Tugas 3: Tugas Praktikum Mandiri 03

Revani – 0110224111 [1]

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

[*]E-mail: 0110224111@student.nurulfikri.ac.id

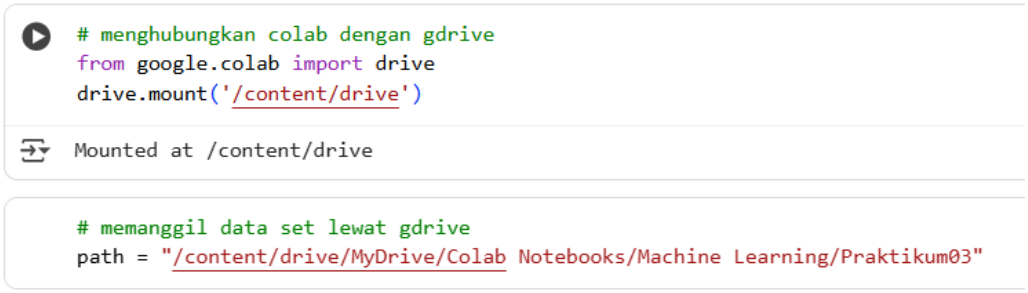
Abstract. Pada tugas praktikum mandiri ini, dilakukan pemodelan *Multiple Linear Regression* untuk memprediksi jumlah peminjaman sepeda harian (*cnt*) dengan metode OLS (*Ordinary Least Squares*). Dataset yang digunakan merupakan *Bike Sharing Dataset* dari Kaggle, yang berisi berbagai variabel seperti musim, kondisi cuaca, suhu, kelembapan, kecepatan angin, serta informasi waktu. Tujuan utama dari tugas praktikum mandiri ini adalah memahami bagaimana membangun model regresi untuk memprediksi nilai kontinu, dalam hal ini jumlah penyewaan sepeda, serta mengevaluasi kinerja model melalui metrik seperti MAE, RMSE, dan R^2. Proses pengerjaan dilakukan menggunakan Python dengan pustaka *pandas*, *statsmodels*, dan *scikit-learn*. Dari hasil pemodelan, diperoleh persamaan regresi dan visualisasi perbandingan antara nilai aktual dan nilai prediksi, sehingga dapat dilihat seberapa baik model dapat menggambarkan data yang ada.

1. Penjelasan Hasil Praktikum Mandiri

Pada praktikum mandiri ini, langkah awal yang dilakukan adalah membaca dataset *day.csv* dari *Bike Sharing Dataset* melalui Google Colab untuk memastikan data dapat diakses dengan baik. Setelah data berhasil dimuat, dilakukan proses pembersihan data dengan menghapus kolom yang tidak relevan untuk pemodelan, yaitu *instant, dteday, casual,* dan *registered.* Langkah selanjutnya adalah menganalisis korelasi antar variabel untuk mengetahui variabel-variabel mana yang paling berpengaruh terhadap *cnt* (jumlah penyewaan sepeda). Berdasarkan hasil heatmap korelasi, variabel yang memiliki pengaruh cukup besar antara lain *temp, atemp, season, yr, weathersit, hum,* dan *windspeed.*

Data kemudian dibagi menjadi dua bagian, yaitu data latih (80%) dan data uji (20%) untuk melatih dan menguji model. Proses pemodelan dilakukan menggunakan metode *Multiple Linear Regression* dengan OLS (*Ordinary Least Squares*). Hasil pemodelan menunjukkan nilai R^2 yang cukup baik, yang berarti model dapat menjelaskan Sebagian besar variasi jumlah penyewaan sepeda dari variabel-variabel predictor. Nilai MAE dan RMSE juga dihitung untuk mengetahui seberapa jauh kesalahan prediksi dari nilai actual. Terakhir, dibuat grafik perbandingan antara nilai actual dengan nilai prediksi. Titik-titik pada grafik menggambarkan hasil prediksi terhadap data aktual. Semakin mendekati garis diagonal, semakin baik akurasi prediksi model. Berikut ini adalah hasil praktikum yang saya kerjakan pada praktikum mandiri 03.

Langkah awal yang dilakukan adalah menghubungkan google colab dengan google drive dan memanggil dataset lewat google drive seperti yang ditunjukkan pada Gambar 1.



