

Tugas 3: Tugas Mandiri 3 – Regresi Linear Sederhana

Amaya Eshia - 0110224102*

¹ Teknik Informatika, STT Terpadu Nurul Fikri, Depok

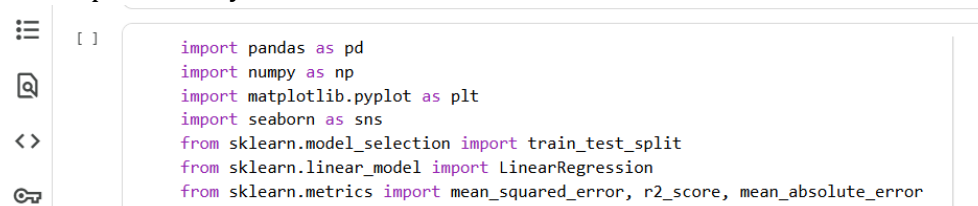
*E-mail: name@institution.edu – 0110224102@student.nurulfikri.ac.id

Abstract. Praktikum ini bertujuan untuk mempelajari teknik regresi linear sederhana dalam pemodelan machine learning menggunakan dataset bike sharing (day.csv). Dataset dianalisis terlebih dahulu melalui statistik deskriptif, pemeriksaan missing values, dan analisis korelasi untuk memilih variabel independen yang paling berpengaruh terhadap variabel target 'cnt' (jumlah penyewaan sepeda). Pembagian data dilakukan dengan rasio 80:20 untuk training dan testing. Model regresi linear dibangun menggunakan library Scikit-learn dan Statsmodels untuk mendapatkan hasil yang detail. Hasil menunjukkan bahwa model memiliki R^2 score sekitar 0.82 pada data testing, menandakan kemampuan model dalam menjelaskan variasi data sebesar 82%. Evaluasi metrik seperti MAE, MSE, dan RMSE dilakukan, disertai visualisasi seperti scatter plot actual vs predicted, residual plot, dan distribusi residuals. Verifikasi dilakukan untuk memastikan model tidak overfitting dan residuals terdistribusi secara acak. Keterampilan ini penting dalam memprediksi variabel kontinu seperti jumlah penyewaan sepeda berdasarkan faktor cuaca dan waktu.

Kata Kunci: Machine Learning, Regresi Linear, Bike Sharing Dataset, Korelasi, Scikit-learn, Statsmodels, Python.

1. Penjelasan Koding Program

1.1 Import Library



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

Mengimpor library Statsmodels dengan alias sm untuk analisis statistik, Pandas dengan alias pd untuk manipulasi data, Matplotlib dengan alias plt untuk visualisasi, dan NumPy dengan alias np untuk operasi matematika dan array. Selain itu, mengimpor library lain seperti Seaborn untuk visualisasi yang lebih baik, serta fungsi-fungsi dari Scikit-learn untuk pembagian data, model regresi, dan evaluasi metrik.

1.2 Menghubungkan Google Colab dengan GDrive

[]

```
# menghubungkan colab dengan google drive
from google.colab import drive
drive.mount('/content/drive')
```



Mounted at /content/drive

Menghubungkan Google Colab dengan Google Drive agar dapat mengakses file dataset yang tersimpan di Drive. Setelah dijalankan, akan muncul link untuk otorisasi akses.

1.3 Membaca File CSV dan Menampilkan Data Awal

```
# Load dataset
df = pd.read_csv('/content/drive/MyDrive/Praktikum Machine Learning_Amaya Eshia_0110224102_Ai02/Praktikum 3/Data/day.csv', sep=',')

# Lihat 5 baris pertama
df.head()
```

- Membaca file CSV dengan nama day.csv menggunakan fungsi `pd.read_csv()` dan menyimpannya dalam DataFrame `df`.
- Menampilkan 5 baris pertama data menggunakan fungsi `.head()` untuk verifikasi awal.
- Menampilkan ringkasan statistik deskriptif menggunakan `.describe()` untuk memahami distribusi data..

