

---

## Praktikum 3 – Machine Learning Regresi dan Evaluasi Model

Prepared By:  
Dr. Sirojul Munir S.Si,M.Kom  
Diah Ayu Puspasari  
Elyas Randi

### Tujuan:

1. Memahami konsep dasar regresi dalam konteks machine learning.
2. Mampu membangun model Linear Regression dan Multiple Linear Regression menggunakan pustaka program scikit-learn dan statmodels (OLS).
3. Mampu mengukur dan mengevaluasi kinerja model regresi menggunakan metrik seperti MAE, MSE, dan  $R^2$  Score.
4. Mampu menampilkan visualisasi hasil prediksi model regresi.

Dateline : 1 Pekan

Gitlab/Github :

Branch Repository : [PRODI ROMBEL]\_[NAMASINGKAT]\_[NIM] ( contoh: ti01\_budi\_0110112001)

### Aturan Pengerjaan:

1. Gunakan text editor yang nyaman bagi anda
2. Diperkenankan mengerjakan langsung bagi yang sudah memahami dan menguasai materi
3. Dilarang melakukan tindakan plagiarisme (asisten lab akan mengecek hasil pekerjaan)
  - a. 1x nilai praktikum terkait bernilai 0
  - b. 2x nilai matakuliah pemrograman web E
  - c. 3x mahasiswa akan di sidang komite etik kampus

---

## Pendahuluan

Dalam dunia machine learning, regresi merupakan salah satu metode paling dasar namun sangat penting. Model regresi memprediksi sebuah variabel yang belum diketahui nilainya menggunakan satu atau lebih variabel lain yang sudah diketahui nilainya. Salah satu penerapan regresi adalah digunakan untuk memprediksi nilai numerik kontinu, seperti:

- Harga rumah berdasarkan luas dan lokasi
- Jumlah penjualan berdasarkan waktu
- Tinggi badan berdasarkan usia
- Suhu udara berdasarkan kelembapan

Dengan kata lain, regresi memodelkan hubungan antara satu atau lebih variabel bebas (independen) dengan variabel target (dependen) yang bersifat kontinu. Terdapat beberapa jenis model regresi linear yang dapat diterapkan dalam membantu memprediksi suatu nilai kontinu, diantaranya adalah:

1. Linear Regression

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

2. Multiple Linear Regression

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \varepsilon_i$$

3. Polynomial Linear Regression

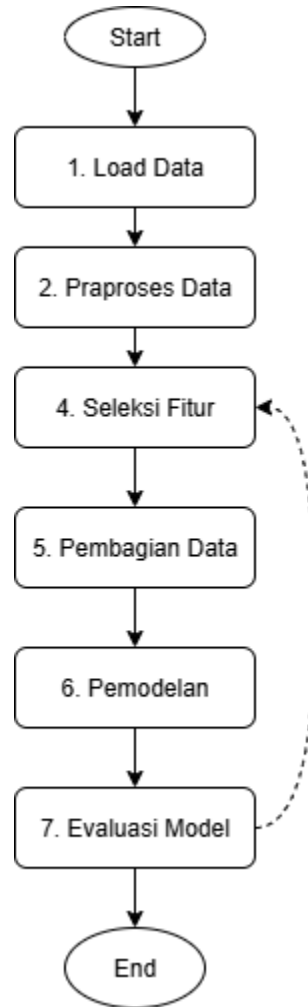
$$Y_i = \beta_0 + \beta_1 X_i + \beta_2 X_i^2 + \varepsilon_i$$

Pada praktikum ini, kita akan mempelajari bagaimana cara membangun model Linear Regression dan Multiple Linear Regression untuk memprediksi nilai berdasarkan data yang ada, melakukan evaluasi performa model, dan menampilkan hasilnya secara visual.

---

## Tahapan Pemodelan Machine Learning

Sebelum melakukan pemodelan machine learning, beberapa tahapan perlu dilakukan agar proses pemodelan berjalan dengan baik dan dapat di duplikasi untuk project machine learning lainnya. Pada Gambar 3.1 adalah tahapan pemodelan machine learning.



Gambar 3.1 Tahapan Pemodelan Machine Learning

Diagram alur (flowchart) pada Gambar 3.1 memperlihatkan tahapan dalam proses pengolahan data hingga evaluasi model pada suatu proyek *machine learning* atau analisis data.

### 1. Load Data

Tahap pertama adalah memuat data (Load Data). Pada langkah ini, data dikumpulkan dari berbagai sumber seperti file CSV, database, API, atau sumber lain yang relevan. Data yang diperoleh menjadi bahan mentah untuk proses selanjutnya.