Gambar 1. Pada bagian ini ditunjukkan untuk menghubungkan *gdrive* dan memanggil *dataset*.

```
import pandas as pd
```

Perintah ini digunakan untuk memanggil library *Pandas* yang digunakan untuk mengolah data berbasis tabel (*dataframe*).

```
df = pd.read_csv(path + '/Data/day.csv')
df.head()
```

Perintah ini digunakan untuk membaca file *day.csv* yang ada di folder data. Hasil pembacaan disimpan dalam variabel *df* sebagai sebuah *dataframe* seperti yang ditunjukkan pada Gambar 2.

```
# membaca file csv menggunakan pandas
import pandas as pd

# buat datagrame read data
df = pd.read_csv(path + '/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

Gambar 2. Pada bagian ini ditunjukkan untuk membaca file *csv*.

```
df.info()
```

Perintah ini digunakan untuk menampilkan ringkasan informasi dari *dataframe*. Output yang ditampilkan adalah jumlah data dengan 731 baris dan 16 kolom, dengan tipe data *int*, *float* dan *object*, serta semua kolom punya 731 *non-null values* yang berarti tidak ada data kosong. Jadi dapat dipahami bahwa *dataset* rapi dan siap untuk dianalisis lebih lanjut tanpa perlu pembersihan data besar-besaran seperti yang ditunjukkan pada Gambar 3.

```
# Cek informasi data
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
```

Gambar 3. Pada bagian ini ditunjukkan untuk cek informasi data.

`df.describe()`
Perintah ini untuk menampilkan *statistic deskriptif* (*mean, min, max, std, kuartil*) untuk kolom numerik.
`df = df.drop(['instant', 'dteday', 'casual', 'registered'], axis=1)`
Perintah ini digunakan untuk menghapus kolom yang tidak diperlukan dari *dataframe*. Kolom *instant* hanya nomor urut, *dteday* bertipe tanggal, sedangkan *casual* dan *registered* merupakan komponen dari *cnt* sehingga tidak boleh dijadikan variabel *predictor* (untuk menghindari data leakage) seperti yang ditunjukkan pada Gambar 4.

```
# Statistik deskriptif
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

```
# Buang kolom yang tidak digunakan
df = df.drop(['instant', 'dteday', 'casual', 'registered'], axis=1)
```

Gambar 4. Pada bagian ini ditunjukkan tabel *statistic deskriptif* dan buang kolom yang tidak digunakan.

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

Perintah ini digunakan untuk mengimpor pustaka visualisasi data. *Seaborn* digunakan untuk membuat grafik statistic, *matplotlib* untuk kontrol tampilan grafik.

```
corr = df.corr()
```

Perintah ini menghitung nilai korelasi antar kolom numerik. Korelasi bernilai antara -1 sampai 1.

```
plt.figure(figsize=(10,6))
```

Perintah ini digunakan untuk mengatur ukuran gambar.

```
sns.heatmap(corr, annot=True, cmap='coolwarm')
```

Perintah ini digunakan untuk menampilkan peta panas (*heatmap*) dari hasil korelasi, dengan *annot=True* yang menampilkan nilai korelasi di dalam kotak dan *cmap='coolwarm'* untuk mengatur gradasi warna dari biru ke merah.

```
plt.title('Heatmap Korelasi')
plt.show()
```

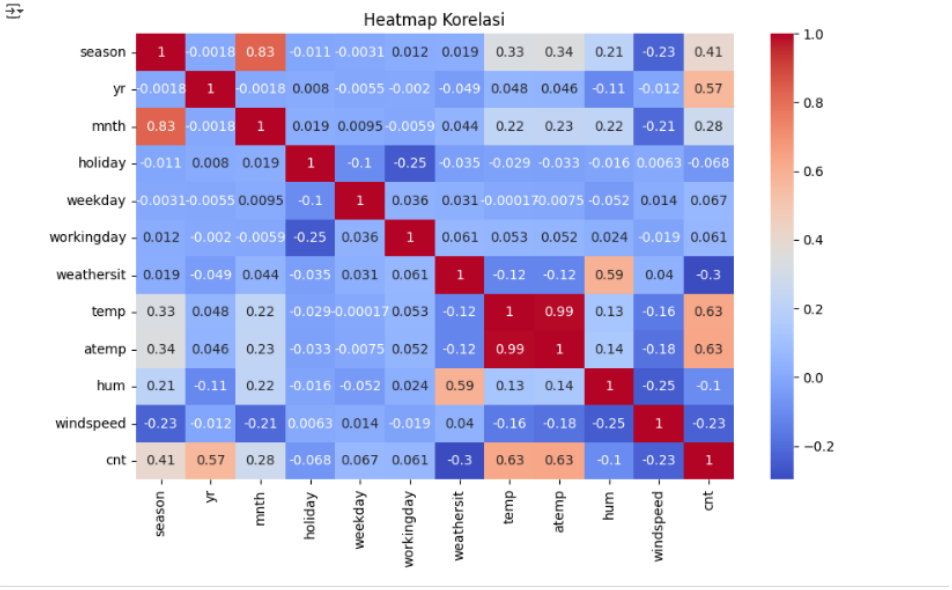
Perintah ini digunakan untuk memberi judul grafik dan menampilkannya.