1.4 Menganalisa Dataset di Awal

≡

[]



```
# Info dataset
print(df.info())
print("\n")
print(df.describe())

# Cek missing values
print("\nMissing values:")
print(df.isnull().sum())

# Lihat korelasi dengan cnt
print("\nKorelasi dengan cnt:")
print(df.select_dtypes(include=np.number).corr()['cnt'].sort_values(ascending=False))
```

- Memeriksa informasi dataset menggunakan `.info()` untuk melihat tipe data dan jumlah entri.
- Memeriksa missing values menggunakan `.isnull().sum()` untuk memastikan tidak ada data yang hilang.
- Menghitung korelasi variabel numerik dengan target 'cnt' dan menampilkannya dalam urutan menurun

1.5 Membuat Heatmap korelasi

```
[ ]
# Heatmap korelasi
plt.figure(figsize=(12, 8))
correlation_matrix = df.select_dtypes(include=np.number).corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix')
plt.show()

# Korelasi spesifik dengan cnt
plt.figure(figsize=(10, 6))
df.select_dtypes(include=np.number).corr()['cnt'].sort_values(ascending=False).plot(kind='bar')
plt.title('Korelasi variabel dengan cnt')
plt.ylabel('Correlation')
plt.show()
```

- Membuat heatmap korelasi menggunakan Seaborn untuk visualisasi matriks korelasi.
- Membuat bar plot korelasi spesifik dengan 'cnt' untuk melihat variabel yang paling berpengaruh.

1.6 Pemilihan Variabel dan Pembagian Dataset

```
[ ]
# Berdasarkan korelasi, pilih variabel yang paling berpengaruh
# Contoh: temp, atemp, hum, windspeed (sesuaikan dengan hasil analisis korelasi)

# Variabel independent (X) - pilih kolom yang relevan
X = df[['temp', 'atemp', 'hum', 'windspeed', 'season', 'yr', 'mnth',
        'holiday', 'weekday', 'workingday', 'weathersit']]

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

print("Shape X:", X.shape)
print("Shape y:", y.shape)
```

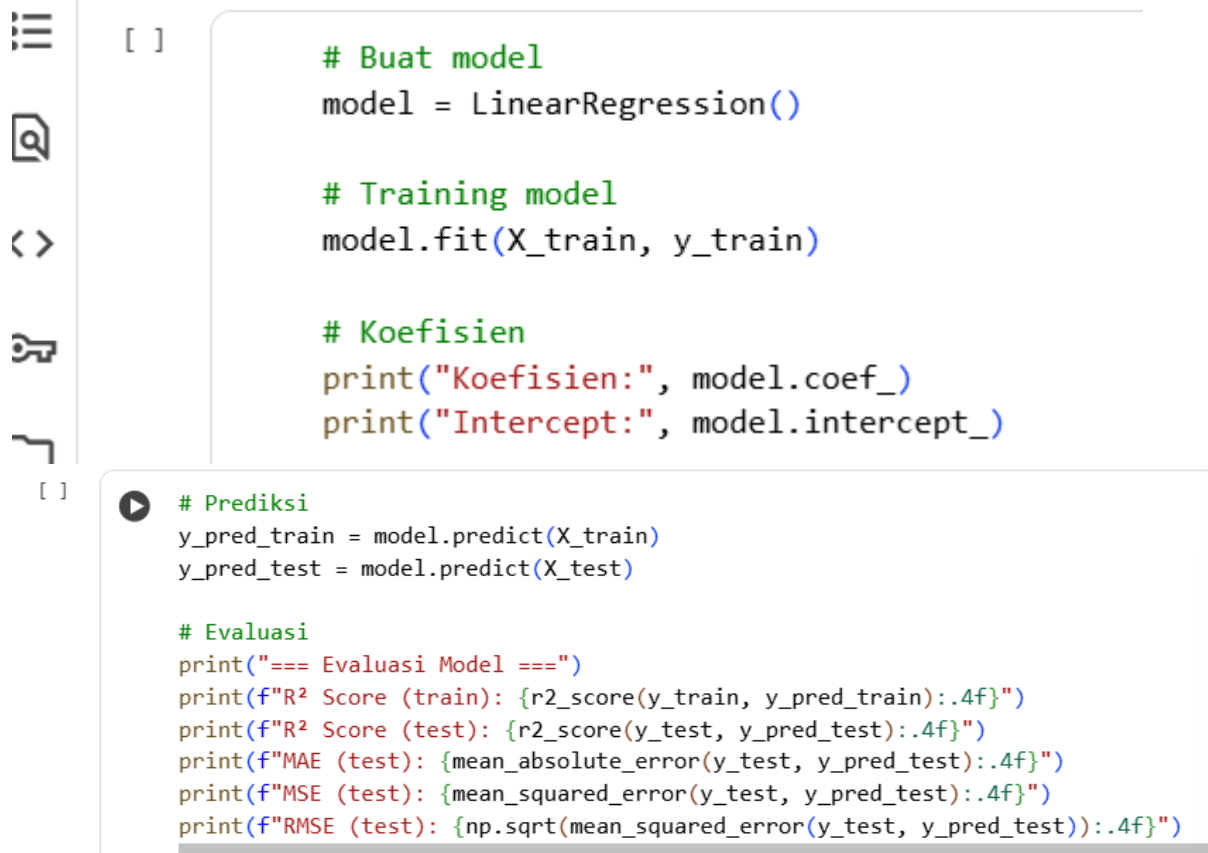
```
⇒ Shape X: (731, 11)
Shape y: (731,)
```

```
[ ]
# Split data 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("Jumlah data training:", len(X_train))
print("Jumlah data testing:", len(X_test))
```

