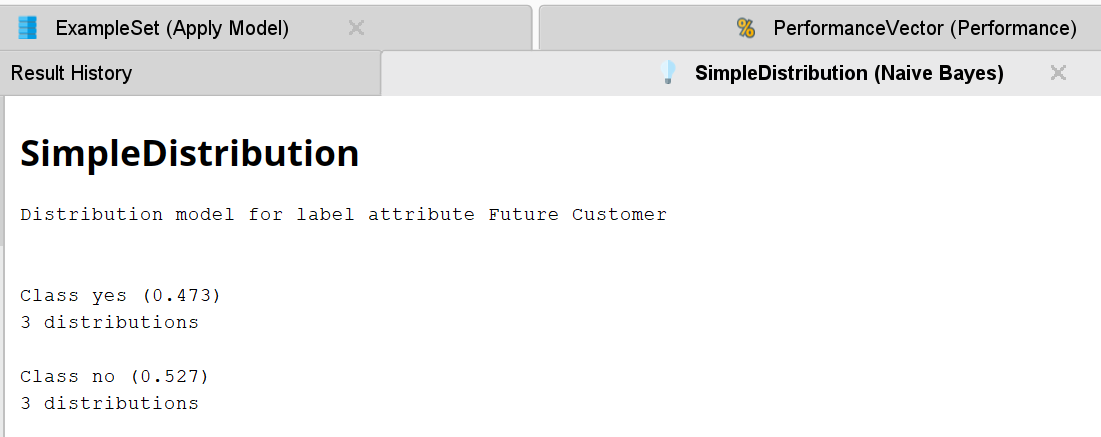
* **Baris 1**:
  + **Future Customer**: "yes" (pelanggan ini benar-benar membeli produk).
  + **Prediction**: "yes" (model memprediksi dengan benar bahwa pelanggan akan membeli).
  + **Confidence(yes)**: 0.836 (model sangat yakin dengan prediksinya).
  + **Confidence(no)**: 0.164 (tingkat kepercayaan bahwa pelanggan tidak akan membeli sangat rendah).
  + **Age**: 44 tahun.
  + **Gender**: Male.
  + **Payment Method**: Credit card.

**Kesimpulan**: Model bekerja dengan baik untuk pelanggan ini. Prediksinya benar, dan tingkat kepercayaannya tinggi.

* **Baris 2**:
  + **Future Customer**: "no" (pelanggan ini tidak membeli produk).
  + **Prediction**: "no" (model memprediksi dengan benar bahwa pelanggan tidak akan membeli).
  + **Confidence(yes)**: 0.006 (model sangat tidak yakin bahwa pelanggan akan membeli).
  + **Confidence(no)**: 0.994 (model sangat yakin bahwa pelanggan tidak akan membeli).
  + **Age**: 86 tahun.
  + **Gender**: Female.
  + **Payment Method**: Credit card.

**Kesimpulan**: Model juga bekerja dengan baik untuk pelanggan ini. Prediksinya benar, dan tingkat kepercayaannya sangat tinggi.

### 5. Simple Distribution model Naïve Bayes



Gambar 3.5. Simple Distribution model Naïve Bayes

**Penjelasan Hasil Simple Distribution**

**1. Apa Itu Simple Distribution?**

Simple Distribution adalah cara untuk melihat bagaimana model Naive Bayes membagi atau mendistribusikan probabilitas untuk setiap kelas (label) berdasarkan fitur-fitur yang ada. Dalam kasus ini, label yang kita lihat adalah **Future Customer**, yang memiliki dua kelas: **yes** (pelanggan akan membeli) dan **no** (pelanggan tidak akan membeli).

**2. Apa yang Ditunjukkan oleh Hasil Ini?**

1. **Class yes (0.473)**:
   * Ini berarti bahwa **47.3%** dari data pelatihan memiliki label **yes** (pelanggan akan membeli produk).
   * Angka ini menunjukkan seberapa sering kelas "yes" muncul dalam dataset.
2. **Class no (0.527)**:
   * Ini berarti bahwa **52.7%** dari data pelatihan memiliki label **no** (pelanggan tidak akan membeli produk).
   * Angka ini menunjukkan seberapa sering kelas "no" muncul dalam dataset.
3. **3 distributions**:
   * Ini menunjukkan bahwa ada **3 fitur** yang digunakan oleh model untuk membedakan antara kelas "yes" dan "no". Misalnya, fitur-fitur tersebut bisa berupa **Usia**, **Jenis Kelamin**, dan **Metode Pembayaran**.
   * Model Naive Bayes menggunakan distribusi probabilitas dari ketiga fitur ini untuk memprediksi apakah pelanggan akan membeli produk atau tidak.

**3. Apa Arti dari Angka-Angka Ini?**

* **47.3% yes dan 52.7% no**: Dataset yang digunakan untuk melatih model ini memiliki distribusi yang cukup seimbang antara pelanggan yang membeli dan tidak membeli. Ini berarti model tidak terlalu bias ke salah satu kelas.
* **3 distributions**: Model menggunakan tiga fitur untuk memprediksi kelas. Misalnya:
  + **Usia**: Apakah pelanggan muda atau tua?
  + **Jenis Kelamin**: Apakah pelanggan laki-laki atau perempuan?
  + **Metode Pembayaran**: Apakah pelanggan menggunakan kartu kredit, tunai, atau cek?

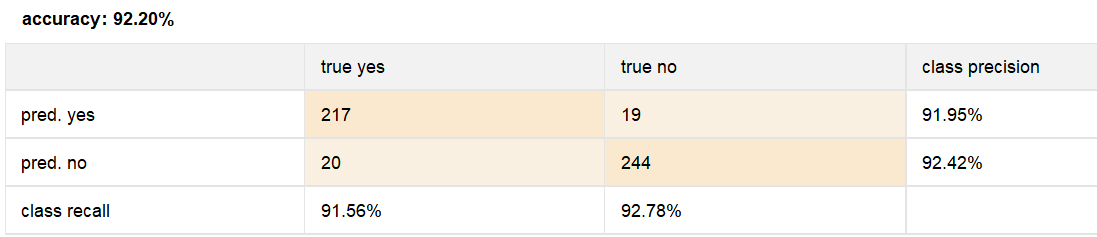
**4. Bagaimana Model Menggunakan Distribusi Ini?**

Model Naive Bayes menggunakan distribusi ini untuk menghitung probabilitas bersyarat. Misalnya:

* Jika seorang pelanggan berusia muda, perempuan, dan menggunakan kartu kredit, model akan melihat distribusi probabilitas dari ketiga fitur ini untuk memprediksi apakah pelanggan tersebut akan membeli produk atau tidak.

Hasil Simple Distribution dari model Naive Bayes menunjukkan bahwa dataset yang digunakan memiliki distribusi yang seimbang antara pelanggan yang membeli dan tidak membeli. Model menggunakan tiga fitur untuk memprediksi kelas, dan distribusi probabilitas ini membantu model dalam membuat prediksi yang lebih akurat.

