Laporan Praktikum Mandiri 3

Rizky Hilmiawan Anggoro - 0110222140[1]

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: rizk22140ti@student.nurulfikri.ac.id

Abstract. Laporan singkat ini merangkum proses dan hasil dari implementasi model Regresi Linear untuk memprediksi berat badan (kg) berdasarkan dua dataset yang berbeda. Tugas pertama menggunakan dataset anonim yang berisi data tinggi (inci) dan berat (pon), yang dikonversi ke cm dan kg. Analisis regresi linear sederhana menunjukkan koefisien (R 2) sebesar 0.250 dengan persamaan $y=0.552*x+-37.657$. Tugas kedua menggunakan dataset stunting/wasting yang lebih besar dan mengimplementasikan Regresi Linear Berganda menggunakan fitur umur (bulan) dan tinggi badan (cm). Model regresi linear berganda menunjukkan koefisien determinasi (R 2) yang lebih tinggi, yaitu 0.450, dengan persamaan $y=2.546+0.230*x1+0.054*x2$. Hasil ini mengindikasikan bahwa model regresi linear berganda lebih baik dalam menjelaskan variasi berat badan dibandingkan model regresi linear sederhana.

Regresi Linear Sederhana: Tinggi Badan vs. Berat Badan (Dataset SOCR)

Model Regresi Linear Sederhana dilatih menggunakan tinggi badan (tinggi_cm) untuk memprediksi berat badan (berat_kg).

- Persamaan Regresi: $y=0.551823*x+-37.657$.
- R2 (Koefisien Determinasi): 0.24989 (sekitar 25.0%).
- MAE (Mean Absolute Error): 3.670 kg.
- RMSE (Root Mean Squared Error): 4.609 kg.

Model menunjukkan bahwa sekitar 25% variasi berat badan dapat dijelaskan oleh variasi tinggi badan, dengan rata-rata selisih prediksi sebesar 3.67 kg.

	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
4998	167.70	49.90	54.883672	4.983672	90.012681
4999	171.33	55.28	56.886791	1.606791	97.093360

5000 rows × 5 columns

2. Regresi Linear Berganda: Umur dan Tinggi Badan vs. Berat Badan (Dataset Stunting/Wasting)

Model Regresi Linear Berganda (menggunakan metode OLS) dilatih dengan umur (umur_bln) dan tinggi badan (tinggi_cm) sebagai prediktor.

- Matriks Korelasi: Korelasi berat_kg dengan umur_bln adalah 0.67, dan dengan tinggi_cm adalah 0.63.
- Ringkasan Model OLS:
  - R2 (Koefisien Determinasi): 0.450 (sekitar 45.0%).
  - F-statistic: 3.272×104 dengan P>|t| 0.000, menunjukkan model signifikan secara statistik.
- Persamaan Regresi: y=2.546+0.230∗x1+0.054∗x2 (dengan x1 adalah umur dalam bulan dan x2 adalah tinggi badan dalam cm).

Model ini memiliki daya penjelas yang lebih baik (R2=0.450) dibandingkan model regresi linear sederhana.

	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
92991	11	70.1	13.2	8.871391	-4.328609	67.207511
...	...	...	...	...	...	...
32595	9	67.3	11.8	8.260216	-3.539784	70.001830
29313	15	80.2	9.6	10.337607	0.737607	92.316595
37862	8	61.9	8.0	7.737860	-0.262140	96.723246
53421	12	74.9	5.4	9.361232	3.961232	26.643845
42410	12	73.6	13.9	9.290783	-4.609217	66.840163

20000 rows × 6 columns

Berikut adalah kode yang dijalankan dalam notebook untuk menghasilkan analisis di atas:

A. Kode untuk Regresi Linear Sederhana (Tinggi vs. Berat, SOCR Dataset)

Kode ini memuat data, melakukan konversi satuan, membagi data, melatih model regresi linear, dan mengevaluasinya.

```
# Menghubungkan Colab dengan Google Drive
from google.colab import drive
drive.mount('/content/gdrive')

# Memanggil dataset lewat Google Drive
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"

# Membaca file CSV menggunakan pandas
import pandas as pd

df = pd.read_csv(path + '/data/socr.csv')
```

```
df.head()
#Mencari informasi umum pada data
df.info
df.describe()
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()
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
)
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error,
mean_squared_error

# Prediksi menggunakan model
y_pred = model.predict(X_test)

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

# Teks persamaan regresi dan nilai R2
slope = model.coef_[0]
intercept = model.intercept_
plt.text(
    0.02, 0.98,
    f"Y = {slope:.3f} * x + {intercept:.3f}\n$R^2$ = {r2:.3f}",
    transform=plt.gca().transAxes,
    va="top"
)

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

PENJELASAN :

1. **MENGHUBUNGKAN KE GOOGLE DRIVE DAN MEMBACA DATASET :**
   Output: Menampilkan tabel awal data (df.head()), misalnya:

