

# Tugas 3: Praktikum & Praktikum Mandiri 3

Pandu Linggar Kumara - 0110221277,  
Link GitHub - [https://github.com/PanduLgg/M\\_Learning.git](https://github.com/PanduLgg/M_Learning.git)

<sup>1</sup> Teknik Informatika, STT Terpadu Nurul Fikri, Depok

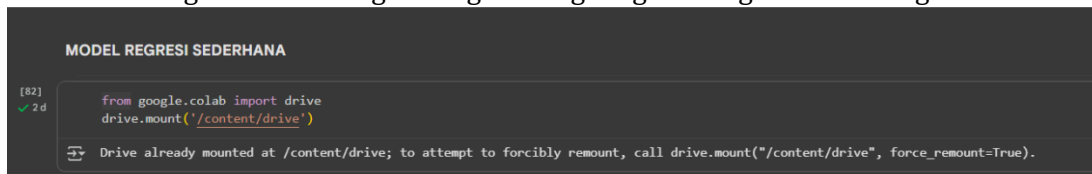
\*E-mail: [pandulinggar1@gmail.com](mailto:pandulinggar1@gmail.com)

**Abstract.** Eksperimen dimulai dari studi kasus sederhana, yaitu prediksi berat badan berdasarkan tinggi dan umur, kemudian dilanjutkan dengan penerapan pada dataset nyata yaitu Bike Sharing Dataset dari Kaggle. Analisis dilakukan menggunakan metode Ordinary Least Squares (OLS) dengan bantuan pustaka statsmodels pada Python untuk memodelkan hubungan antara variabel dependen (cnt, total penyewaan sepeda) dan beberapa variabel independen seperti suhu, kelembapan, musim, serta kecepatan angin. Proses pemodelan mencakup tahap persiapan data, pemilihan fitur, pembuatan model, serta evaluasi menggunakan metrik statistik seperti koefisien determinasi ( $R^2$ ), koefisien regresi, dan nilai signifikansi (p-value). Hasil analisis menunjukkan bahwa regresi linear berganda mampu menjelaskan variasi jumlah penyewaan sepeda berdasarkan faktor lingkungan dan musiman secara efektif.

## 1. Connecting Google Colab & Drive

### 1.1 Menghubungkan lingkungan Google Colab dengan akun Google Drive

Sel ini berfungsi untuk menghubungkan lingkungan Google Colab dengan akun Google Drive



```
MODEL REGRESI SEDERHANA

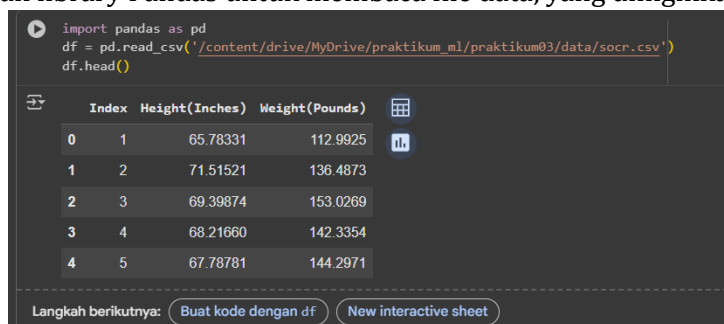
[82]
✓ 2d
from google.colab import drive
drive.mount('/content/drive')

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

Gambar 1.1. Proses ini hanya perlu dilakukan satu kali per sesi.

### 1.2 Memanggil Data set dari Gdrive dan Membaca file .CSV menggunakan Pandas

Sel ini menggunakan library Pandas untuk membaca file data, yang diinginkan



```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum03/data/socr.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

Langkah berikutnya: [Buat kode dengan df](#) [New interactive sheet](#)

Gambar 1.2. Proses ini hanya perlu dilakukan satu kali per sesi.