### 6. Hasil Akurasi Model Naïve Bayes



Gambar 3.6. Akurasi model Naïve Bayes

**Penjelasan Hasil Akurasi Model Naive Bayes**

**1. Apa yang Terjadi?**

Setelah model Naive Bayes dilatih, kita perlu mengevaluasi seberapa baik model tersebut bekerja. Salah satu cara untuk mengevaluasi model adalah dengan melihat **akurasi**, yaitu seberapa sering model membuat prediksi yang benar. Dalam kasus ini, akurasi model adalah **92.20%**, yang artinya model benar dalam memprediksi **92.20%** dari total data yang diuji.

**2. Tabel Confusion Matrix**

Tabel yang diberikan adalah **confusion matrix**, yang membantu kita memahami bagaimana model melakukan prediksi. Mari kita bahas kolom dan baris dalam tabel ini:

1. **true yes**: Ini adalah jumlah data yang sebenarnya memiliki label **yes** (pelanggan akan membeli produk).
2. **true no**: Ini adalah jumlah data yang sebenarnya memiliki label **no** (pelanggan tidak akan membeli produk).
3. **pred. yes**: Ini adalah jumlah data yang diprediksi oleh model sebagai **yes**.
4. **pred. no**: Ini adalah jumlah data yang diprediksi oleh model sebagai **no**.

**3. Detail Hasil dalam Tabel**

* **pred. yes**:
  + **217** data diprediksi sebagai **yes** dan benar-benar **yes** (true yes).
  + **19** data diprediksi sebagai **yes** tetapi sebenarnya **no** (false yes).
  + **Class Precision**: 91.95%  
    Artinya, dari semua data yang diprediksi sebagai **yes**, **91.95%** benar-benar **yes**.
* **pred. no**:
  + **244** data diprediksi sebagai **no** dan benar-benar **no** (true no).
  + **20** data diprediksi sebagai **no** tetapi sebenarnya **yes** (false no).
  + **Class Precision**: 92.42%  
    Artinya, dari semua data yang diprediksi sebagai **no**, **92.42%** benar-benar **no**.
* **class recall**:
  + **91.56%** untuk kelas **yes**: Artinya, dari semua data yang sebenarnya **yes**, model berhasil mengidentifikasi **91.56%**-nya.
  + **92.78%** untuk kelas **no**: Artinya, dari semua data yang sebenarnya **no**, model berhasil mengidentifikasi **92.78%**-nya.

**4. Apa yang Bisa Kita Pelajari dari Hasil Ini?**

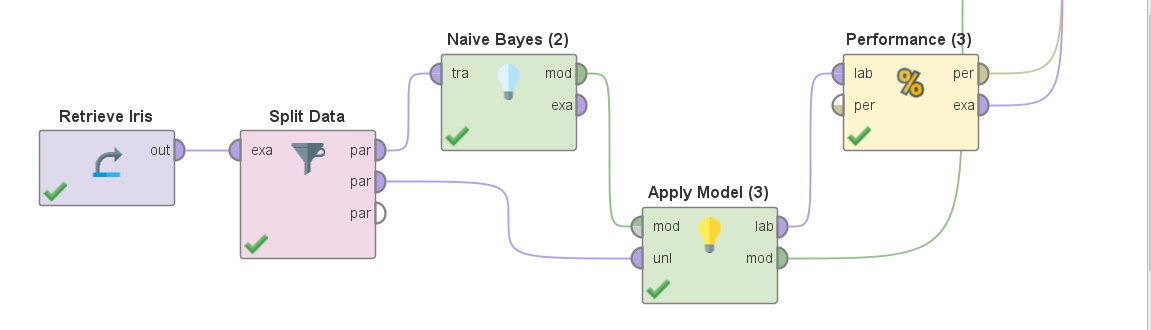
1. **Akurasi Tinggi (92.20%)**: Model ini bekerja sangat baik karena mampu memprediksi dengan benar **92.20%** dari total data. Ini berarti model cukup andal untuk digunakan.
2. **Precision dan Recall yang Seimbang**:
   * **Precision**: Model memiliki precision yang tinggi untuk kedua kelas (91.95% untuk "yes" dan 92.42% untuk "no"). Ini berarti ketika model memprediksi suatu kelas, kemungkinan besar prediksinya benar.
   * **Recall**: Model juga memiliki recall yang tinggi (91.56% untuk "yes" dan 92.78% untuk "no"). Ini berarti model mampu mengidentifikasi sebagian besar data yang sebenarnya termasuk dalam setiap kelas.
3. **Kesalahan Model**:
   * Model membuat **19 kesalahan** dengan memprediksi "yes" padahal sebenarnya "no" (false yes).
   * Model juga membuat **20 kesalahan** dengan memprediksi "no" padahal sebenarnya "yes" (false no).
   * Meskipun jumlah kesalahannya kecil, ini bisa menjadi titik untuk investigasi lebih lanjut. Misalnya, apakah ada pola tertentu yang membuat model salah memprediksi data-data ini?