### 2. Praproses Data

Selanjutnya dilakukan praproses data (Data Preprocessing). Tahap ini bertujuan untuk membersihkan dan menyiapkan data agar siap digunakan dalam pemodelan. Langkah-langkahnya meliputi:

- 
- Menangani data hilang (*missing values*),
  - Menghapus duplikasi,
  - Menormalisasi atau *standardize* data numerik,
  - Melakukan *encoding* pada data kategorikal.

### 3. Seleksi Fitur

Tahap berikutnya adalah seleksi fitur (Feature Selection). Pada langkah ini, dilakukan pemilihan fitur-fitur (variabel) yang paling relevan dan berpengaruh terhadap target yang akan diprediksi. Seleksi fitur membantu meningkatkan performa model dan mengurangi kompleksitas komputasi.

### 4. Pembagian Data

Setelah fitur dipilih, data dibagi menjadi dua bagian utama dalam tahap pembagian data (Data Splitting):

- Data latih (training set) untuk melatih model,
- Data uji (testing set) untuk menguji performa model.

Biasanya, pembagian dilakukan dengan rasio seperti 80:20 atau 70:30.

### 5. Pemodelan

Tahap selanjutnya adalah pemodelan (Modeling). Pada langkah ini, algoritma *machine learning* diterapkan pada data latih untuk membangun model prediksi. Contoh algoritma yang digunakan meliputi regresi linier, *decision tree*, *random forest*, *SVM*, dan lainnya.

### 6. Evaluasi Model

Tahap evaluasi model (Model Evaluation) digunakan untuk mengukur seberapa baik model yang telah dibuat. Evaluasi dilakukan menggunakan data uji dengan metrik seperti akurasi, *precision*, *recall*, F1-score, atau MSE tergantung pada jenis masalahnya (klasifikasi/regresi).

### 7. Umpan Balik ke Seleksi Fitur

Dari hasil evaluasi, jika performa model belum optimal, terdapat panah umpan balik (feedback loop) menuju tahap seleksi fitur. Ini menandakan bahwa proses bersifat iteratif: fitur dapat disesuaikan atau diperbaiki untuk meningkatkan kinerja model.

Setelah model dievaluasi dan hasilnya memuaskan, proses berakhir di tahap ini, dan model siap untuk diimplementasikan atau digunakan dalam sistem prediksi nyata.

---

## Langkah Awal

Sebelum memulai praktikum ini, pastikan Anda telah menyiapkan struktur folder di Google Drive / Jupyter seperti pada praktikum sebelumnya. Untuk praktikum ke-3, buat sub-folder baru dengan nama praktikum03/ di dalam direktori utama praktikum\_ml/.

Struktur folder yang digunakan adalah sebagai berikut:

```
praktikum_ml/  
├─ praktikum01/  
├─ praktikum02/  
├─ praktikum03/  
│   └─ data/  
│   └─ notebooks/  
│   └─ model/  
│   └─ reports/
```

### Catatan:

1. Jika telah memiliki folder praktikum\_ml, cukup tambahkan folder praktikum03.
2. Semua file notebook pada praktikum ini disimpan di dalam folder notebooks/ dengan nama praktikum03.ipynb.
3. Dataset ditempatkan di folder data/.
4. Model hasil training disimpan di folder model/ (format .pkl atau .joblib).
5. Visualisasi atau laporan hasil analisis disimpan di folder reports/.Jalankan file installer yang sudah diunduh.

## 3.1 Model Regresi Sederhana

### 1. Membaca data file CSV

```
[1] ✓ 35s # menghubungkan colab dengan google drive  
from google.colab import drive  
drive.mount('/content/gdrive')  
Mounted at /content/gdrive  
[2] ✓ 0s # mengambil data set lewat gdrive  
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"
```

▼ Membaca File CSV

```
[6] ✓ 0s # membaca file csv menggunakan pandas  
import pandas as pd  
df = pd.read_csv(path + '/data/SOCR-HeightWeight.csv')  
df.head()
```

	Index	Height(Inches)	Weight(Pounds)
0	1	65.78331	112.9925
1	2	71.51521	136.4873
2	3	69.39874	153.0269
3	4	68.21660	142.3354
4	5	67.78781	144.2971

Sel pertama berfungsi untuk menghubungkan lingkungan Google Colab dengan akun Google Drive-mu. Setelah kode ini dijalankan, akan muncul tautan otorisasi. Kamu harus mengklik tautan tersebut, memilih akun Google, dan memberikan izin agar Colab bisa mengakses file-file yang tersimpan di Google Drive-mu. Proses ini hanya perlu dilakukan satu kali per sesi.

---

Selanjutnya, menggunakan library Pandas untuk membaca file data. Variabel path menyimpan lokasi folder di Google Drive tempat file socr berada. Fungsi `pd.read_csv()` kemudian membaca file tersebut dan menyimpannya ke dalam sebuah DataFrame.