Output yang muncul jika grafik *heatmap* dengan warna merah menunjukkan korelasi positif kuat (missal *temp* dengan *cnt*), sedangkan warna biru menunjukkan korelasi negatif (misalnya *windspeed* dengan *cnt*). Hal ini membantu memilih variabel yang signifikan untuk pemodelan seperti yang ditunjukkan pada Gambar 5.

Analisis Korelasi

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

corr = df.corr()
plt.figure(figsize=(10,6))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Heatmap Korelasi')
plt.show()
```



Gambar 5. Pada bagian ini ditunjukkan analisis korelasi.

```
from sklearn.model_selection import train_test_split
```

Perintah ini digunakan untuk mengimpor fungsi pembagi data dari pustaka *Scikit-learn*.

```
X = df.drop(['cnt'], axis=1)
```

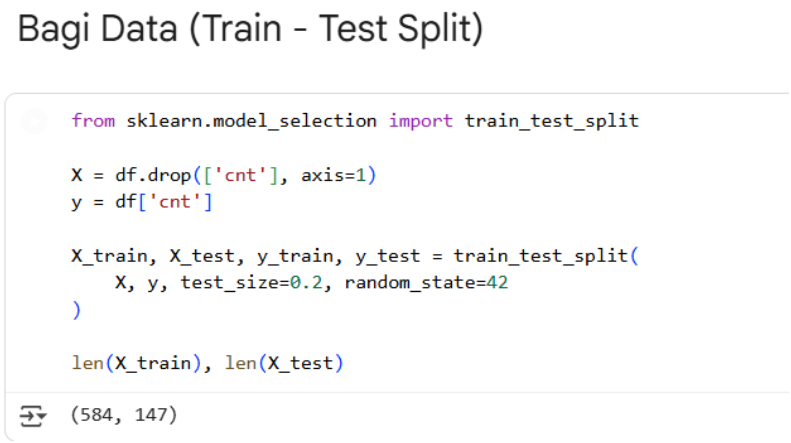
Perintah ini digunakan untuk menyimpan semua kolom *predictor* ke variabel X.

```
y = df['cnt']
```

Perintah ini digunakan untuk menyimpan target jumlah penyewaan sepeda harian ke variabel y.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
len(X_train), len(X_test)
```

Perintah ini digunakan untuk membagi *dataset* menjadi data latih (80%) dan data uji (20%). Output yang di hasilkan yaitu *x_train* berisi sekitar 584 baris, *X_test* sekitar 147 baris, *y_train* dan *y_test* berisi jumlah target yang sesuai dengan pembagian seperti yang ditunjukkan pada Gambar 6.



Gambar 6. Pada bagian ini ditunjukkan bagi data (*train-test split*).

`import statsmodels.api as sm`
Perintah ini digunakan untuk mengimpor pustaka *statsmodels* untuk analisis *statistic* dan *regresi*.
`X_train_const = sm.add_constant(X_train)`
Perintah ini digunakan menambahkan kolom konstanta (*bias/intercept*) ke variabel X. Model regresi linear membutuhkan *intercept* agar persamaan regresinya lengkap.
`model = sm.OLS(y_train, X_train_const).fit()`
Perintah ini digunakan untuk membuat model regresi linier berganda menggunakan metode OLS (*Ordinary Least Squares*). Perintah *fit*() digunakan untuk melatih model menggunakan data latih.
`print(model.summary())`
Perintah ini digunakan untuk menampilkan ringkasan model dalam bentuk *table statistic*.
Output yang dihasilkan yaitu tabel ringkasan dengan koefisien untuk setiap variabel (*season*, *yr*, *temp*, dll). Nilai R^2 dan *Adjusted* R^2. P-Value menunjukkan variabel signifikan seperti yang ditunjukkan pada Gambar 7.

