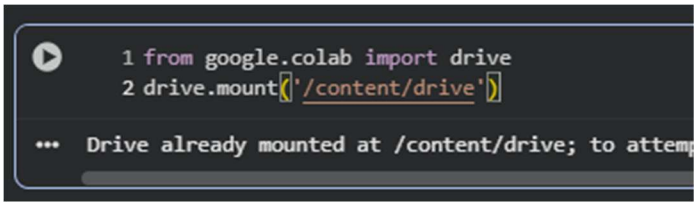
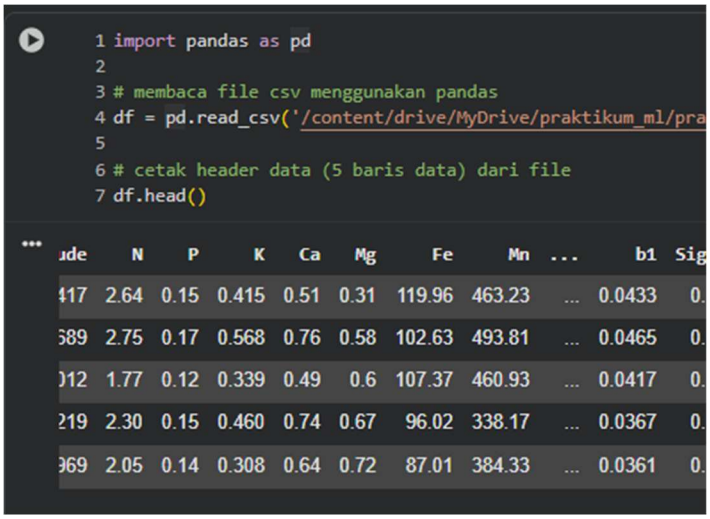
PRAKTIKUM 07

1. Menghubungkan google collab dengan google drive



```
1 from google.colab import drive
2 drive.mount('/content/drive')

Drive already mounted at /content/drive; to attemp
```

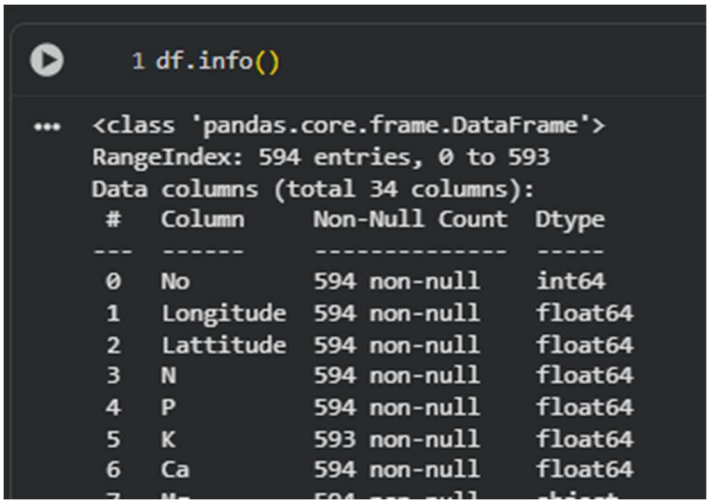
2. Import library yang dibutuhkan



```
1 import pandas as pd
2
3 # membaca file csv menggunakan pandas
4 df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/pra
5
6 # cetak header data (5 baris data) dari file
7 df.head()
```

ude	N	P	K	Ca	Mg	Fe	Mn	...	b1	Sig
417	2.64	0.15	0.415	0.51	0.31	119.96	463.23	...	0.0433	0.
589	2.75	0.17	0.568	0.76	0.58	102.63	493.81	...	0.0465	0.
012	1.77	0.12	0.339	0.49	0.6	107.37	460.93	...	0.0417	0.
219	2.30	0.15	0.460	0.74	0.67	96.02	338.17	...	0.0367	0.
969	2.05	0.14	0.308	0.64	0.72	87.01	384.33	...	0.0361	0.