## 2. Melihat informasi umum pada data.

```
# Mencari informasi umum pada data
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 3 columns):
#   Column                Non-Null Count  Dtype
---  ---
0   Index                  25000 non-null int64
1   Height(Inches)         25000 non-null float64
2   Weight(Pounds)         25000 non-null float64
dtypes: float64(2), int64(1)
memory usage: 586.1 KB
```

Metode `.info()` memberikan ringkasan singkat tentang DataFrame. Ini sangat penting untuk langkah awal analisis data karena menampilkan informasi berikut:

- Jumlah baris 25000 entri dan 3 kolom
- Nama-nama kolom.
- Jumlah data non-null (data yang tidak kosong) per kolom.
- Tipe data (Dtype) dari setiap kolom, seperti int64 (integer) dan float64 (float).

## 3. Menghitung statistik deskriptif pada kolom numeric dengan describe.

```
[2]: df.describe()

[2]:
```

	Index	Height(Inches)	Weight(Pounds)
count	25000.000000	25000.000000	25000.000000
mean	12500.500000	67.993114	127.079421
std	7217.022701	1.901679	11.660898
min	1.000000	60.278360	78.014760
25%	6250.750000	66.704397	119.308675
50%	12500.500000	67.995700	127.157750
75%	18750.250000	69.272958	134.892850
max	25000.000000	75.152800	170.924000

Metode `.describe()` secara otomatis menghitung statistik deskriptif dasar untuk semua kolom numerik. Ini memberikan gambaran cepat tentang distribusi data. Hasilnya adalah sebuah tabel yang berisi metrik-metrik berikut untuk setiap kolom numerik:

- count: Jumlah data non-null.
- mean: Rata-rata.
- std: Standar deviasi.

- 
- min: Nilai minimum.
  - 25%, 50%, 75%: Kuartil pertama, median, dan kuartil ketiga.
  - max: Nilai maksimum.

## 4. Data Pre-processing

```
[3]: df1 = (
      df[["Height(Inches)", "Weight(Pounds)"]]
      .rename(columns={"Height(Inches)": "tinggi_cm",
                       "Weight(Pounds)": "berat_kg"})
      .assign(
          tinggi_cm=lambda d: d["tinggi_cm"] * 2.54,      # in → cm
          berat_kg=lambda d: d["berat_kg"] * 0.45359237   # lb → kg
      )
      .round({"tinggi_cm": 2, "berat_kg": 2})
      ).copy()

      df1.head()
```

```
[3]:
```

	tinggi_cm	berat_kg
0	167.09	51.25
1	181.65	61.91
2	176.27	69.41
3	173.27	64.56
4	172.18	65.45

Dalam hal ini kita membuat DataFrame baru bernama df1 yang berisi data tinggi dan berat badan yang telah dikonversi dari satuan inci dan pon menjadi sentimeter dan kilogram. Kolom Height (Inches) dan Weight (Pounds) diambil, kemudian diubah namanya menjadi tinggi\_cm dan berat\_kg, dikalikan dengan faktor konversi (2.54 untuk inci → cm dan 0.45359237 untuk pon → kg), lalu hasilnya dibulatkan dua angka di belakang koma. Hasil akhirnya adalah tabel tinggi dan berat dalam satuan metrik yang lebih mudah dibaca.

## 5. Membagi dataset untuk Training dan Test

```
[4]: from sklearn.model_selection import train_test_split

X = df1[["tinggi_cm"]]
y = df1[["berat_kg"]]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=7
)
```

Pertama kita menentukan terlebih dahulu variabel dependen/target dan independen:

- Kolom **tinggi\_cm** -> Variable independen
- Kolom **berat\_cm** -> Variable dependen

Lalu, masing-masing variabel dependen dan independen dibagi menjadi 2 yaitu train dan test. Yang mana pembagian dengan nilai:

- Data train -> 80%
- Data test -> 20%

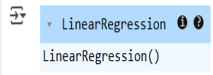
---

Dan parameter Parameter `random_state=7` berfungsi untuk **mengatur angka acak (seed)** yang digunakan saat proses pembagian data agar hasilnya **selalu konsisten** setiap kali kode dijalankan.