**5. Apa yang Harus Dilakukan Selanjutnya?**

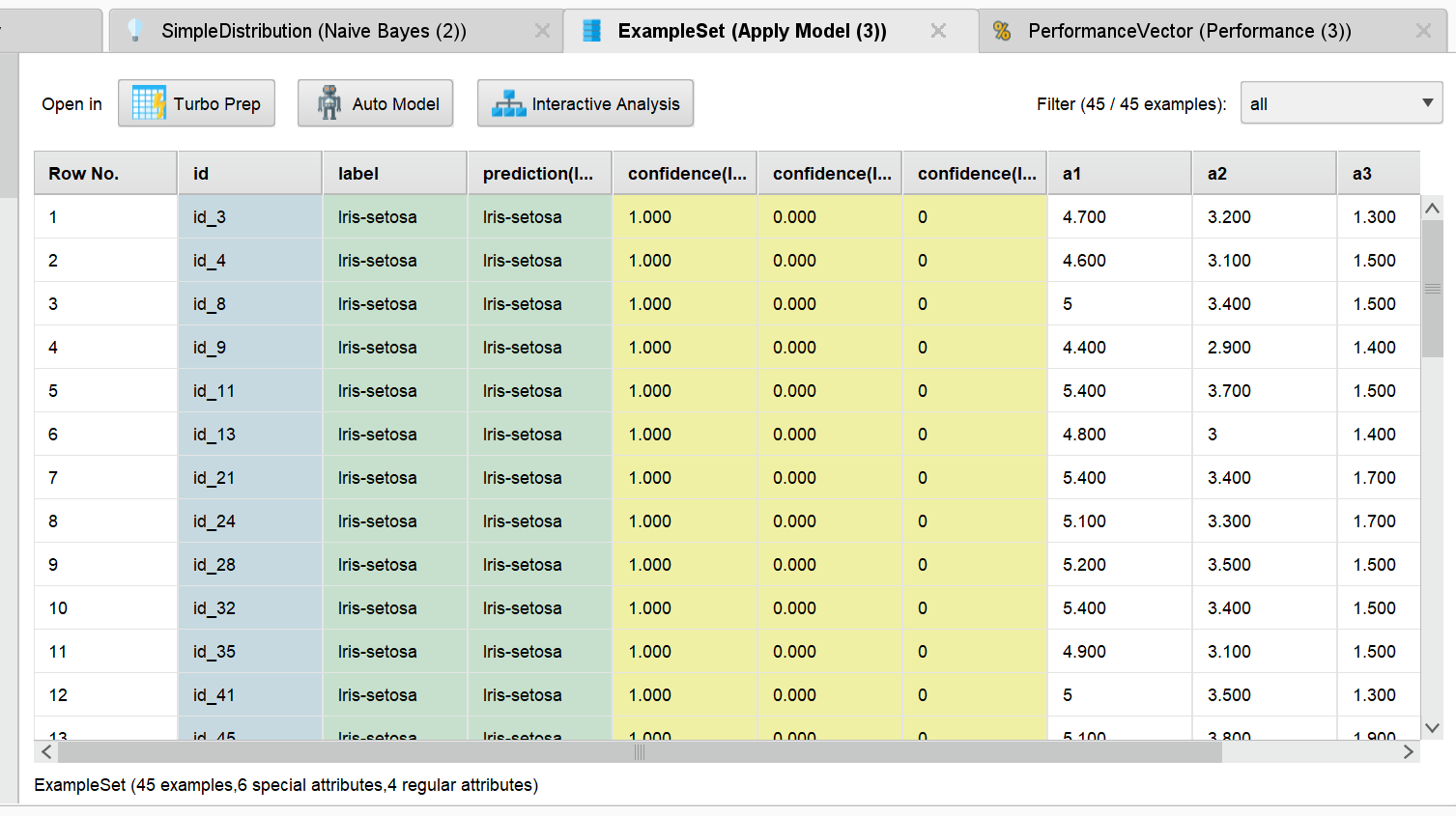
* **Analisis Kesalahan**: Kita bisa melihat lebih detail data-data yang salah diprediksi. Apakah ada pola tertentu yang membuat model salah? Misalnya, apakah pelanggan dengan usia tertentu atau metode pembayaran tertentu lebih sulit diprediksi?
* **Optimasi Model**: Jika kita ingin meningkatkan performa model, kita bisa mencoba:
  + Menambahkan fitur-fitur baru yang mungkin lebih relevan.
  + Menggunakan teknik seperti cross-validation untuk memastikan model tidak overfitting.
  + Mencoba algoritma lain untuk membandingkan performa.
* **Evaluasi Lebih Lanjut**: Selain akurasi, kita juga bisa melihat metrik lain seperti **F1-score** atau **ROC-AUC** untuk mendapatkan gambaran yang lebih lengkap tentang performa model.

Hasil akurasi model Naive Bayes sebesar **92.20%** menunjukkan bahwa model ini bekerja sangat baik dalam memprediksi apakah pelanggan akan membeli produk atau tidak. Precision dan recall yang seimbang juga menunjukkan bahwa model tidak hanya akurat tetapi juga mampu mengidentifikasi sebagian besar data dengan benar. Meskipun ada beberapa kesalahan, secara keseluruhan model ini cukup andal untuk digunakan.

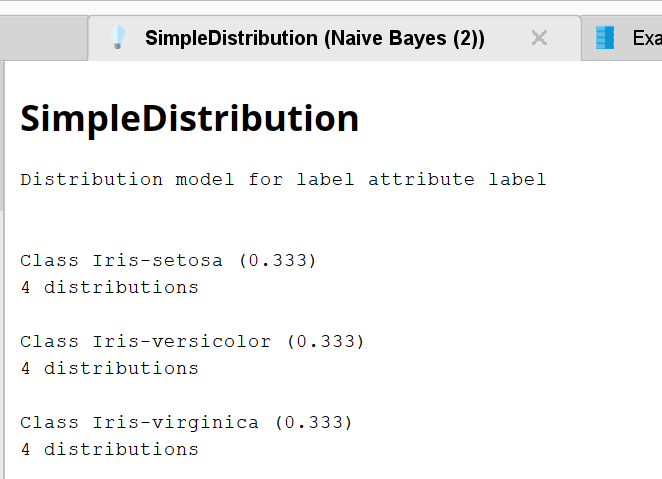
### 7. Implementasi Model Naïve Bayes pada Dataset Iris



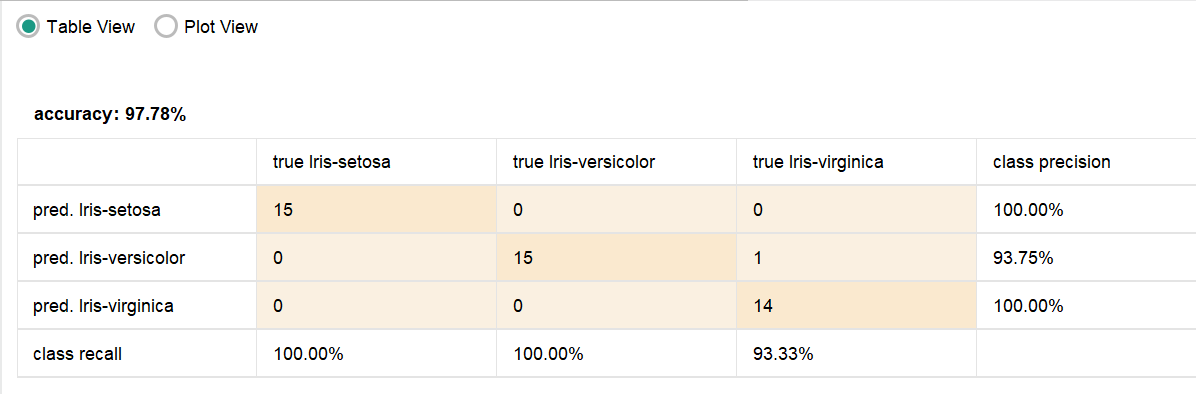
Gambar 3.7. Implementasi Model Naïve Bayes pada Dataset Iris



