TUGAS METODE NUMERIK

IMPLEMENTASI REGRESI

MODEL LINEAR DAN PANGKAT SEDERHANA

ZEKA EMO

21120122130075

2024/2025

Link GitHub: https://github.com/zekaemo/Implementasi-Regresi_Metode-Numerik

Mencari hubungan antara jumlah latihan soal (NL) terhadap nilai ujian (NL) pada dataset yang tersedia secara open-source di Kaggle dengan judul Student Performance.

Berikut adalah tampilan dari dataset Student Performance:

	Hours Studied	Previous Scores	Extracurricular Activities	Sleep Hours	Sample Question Papers Practiced	Performance Index
0	7	99	Yes	9	1	91.0
1	4	82	No	4	2	65.0
2	8	51	Yes	7	2	45.0
3	5	52	Yes	5	2	36.0
4	7	75	No	8	5	66.0
...	...	...	...	...	...	...
9995	1	49	Yes	4	2	23.0
9996	7	64	Yes	8	5	58.0
9997	6	83	Yes	8	5	74.0
9998	9	97	Yes	7	0	95.0
9999	7	74	No	8	1	64.0

10000 rows × 6 columns

Regresi Linear

Analisis regresi linear sederhana adalah analisis regresi linear yang hanya melibatkan dua variabel, yaitu satu variabel independen dan satu variabel dependen. Disebut linear sederhana karena variabel dependen diasumsikan berhubungan linear dalam parameter dan linear dengan variabel independen. Secara umum, model regresi linear sederhana dengan satu variabel independen dan fungsi linear dalam X dapat ditulis :

$$Y_i = \beta_0 + \beta_1 X_{1i} + \varepsilon_i$$

dengan :

Y_i merupakan nilai variabel independen observasi ke – i.

$\beta_i$merupakan parameter koefisien regresi.

$X_{\{ki\}}$ merupakan nilai variabel independen ke-k, observasi ke-i

ε_i merupakan nilai random error

Linearisasi Pangkat Sederhana

Linearisasi pangkat sederhana merupakan model regresi non-linear. Linearisasi pangkat sederhana adalah metode untuk mendekati fungsi non-linear (misalnya, ($y = x^n$)) dengan fungsi linear di sekitar titik tertentu. Ini dilakukan dengan mengambil turunan pertama fungsi pada titik tersebut untuk mendapatkan kemiringan (gradien), kemudian menggunakan persamaan garis lurus ($y = mx + c$) untuk menggantikan fungsi asli dalam rentang nilai yang sempit di sekitar titik tersebut. Linearisasi pangkat sederhana ini juga disebut sebagai **Power Law Model.**

I. Source Code

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

# Menyiapkan data
df = pd.read_csv('Student_Performance.csv')
# Buang kolom yang tidak penting
df = df.drop(["Hours Studied", "Previous Scores",
"Extracurricular Activities", "Sleep Hours"], axis=1)
```

```
# Mengubah kolom menjadi array
NL = df['Sample Question Papers Practiced'].values
NT = df['Performance Index'].values
x = NL.reshape(-1, 1)

# Pembagian data set
NL_train, NL_test, NT_train, NT_test = train_test_split(x, NT,
test_size=0.2, random_state=1)

# Inisialisasi Power Law Model class
class PowerLawModel:
    def __init__(self):
        self.params = None
        self.rms_error = None

    def fit(self, X, y):
        def power_law(x, a, b):
            return a * np.power(x, b)
        self.params, _ = curve_fit(power_law, X.flatten(), y,
p0=[1, 1])

    def predict(self, X):
        a, b = self.params
        return a * np.power(X.flatten(), b)

    def calculate_rms_error(self, y_true, y_pred):
        self.rms_error  =  np.sqrt(mean_squared_error(y_true,
y_pred))
        return self.rms_error

# Melatih model
power_model = PowerLawModel()
power_model.fit(NL_train, NT_train)

# Menguji model
hasil_uji_power = power_model.predict(NL_test)

# Perhitungan RMS error
```

```
test_rms_error   =   power_model.calculate_rms_error(NT_test,
hasil_uji_power)
test_rms_error