## 6. Training Model

```
[10]: from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```



Selanjutnya melatih model dengan menggunakan regresi linear. Untuk menerapkan regresi linear kita perlu menggunakan **LinearRegression** milik scikit-learn. Lalu menginput data train milik variabel dependen dan independen.

## 7. Evaluasi Model

```
[6]: import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)

print("Koefisien (kg per cm):", model.coef_[0])
print("Intersep (kg):", model.intercept_)
print("R2 (test):", r2)
print("MAE (kg):", mean_absolute_error(y_test, y_pred))
mse = mean_squared_error(y_test, y_pred) # default squared=True
rmse = np.sqrt(mse)
print("RMSE (kg):", rmse)
```

```
Koefisien (kg per cm): 0.5518232618278273
Intersep (kg): -37.65788878383336
R2 (test): 0.24989263013277574
MAE (kg): 3.6704107898943548
RMSE (kg): 4.609006140308042
```

- Koefisien (0.55 kg/cm): Menunjukkan bahwa setiap kenaikan 1 cm pada variabel independen (misalnya tinggi badan) akan meningkatkan prediksi berat badan sekitar 0.55 kg. Artinya terdapat hubungan positif, namun tidak terlalu kuat antara variabel tersebut.
- Intersep (-37.65 kg): Titik potong garis regresi pada sumbu Y. Nilai ini berarti jika variabel tinggi bernilai 0 cm (secara matematis), maka model memprediksi berat -37.65 kg. Nilai ini tidak bermakna secara fisik, tetapi penting dalam menentukan posisi garis regresi.
- $R^2$  (0.25): Nilai ini menunjukkan bahwa hanya sekitar 25% variasi data target (berat badan) dapat dijelaskan oleh model regresi. Dengan kata lain, 75% variasi sisanya dipengaruhi oleh faktor lain yang tidak dimasukkan ke dalam model. Nilai  $R^2$  yang rendah ini menandakan bahwa model masih belum menjelaskan data dengan baik.
- MAE (3.67 kg): Rata-rata kesalahan absolut model adalah sekitar 3–4 kg. Artinya, secara umum, prediksi berat badan meleset sekitar 3 hingga 4-kilogram dari nilai sebenarnya.



- RMSE (4.61 kg): Nilai RMSE yang lebih tinggi daripada MAE menunjukkan adanya outlier atau prediksi yang meleset cukup jauh dari data aktual. Semakin kecil nilai RMSE, semakin baik performa model.

## 8. Persamaan Regresi

```
[7]: slope = model.coef_[0]
intercept = model.intercept_
print(f"Persamaan: y = {slope:.3f} * x + {intercept:.3f}")

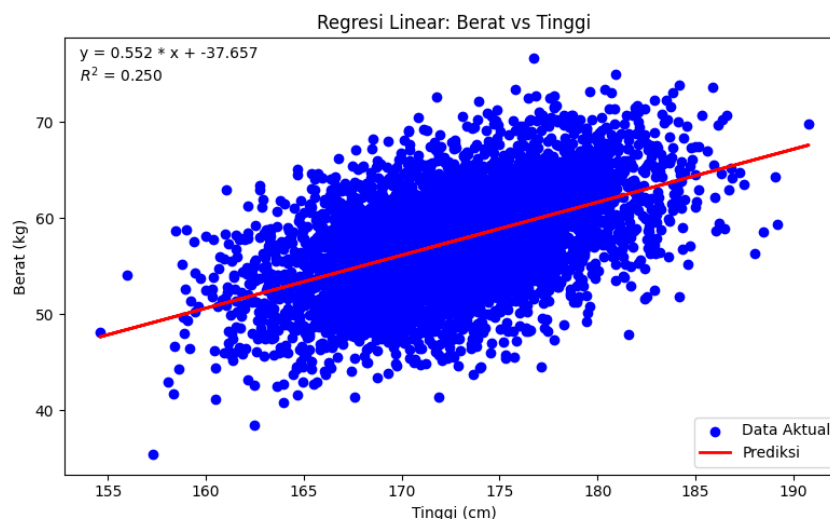
Persamaan: y = 0.552 * x + -37.657
```

Selanjutnya kita ingin melihat dan memodelkan hubungan antara 2 variabel yaitu dengan cara mengambil nilai koefisien (`model.coef_[0]`) dan intercept (`model.intercept_`) dari model regresi linear yang telah dilatih, lalu menampilkannya dalam bentuk persamaan garis regresi. Hasil persamaan  $y = 0.552 * x - 37.657$  menunjukkan bahwa setiap kenaikan tinggi badan 1 cm akan meningkatkan berat badan sekitar 0.552 kg, sedangkan nilai -37.657 merupakan titik potong garis dengan sumbu y yang bermakna secara matematis. Secara keseluruhan, hasil ini menunjukkan adanya hubungan positif antara tinggi dan berat badan — semakin tinggi seseorang, maka berat badannya cenderung lebih besar.

## 9. Plot Regresi