Gambar 3.8. Hasil Klasifikasi Model Naïve Bayes pada Dataset Iris



Gambar 3.9. Simple Distribution Model Naïve Bayes pada Dataset Iris

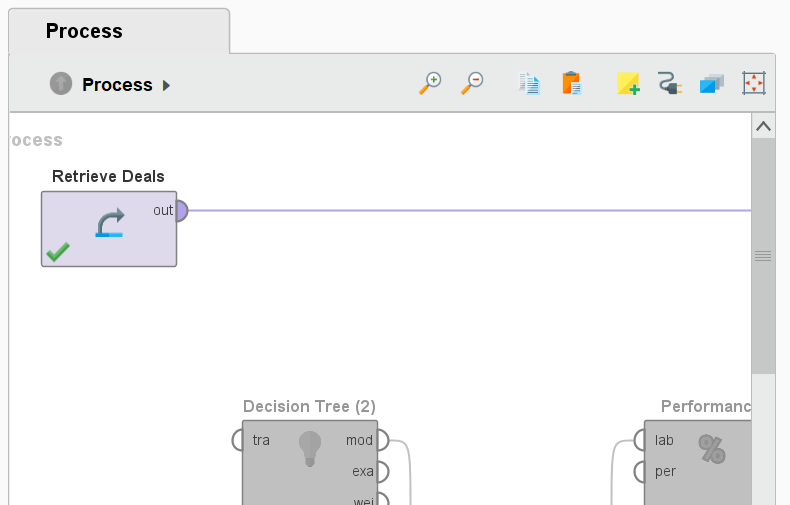


Gambar 3.10. Akurasi model Naïve Bayes pada Dataset Iris

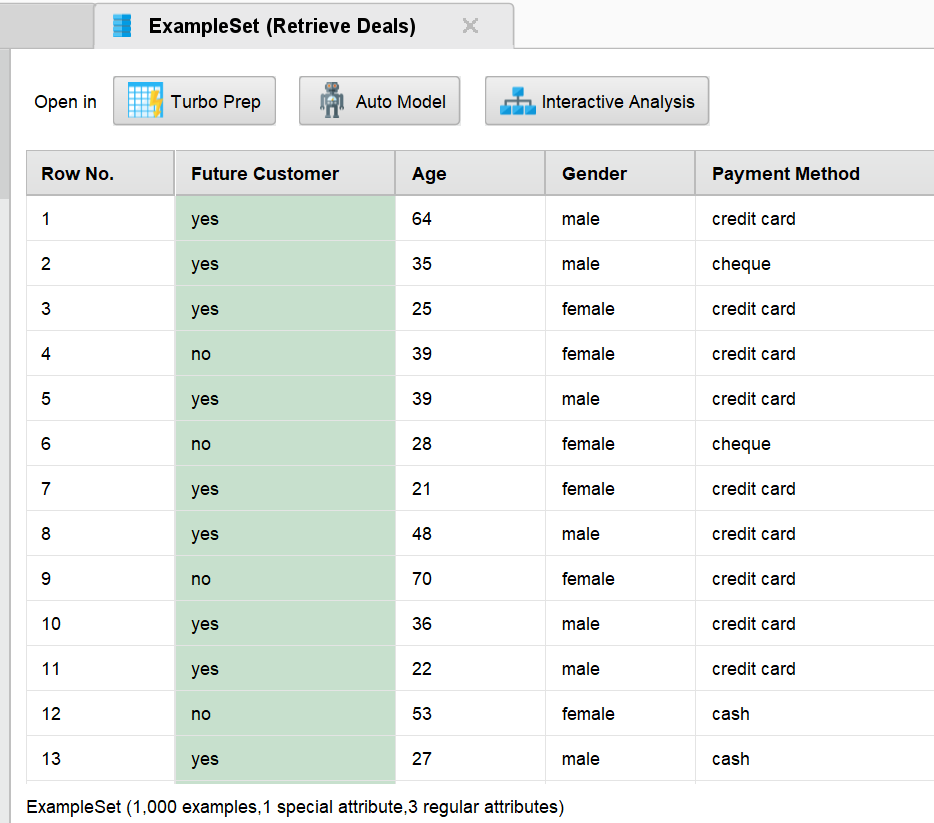
## 

## B. Decision Tree

### 1. Retrieve dan Melihat Isi Dataset Deals

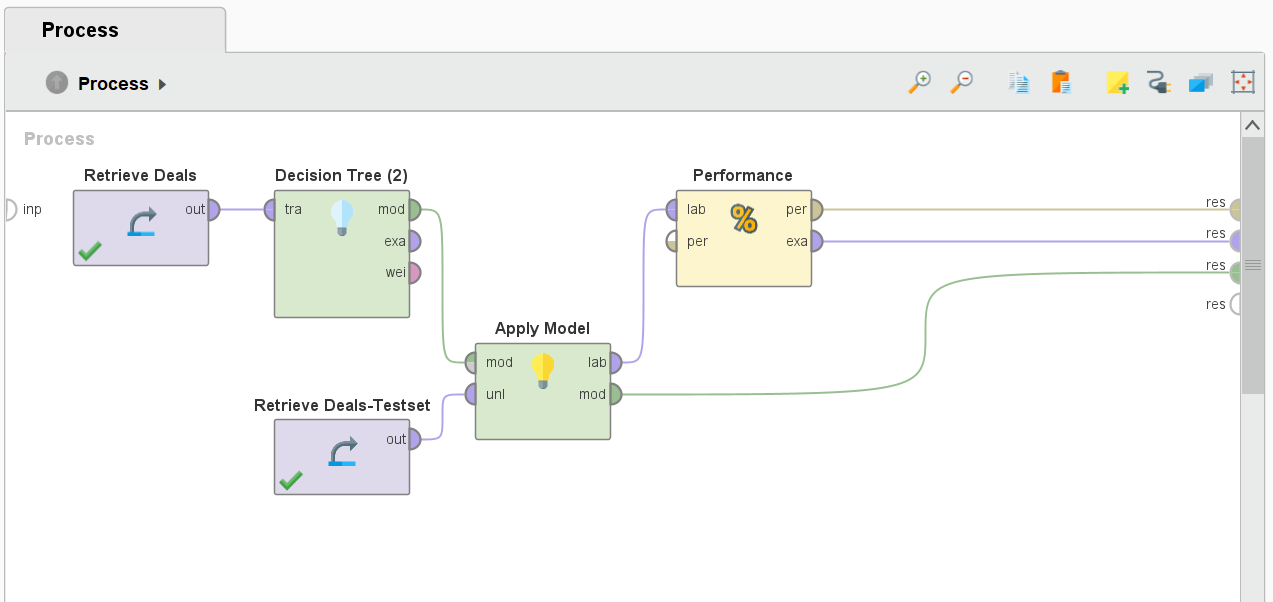


Disini kita menggunakan dataset Deals yang berasal dari folder sample data di Rapidminer ke dalam process. Pada gambar diatas, kita akan mencoba untuk melihat dan memahami isi dataset Deals terlebih dahulu sebelum memulai proses prediksi atau implementasi model algoritma Naïve Bayes di Rapidminer ini.



Nah, bisa kita lihat isi dataset Deals di bagian Result, dalam dataset ini terdapat 1000 baris dan 4 kolom, yang dimana kolom-kolomnya meliputi *Future Customer*, *Age*, *Gender*, *Payment Method*. Lalu untuk target feature-nya adalah kolom Feature Customer.

### 2. Implementasi Model Decision Tree di Rapidminer



### 3. Penjelasan Implementasi Decision Tree di RapidMiner

**1. Retrieve Deals**

* Langkah pertama adalah mengambil dataset yang akan digunakan untuk melatih model. Dataset ini biasanya berisi data historis atau data yang sudah dikumpulkan sebelumnya.
* **Contoh**: Misalnya, dataset ini berisi informasi tentang transaksi pelanggan, seperti usia, pendapatan, dan apakah mereka membeli produk tertentu atau tidak.
* **Tujuan**: Dataset ini akan digunakan untuk melatih model Decision Tree agar bisa memprediksi kelas (misalnya, "Beli" atau "Tidak Beli") berdasarkan fitur-fitur yang ada.