### 1.3 Mencari informasi data yang ada pada file

Sel ini menampilkan informasi yang ada di dalam file dari mulai tipe data nama kolom, dsb.

```
# @title
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

df.describe()

           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
```

**Gambar 1.3.** Mencari info data pada file

### 1.4 Data Pre-processing

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

```
# Menghitung mean semua kolom numerik
df['Height'].mean()

np.float64(169.944)

# Menghitung Median semua kolom numerik
df['Height'].median()

170.5

# Mencari Modus bisa lebih dari satu
df['Height'].mode()

Height
0      188
dtype: int64

# Menghitung Variansi & Standard Deviasi
df.var(numeric_only=True)

Height      268.149162
Weight    1048.633267
Index       1.836168
dtype: float64

# Menghitung Standar Deviasi
df.std(numeric_only=True)

Height      16.375261
Weight      32.382607
Index        1.355053
dtype: float64
```

**Gambar 1.4.** Tabel tinggi dan berat dalam satuan metrik yang lebih mudah dibaca

### 1.5 Membagi dataset untuk Training dan Test

Sel ini membagi variabel dependen dan independen dibagi menjadi 2 yaitu train dan test

```
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
)
```

**Gambar 1.5.** Variabel dependen dan independen dibagi menjadi 2

### 1.6 Melatih model dengan menggunakan regresi linear

Untuk menerapkan regresi linear kita perlu menggunakan *LinearRegression* milik *scikit-learn*

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

▼ LinearRegression ⓘ ?  
LinearRegression()

**Gambar 1.6.** LinearRegression

### 1.7 Evaluasi Model dan Menghubungkan 2 Variabel

Menilai seberapa baik model *Multiple Linear Regression* bekerja dan membentuk persamaan regresi

```
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)
rmse = np.sqrt(mse)
```

➦ Koefisien (kg per cm): [0.5518218]  
Intersep (kg): [-37.65688233]  
R2 (test): 0.24989238901493693  
MAE (kg): 3.6704108331736673

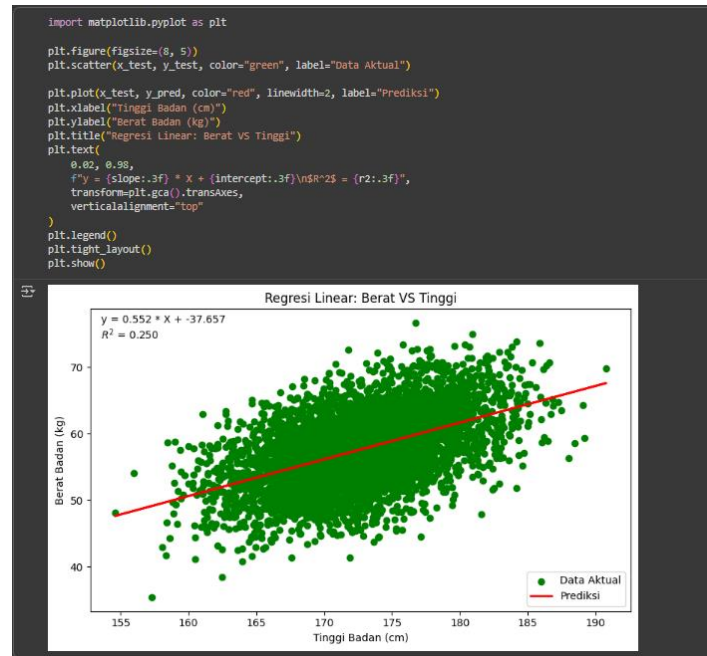
```
slope = model.coef_[0][0]
intercept = model.intercept_[0]
print(f"Persamaan: y = {slope:.3f} * X + {intercept:.3f}")
```

