Laporan Tugas Praktikum 03

Muhammad Bintang Gunawan - 0110224003

Teknik Informatika, STT Terpadu Nurul Fikri, Depok

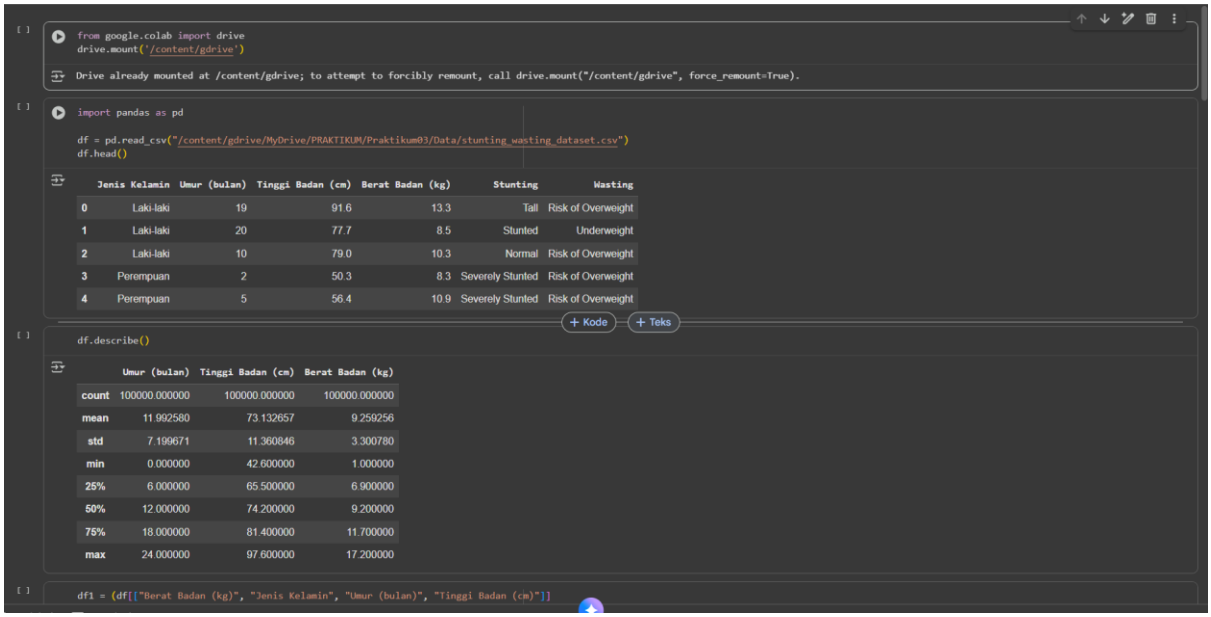
0110224003@ student.nurulfikri.ac.id

Abstract

Pada praktikum ketiga ini, pembahasan berfokus pada penerapan regresi linear untuk memprediksi data numerik berdasarkan hubungan antar variabel.

1. Multiple Linear Regresi

Berikut adalah kode berserta penjelasan dari praktikum Multiple Linear Regresi.