**2. Decision Tree**

* Di sini, model Decision Tree dibangun menggunakan dataset yang telah diambil. Decision Tree adalah algoritma klasifikasi yang membagi data menjadi subset-subset berdasarkan nilai fitur tertentu.
* **Cara Kerja**:
  + Algoritma memilih fitur terbaik untuk memisahkan data menjadi subset yang lebih homogen (murni).
  + Misalnya, jika fitur "Usia" adalah yang terbaik untuk memisahkan data, maka algoritma akan membuat cabang berdasarkan kategori usia.
  + Proses ini berlanjut hingga subset yang dihasilkan cukup murni atau kriteria tertentu terpenuhi.
* **Output**: Model yang sudah dilatih (disebut **mod**) dan data pelatihan (disebut **tra**).

**3. 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.

**4. Performance**

* Setelah model dilatih, langkah selanjutnya adalah mengevaluasi performanya. Ini dilakukan dengan menggunakan dataset uji.
* **Cara Kerja**:
  + Model akan memprediksi kelas untuk setiap contoh dalam dataset uji.
  + 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.

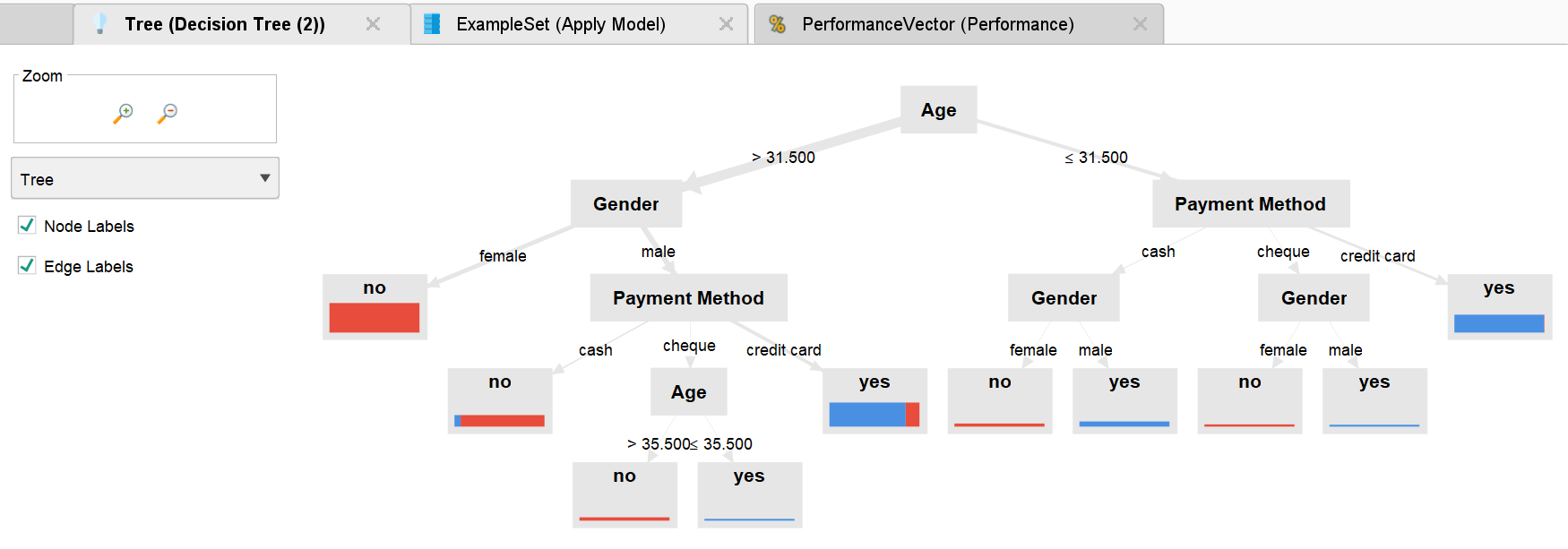
**5. Apply Model**

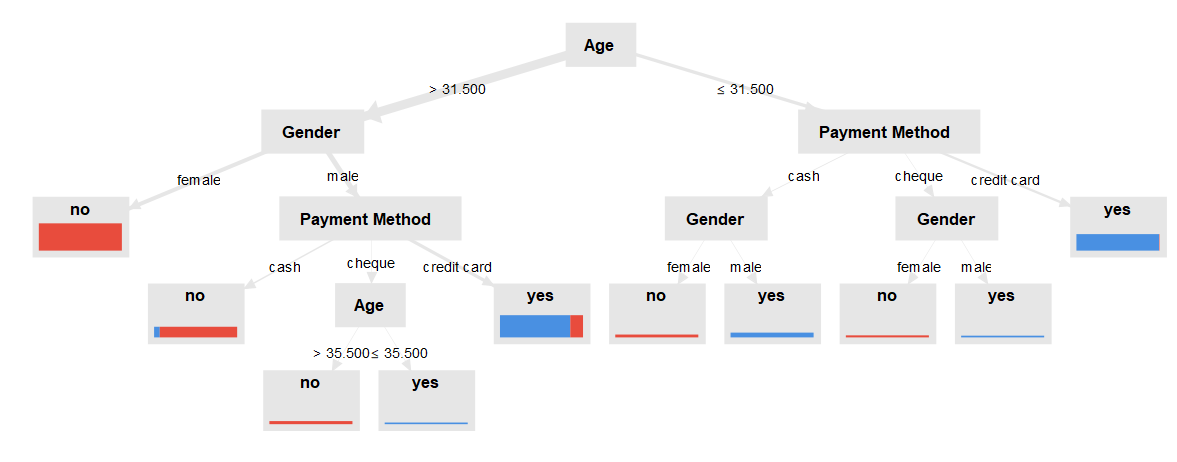
* Setelah model dievaluasi, langkah berikutnya adalah menerapkan model ini ke dataset baru yang belum diketahui labelnya (unlabeled data).
* **Cara Kerja**:
  + Model akan menggunakan fitur-fitur dalam dataset baru untuk memprediksi kelas.
  + Misalnya, jika dataset baru berisi informasi tentang pelanggan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