➦ Persamaan: y =0.552 \* X + -37.657

**Gambar 1.7.** Evaluasi Model menunjukkan Koef, Intercept, R2, MAE

## 1.8 Visualisasi Regresi

### Menghubungkan Berat dan Tinggi badan



**Gambar 1.8.** grafik menunjukkan bahwa sekitar 25% variasi berat badan dapat dijelaskan oleh tinggi badan

## 1.9 Model Prediksi

### Menghitung prediksi dan akurasi data hasil prediksi per-baris

```
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().flatten(),
    "Berat Prediksi (kg)": y_pred_test.flatten(),
})

#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 pembagian 0
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	58.762940	8.602940	82.849003
1	171.31	50.33	56.875710	6.545710	86.994417
2	169.29	58.22	55.761030	-2.458970	95.776417
3	163.30	58.92	52.455617	-6.464383	89.028542
4	170.52	63.06	56.439771	-6.620229	89.501698
...	...	...	...	...	...
4995	178.75	56.59	60.981264	4.391264	92.240212
4996	163.05	47.45	52.317662	4.867662	89.741493
4997	166.51	52.46	54.226965	1.766965	96.631786
4998	167.70	49.90	54.883633	4.983633	90.012759
4999	171.33	55.28	56.886746	1.606746	97.093440

5000 rows x 5 columns

Langkah berikutnya: [Buat kode dengan hasil](#) [New interactive sheet](#)

**Gambar 1.8.** Tabel hasil menampilkan 4 kolom utama

## 2. MULTIPLE LINEAR REGRESI

### 2.1 Membaca Data

Membaca data `stunting_wasting_dataset.csv` menggunakan library `pandas` dan menggunakan `sep` untuk memisahkan koma

**MULTIPLE LINEAR REGRESI**

```
import pandas as pd
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum03/data/stunting_wasting_dataset.csv', sep=',')
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

Langkah berikutnya: [Buat kode dengan df](#) [New interactive sheet](#)

**Gambar 2.1.** Menampilkan Kolom data

### 2.2 Menghitung statistik deskriptif pada kolom numeric dengan `describe`

Metode `.describe()` secara otomatis menghitung statistik deskriptif dasar untuk semua kolom numerik. Ini memberikan gambaran cepat tentang distribusi data

```
df.describe()
```

	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

**Gambar 2.2.** Tabel yang berisi metrik-metrik

### 2.3 Data Pre-processing

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*.

```
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()  
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

Gambar 2.3. Kolom hasil filter data yang diinginkan

### 2.4 Analisis Korelasi

Tahapan ini melakukan evaluasi terhadap beberapa variabel indepen (*x*) untuk memprediksi variable dependen (*y*).

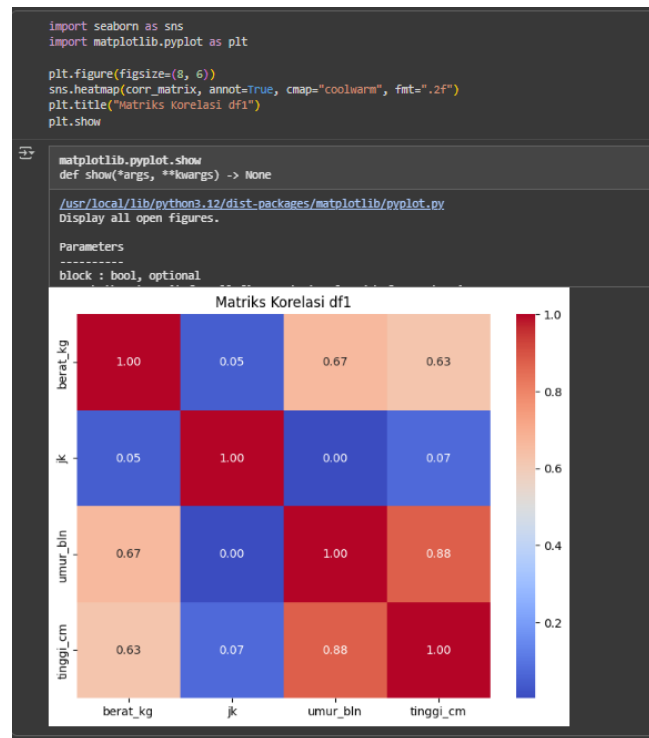
```
corr_matrix = df1.corr()  
print (corr_matrix)
```

	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

Gambar 2.4. Evaluasi variable

## 2.5 Heatmap

Memilih variabel yang dominan untuk dimasukkan dalam model prediksi dengan menggunakan fungsi koefisien korelasi `.corr()`, untuk selanjutnya divisualisasikan dalam bentuk grafik Heatmap.



**Gambar 2.5.** Heatmap Variabel yang paling berpengaruh

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

1. Umur: 0.67 → berpengaruh dominan
2. Tinggi: 0.63 → berpengaruh dominan
3. Jenis Kelamin: 0.05 → tidak berpengaruh signifikan

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

Variabel independen (X):

o  $X_1$  = Umur

o  $X_2$  = Tinggi

Variabel dependen (Y):

o Y = Berat

## 2.6 Membagi dataset untuk Training dan Test

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