```
[8]: import matplotlib.pyplot as plt

# Plot data scatter
plt.figure(figsize=(8, 5))
plt.scatter(X_test, y_test, color="blue", label="Data Aktual")
# Garis regresi
plt.plot(X_test, y_pred, color="red", linewidth=2, label="Prediksi")
plt.xlabel("Tinggi (cm)")
plt.ylabel("Berat (kg)")
plt.title("Regresi Linear: Berat vs Tinggi")
plt.text(
    0.02, 0.98,
    f"y = {slope:.3f} * x + {intercept:.3f}\nR^2 = {r2:.3f}",
    transform=plt.gca().transAxes, va="top"
)
plt.legend()
plt.tight_layout()
plt.show()
```



Dalam visualisasi tersebut didapatkan hubungan antara **tinggi badan (sumbu x)** dan **berat badan (sumbu y)** dalam bentuk grafik regresi linear. Titik-titik **berwarna biru** merepresentasikan **data aktual** hasil pengamatan, sedangkan **garis merah** menunjukkan **garis prediksi** dari model regresi linear yang menggambarkan tren umum hubungan antara kedua variabel. Terlihat bahwa semakin tinggi seseorang, berat badannya cenderung meningkat, menandakan adanya **hubungan positif** antara tinggi dan berat badan. Nilai  $R^2 = 0.25$  pada grafik menunjukkan bahwa sekitar **25% variasi berat badan dapat dijelaskan oleh tinggi badan**, sedangkan sisanya dipengaruhi faktor lain di luar model.

```
[9]: y_pred_test = model.predict(X_test)

# Buat tabel hasil (tinggi, aktual, prediksi, dan error)
hasil = pd.DataFrame({
    "Tinggi (cm)": X_test["tinggi_cm"].to_numpy(),
    "Berat aktual (kg)": y_test.to_numpy(),
    "Berat Prediksi (kg)": y_pred_test,
})

# 1) Selisih error (positif = overpredict)
hasil["Selisih error (kg)"] = hasil["Berat Prediksi (kg)"] - hasil["Berat aktual (kg)"]

# 2) Akurasi per-baris (100 * (1 - |error|/aktual)), dibatasi 0-100
denom = hasil["Berat aktual (kg)"].replace(0, np.nan) # antisipasi pembagi nol
hasil["Akurasi (%)"] = (1 - (hasil["Selisih error (kg)"].abs() / denom)).clip(lower=0, upper=1) * 100

hasil
```

	Tinggi (cm)	Berat aktual (kg)	Berat Prediksi (kg)	Selisih error (kg)	Akurasi (%)
0	174.73	50.16	58.762990	8.602990	82.848904
1	171.31	50.33	56.875754	6.545754	86.994329
2	169.29	58.22	55.761071	-2.458929	95.776488
3	163.30	58.92	52.455650	-6.464350	89.028598
4	170.52	63.06	56.439814	-6.620186	89.501766
...	...	...	...	...	...
4995	178.75	56.59	60.981319	4.391319	92.240114
4996	163.05	47.45	52.317694	4.867694	89.741425
4997	166.51	52.46	54.227003	1.767003	96.631715

- 1) `y_pred_test = model.predict(X_test)` → menghitung nilai berat **prediksi** berdasarkan model regresi dan data tinggi pada set uji (`X_test`).
- 2) Tabel hasil menampilkan 4 kolom utama:
  - **Tinggi (cm)** → data input.
  - **Berat aktual (kg)** → nilai sebenarnya dari data uji.
  - **Berat Prediksi (kg)** → hasil prediksi dari model.
  - **Selisih error (kg)** → selisih antara hasil prediksi dan nilai aktual (positif berarti prediksi terlalu tinggi, negatif berarti terlalu rendah).
- 3) Kolom **Akurasi (%)** dihitung berdasarkan seberapa kecil selisih error terhadap nilai aktual (semakin kecil error, semakin tinggi akurasi). Nilai akurasi dibatasi antara 0–100%.

---

## 3.2 Multiple Linear Regresi

### 1. Membaca data file CSV

```
[1]: import pandas as pd

# Read the CSV file with a comma delimiter
df = pd.read_csv('../data/stunting_wasting_dataset.csv', sep=',')

# cetak header data (5 baris data) dari file
df.head()
```

```
[1]:
```

	Jenis Kelamin	Umur (bulan)	Tinggi Badan (cm)	Berat Badan (kg)	Stunting	Wasting
0	Laki-laki	19	91.6	13.3	Tall	Risk of Overweight
1	Laki-laki	20	77.7	8.5	Stunted	Underweight
2	Laki-laki	10	79.0	10.3	Normal	Risk of Overweight
3	Perempuan	2	50.3	8.3	Severely Stunted	Risk of Overweight
4	Perempuan	5	56.4	10.9	Severely Stunted	Risk of Overweight

Selanjutnya, menggunakan library Pandas untuk membaca file data dengan menggunakan Fungsi `pd.read_csv()` dan `sep ','` untuk pemisah antar kolom.

### 2. Menghitung statistik deskriptif pada kolom numeric dengan `describe`.