**Alur Kerja Secara Keseluruhan**

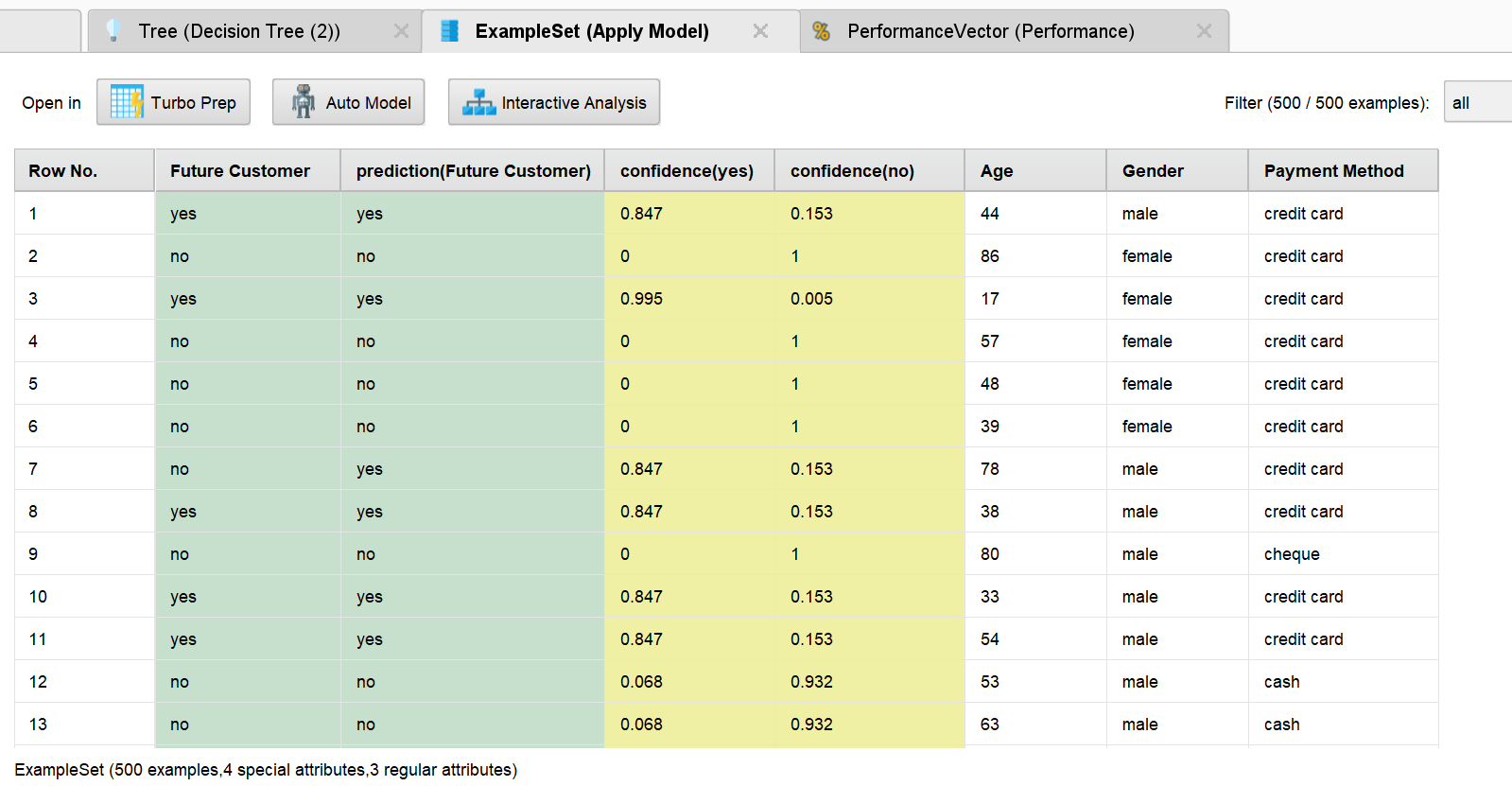
* Ambil Data Pelatihan: Dataset diambil untuk melatih model.
* Latih Model: Model Decision Tree dibangun menggunakan dataset pelatihan.
* Ambil Data Uji: Dataset uji diambil untuk menguji model pada data yang belum pernah dilihat sebelumnya.
* Evaluasi Performa: Model diuji menggunakan dataset uji untuk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

### 4. Gambar Model Decision Tree





### 5. Hasil Klasifikasi Model Decision Tree



**Penjelasan Hasil Klasifikasi Model Decision Tree**

**1. Apa yang Terjadi?**

Setelah model Decision Tree dilatih, model tersebut digunakan untuk memprediksi kelas pada dataset baru. Hasil prediksi ini ditampilkan dalam bentuk tabel yang berisi beberapa kolom penting. Mari kita bahas kolom-kolom tersebut satu per satu.

**2. Kolom-Kolom dalam Tabel**

* **Row No.**: Nomor baris data. Ini hanya penanda untuk setiap contoh (data point) dalam dataset.
* **Future Customer**: Ini adalah label sebenarnya (ground truth) dari data. Misalnya, "yes" berarti pelanggan tersebut benar-benar membeli produk, dan "no" berarti tidak.
* **prediction(Future Customer)**: Ini adalah hasil prediksi dari model Decision Tree. Model mencoba menebak apakah pelanggan akan membeli produk ("yes") atau tidak ("no").
* **confidence(yes)**: Ini adalah tingkat kepercayaan model bahwa pelanggan akan membeli produk. Nilainya antara 0 dan 1. Semakin mendekati 1, semakin yakin model dengan prediksinya.
* **confidence(no)**: Ini adalah tingkat kepercayaan model bahwa pelanggan tidak akan membeli produk. Nilainya juga antara 0 dan 1.
* **Age**: Usia pelanggan. Ini adalah salah satu fitur yang digunakan model untuk membuat prediksi.
* **Gender**: Jenis kelamin pelanggan. Fitur lain yang digunakan model.
* **Payment Method**: Metode pembayaran yang digunakan pelanggan. Fitur lain yang digunakan model.

**3. Contoh Baris dalam Tabel**

Mari kita lihat beberapa baris dalam tabel untuk memahami hasilnya:

* **Baris 1**:
  + **Future Customer**: "yes" (pelanggan ini benar-benar membeli produk).
  + **Prediction**: "yes" (model memprediksi dengan benar bahwa pelanggan akan membeli).
  + **Confidence(yes)**: 0.847 (model sangat yakin dengan prediksinya).
  + **Confidence(no)**: 0.153 (tingkat kepercayaan bahwa pelanggan tidak akan membeli sangat rendah).
  + **Age**: 44 tahun.
  + **Gender**: Male.
  + **Payment Method**: Credit card.

**Kesimpulan**: Model bekerja dengan baik untuk pelanggan ini. Prediksinya benar, dan tingkat kepercayaannya tinggi.

* **Baris 2**:
  + **Future Customer**: "no" (pelanggan ini tidak membeli produk).
  + **Prediction**: "no" (model memprediksi dengan benar bahwa pelanggan tidak akan membeli).
  + **Confidence(yes)**: 0 (model sangat tidak yakin bahwa pelanggan akan membeli).
  + **Confidence(no)**: 1 (model sangat yakin bahwa pelanggan tidak akan membeli).
  + **Age**: 86 tahun.
  + **Gender**: Female.
  + **Payment Method**: Credit card.