Pemodelan OLS (Ordinary Least Squares)

```
[ ]
```

```python
import statsmodels.api as sm

# Tambahkan konstanta (intercept)
X_train_const = sm.add_constant(X_train)

# Buat model OLS
model = sm.OLS(y_train, X_train_const).fit()

# Ringkasan model
print(model.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.791
Model:                            OLS   Adj. R-squared:                  0.787
Method:                 Least Squares   F-statistic:                     196.9
Date:                Wed, 08 Oct 2025   Prob (F-statistic):          2.56e-186
Time:                        13:03:10   Log-Likelihood:                -4784.9
No. Observations:                 584   AIC:                             9594.
Df Residuals:                     572   BIC:                             9646.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       1248.3209    272.690      4.578      0.000     712.724    1783.918
season       524.7225     65.596      7.999      0.000     395.885     653.560
yr          2023.9975     73.972     27.362      0.000    1878.708    2169.287
mnth         -38.4447     20.537     -1.872      0.062     -78.783       1.893
holiday     -391.5508    240.168     -1.630      0.104    -863.269      80.167
weekday       72.9370     18.310      3.984      0.000      36.975     108.900
workingday   160.8049     80.523      1.997      0.046       2.647     318.963
weathersit  -632.8563     87.718     -7.215      0.000    -805.145    -460.568
temp        2097.2478   1480.268      1.417      0.157    -810.176    5004.672
atemp       3488.0422   1675.527      2.082      0.038     197.106    6778.979
hum         -865.4394    354.490     -2.441      0.015   -1561.701    -169.178
windspeed  -2080.5404    519.038     -4.008      0.000   -3099.994   -1061.087
==============================================================================
Omnibus:                       54.728   Durbin-Watson:                   1.967
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               99.315
Skew:                          -0.595   Prob(JB):                     2.72e-22
Kurtosis:                       4.633   Cond. No.                         519.
==============================================================================

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

Gambar 7. Pada bagian ini ditunjukkan pemodelan OLS.

`X_test_const = sm.add_constant(X_test)`

Perintah ini digunakan untuk menambahkan konstanta juga pada data uji agar struktur sama dengan data latih.

`y_pred = model.predict(X_test_const)`

Perintah ini digunakan untuk menghasilkan nilai prediksi *cnt* berdasarkan model yang sudah dilatih.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
import numpy as np
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)
```
Perintah *mean_absolut_error* digunakan untuk menghitung rata-rata kesalahan *absolut* antara prediksi dan data aktual. Perintah *mean_squared_error* digunakan untuk menghitung rata-rata kuadrat kesalahan. Perintah *np.sqrt()* digunakan untuk mengambil akar dari MSE untuk mendapat RMSE. Perintah *r^2_score* digunakan untuk menghitung koefisien determinasi (seberapa baik model menjelaskan data).
```
print(f"MAE  : {mae:.2f}")
print(f"RMSE: {rmse:.2f}")
print(f"R²   : {r2:.3f}")
```
Output yang dihasilkan adalah nilai MAE kecil dengan selisih rata-rata antara prediksi dan aktual kecil, nilai RMSE dengan ukuran kesalahan prediksi secara kuadrat, serta nilai R^2 dengan semakin mendekati 1 maka semakin baik model seperti yang ditunjukkan pada Gambar 8.

Evaluasi Model

```
[ ]
X_test_const = sm.add_constant(X_test)
y_pred = model.predict(X_test_const)

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

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

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

Gambar 8. Pada bagian ini ditunjukkan evaluasi model.