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

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

```
import pandas as pd

df = pd.read_csv("/content/gdrive/MyDrive/PRAKTIKUM/Praktikum03/Data/stunting_wasting_dataset.csv")
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

+ Kode + Teks

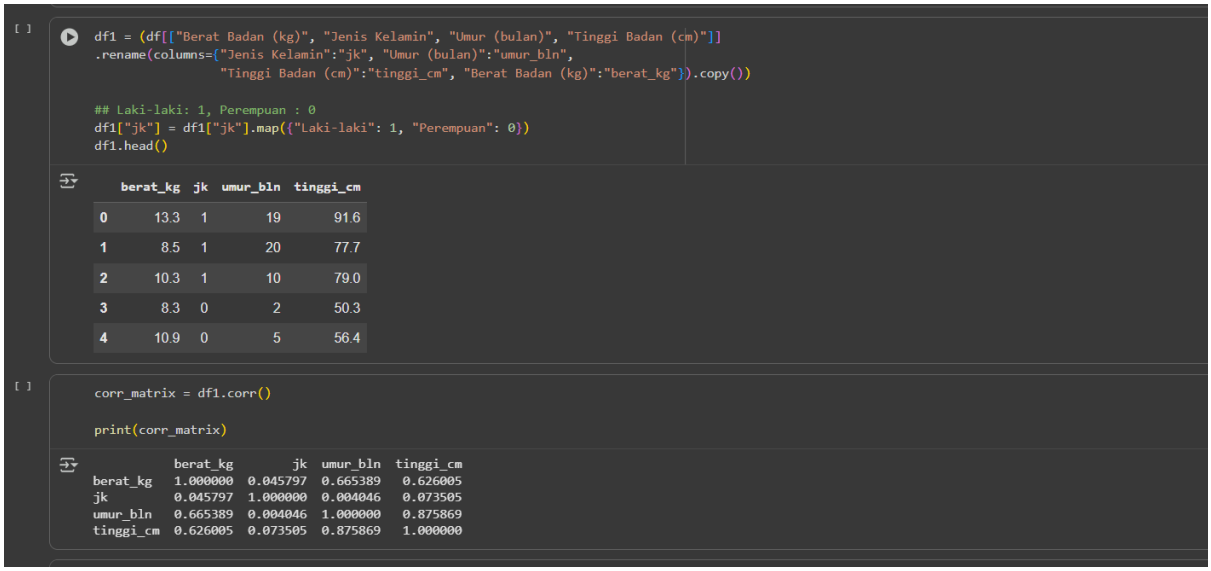
```
df.describe()
```

	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

```
df1 = (df[["Berat Badan (kg)", "Jenis Kelamin", "Umur (bulan)", "Tinggi Badan (cm)"]]
```

Gambar 1

Pada bagian awal ini dilakukan proses mount Google Drive agar file dataset bisa diakses dari Colab.
Kemudian library *pandas* digunakan untuk membaca dataset stunting_wasting_dataset.csv dan menampilkan beberapa baris pertama dengan `df.head()` untuk memastikan data berhasil dimuat.
Selanjutnya, fungsi `df.describe()` digunakan untuk menampilkan statistik deskriptif dari kolom numerik seperti umur, tinggi badan, dan berat badan, sehingga dapat diketahui rentang nilai dan rata-rata 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

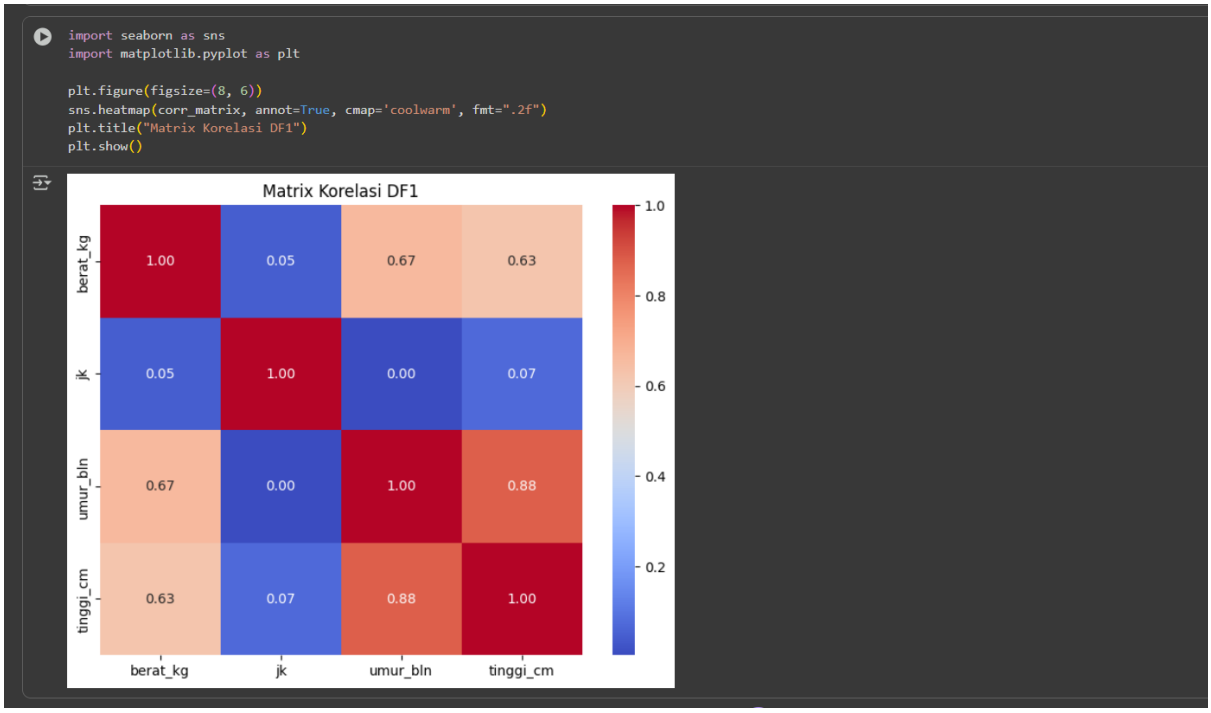
```
corr_matrix = df1.corr()