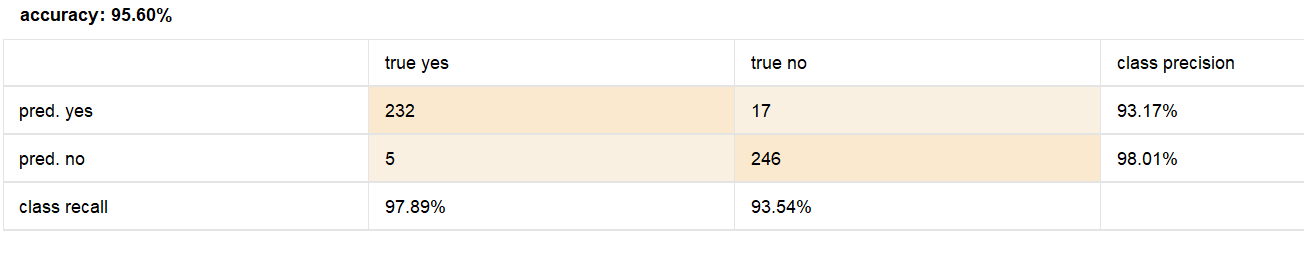
**Kesimpulan**: Model juga bekerja dengan baik untuk pelanggan ini. Prediksinya benar, dan tingkat kepercayaannya sangat tinggi.

**4. Apa yang Bisa Kita Pelajari dari Hasil Ini?**

* **Akurasi Model**: Dari contoh di atas, model Decision Tree tampaknya bekerja cukup baik. Sebagian besar prediksinya sesuai dengan label sebenarnya (Future Customer).
* **Tingkat Kepercayaan**: Nilai confidence menunjukkan seberapa yakin model dengan prediksinya. Semakin tinggi nilai confidence, semakin baik model bekerja.
* **Pengaruh Fitur**: Fitur seperti usia, jenis kelamin, dan metode pembayaran digunakan oleh model untuk membuat prediksi. Misalnya, pelanggan dengan usia lebih muda (Baris 3) cenderung diprediksi akan membeli produk.
* **Kasus yang Menarik**: Pada Baris 7, meskipun model memprediksi "yes", nilai confidence(yes) cukup tinggi (0.847), tetapi prediksinya salah. Ini bisa menjadi titik untuk investigasi lebih lanjut. Mungkin ada pola tertentu yang membuat model agak ragu.

Hasil klasifikasi model Decision Tree di RapidMiner menunjukkan bahwa model ini cukup baik dalam memprediksi apakah pelanggan akan membeli produk atau tidak. Tingkat kepercayaan (confidence) yang tinggi menunjukkan bahwa model yakin dengan prediksinya. Namun, tetap penting untuk mengevaluasi model lebih lanjut dan memeriksa kasus-kasus di mana model mungkin ragu atau salah.

### 6. Hasil Akurasi Model Decision Tree



**Penjelasan Hasil Akurasi Model Decision Tree**

**1. Apa yang Terjadi?**

Setelah model Decision Tree dilatih, kita perlu mengevaluasi seberapa baik model tersebut bekerja. Salah satu cara untuk mengevaluasi model adalah dengan melihat **akurasi**, yaitu seberapa sering model membuat prediksi yang benar. Dalam kasus ini, akurasi model adalah **95.60%**, yang artinya model benar dalam memprediksi **95.60%** dari total data yang diuji.

**2. Tabel Confusion Matrix**

Tabel yang diberikan adalah **confusion matrix**, yang membantu kita memahami bagaimana model melakukan prediksi. Mari kita bahas kolom dan baris dalam tabel ini:

1. **true yes**: Ini adalah jumlah data yang sebenarnya memiliki label **yes** (pelanggan akan membeli produk).
2. **true no**: Ini adalah jumlah data yang sebenarnya memiliki label **no** (pelanggan tidak akan membeli produk).
3. **pred. yes**: Ini adalah jumlah data yang diprediksi oleh model sebagai **yes**.
4. **pred. no**: Ini adalah jumlah data yang diprediksi oleh model sebagai **no**.

**3. Detail Hasil dalam Tabel**

* **pred. yes**:
  + **232** data diprediksi sebagai **yes** dan benar-benar **yes** (true yes).
  + **17** data diprediksi sebagai **yes** tetapi sebenarnya **no** (false yes).
  + **Class Precision**: 93.17%  
    Artinya, dari semua data yang diprediksi sebagai **yes**, **93.17%** benar-benar **yes**.
* **pred. no**:
  + **246** data diprediksi sebagai **no** dan benar-benar **no** (true no).
  + **5** data diprediksi sebagai **no** tetapi sebenarnya **yes** (false no).
  + **Class Precision**: 98.01%  
    Artinya, dari semua data yang diprediksi sebagai **no**, **98.01%** benar-benar **no**.
* **class recall**:
  + **97.89%** untuk kelas **yes**: Artinya, dari semua data yang sebenarnya **yes**, model berhasil mengidentifikasi **97.89%**-nya.
  + **93.54%** untuk kelas **no**: Artinya, dari semua data yang sebenarnya **no**, model berhasil mengidentifikasi **93.54%**-nya.

**4. Apa yang Bisa Kita Pelajari dari Hasil Ini?**

* **Akurasi Tinggi (95.60%)**: Model ini bekerja sangat baik karena mampu memprediksi dengan benar **95.60%** dari total data. Ini berarti model cukup andal untuk digunakan.
* **Precision dan Recall yang Seimbang**:
  + **Precision**: Model memiliki precision yang tinggi untuk kedua kelas (93.17% untuk "yes" dan 98.01% untuk "no"). Ini berarti ketika model memprediksi suatu kelas, kemungkinan besar prediksinya benar.
  + **Recall**: Model juga memiliki recall yang tinggi (97.89% untuk "yes" dan 93.54% untuk "no"). Ini berarti model mampu mengidentifikasi sebagian besar data yang sebenarnya termasuk dalam setiap kelas.
* **Kesalahan Model**:
  + Model membuat **17 kesalahan** dengan memprediksi "yes" padahal sebenarnya "no" (false yes).
  + Model juga membuat **5 kesalahan** dengan memprediksi "no" padahal sebenarnya "yes" (false no).
  + Meskipun jumlah kesalahannya kecil, ini bisa menjadi titik untuk investigasi lebih lanjut. Misalnya, apakah ada pola tertentu yang membuat model salah memprediksi data-data ini?