- Memilih variabel independen (X) berdasarkan analisis korelasi (contoh: 'temp', 'atemp', 'hum', dll.).
- Variabel dependen (y) adalah 'cnt'.
- Membagi data menjadi training (80%) dan testing (20%) menggunakan `train_test_split` dari Scikit-learn dengan `random_state=42` untuk reproduktibilitas.
- Menampilkan ukuran data training dan testing untuk verifikasi

1.7 Pembuatan Model Regresi Linear Menggunakan Scikit-learn



```
[ ]  
  
# Buat model  
model = LinearRegression()  
  
# Training model  
model.fit(X_train, y_train)  
  
# Koefisien  
print("Koefisien:", model.coef_)  
print("Intercept:", model.intercept_)  
  
[ ]  
# Prediksi  
y_pred_train = model.predict(X_train)  
y_pred_test = model.predict(X_test)  
  
# Evaluasi  
print("=== Evaluasi Model ===")  
print(f"R2 Score (train): {r2_score(y_train, y_pred_train):.4f}")  
print(f"R2 Score (test): {r2_score(y_test, y_pred_test):.4f}")  
print(f"MAE (test): {mean_absolute_error(y_test, y_pred_test):.4f}")  
print(f"MSE (test): {mean_squared_error(y_test, y_pred_test):.4f}")  
print(f"RMSE (test): {np.sqrt(mean_squared_error(y_test, y_pred_test)):.4f}")
```

- Membuat objek model LinearRegression dari Scikit-learn.
- Melatih model menggunakan data training dengan .fit().
- Menampilkan koefisien regresi dan intercept.
- Melakukan prediksi pada data training dan testing.
- Mengevaluasi model dengan metrik R² score, MAE, MSE, dan RMSE.

1.8 Visualisasi Hasil Prediksi (Scikit-learn)

[]

```
# Scatter plot actual vs predicted
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred_test, alpha=0.5, color='blue', label='Data Test')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
         'r--', lw=2, label='Perfect Prediction')
plt.xlabel('Actual cnt')
plt.ylabel('Predicted cnt')
plt.title('Actual vs Predicted - Bike Sharing')
plt.legend()
plt.grid(True)
plt.show()

# Residual plot
residuals = y_test - y_pred_test
plt.figure(figsize=(10, 6))
plt.scatter(y_pred_test, residuals, alpha=0.5)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.grid(True)
plt.show()
```

[]

```
# Buat dataframe hasil
hasil = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted': y_pred_test,
    'Error': y_test.values - y_pred_test
})

print(hasil.head(10))

# Statistik error
print("\n=== Statistik Error ===")
print(f"Mean Error: {hasil['Error'].mean():.4f}")
print(f"Std Error: {hasil['Error'].std():.4f}")
```

- Membuat scatter plot actual vs predicted untuk data testing.
- Membuat residual plot untuk memeriksa distribusi error.
- Membuat DataFrame hasil prediksi dengan kolom actual, predicted, dan error untuk sampel verifikasi.