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

Pada cell pertama, data disederhanakan dengan memilih kolom penting yaitu berat badan, jenis kelamin, umur, dan tinggi badan. Nama kolom juga diubah agar lebih ringkas dan mudah digunakan dalam analisis.
olom jenis kelamin kemudian dikonversi ke bentuk numerik: Laki-laki 1 dan Perempuan 0, supaya dapat digunakan dalam perhitungan regresi. Hasilnya ditampilkan menggunakan `df1.head()`.

Pada cell kedua, digunakan fungsi `corr()` untuk menghitung korelasi antar variabel numerik. Nilai korelasi ini menunjukkan seberapa kuat hubungan antar kolom — misalnya, berat badan memiliki korelasi yang cukup tinggi dengan tinggi badan dan umur, menandakan bahwa kedua faktor tersebut berpengaruh besar terhadap berat badan.



Gambar 3

Pada bagian ini digunakan seaborn dan matplotlib untuk membuat *heatmap* dari matriks korelasi yang sebelumnya dihitung. Visualisasi ini membantu melihat hubungan antar variabel secara lebih jelas melalui warna.

Warna merah menunjukkan korelasi positif yang kuat (semakin tinggi nilainya, semakin besar pengaruhnya antar variabel), sedangkan biru menunjukkan korelasi yang lemah atau negatif.

```python
from sklearn.model_selection import train_test_split

y = df1["berat_kg"]

X = df1[["umur_bln", "tinggi_cm"]]

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
)

print("Jumlah data train :", len(X_train))
print("Jumlah data test :", len(X_test))

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

```python
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 4

Pada bagian ini data dibagi menjadi dua bagian data latih dan data uji menggunakan fungsi `train_test_split` dari library sklearn.

Sebanyak 80% data digunakan untuk melatih model, sedangkan 20% sisanya digunakan untuk menguji hasil prediksi agar model tidak hanya menghafal data. Variabel y berisi berat badan, sementara variabel X berisi umur dan tinggi badan sebagai faktor yang memengaruhi berat.

```python
import statsmodels.api as sm


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(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 5

Model Regresi OLS berhasil diestimasi menggunakan statsmodels.
Model tersebut menunjukkan bahwa variabel y dapat diprediksi oleh variabel umur_bln (x1) dan tinggi_cm (x2) berdasarkan persamaan regresi

```
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:              Sat, 11 Oct 2025   Prob (F-statistic):               0.00
Time:                      02:47:58   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.
```

Gambar 6

Dari output ini hasil evaluasi model regresi linear ganda. Nilai **R^2 = 0.450** menunjukkan model mampu menjelaskan sekitar 45% variasi berat badan. Koefisien **umur (0.2297)** dan **tinggi (0.0542)** bernilai positif, artinya keduanya berpengaruh menaikkan berat badan. Nilai **p-value = 0.000** menandakan kedua variabel signifikan secara statistik.