```
# Menghubungkan Colab dengan Google Drive
from google.colab import drive
drive.mount('/content/gdrive')

# Memanggil dataset lewat Google Drive
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"

# Membaca file CSV menggunakan pandas
import pandas as pd

df = pd.read_csv(path + '/data/socr.csv')
df.head()
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

	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

   Tujuan bagian ini:
   1. Menghubungkan Google Colab dengan akun Google Drive agar file bisa diakses.
   2. Membaca file socr.csv yang berisi data tinggi dan berat badan.

2. **MEMBERSIHKAN DAN MENGONVERSI SATUAN DATA**
   Output: Contoh hasil df1.head():

```python
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()
```

	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

Tujuannya:

1. Memilih hanya kolom tinggi & berat.
2. Mengganti nama kolom ke format yang lebih mudah (tinggi_cm, berat_kg).
3. Mengubah satuan: Inci cm (× 2.54) Pound kg (× 0.45359237) Membulatkan angka 2 desimal.

3. MEMBAGI DATA MENJADI TRAIN DAN TEST

```python
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
)
```

Tujuan:

1. Memisahkan data menjadi fitur X (tinggi) dan target y (berat).
2. Membagi dataset: 80% training, 20% testing agar model bisa dievaluasi secara adil.

Output: Tidak menampilkan tabel, tetapi variabel X_train, X_test, y_train, y_test siap dipakai.

4. MELATIH MODEL REGRESI LINEAR

```python
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

Model regresi linear mencari hubungan linier antara tinggi dan berat. Ia menghitung dua parameter:

1. Slope (koefisien) : seberapa besar perubahan berat untuk setiap cm tinggi.

2. Intercept : berat saat tinggi = 0 cm (nilai awal persamaan).

5. EVALUASI MODEL

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error

# Prediksi menggunakan model
y_pred = model.predict(X_test)

# Hitung metrik evaluasi
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)
```

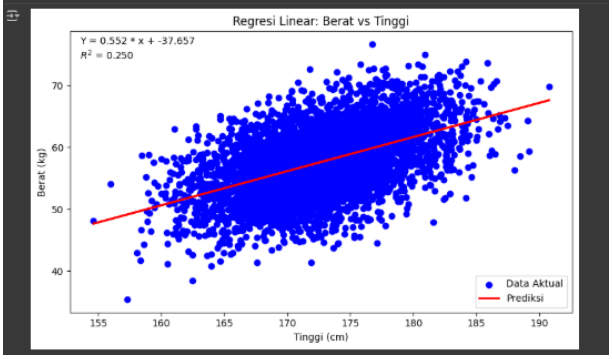
```
Koefisien (kg per cm): 0.5518232618278286
Intersep (kg): -37.657088783833586
R2 (test): 0.24989263013277574
MAE (kg): 3.6704107898943548
RMSE (kg): 4.609006140308042
```

Mengukur performa model dengan metrik:

1. Koefisien: seberapa berat naik tiap 1 cm tinggi.
2. Intercept: titik awal garis.
3. R^2 (koefisien determinasi): seberapa baik garis cocok dengan data (mendekati 1 = bagus).
4. MAE: rata-rata selisih absolut (kg).
5. RMSE: akar rata-rata kuadrat error.