# Menguhitung skor R-kuadrat
r2_score_power = r2_score(NT_test, hasil_uji_power)
r2_score_power

# Melatih model regresi linear
linreg = LinearRegression()
linreg.fit(NL_train, NT_train)

# Menguji model regresi linear
hasil_uji_linreg = linreg.predict(NL_test)

# Perhitungan RMS error
test_rms_error_linear  =  np.sqrt(mean_squared_error(NT_test,
hasil_uji_linreg))

# Menguhitung skor R-kuadrat
test_r2_linear = r2_score(NT_test, hasil_uji_linreg)

# Visualisasi data
plt.figure(figsize=(16, 8))

# visualisasi data linear
plt.subplot(1, 2, 1)
x_garis=np.linspace(0, 9, 100)
x_test=x_garis.reshape(-1, 1)
y_garis= linreg.predict(x_test)
plt.scatter(NL, NT, color='black')
plt.plot(x_garis, y_garis, linewidth=2)
plt.ylabel('NT')
plt.xlabel('NL')
plt.title('Linear Regression')

# Visualisasi regresi pangkat sederhana
plt.subplot(1, 2, 2)
```

```
x_garis=np.linspace(0, 9, 100)
y_garis= power_model.predict(x_garis)
plt.scatter(NL, NT, color='black')
plt.plot(x_garis, y_garis, linewidth=2, color="red")

plt.ylabel('NT')
plt.xlabel('NL')
plt.title('Power Law Regression')

plt.suptitle('Linear and Power Law Regression Results',
fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

II. Alur Kode

1. Import library

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

- Library numpy digunakan untuk menjalankan operasi-operasi numerik
- Library pandas digunakan untuk menjalankan proses manipulasi data
- Library matplotlib.pyplot digunakan untuk visualisasi data dalam bentuk grafik.
- Library sklearn digunakan untuk menjalankan proses *statistical modelling*, seperti regresi dan perhitungan-perhitungan performa dari model statistik yang dijalankan.

2. Menyiapkan data

```
# Menyiapkan data
df = pd.read_csv('Student_Performance.csv')

# Buang kolom yang tidak penting
```

```
df = df.drop(["Hours Studied", "Previous Scores",
"Extracurricular Activities", "Sleep Hours"], axis=1)

# Mengubah kolom menjadi array
NL = df['Sample Question Papers Practiced'].values
NT = df['Performance Index'].values
x = NL.reshape(-1, 1)
```

- Mengupload data pada sesi
- Membuang kolom selain NL dan NT
- Mengubah bentuk kolom menjadi array melalui instruksi df[‘nama-kolom].values
- Mengubah dimensi pada NL dan menyimpannya pada variabel yang baru `NL.reshape(-1, 1)`.
- Mengubah nama kolom Sample Question Papers Practiced (jumlah latihan soal) menjadi NL
- Mengubah nama kolom Performance Index (nilai ujian siswa) menjadi NT

3. Pembagian dataset

```
NL_train, NL_test, NT_train, NT_test = train_test_split(x, NT,
test_size=0.2, random_state=1)
```

- Membagi data latih (train set) dan data uji (test set) dengan perbandingan 8:2.
- Pembagian ini digunakan untuk perhitungan performa model yang lebih akurat.

4. Inisialisasi class mode Power Law

```
# Inisialisasi class model Power Law
class PowerLawModel:
    def __init__(self):
        self.params = None
        self.rms_error = None

    def fit(self, X, y):
        def power_law(x, a, b):
            return a * np.power(x, b)
```

```
        self.params, _ = curve_fit(power_law, X.flatten(), y,
p0=[1, 1])

    def predict(self, X):
        a, b = self.params
        return a * np.power(X.flatten(), b)

    def calculate_rms_error(self, y_true, y_pred):
        self.rms_error = np.sqrt(mean_squared_error(y_true,
y_pred))
        return self.rms_error
```

- Inisialisasi class model Power Law dengan empat fungsi:
  - Fungsi __init__ sebagai penyimpan parameter pada class
  - Fungsi fit untuk melakukan pelatihan pada model dengan data latih (train set)
  - Fungsi predict untuk melakukan prediksi pada set data asing (data uji)
  - Fungsi calculate_rms_error sebagai fungsi untuk menghitung performa model dengan menggunakan perhitungan Root Mean Square (RMS) error.