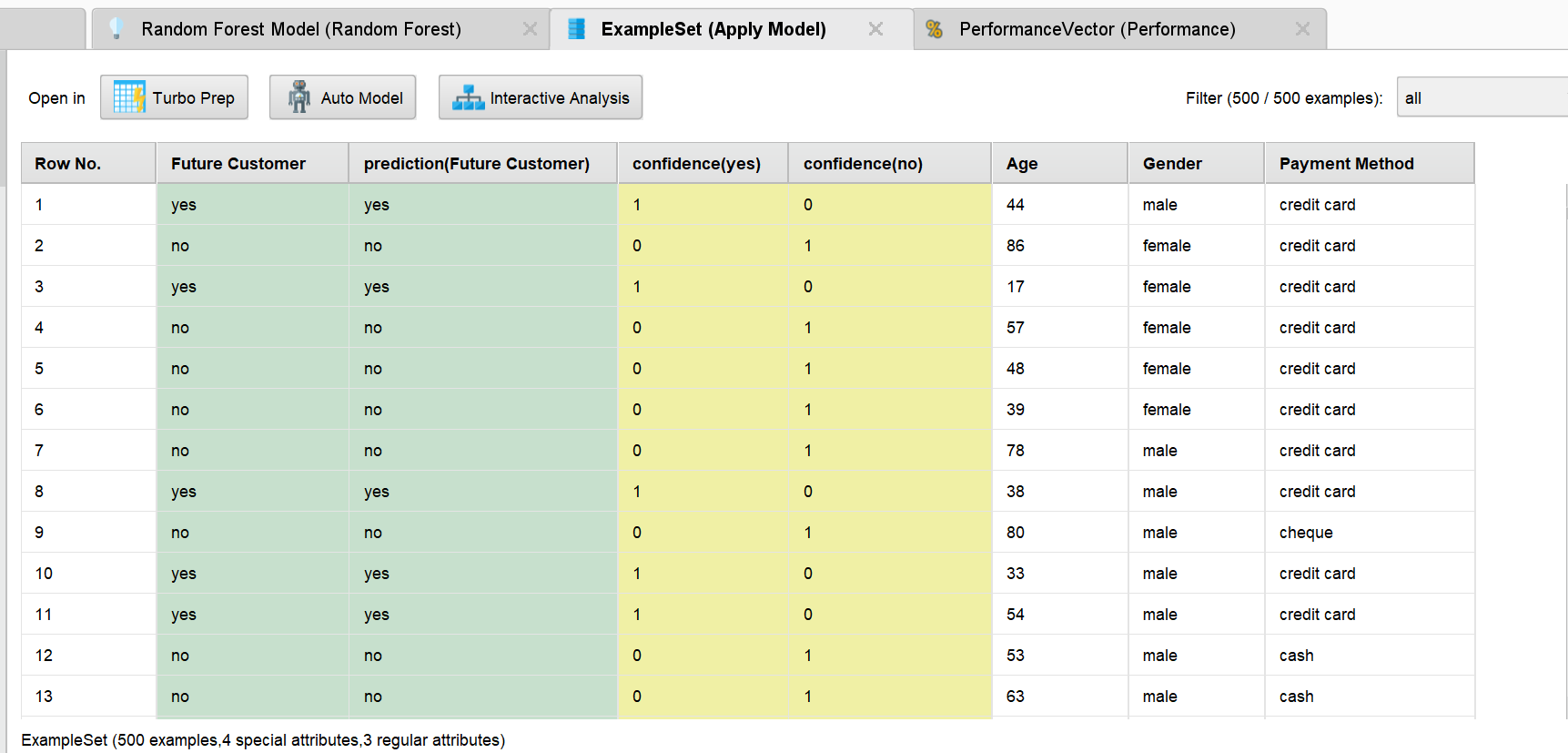
**5. Apa yang Harus Dilakukan Selanjutnya?**

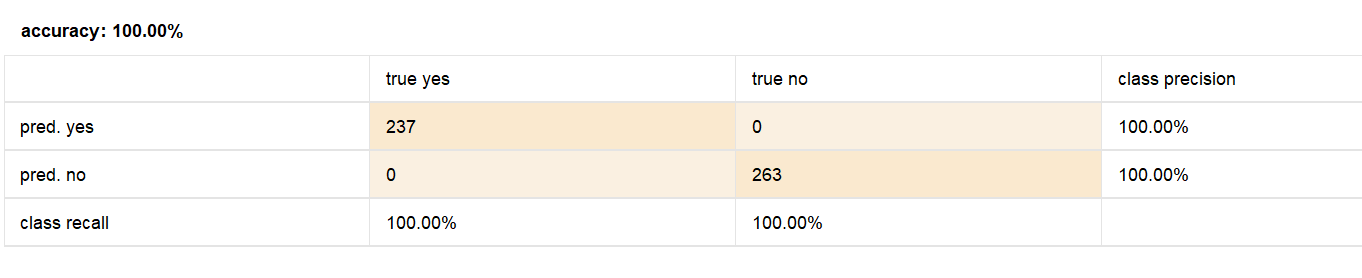
* **Analisis Kesalahan**: Kita bisa melihat lebih detail data-data yang salah diprediksi. Apakah ada pola tertentu yang membuat model salah? Misalnya, apakah pelanggan dengan usia tertentu atau metode pembayaran tertentu lebih sulit diprediksi?
* **Optimasi Model**: Jika kita ingin meningkatkan performa model, kita bisa mencoba:
  + Menambahkan fitur-fitur baru yang mungkin lebih relevan.
  + Menggunakan teknik seperti cross-validation untuk memastikan model tidak overfitting.
  + Mencoba algoritma lain untuk membandingkan performa.
* **Evaluasi Lebih Lanjut**: Selain akurasi, kita juga bisa melihat metrik lain seperti **F1-score** atau **ROC-AUC** untuk mendapatkan gambaran yang lebih lengkap tentang performa model.

Hasil akurasi model Decision Tree sebesar **95.60%** menunjukkan bahwa model ini bekerja sangat baik dalam memprediksi apakah pelanggan akan membeli produk atau tidak. Precision dan recall yang seimbang juga menunjukkan bahwa model tidak hanya akurat tetapi juga mampu mengidentifikasi sebagian besar data dengan benar. Meskipun ada beberapa kesalahan, secara keseluruhan model ini cukup andal untuk digunakan.

### 7. Implementasi Model Random Forest

### 





# Referensi

* <https://images.app.goo.gl/3HdSSQYbvA19gUXc9>
* [https://miro.medium.com/v2/resize:fit:1046/1\*l59BUnPwWHMf1H-GNxgZHQ.png](https://miro.medium.com/v2/resize:fit:1046/1*l59BUnPwWHMf1H-GNxgZHQ.png)
* <https://docs.rapidminer.com/2025.0/studio/operators/modeling/predictive/functions/linear_regression.html>
* <https://docs.rapidminer.com/2025.0/studio/operators/modeling/predictive/logistic_regression/logistic_regression_evolutionary.html>
* <https://docs.rapidminer.com/2025.0/studio/operators/modeling/predictive/functions/generalized_linear_model.html>
* <https://docs.rapidminer.com/2025.0/studio/operators/modeling/predictive/bayesian/naive_bayes.html>
* <https://docs.rapidminer.com/2025.0/studio/operators/modeling/predictive/neural_nets/neural_net.html>
* <https://docs.rapidminer.com/2025.0/studio/operators/modeling/predictive/trees/parallel_decision_tree.html>
* <https://drive.google.com/file/d/1tUU9u0ZhUF50UD82L38wbq1aOVHTQbY2/view>
* https://drive.google.com/file/d/1XeDRZBXgSty0\_0sqZ-hT\_LAvaoUAk7Wc/view