```
intercept = model.params['const']
coef = model.params.drop('const')
```
Perintah *model.params* digunakan untuk menyimpan semua nilai koefisien model regresi. Perintah *['const']* digunakan untuk mengambil nilai *intercept*. Perintah *.drop('const')* digunakan untuk menampilkan hanya koefisien variabel X.
```
print(f"Intercept: {intercept}")
print("Koefisien:")
print(coef)
```
Output yang dihasilkan adalah nilai *intercept* dan koefisien dari setiap variabel seperti yang ditunjukkan pada Gambar 9.

Persamaan Regresi

```
[ ]
```

```
intercept = model.params['const']
coef = model.params.drop('const')

print(f"Intercept: {intercept}")
print("Koefisien:")
print(coef)
```

```
Intercept: 1248.3209284778216
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
dtype: float64
```

Gambar 9. Pada bagian ini ditunjukkan persamaan regresi.

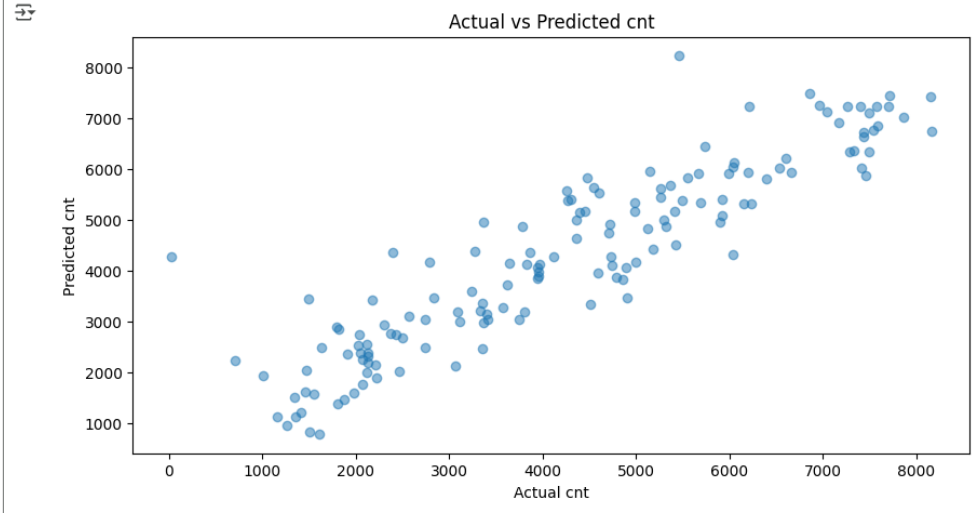
```
plt.figure(figsize=(10,5))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual cnt')
plt.ylabel('Predicted cnt')
plt.title('Actual vs Predicted cnt')
plt.show()
```

Perintah *plt.figure(figsize=(10,5))* digunakan untuk mengatur ukuran grafik. Perintah *plt.scatter(y_test, y_pred, alpha=0.5)* digunakan untuk membuat grafik *scatter plot* dengan titik-titik perbandingan antara data aktual (*y_test*) dan hasil prediksi (*y_pred*). Perintah *alpha=0.5* digunakan untuk membuat titik semi-transparan. Perintah *plt.xlabel()* dan *plt.ylabel()* digunakan untuk memberi label sumbu X dan Y. Perintah *plt.title()* digunakan untuk memberi judul pada grafik.

Output yang dihasilkan adalah titik-titik menyebar di sekitar garis diagonal dan semakin rapat titik ke garis diagonal maka semakin baik prediksi model seperti yang ditunjukkan pada Gambar 10.

Visualisasi

```
plt.figure(figsize=(10,5))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual cnt')
plt.ylabel('Predicted cnt')
plt.title('Actual vs Predicted cnt')
plt.show()
```



Gambar 10. Pada bagian ini ditunjukkan hasil visualisasi.

2. Kesimpulan dan Hasil Implementasi

Dari implementasi ini, terlihat bahwa *Multiple Linear Regression* menggunakan OLS mampu membangun model prediksi yang cukup baik untuk kasus jumlah penyewaan sepeda harian. Variabel (*temp, atemp, season, weathersit, hum, windspeed*) terbukti memiliki pengaruh terhadap jumlah peminjaman sepeda. Nilai R^2 dan metrik error menunjukkan bahwa model memiliki performa prediksi yang cukup baik. Tahapan analisis ini juga mencerminkan alur standar proyek *machine learning:* mulai dari *load data, cleaning, analisis korelasi, split data, training model, evaluasi,* hingga visualisasi hasil.

3. Link GitHub (Praktikum dan Latihan Mandiri 03)

https://github.com/revani18/ML_2025_Revani_3AI01/tree/666bb54d341bb6c7cffc839a06471e12dd1eaf9c/Praktikum03