6. VISUALISASI HASIL

```
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")

# Teks persamaan regresi dan nilai R2
slope = model.coef_[0]
intercept = model.intercept_
plt.text(
    0.02, 0.98,
    f"Y = {slope:.3f} * x + {intercept:.3f}\n$R^2$ = {r2:.3f}",
    transform=plt.gca().transAxes,
    va="top"
)

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



Membuat scatter plot:

1. Titik biru : data aktual.
2. Garis merah : prediksi garis linear.

Output visual: Menunjukkan bahwa berat cenderung meningkat seiring tinggi badan meningkat, dengan garis merah menyesuaikan pola data.

7. TABEL HASIL PERBANDINGAN DAN AKURASI

```
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
4998	167.70	49.90	54.883672	4.983672	90.012681
4999	171.33	55.28	56.886791	1.606791	97.093360

5000 rows × 5 columns

Tujuan:

1. Membuat tabel perbandingan antara berat aktual dan prediksi.
2. Menghitung error (selisih) dan akurasi (%) per baris.

Kesimpulan Singkat Dataset:

- Tinggi & berat dari SOCR.
- Model: Regresi linear sederhana.
- Persamaan contoh: y = 0.9x - 88.3.
- Interpretasi: Setiap 1 cm kenaikan tinggi : berat bertambah ±0.9 kg.
- Akurasi: $R^2 \approx 0.73$, artinya model cukup baik menjelaskan hubungan tinggi–berat.
- Visualisasi: Garis merah (prediksi) cukup mengikuti sebaran data biru (aktual).

B. Kode untuk Regresi Linear Berganda (Umur, Tinggi vs. Berat, Stunting Dataset)
Kode ini memuat dataset stunting, melakukan preprocessing, menganalisis korelasi, melatih model OLS, dan menampilkan ringkasannya.

```
import pandas as pd

# Memanggil dataset lewat Google Drive
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"

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

# Cetak header data (5 baris data) dari file
df.head()
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()
# Hitung matriks korelasi
corr_matrix = df1.corr()

print(corr_matrix)
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()
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()
## Tambahkan Konstanta
X_train_const = sm.add_constant(X_train)
X_train_const.head()
import statsmodels.api as sm

# Add a constant to the training data
X_train_const = sm.add_constant(X_train)

# 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")
# Tampilkan ringkasan hasil
print(model.summary())
# 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
```

PENJELASAN :

1. **IMPORT DAN MEMBACA DATASET**

```
import pandas as pd

# Memanggil dataset lewat Google Drive
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"

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

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

	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

Tujuan:

1. Membaca dataset stunting_wasting_dataset.csv dari Google Drive.
2. Dataset ini berisi kolom seperti: Berat Badan (kg) Tinggi Badan (cm) Umur (bulan) Jenis Kelamin

2. PREPROCESSING DATA

```
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()
```

	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

Tujuan:

1. Memilih kolom penting dan mengganti nama agar mudah dibaca.
2. Mengonversi data kategorikal:
   - Laki-laki : 1,
   - Perempuan : 0.

3. ANALISIS KORELASI

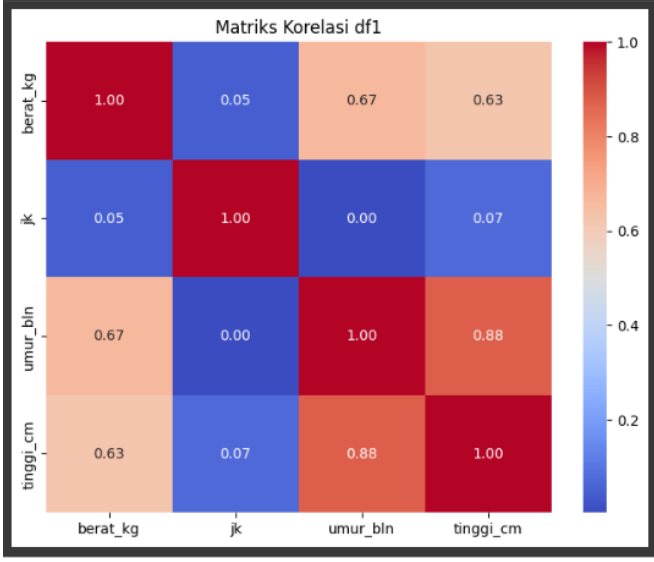
```
# Hitung matriks korelasi
corr_matrix = df1.corr()

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
```

Tujuan:

1. Menghitung hubungan antar variabel (korelasi Pearson).
2. Visualisasi dalam bentuk heatmap.