```
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 konstanta 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

Langkah berikutnya: [Buat kode dengan X\\_train](#) [New interactive sheet](#)

**Gambar 2.6.** Membagi dataset menjadi 80% data training dan 20% data testing

## 2.7 Pemodelan

Melakukan pemodelan dengan Pustaka program OLS, dan jalankan training data, kemudian cetak parameter constan, x1 dan x2 dan tampilkan persamaan regresi nya. Cek apakah data training telah memiliki nilai konstan, jika belum ada tambahkan variabel konstan bernilai 1.0

```
import statsmodels.api as sm

X_train_const = sm.add_constant(X_train)
X_train_const.head()
```

	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

**Gambar 2.7.1** Menambahkan nilai konstan



```
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
```

**Gambar 2.7.2** Persamaan Regrsi X1 dan X2

## 2.8 Model Regresi OLS

Cetak informasi model regresi OLS

```
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, 12 Oct 2025	Prob (F-statistic):	0.00
Time:	07:10:14	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	0.001	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.

**Gambar 2.8.** Model OLS

## 2.9 Pengujian model dengan data testing

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

```
# 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 x (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 x 6 columns

Gambar 2.9. Hasil Uji Model dan Data

### 3. Praktikum Mandiri

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()
```

	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 !!!

#### 3.1 Baca Dataset

Sel ini membaca dataset yang diinginkan yaitu day.csv

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np

# Baca dataset
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/praktikum03/data/day.csv')

df.head()
```

Gambar 3.1. Proses ini hanya perlu dilakukan satu kali per sesi.

#### 3.2 Variabel Independen (X) dan Dependen (Y)

Menentukan Variabel X dan Y

```
# Variabel dependen (Y)
y = df['cnt']

# Variabel independen (X)
X = df[['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
        'weathersit', 'temp', 'atemp', 'hum', 'windspeed']]
```

Gambar 3.2. Hasil Variabel

### 3.3 Membuat Data Training dan Testing serta melatih Model Linear Regression

Membagi data training dan melatih model Multiple Line Regression

```
x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Inisialisasi model
model = LinearRegression()

# Latih model
model.fit(x_train, y_train)
```

LinearRegression

LinearRegression()

Gambar 3.3. Linear Reression

### 3.4 Evaluasi Model

Evaluasi model mulai dari hasil prediksi dan skor R2 RMSE

```
# Prediksi
y_pred = model.predict(x_test)

# Hitung skor R2 dan RMSE
r2 = r2_score(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

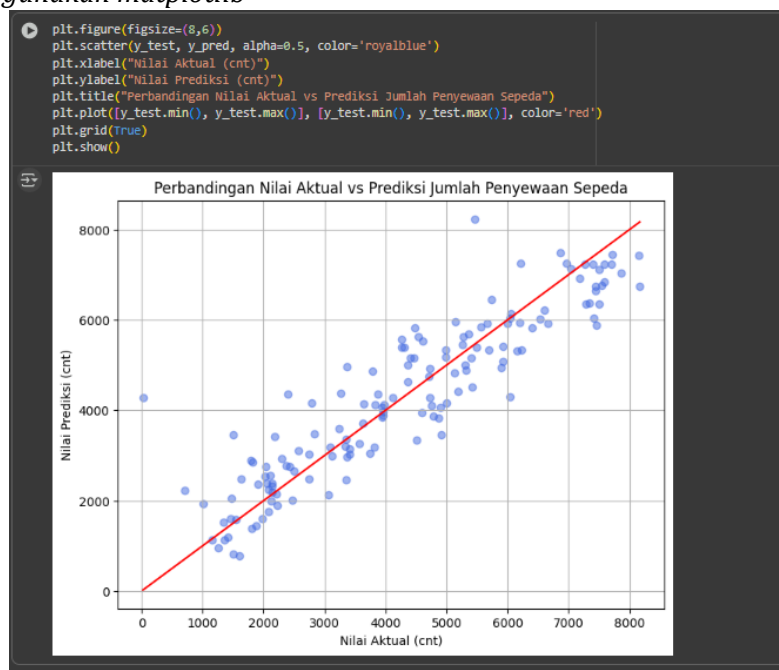
print("R2 Score:", r2)
print("RMSE:", rmse)
```