1.9 Pembuatan Model Regresi Linear Menggunakan Statsmodels

```
# Pilih variabel independent dengan korelasi tinggi
# Misalnya: temp, atemp, hum, yr, season
X = df[['temp', 'atemp', 'hum', 'windspeed', 'yr', 'season', 'weathersit']]
y = df['cnt']

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print(f"\nJumlah data training: {len(X_train)}")
print(f"Jumlah data testing: {len(X_test)}")

# METODE 1: Menggunakan Statsmodels untuk hasil detail OLS
print("\n" + "="*80)
print("METODE 1: STATSMODELS OLS REGRESSION")
print("="*80)

# Tambahkan konstanta untuk statsmodels
X_train_sm = sm.add_constant(X_train)
X_test_sm = sm.add_constant(X_test)

# Fit model
model_sm = sm.OLS(y_train, X_train_sm)
results = model_sm.fit()

# Tampilkan hasil lengkap
print(results.summary())
```

- Memilih variabel independen yang lebih spesifik berdasarkan korelasi.
- Menambahkan konstanta untuk model OLS.
- Melatih model menggunakan `sm.OLS` dan `.fit()`.
- Menampilkan `summary` hasil OLS yang detail, termasuk p-value dan statistik lainnya.

1.10 Pembuatan Model Regresi Linear Menggunakan Scikit-Learn (Versi Kedua) dan Evaluasi

```
# METODE 2: Menggunakan Sklearn (lebih sederhana)
print("\n" + "="*80)
print("METODE 2: SKLEARN LINEAR REGRESSION")
print("="*80)

model_sk = LinearRegression()
model_sk.fit(X_train, y_train)

# Prediksi
y_pred_train = model_sk.predict(X_train)
y_pred_test = model_sk.predict(X_test)

# Evaluasi
r2_train = r2_score(y_train, y_pred_train)
r2_test = r2_score(y_test, y_pred_test)
mae = mean_absolute_error(y_test, y_pred_test)
mse = mean_squared_error(y_test, y_pred_test)
rmse = np.sqrt(mse)

print(f"\nR2 Score (Training): {r2_train:.4f}")
print(f"R2 Score (Testing): {r2_test:.4f}")
print(f"MAE: {mae:.4f}")
print(f"MSE: {mse:.4f}")
print(f"RMSE: {rmse:.4f}")

print("\nKoefisien Regresi:")
print(f"Intercept: {model_sk.intercept_:.4f}")
for i, col in enumerate(X.columns):
    print(f" {col}: {model_sk.coef_[i]:.4f}")
```

- Membuat model LinearRegression kedua dengan variabel yang disesuaikan.
- Melatih dan memprediksi.
- Mengevaluasi dengan R², MAE, MSE, RMSE.
- Menampilkan koefisien regresi untuk interpretasi pengaruh variabel.