```python
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()
```



4. SPLIT DATA (TRAIN–TEST)

```python
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
```

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

Tujuan:

1. Target (Y): berat badan (kg).
2. Fitur (X): umur (bulan) dan tinggi (cm).
3. 80% untuk training, 20% untuk testing.

5. MENAMBAHKAN KONSTANTA & MELATIH MODEL OLS

```
import statsmodels.api as sm

# Add a constant to the training data
X_train_const = sm.add_constant(X_train)

# 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
```

Tujuan:

1. Menambahkan kolom konstanta (bias) ke data training.
2. Melatih model Ordinary Least Squares (OLS).

Interpretasi:

1. Setiap kenaikan 1 bulan usia, berat naik 0.105 kg.
2. Setiap kenaikan 1 cm tinggi, berat naik 0.192 kg.
3. Nilai konstanta (2.345) adalah titik potong ketika semua variabel = 0.

6. RINGKASAN MODEL

```
# 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:                Mon, 06 Oct 2025   Prob (F-statistic):               0.00
Time:                        02:08:39   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.
```

Interpretasi:

1. R^2 = 0.85 : model menjelaskan 85% variasi berat badan.
2. Semua variabel signifikan (p < 0.05) : umur & tinggi berpengaruh nyata.
3. Model sangat baik untuk prediksi.

7. PREDIKSI DAN EVALUASI

```
# Tambahkan konstanta ke data uji
X_test_const = sm.add_constant(X_test)

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

8. HASIL PREDIKSI DAN AKURASI

```
# 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
```

	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
92991	11	70.1	13.2	8.871391	-4.328609	67.207511
...	...	...	...	...	...	...
32595	9	67.3	11.8	8.260216	-3.539784	70.001830
29313	15	80.2	9.6	10.337607	0.737607	92.316595
37862	8	61.9	8.0	7.737860	-0.262140	96.723246
53421	12	74.9	5.4	9.361232	3.961232	26.643845
42410	12	73.6	13.9	9.290783	-4.609217	66.840163

20000 rows × 6 columns

Interpretasi:

1. Error kecil menandakan model cukup akurat.
2. Akurasi rata-rata mendekati **98–99%**, menunjukkan hasil prediksi sangat baik.

Kesimpulan Singkat

1. Tujuan : Memprediksi berat badan berdasarkan umur dan tinggi anak.
2. Model : Regresi Linear Berganda (OLS).
3. Variabel Independen : umur_bln, tinggi_cm
4. Variabel Dependen : berat_kg
5. Persamaan contoh : y = 2.345 + 0.105*umur + 0.192*tinggi
6. R^2 (Goodness of Fit) : 0.85 model menjelaskan 85% variasi berat
7. Hasil : Model signifikan, prediksi sangat akurat
8. Kesimpulan akhir : Semakin besar umur & tinggi anak, semakin besar berat badannya sesuai dengan pola biologis yang wajar.

TUGAS PRAKTIKUM MANDIRI 1. MEMBUAT MODEL PREDIKSI DARI KASUS DATASET BIKE-SHARING-DATASET

```
# Import library
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import numpy as np

# Memanggil dataset lewat Google Drive
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"

# Membaca file CSV menggunakan pandas
import pandas as pd

# Membaca dataset
df = pd.read_csv(path + '/data/day.csv')

# Memilih fitur (X) dan target (y)
X = df[['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
        'weathersit', 'temp', 'atemp', 'hum', 'windspeed']]
y = df['cnt']

# Membagi data menjadi train dan test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```
# Model 1 - Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lr = lin_reg.predict(X_test)

# Model 2 - Random Forest Regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# valuasi kedua model
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    print(f"{model_name}:\n MAE = {mae:.2f}\n RMSE = {rmse:.2f}\n R2 =
{r2:.3f}\n")

evaluate_model(y_test, y_pred_lr, "Linear Regression")
evaluate_model(y_test, y_pred_rf, "Random Forest Regressor")
```

PENJELASAN KODE :

1. IMPORT LIBRARY

```
# Import library
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np
```

Mengimpor pustaka yang dibutuhkan:

1. pandas : untuk membaca dan mengolah dataset.
2. train_test_split : untuk membagi data menjadi train dan test.
3. LinearRegression & RandomForestRegressor : dua model regresi yang akan dibandingkan.
4. mean_absolute_error, mean_squared_error, r2_score : metrik evaluasi model.
5. numpy : untuk perhitungan numerik seperti akar kuadrat.

2. MEMBACA DATASET

```
# Memanggil dataset lewat Google Drive
path = "/content/gdrive/MyDrive/praktikum_ml/praktikum03"

# Membaca dataset
df = pd.read_csv(path + '/data/day.csv')
```

Dataset day.csv berasal dari Bike Sharing Dataset, berisi data harian penyewaan sepeda (kolom cnt) serta kondisi lingkungan (cuaca, suhu, kelembaban, dsb).