R<sup>2</sup> Score: 0.8276670090367212  
RMSE: 831.2851545662686

Gambar 3.4. Hasil Model Evaluasi

### 3.5 Visualisasi Hasil Prediksi dan Aktual

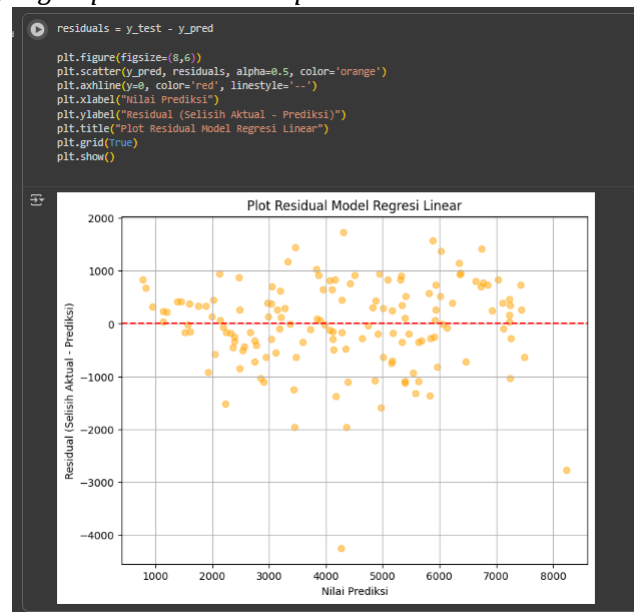
Visualisasi menggunakan matplotlib



Gambar 3.5. Hasil Visualisasi

### 3.6 Visualisasi Residual (Kesalahan Prediksi)

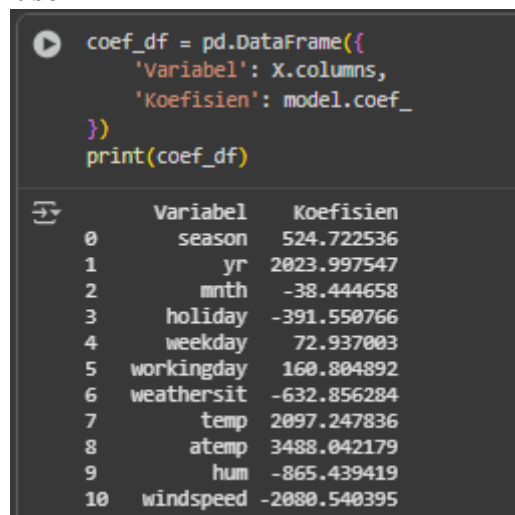
Visualisasi kesalahan yang dapat muncul saat prediksi



Gambar 3.6. Hasil Visualisasi Residual

### 3.7 Koefisien Setiap Variabel

Melihat Koefisien Setiap Variabel



Gambar 3.7. Nilai Koefisien variable X

Hasil :

1. Nilai  $R^2$  menunjukkan seberapa baik model menjelaskan variasi data (semakin mendekati 1 semakin baik).
2. Grafik pertama menunjukkan seberapa dekat prediksi dengan data aktual.
3. Grafik kedua menunjukkan distribusi error (residual), idealnya tersebar acak di sekitar 0.

## Referensi:

- Munir, S., Seminar, K. B., Sudradjat, Sukoco, H., & Buono, A. (2022). The Use of Random Forest Regression for Estimating Leaf Nitrogen Content of Oil Palm Based on Sentinel 1-A Imagery. *Information*, 14(1), 10. <https://doi.org/10.3390/info14010010>
- Seminar, K. B., Imantho, H., Sudradjat, Yahya, S., Munir, S., Kaliana, I., Mei Haryadi, F., Noor Baroroh, A., Supriyanto, Handoyo, G. C., Kurnia Wijayanto, A., Ijang Wahyudin, C., Liyantono, Budiman, R., Bakir Pasaman, A., Rusiawan, D., & Sulastri. (2024). PreciPalm: An Intelligent System for Calculating Macronutrient Status and Fertilizer Recommendations for Oil Palm on Mineral Soils Based on a Precision Agriculture Approach. *Scientific World Journal*, 2024(1). <https://doi.org/10.1155/2024/1788726>