3. Melihat ringkasan struktur DataFrame secara cepat.



```
1 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594 entries, 0 to 593
Data columns (total 34 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   No         594 non-null    int64
 1   Longitude  594 non-null    float64
 2   Lattitude  594 non-null    float64
 3   N          594 non-null    float64
 4   P          594 non-null    float64
 5   K          593 non-null    float64
 6   Ca         594 non-null    float64
```

4. Menampilkan statistik deskriptif dari kolom-kolom numerik dalam DataFrame.

```
1 df.describe()
```

	No	Longitude	Lattitude	N	P
count	594.000000	594.000000	594.000000	594.000000	594.000000
mean	297.500000	106.878644	-1.024933	2.259091	0.141380
std	171.617307	4.949840	0.965349	0.395499	0.019782
min	1.000000	102.760857	-2.333750	1.140000	0.090000
25%	149.250000	102.927811	-2.233338	1.982500	0.130000
50%	297.500000	103.581969	-0.602276	2.280000	0.140000
75%	445.750000	113.403797	-0.257349	2.570000	0.150000
max	594.000000	113.434700	0.069251	3.230000	0.220000

8 rows × 33 columns

5. Menghilangkan missing value dan menentukan atribut dan target

```
1 from sklearn.model_selection import train_test_split
2 df['Mg'] = pd.to_numeric(df['Mg'], errors='coerce')
3 df['Mg'].fillna(df['Mg'].mean(), inplace=True)
4 df['K'].fillna(df['K'].mean(), inplace=True)
5
6 x = df[['P', 'K', 'Ca', 'Mg', 'Fe', 'Mn',
7        'Cu', 'Zn', 'B', 'b12', 'b11', 'b9', 'b8a', 'b8', 'b7', 'b6', 'b5',
8        'b4', 'b3', 'b2', 'b1', 'Sigma_VV', 'Sigma_VH', 'plia', 'lia', 'iafe',
9        'gamma0_vv', 'gamma0_vh', 'beta0_vv', 'beta0_vh']]
10 y = df["N"]
11
12 x_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=7)
```

- from sklearn.model_selection import train_test_split => membagi data menjadi data latih dan data uji.
- df['Mg'] = pd.to_numeric(df['Mg'], errors='coerce') => Mengubah kolom 'Mg' menjadi tipe numerik
    - df['Mg'].fillna(df['Mg'].mean(), inplace=True)
    - df['K'].fillna(df['K'].mean(), inplace=True)
- Mengisi nilai NaN dengan nilai rata-rata kolom, tujuannya supaya tidak ada missing value yang mengganggu proses training model.
- X = df[[]] => Menentukan atribut
- Y = df[[]] => Menentukan target prediksi
- x_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=7)
    - 80% data jadi data latih (train) → untuk membangun model.
    - 20% data jadi data uji (test) → untuk mengevaluasi performa model.
    - random_state=7 menjaga agar pembagian data selalu konsisten setiap kali dijalankan.

6. Membuat model linear regression

```
1 from sklearn.linear_model import LinearRegression
2
3 model = LinearRegression()
4 model.fit(x_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

- model = LinearRegression() => Membuat objek model regresi linear
- model.fit(x_train, y_train) => Mencari hubungan matematis antara fitur (x) dan target (y).

7. Menghitung R2, MAE, RMSE