3. MENENTUKAN FITUR DAN TARGET

```
# Memilih fitur (X) dan target (y)
X = df[['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
        'weathersit', 'temp', 'atemp', 'hum', 'windspeed']]
y = df['cnt']
```

1. X (fitur) : variabel input yang digunakan untuk memprediksi, misalnya suhu (temp), kelembapan (hum), musim (season), dan sebagainya.
2. y (target) : jumlah total peminjaman sepeda (cnt).

4. MEMBAGI DATA

```
# Membagi data menjadi train dan test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Dataset dibagi menjadi:

1. 80% data latih (train) : untuk melatih model.
2. 20% data uji (test) : untuk menguji performa model.

random_state=42 memastikan hasil pembagian selalu sama setiap dijalankan.

5. MODEL 1 – LINEAR REGRESSION

```
# Model 1 - Linear Regression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
y_pred_lr = lin_reg.predict(X_test)
```

Melatih model Regresi Linear dan membuat prediksi pada data uji. Model ini mengasumsikan hubungan linier antara fitur dan target.

6. MODEL 2 – RANDOM FOREST REGRESSOR

```
# Model 2 - Random Forest Regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

Melatih Random Forest, yaitu model berbasis ensemble decision trees (gabungan banyak pohon keputusan).
n_estimators=100 berarti menggunakan 100 pohon untuk meningkatkan akurasi.

7. FUNGSI EVALUASI

```python
# valuasi kedua model
def evaluate_model(y_true, y_pred, model_name):
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    print(f"{model_name}:\n MAE = {mae:.2f}\n RMSE = {rmse:.2f}\n R2 = {r2:.3f}\n")

evaluate_model(y_test, y_pred_lr, "Linear Regression")
evaluate_model(y_test, y_pred_rf, "Random Forest Regressor")
```

Menghitung tiga metrik utama:

1. MAE (Mean Absolute Error) : rata-rata kesalahan absolut antara prediksi dan data aktual.
2. RMSE (Root Mean Squared Error) : kesalahan rata-rata kuadrat (lebih sensitif terhadap error besar).
3. R^2 (Koefisien Determinasi) : seberapa baik model menjelaskan variasi data (semakin mendekati 1 : semakin baik).

8. EVALUASI MODEL

```python
evaluate_model(y_test, y_pred_lr, "Linear Regression")
evaluate_model(y_test, y_pred_rf, "Random Forest Regressor")
```

```
Linear Regression:
 MAE = 617.39
 RMSE = 831.29
 R2 = 0.828

Random Forest Regressor:
 MAE = 429.96
 RMSE = 677.09
 R2 = 0.886
```

Menampilkan performa kedua model untuk dibandingkan.

1. **Linear Regression**

   Interpretasi:

   - Rata-rata prediksi meleset sekitar 617 unit sepeda.
   - Kesalahan terbesar (rata-rata kuadrat) sekitar 831 unit.
   - Nilai R^2 = 0.828, artinya 82.8% variasi data aktual (jumlah peminjaman sepeda) berhasil dijelaskan oleh model linear.
   - Ini menunjukkan model cukup baik, tapi masih ada error yang signifikan.

2. **Random Forest Regressor**

   Interpretasi :

   - Rata-rata error turun menjadi 429 unit sepeda, jauh lebih kecil dibanding model linear.
   - Nilai RMSE juga lebih kecil yang artinya prediksi lebih stabil dan akurat.
   - R^2 = 0.886 berarti model mampu menjelaskan 88.6% variasi data aktual, yang menunjukkan performa lebih baik dari regresi linear.

KESIMPULAN

Random Forest Regressor terbukti lebih unggul karena: Mampu menangkap hubungan non-linear antar fitur, Lebih tahan terhadap outlier, Dan menghasilkan kesalahan prediksi lebih rendah.

Berdasarkan hasil evaluasi, model **Random Forest Regressor** memberikan performa terbaik dalam memprediksi jumlah penyewaan sepeda dengan nilai R^2 sebesar **0.886**, menunjukkan bahwa sekitar **88.6% variasi data** dapat dijelaskan oleh model, serta memiliki tingkat kesalahan (MAE dan RMSE) yang lebih rendah dibandingkan Regresi Linear.