1.11 Visualisasi Hasil Lengkap

```
# VISUALISASI HASIL
fig, axes = plt.subplots(2, 2, figsize=(15, 12))

# Plot 1: Actual vs Predicted (Training)
axes[0, 0].scatter(y_train, y_pred_train, alpha=0.5, color='blue', s=20)
axes[0, 0].plot([y_train.min(), y_train.max()],
                [y_train.min(), y_train.max()],
                'r--', lw=2, label='Perfect Prediction')
axes[0, 0].set_xlabel('Actual cnt (Training)', fontsize=12)
axes[0, 0].set_ylabel('Predicted cnt', fontsize=12)
axes[0, 0].set_title(f'Training Data: Actual vs Predicted\nR2 = {r2_train:.4f}',
                    fontsize=12, fontweight='bold')
axes[0, 0].legend()
axes[0, 0].grid(True, alpha=0.3)

# Plot 2: Actual vs Predicted (Testing)
axes[0, 1].scatter(y_test, y_pred_test, alpha=0.5, color='green', s=20)
axes[0, 1].plot([y_test.min(), y_test.max()],
                [y_test.min(), y_test.max()],
                'r--', lw=2, label='Perfect Prediction')
axes[0, 1].set_xlabel('Actual cnt (Testing)', fontsize=12)
axes[0, 1].set_ylabel('Predicted cnt', fontsize=12)
axes[0, 1].set_title(f'Testing Data: Actual vs Predicted\nR2 = {r2_test:.4f}',
                    fontsize=12, fontweight='bold')
axes[0, 1].legend()
axes[0, 1].grid(True, alpha=0.3)

# Plot 3: Residual Plot
residuals_test = y_test - y_pred_test
axes[1, 0].scatter(y_pred_test, residuals_test, alpha=0.5, color='purple')
axes[1, 0].axhline(y=0, color='r', linestyle='--', lw=2)
axes[1, 0].set_xlabel('Predicted Values', fontsize=12)
axes[1, 0].set_ylabel('Residuals', fontsize=12)
axes[1, 0].set_title('Residual Plot (Testing Data)', fontsize=12, fontweight='bold')
axes[1, 0].grid(True, alpha=0.3)

# Plot 4: Distribution of Residuals
axes[1, 1].hist(residuals_test, bins=50, color='orange', edgecolor='black', alpha=
axes[1, 1].axvline(x=0, color='r', linestyle='--', lw=2)
axes[1, 1].set_xlabel('Residuals', fontsize=12)
axes[1, 1].set_ylabel('Frequency', fontsize=12)
axes[1, 1].set_title('Distribution of Residuals', fontsize=12, fontweight='bold')
axes[1, 1].grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('regression_results.png', dpi=300, bbox_inches='tight')
plt.show()
```



```

# VISUALISASI KOEFISIEN
plt.figure(figsize=(10, 6))
coef_df = pd.DataFrame({
    'Feature': X.columns,
    'Coefficient': model_sk.coef_
}).sort_values('Coefficient', ascending=True)

plt.barh(coef_df['Feature'], coef_df['Coefficient'], color='steelblue')
plt.xlabel('Coefficient Value', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.title('Feature Coefficients in Linear Regression Model',
         fontsize=14, fontweight='bold')
plt.axvline(x=0, color='red', linestyle='--', linewidth=1)
plt.grid(axis='x', alpha=0.3)
plt.tight_layout()
plt.savefig('coefficients.png', dpi=300, bbox_inches='tight')
plt.show()

```

- Membuat subplot untuk actual vs predicted (training dan testing), residual plot, dan distribusi residuals.
- Menyimpan gambar sebagai 'regression_results.png'.
- Membuat bar plot koefisien fitur untuk melihat pengaruh variabel.
- Menyimpan gambar sebagai 'coefficients.png'.

1.12 Menampilkan Hasil Prediksi dan Interpretasi Model

```

# TABEL HASIL PREDIKSI
hasil = pd.DataFrame({
    'Actual': y_test.values,
    'Predicted': y_pred_test,
    'Error': residuals_test,
    'Error_Pct': (residuals_test / y_test.values * 100)
})

print("\n" + "="*80)
print("SAMPLE HASIL PREDIKSI (10 data pertama):")
print("="*80)
print(hasil.head(10).to_string())

print("\n" + "="*80)
print("STATISTIK ERROR:")
print("="*80)
print(f"Mean Error: {hasil['Error'].mean():.4f}")
print(f"Std Error: {hasil['Error'].std():.4f}")
print(f"Min Error: {hasil['Error'].min():.4f}")
print(f"Max Error: {hasil['Error'].max():.4f}")
print(f"Mean Absolute Error: {hasil['Error'].abs().mean():.4f}")
print(f"Mean Error Percentage: {hasil['Error_Pct'].mean():.2f}%")

```

```

# INTERPRETASI MODEL
print("\n" + "="*80)
print("INTERPRETASI MODEL:")
print("="*80)
print(f"1. Model mampu menjelaskan {r2_test*100:.2f}% variasi dalam data cnt")
print(f"2. Rata-rata kesalahan prediksi (MAE): {mae:.2f} unit cnt")
print(f"3. Root Mean Squared Error (RMSE): {rmse:.2f} unit cnt")
print(f"4. Variabel dengan pengaruh terbesar:")
for i, col in enumerate(X.columns):
    print(f"    - {col}: {model_sk.coef_[i]:.4f}")

```

-
- Membuat DataFrame hasil prediksi dengan tambahan error percentage.
 - Menampilkan sampel 10 data pertama.
 - Menampilkan statistik error seperti mean, std, min, max, dan MAE.
 - Memberikan interpretasi model berdasarkan metrik evaluasi dan koefisien.

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., Kaliaana, 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>