```
 1 import numpy as np
 2 from sklearn.linear_model import LinearRegression
 3 from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
 4
 5
 6 y_pred = model.predict(X_test)
 7 r2 = r2_score(y_test, y_pred)
 8
 9 print("R2 :", r2)
10 print("MAE :", mean_absolute_error(y_test, y_pred))
11
12 mse = mean_squared_error(y_test, y_pred)  # default squared=True
13 rmse = np.sqrt(mse)
14 print("RMSE :", rmse)
```

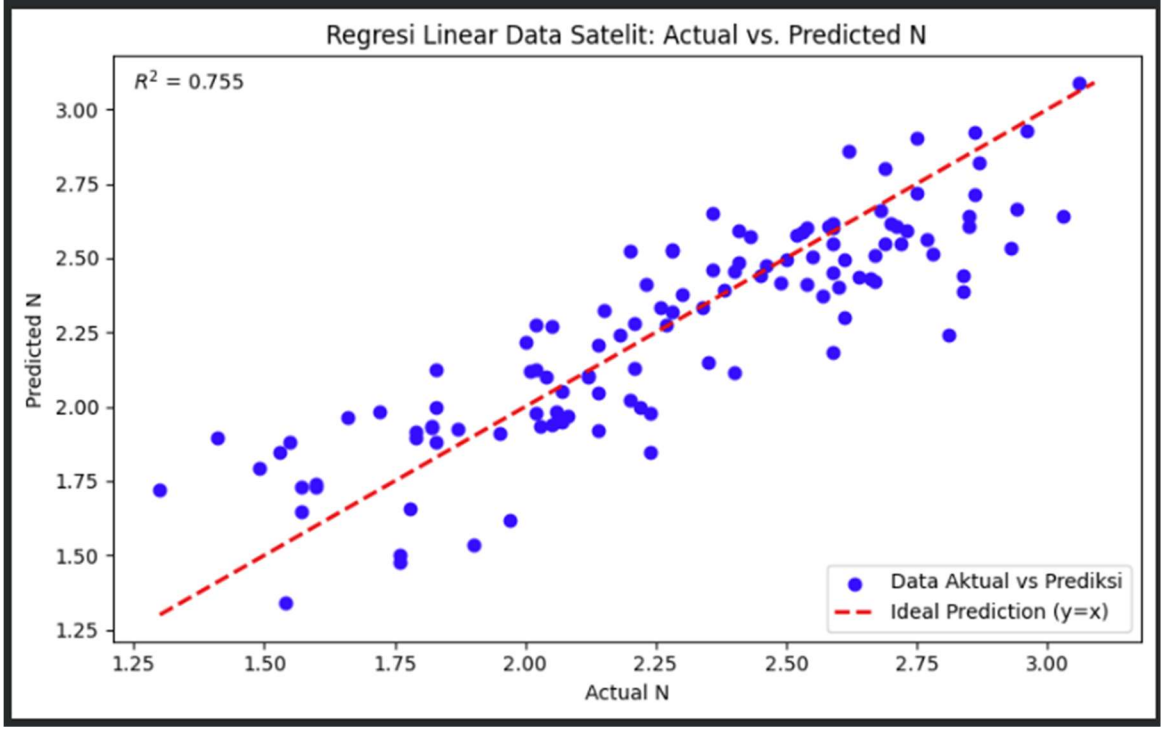
```
R2 : 0.754502518410239
MAE : 0.16239646777012268
RMSE : 0.2027206397565257
```

- import numpy as np => Mengimpor library NumPy dengan alias np. NumPy digunakan untuk operasi matematika seperti akar kuadrat nanti.
- from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
  Mengimpor fungsi evaluasi:
  - r2_score untuk mengukur seberapa baik model menjelaskan variasi data.
  - mean_absolute_error untuk menghitung kesalahan rata-rata absolut.
  - mean_squared_error untuk menghitung kesalahan rata-rata kuadrat.
- y_pred = model.predict(X_test) => Melakukan prediksi menggunakan data uji. Hasilnya disimpan dalam y_pred.
- r2 = r2_score(y_test, y_pred)
  Menghitung R^2 (R-squared). Nilai R^2 menunjukkan seberapa besar model mampu menjelaskan variasi nilai target.

- Nilai mendekati 1 berarti model bagus.
- Nilai 0 berarti model tidak menjelaskan data.

8. Visualisasi scatterplot

```python
import matplotlib.pyplot as plt

# Plot data scatter - Plotting actual vs predicted values
plt.figure(figsize=(8, 5))
plt.scatter(y_test, y_pred, color="blue", label="Data Aktual vs Prediksi")

# Ideal prediction line (y = x)
min_val = min(y_test.min(), y_pred.min())
max_val = max(y_test.max(), y_pred.max())
plt.plot([min_val, max_val], [min_val, max_val], color="red", linestyle='--', linewidth=2, label="I

plt.xlabel("Actual N")
plt.ylabel("Predicted N")
plt.title("Regresi Linear Data Satelit: Actual vs. Predicted N")

# Display R^2 score
plt.text(
    0.02, 0.98,
    f"$R^2$ = {r2:.3f}",
    transform=plt.gca().transAxes,
    va="top"
)

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



Model regresi linear yang dibuat dapat memperkirakan kandungan Nitrogen (N) dengan akurasi yang cukup baik, ditunjukkan oleh R^2 sebesar 0.755 dan pola sebaran data yang mengikuti garis prediksi ideal. Meski masih terdapat error, model sudah layak digunakan sebagai pendekatan awal untuk analisis estimasi unsur Nitrogen menggunakan data satelit.

9. Visualisasi Seaborn

```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8, 5))
errors = y_test - y_pred
sns.histplot(errors, bins=20, kde=True, color="#00BFFF", alpha=0.7)
plt.axvline(0, color='red', linestyle='--', linewidth=2)
plt.title("Distribusi Error (y_test - y_pred)")
plt.xlabel("Error (Selisih Prediksi dan Aktual)")
plt.ylabel("Frekuensi")
plt.tight_layout()
plt.show()

```