```python
import numpy as np

X_test_const = sm.add_constant(X_test)
y_pred_test = model.predict(X_test_const)

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
})

hasil["Selisih error (kg)"] = hasil["Berat Prediksi (kg)"] - hasil["Berat Aktual (kg)"]

denom = hasil["Berat Aktual (kg)"].replace(0, np.nan)
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

Gambar 7

Dari tabel hasil prediksi model regresi. Bagian atas menambahkan kolom konstanta ke data uji, lalu menghasilkan prediksi berat badan. Setelah itu dibuat DataFrame berisi umur, tinggi, berat aktual, berat prediksi, serta selisih antara keduanya. Kolom **“Akurasi (%)”** dihitung dari selisih relatif terhadap berat aktual.

2. Tugas Praktikum Mandiri

Berikut adalah penjelsan kode dan output dari hasil Tugas Praktikum.
Pada bagian tugas ini, dataset *Bike Sharing* digunakan untuk membuat model prediksi jumlah penyewaan sepeda harian menggunakan regresi linear berganda.

```
[ ]
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ]
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/PRAKTIKUM/Praktikum03/Data/day.csv")
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

Langkah berikutnya: Buat kode dengan df | New interactive sheet

Gambar 8

Bagian ini menghubungkan Google Drive ke Colab dan membaca dataset day.csv menggunakan Pandas untuk menampilkan lima baris data pertama

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

```
df.describe()
```

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.683995	1.395349	0.495385	0.474354	0.627894	0.190486	848.176471	3656.172367	4504.348837
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.465233	0.544894	0.183051	0.162961	0.142429	0.077498	686.622488	1560.256377	1937.211452
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.059130	0.079070	0.000000	0.022392	2.000000	20.000000	22.000000
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	0.337083	0.337842	0.520000	0.134950	315.500000	2497.000000	3152.000000
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.000000	1.000000	0.498333	0.486733	0.626667	0.180975	713.000000	3662.000000	4548.000000
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	0.655417	0.608602	0.730209	0.233214	1096.000000	4776.500000	5956.000000
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	0.861667	0.840896	0.972500	0.507463	3410.000000	6946.000000	8714.000000

Gambar 9

Bagian ini menampilkan informasi struktur dataset menggunakan `df.info()` serta statistik deskriptif setiap kolom dengan `df.describe()`, seperti jumlah data, nilai rata-rata, standar deviasi, dan nilai minimum–maksimum untuk memahami karakteristik awal dataset.

Gambar 10

Di bagian ini dilakukan proses pembagian data, pelatihan model regresi linear, dan evaluasi hasil prediksi.
Pertama, data dibagi jadi dua bagian pakai fungsi `train_test_split dari sklearn.model_selection`, yaitu **data training (80%)** dan **data testing (20%)**. Data training dipakai buat ngajarin model supaya bisa ngerti pola hubungan antara variabel input (X) dan target (y), sedangkan data testing dipakai buat nguji seberapa bagus model tersebut kalau dikasih data baru yg belum pernah dilihat. Dari hasil pembagian, total ada 584 data training dan 147 data testing.

Lanjut ke bagian kedua, di sini digunakan kelas LinearRegression dari sklearn.linear_model. Model ini termasuk jenis regresi linear sederhana, yg fungsinya buat nyari hubungan linier antara variabel bebas dan variabel target. Setelah objek model dibuat `(model = LinearRegression())`, model dilatih dengan data training lewat perintah `model.fit(X_train, y_train).`

Setelah model selesai dilatih, langkah berikutnya yaitu prediksi data testing dengan model.predict(X_test). Hasil prediksi (y_pred) dibandingkan dengan nilai asli (y_test) buat ngukur performa model.