5. Melatih model Power Law

```
power_model = PowerLawModel()
power_model.fit(NL_train, NT_train)
```

- Inisialisasi objek dari kelas `PowerLawModel` dengan variabel `power_model`.
- Melatih model power law dengan data latih yaitu `NL_train` dan `NT_train`

6. Menguji model Power Law

```
hasil_uji_power = power_model.predict(NL_test)
```

- Melakukan pengujian pada model Power Law menggunakan data jumlah latihan soal untuk melihat prediksi nilai ujian siswa.
- Hasil dari prediksi ini nantinya akan disimpan pada variabel `hasil_uji_power`

7. Perhitungan performa model Power Law

```
# Perhitungan RMS error pada model Power Law
test_rms_error     =     power_model.calculate_rms_error(NT_test,
hasil_uji_power)
test_rms_error

# Menguhitung skor R-kuadrat pada model Power Law
r2_score_power = r2_score(NT_test, hasil_uji_power)
r2_score_power
```

- Melakukan perhitungan pada performa model Power Law menggunakan dua perhitungan:
  - RMS error dengan memanggil fungsi pada class PowerLawModel. Perhitungan RMS error dilakukan dengan membandingkan nilai dari variabel NT pada set data uji (NT_test) dengan prediksi pada data uji (hasil_uji_power).
  - Mengihtung performa model Power Law dengan skor R kuadrat pada nilai dari variabel NT pada set data uji (NT_test) dengan prediksi pada data uji (hasil_uji_power).
- Hasil perhitungan model Power Law:
  - RMS error: 25.960431604432294
  - R-kuadrat: -0.8371257354195636

8. Melatih model Regresi Linear

```
linreg = LinearRegression()
linreg.fit(NL_train, NT_train)
```

- Inisialisasi objek `linreg` yang memanggil fungsi `LinearRegression` dari library Sklearn.
- Melakukan pelatihan pada model Regresi Linear dengan data latih.

9. Menguji model Regresi Linear

```
hasil_uji_linreg = linreg.predict(NL_test)
```

- Melakukan pengujian pada data latih menggunakan set data uji (`NL_test`) dan menyimpan hasilnya pada variabel `hasil_uji_linreg`.

10. Perhitungan performa model Regresi Linear

```
# Perhitungan RMS error pada model Regresi Linear
```

```
test_rms_error_linear    =    np.sqrt(mean_squared_error(NT_test,
hasil_uji_linreg))
# Menguhitung skor R-kuadrat pada model Regresi Linear
test_r2_linear = r2_score(NT_test, hasil_uji_linreg)
```

- Melakukan perhitungan pada performa model Regresi Linear menggunakan dua perhitungan:
  - RMS error dengan memanggil fungsi pada class PowerLawModel. Perhitungan RMS error dilakukan dengan membandingkan nilai dari variabel NT pada set data uji (`NT_test`) dengan prediksi pada data uji (`hasil_uji_linreg`).
  - Mengihtung performa model Power Law dengan skor R kuadrat pada nilai dari variabel NT pada set data uji (`NT_test`) dengan prediksi pada data uji (`hasil_uji_linreg`).
- Hasil perhitungan model Regresi Linear:
  - RMS error: 19.12713195288997
  - R-kuadrat: 19.12713195288997

11. Visualisasi data

```
plt.figure(figsize=(16, 8))

# Visualisasi data linear
plt.subplot(1, 2, 1)
x_garis=np.linspace(0, 9, 100)
x_test=x_garis.reshape(-1, 1)
y_garis= linreg.predict(x_test)
plt.scatter(NL, NT, color='black')
plt.plot(x_garis, y_garis, linewidth=2)
plt.ylabel('NT')
plt.xlabel('NL')
plt.title('Linear Regression')

# Visualisasi regresi pangkat sederhana
plt.subplot(1, 2, 2)
x_garis=np.linspace(0, 9, 100)
y_garis= power_model.predict(x_garis)
```