```
[2]: df.describe()
```

```
[2]:
```

	Umur (bulan)	Tinggi Badan (cm)	Berat Badan (kg)
count	100000.000000	100000.000000	100000.000000
mean	11.992580	73.132657	9.259256
std	7.199671	11.360846	3.300780
min	0.000000	42.600000	1.000000
25%	6.000000	65.500000	6.900000
50%	12.000000	74.200000	9.200000
75%	18.000000	81.400000	11.700000
max	24.000000	97.600000	17.200000

Metode `.describe()` secara otomatis menghitung statistik deskriptif dasar untuk semua kolom numerik. Ini memberikan gambaran cepat tentang distribusi data. Hasilnya adalah sebuah tabel yang berisi metrik-metrik berikut untuk setiap kolom numerik:

3. count: Jumlah data non-null.
4. mean: Rata-rata.
5. std: Standar deviasi.
6. min: Nilai minimum.
7. 25%, 50%, 75%: Kuartil pertama, median, dan kuartil ketiga.
8. max: Nilai maksimum.

---

### 3. Data Pre-processing

Selanjutnya kita ingin mengcopy variabel df dan hanya memakai pada variabel/kolom Berat Badan (kg), Jenis Kelamin, Umur (bulan), Tinggi Badan (cm) dengan menamai dengan df1. Namun dilanjutkan dengan mengubah nama kolom dengan function rename.

```
[3]: df1 = (df[["Berat Badan (kg)", "Jenis Kelamin", "Umur (bulan)", "Tinggi Badan (cm)"]]  
        .rename(columns={"Jenis Kelamin": "jk", "Umur (bulan)": "umur_bln",  
                        "Tinggi Badan (cm)": "tinggi_cm", "Berat Badan (kg)": "berat_kg"}).copy())  
      ## Laki-Laki: 1, Perempuan : 0  
      df1["jk"] = df1["jk"].map({"Laki-laki": 1, "Perempuan": 0})  
      df1.head()
```

```
[3]:
```

	berat_kg	jk	umur_bln	tinggi_cm
0	13.3	1	19	91.6
1	8.5	1	20	77.7
2	10.3	1	10	79.0
3	8.3	0	2	50.3
4	10.9	0	5	56.4

### 4. Analisis Korelasi

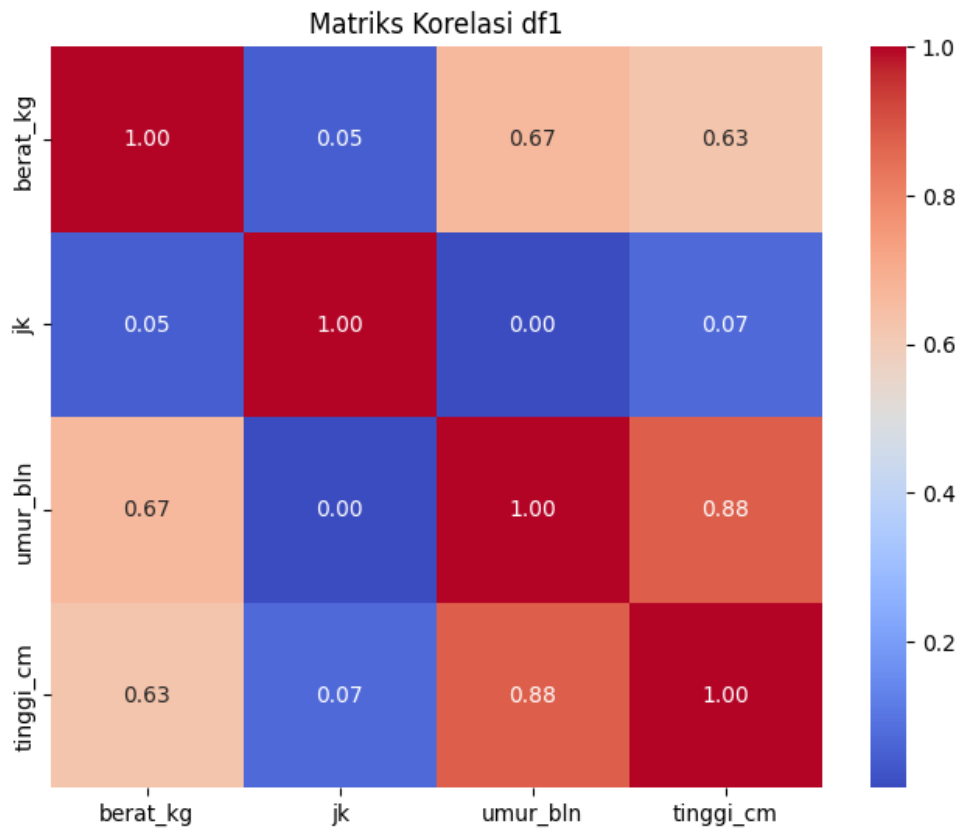
Tahapan ini melakukan evaluasi terhadap beberapa variabel indepen (x) untuk memprediksi variabel dependen (y). Kemudian dipilih variabel yang dominan untuk dimasukkan dalam model prediksi dengan menggunakan fungsi koefisien korelasi .corr(), untuk selanjutnya divisualisasikan dalam bentuk grafik Heatmap.