```
coeff_df = pd.DataFrame(model.coef_, X.columns, columns=['Koefisien'])
print(coeff_df)
print("\nIntercept:", model.intercept_)
```

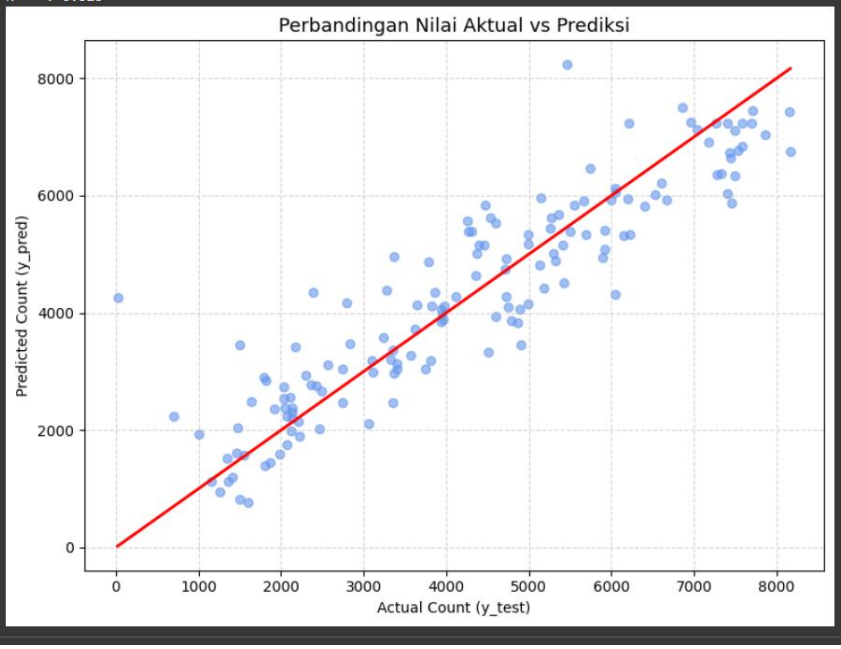
```
              Koefisien
season       524.722536
yr          2023.997547
mnth         -38.444658
holiday     -391.550766
weekday       72.937003
workingday   160.804892
weathersit  -632.856284
temp        2097.247836
atemp       3488.042179
hum         -865.439419
windspeed  -2080.540395

Intercept: 1248.3209284778172
```

Gambar 11

Bagian ini digunakan buat melihat hasil koefisien dan intercept dari model regresi linear yg udah dilatih sebelumnya.

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2 = r2_score(y_test, y_pred)

print("Evaluasi Model:")
print(f"MAE  : {mae:.2f}")
print(f"RMSE : {rmse:.2f}")
print(f"R²   : {r2:.3f}")

plt.figure(figsize=(8, 6))

plt.scatter(y_test, y_pred, color='cornflowerblue', alpha=0.6)

x_line = np.linspace(min(y_test), max(y_test), 100)
plt.plot(x_line, x_line, color='red', linewidth=2)

plt.title("Perbandingan Nilai Prediksi", fontsize=13)
plt.xlabel("Actual Count (y_test)")
plt.ylabel("Predicted Count (y_pred)")

plt.grid(True, linestyle='--', alpha=0.5)

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

```
Evaluasi Model:
MAE  : 617.39
RMSE : 831.29
R²   : 0.828
```



Bagian ini digunakan buat evaluasi performa model regresi linear yg udah dilatih sebelumnya. Tujuannya buat ngeliat seberapa akurat model dalam memprediksi data testing dan seberapa dekat hasil prediksi dengan nilai aslinya.

3. Kesimpulan hasil implementasi algoritma

Pada praktikum ini, model regresi linear berhasil diterapkan untuk menganalisis hubungan antar variabel pada dua kasus berbeda. Hasil menunjukkan bahwa model mampu memprediksi nilai target dengan tingkat akurasi yang cukup baik, terlihat dari nilai R^2 yang tinggi dan error yang relatif kecil. Melalui proses analisis korelasi, pelatihan, dan evaluasi model, dapat disimpulkan bahwa algoritma regresi linear mampu menggambarkan pola hubungan antar variabel dengan efektif, serta menjadi dasar yang baik untuk pengembangan model prediksi lebih lanjut.

Link GitHub:
https://github.com/XiaoMao15/ML_Praktikum-dan-Tugas.git