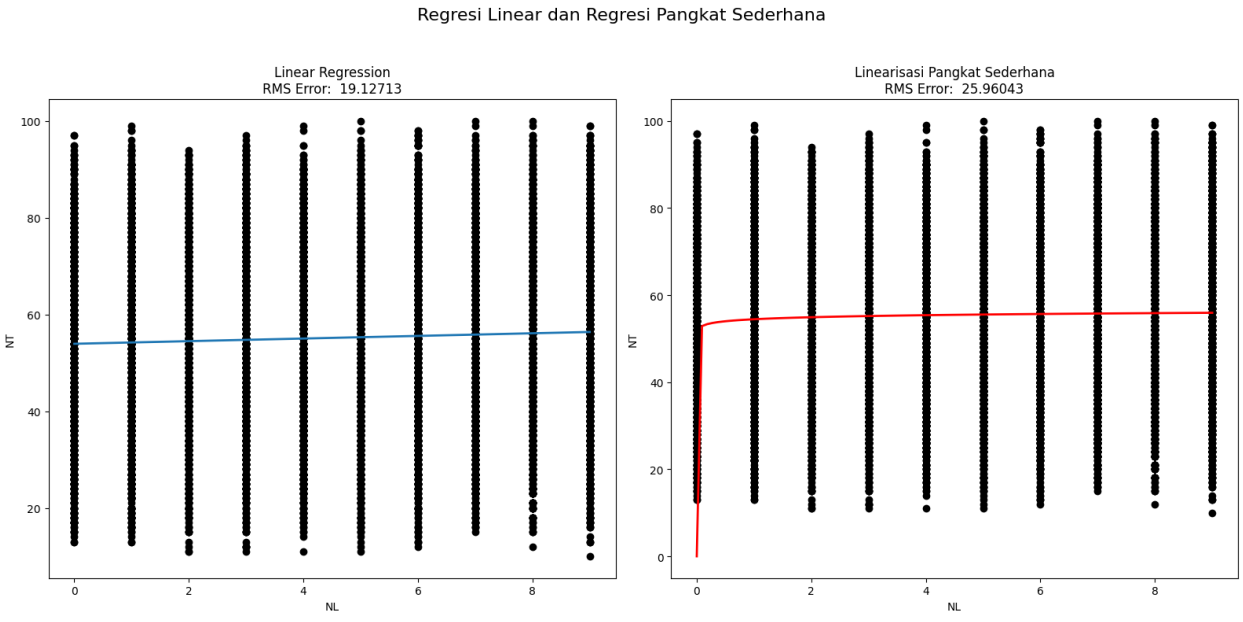
```
plt.scatter(NL, NT, color='black')
plt.plot(x_garis, y_garis, linewidth=2, color="red")

plt.ylabel('NT')
plt.xlabel('NL')
plt.title('Power Law Regression')

plt.suptitle('Linear and Power Law Regression Results',
fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

- Variabel x_garis pada kedua grafik berisi tiap titik pada data yang akan terregresi. Dengan instruksi ... , titik-titik akan terdistribusi secara merata dari 0 hingga 9 sebanyak 100 titik.
- Menghitung nilai y_garis pada masing-masing sebagai hasil regresi tiap model.
- Visualisasi titik-titik data (NL, NT) digunakan instruksi plt.scatter.
- Visualisasi garis regresi (x_garis, y_garis) digunakan dengan instruksi plt.plot.

III. Analisis Hasil



Pada proses pencarian hubungan antara kedua variabel, NL dan NT, dengan dua model berbeda. Dalam mengimplementasikan model **Regresi Linear** ditemukan bahwa nilai RMS error yang dihasilkan adalah 19, 13. Sedangkan pada model **Linearisasi Pangkat Sederhana**, didapatkan perhitungan RMS error senlai 25, 96. Kedua model ini dapat dikatakan sebagai model yang sangat buruk performanya bila dinilai dari RMS errornya. Sebuah model dapat dikatakan memiliki performa yang baik dalam mengahadapi sebuah dataset memiliki nilai RMS dibawah 0,5. Dengan begitu nilai RMS yang didapatkan dari kedua model dapat dikatakan sangat jauh dari nilai yang baik.

Hal ini dapat memunculkan satu-satunya kemungkinan, kedua model tidak bisa memproses dataset yang diberikan. Atau dengan kata lain, kedua model ini tidak cocok untuk digunakan dalam menghadapi dataset Student Performance. Beberapa hal yang dapat menjadi alasan dalam argumentasi ini merupakan nilai baseline pada dataset yang sama atau tidak jauh beda dengan nilai RMS error pada kedua model. Nilai baseline pada dataset Student Performance sebesar 19,15. Hal ini dapat memberi pengertian bahwa nilai yang diprediksikan oleh kedua model berbeda dengan nilai yang sebenarnya. Dan bila kita melihat kembali bentuk persebaran nilai NL terhadap NT, terdapat variasi yang signifikan terhadap nilai NT. Ini artinya berapapun nilai NL yang diuji, hasilnya sangat bervariasi dan model tidak cocok untuk memproses kemungkinan nilai prediksi yang sangat bervariasi tersebut.