```
[4]: # Hitung matriks korelasi  
      corr_matrix = df1.corr()  
  
      print(corr_matrix)
```

	berat_kg	jk	umur_bln	tinggi_cm
berat_kg	1.000000	0.045797	0.665389	0.626005
jk	0.045797	1.000000	0.004046	0.073505
umur_bln	0.665389	0.004046	1.000000	0.875869
tinggi_cm	0.626005	0.073505	0.875869	1.000000

### Buat Heatmap Grafik

```
[5]: import seaborn as sns  
      import matplotlib.pyplot as plt  
  
      plt.figure(figsize=(8,6))  
      sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")  
      plt.title("Matriks Korelasi df1")  
      plt.show()
```



Hasil analisis koefisien korelasi menunjukkan bahwa variabel yang paling berpengaruh dalam prediksi berat badan balita adalah sebagai berikut:

- Umur: 0.67 → berpengaruh dominan
- Tinggi: 0.63 → berpengaruh dominan
- Jenis Kelamin: 0.05 → tidak berpengaruh signifikan

Berdasarkan hasil korelasi tersebut, variabel yang digunakan untuk membangun model regresi adalah:

- Variabel independen (X):
  - $X_1$  = Umur
  - $X_2$  = Tinggi
- Variabel dependen (Y):
  - Y = Berat

---

## 5. Membagi dataset untuk Training dan Test

Pada tahapan ini membagi dataset menjadi 80% data training dan 20% data testing.

```
[19]: from sklearn.model_selection import train_test_split

# Misalkan target (Y) adalah berat badan, # Variabel dependen
y = df1["berat_kg"]

# Fitur (X) adalah umur dan tinggi, # Variabel independen
X = df1[["umur_bln", "tinggi_cm"]]

# Bagi data 80% train, 20% test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42 # random_state supaya hasil konsisten
)

## Cetak Pembagian Data
print("Jumlah data train :", len(X_train))
print("Jumlah data test  :", len(X_test))

## cek apakah sudah ada constanta pada data training
X_train.head()
```

```
Jumlah data train : 80000
Jumlah data test  : 20000
```

```
[19]:      umur_bln  tinggi_cm
75220         2        51.9
48955        13        74.3
44966        17        86.7
13568        16        76.8
92727        20        78.5
```

Pertama kita menentukan terlebih dahulu variabel dependen/target dan independen:

- Kolom **tinggi\_cm** -> Variable independen
- Kolom **umur\_bln** -> Variable independen
- Kolom **berat\_cm** -> Variable dependen

Lalu, masing-masing variabel dependen dan independen dibagi menjadi 2 yaitu train dan test. Yang mana pembagian dengan nilai:

- Data train -> 80%
- Data test -> 20%

Dan parameter Parameter `random_state=42` berfungsi untuk **mengatur angka acak (seed)** yang digunakan saat proses pembagian data agar hasilnya **selalu konsisten setiap kali kode dijalankan**.

Selanjutnya dilanjutkan dengan menghitung jumlah data train dan test pada variabel X menggunakan fungsi `len()`.

---

## 6. Pemodelan

- Cek apakah data training telah memiliki nilai konstan, jika belum ada tambahkan variabel konstan bernilai 1.0.

```
[20]: ## Tambahkan konstanta
X_train_const = sm.add_constant(X_train)
X_train_const.head()
```

```
[20]:
```

	const	umur_bln	tinggi_cm
75220	1.0	2	51.9
48955	1.0	13	74.3
44966	1.0	17	86.7
13568	1.0	16	76.8
92727	1.0	20	78.5

- Melakukan pemodelan dengan Pustaka program OLS, dan jalankan training data, kemudian cetak parameter constan, x1 dan x2 dan tampilkan persamaan regresi nya.

```
[34]: import statsmodels.api as sm

# Buat model OLS
model = sm.OLS(y_train, X_train_const).fit()
print('-----')
print(model.params)
print('-----')
const = model.params['const']
x1_umur = model.params['umur_bln']
x2_tinggi = model.params['tinggi_cm']
#print persamaan regresi
print(f"y = {const:.3f} + {x1_umur:.3f}*x1 + {x2_tinggi:.3f}*x2")

-----
const          2.545617
umur_bln       0.229719
tinggi_cm      0.054192
dtype: float64
-----
y = 2.546 + 0.230*x1 + 0.054*x2
```

- Cetak informasi model regresi OLS

```
[35]: # Tampilkan ringkasan hasil
print(model.summary())
```

```

                                OLS Regression Results
=====
Dep. Variable:                  berat_kg    R-squared:                  0.450
Model:                        OLS          Adj. R-squared:            0.450
Method:                    Least Squares   F-statistic:              3.272e+04
Date:                Sun, 05 Oct 2025     Prob (F-statistic):       0.00
Time:                  20:20:06           Log-Likelihood:          -1.8505e+05
No. Observations:          80000          AIC:                    3.701e+05
Df Residuals:              79997          BIC:                    3.701e+05
Df Model:                   2
Covariance Type:            nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
const          2.5456      0.091     28.039      0.000      2.368      2.724
umur_bln       0.2297      0.002     92.330      0.000      0.225      0.235
tinggi_cm      0.0542      0.002     34.359      0.000      0.051      0.057
=====
Omnibus:                 16501.255    Durbin-Watson:              2.006
Prob(Omnibus):            0.000    Jarque-Bera (JB):          3202.586
Skew:                    0.015    Prob(JB):                  0.00
Kurtosis:                 2.020    Cond. No.                  789.
=====

```

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

## 7. Evaluasi Model

Ordinary Least Squares (OLS) adalah metode statistik yang banyak digunakan untuk memperkirakan parameter model regresi linier.

- Hasil regresi OLS menunjukkan bahwa model memiliki R-squared sebesar 0.450, artinya 45,0% variasi pada variabel berat badan (berat\_kg) dapat dijelaskan oleh variabel umur (umur\_bln) dan tinggi (tinggi\_cm), sisanya 55% dijelaskan oleh faktor lain yang tidak termasuk dalam model.
- Kedua variabel tersebut signifikan secara statistik (p-value < 0.05), dengan koefisien umur 0.2297 dan tinggi 0.0542, yang berarti setiap kenaikan satu satuan umur atau tinggi akan meningkatkan berat badan secara significant.
- Nilai Durbin-Watson 2.006 dengan nilai mendekati 2 yang menunjukkan tidak ada autokorelasi serius pada residual. Secara keseluruhan, model ini memiliki kemampuan prediksi yang sangat baik dan signifikan.
- Persamaan regresi:

$$\text{Berat (kg)} = 2.5456 + 0.2297 \times \text{Umur (bulan)} + 0.0542 \times \text{Tinggi (cm)}$$



## 8. Pengujian model dengan data testing

```
[45]: # Tambahkan konstanta ke data uji
X_test_const = sm.add_constant(X_test)

# Prediksi berat badan
y_pred_test = model.predict(X_test_const)

# Buat tabel hasil prediksi
hasil = pd.DataFrame({
    "Umur (bulan)": X_test["umur_bln"].to_numpy(),
    "Tinggi (cm)": X_test["tinggi_cm"].to_numpy(),
    "Berat Aktual (kg)": y_test.to_numpy(),
    "Berat Prediksi (kg)": y_pred_test
})

# 1) Selisih error (positif = overpredict)
hasil["Selisih error (kg)"] = hasil["Berat Prediksi (kg)"] - hasil["Berat Aktual (kg)"]

# 2) Akurasi per-baris (100 * (1 - |error|/aktual)), dibatasi 0-100
denom = hasil["Berat Aktual (kg)"].replace(0, np.nan) #antisipasi pembagi nol
hasil["Akurasi (%)"] = (1 - (hasil["Selisih error (kg)"].abs() / denom)).clip(lower=0, upper=1) * 100

hasil
```

```
[45]:
```

	Umur (bulan)	Tinggi (cm)	Berat Aktual (kg)	Berat Prediksi (kg)	Selisih error (kg)	Akurasi (%)
75721	1	54.6	7.0	5.734226	-1.265774	81.917510
80184	8	66.0	12.2	7.960047	-4.239953	65.246290
19864	20	90.0	10.9	12.017284	1.117284	89.749692
76699	13	82.4	9.6	9.997392	0.397392	95.860500

## 3.3 Tugas Praktikum Mandiri

1. Buat model prediksi dari kasus dataset berikut ini:

<https://www.kaggle.com/datasets/lakshmi25npathi/bike-sharing-dataset>

```
[1]: import pandas as pd

# Read the CSV file with a comma delimiter
df = pd.read_csv('../data/day.csv', sep=',')

# cetak header data (5 baris data) dari file
df.head()
```

```
[1]:
```

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

dengan variable dependen (Y) kolom cnt, tentukan variabel independent (x) dari kolom2 yang tersedia !!!