Regresi Linear

Mohammad Febryan Khamim

1 Pengantar Regresi Linear

Linear Regression atau regresi linear adalah algoritma yang berupaya mengoptimasi fungsi linear untuk memprediksi *dataset* baru. Algoritma ini mengasumsikan terdapat hubungan linear antara input dan *output*. Hubungan linearnya direpresentasikan dengan garis lurus.

2 Garis / Fungsi Linear Terbaik Linear Regression

1. **Tujuan / Objektif Garis Linear:** Meminimalkan jarak antara titik nyata dan hasil prediksi. Garis tersebut membantu untuk memprediksi nilai yang belum ada.
   - θ_1: merepresentasikan *intercept* (nilai Y saat $x = 0$).
   - θ_2: merepresentasikan kemiringan (perubahan Y seiring perubahan X).
2. **Rumus Garis Terbaik:**
   $$y = mx + b$$
   - y: *predicted value*
   - x: *input*
   - m: *Gradien*
   - b: *Intercept*

   Garis terbaik berupaya mengoptimasi nilai m dan b agar *predicted* y akan semakin dekat dengan nilai seharusnya.
3. **Minimizing Error: Metode Least Square** Idenya adalah meminimalkan *sum of squared differences* antara x dan y.
   - *Residual*: $y_i - \hat{y}_i$
   - *Least square method* meminimalkan: $\sum(y_i - \hat{y}_i)^2$

Keterbatasan Regresi Linear

- Mengasumsikan linearitas (tak bekerja baik saat non-linear).
- Sensitif terhadap *outlier*.

Asumsi pada Regresi Linear

1. **Linearitas**
2. **Kemandirian Error:** Kesalahan prediksi satu data tak bergantung pada kesalahan data lain.
3. **Varians Konstan:** Varians error konstan.
4. **Normalitas Error** : Pola kesalahan prediksi mengikuti distribusi normal.
5. **No Multicolinearity** : Tiap variabel tak berkorelasi satu sama lain.
6. **Tak ada autokorelasi**
7. **Additivity**

3 Jenis-jenis Regresi Linear

Terdapat beberapa jenis Regresi Liner, yakni sebagai berikut.

1. **Simple Linear Regression**

   Simple LR digunakan untuk memprediksi nilai tujuan menggunakan satu fitur input.
   Persamaan:

$$\hat{y} = \theta_0 + \theta_1 x$$

2. **Multiple Linear Regression**

   Multiple LR melibatkan lebih dari 1 variabel independen dan 1 variabel dependen.
   Persamaan:

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

4 Mekanisme Pelatihan Regresi Linear

Berikut adalah proses iteratif dalam pelatihan regresi linear:

① **Menentukan Nilai Acak: Gradien dan Intercept**

Langkah pertama adalah menentukan nilai awal untuk parameter model.

$$\boxed{y = mx + b}$$

Ambil sembarang nilai acak, misalnya: $m = 0,5$ dan $b = 2$.

② **Menghitung Error antara Prediksi & Nilai Aktual**
Langkah ini biasanya menggunakan *Mean Squared Error* (MSE) untuk menilai seberapa besar kesalahan prediksi.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i^{pred} - y_i)^2$$

Dengan MSE, dapat diketahui seberapa akurat garis dalam memprediksi semua titik data.

③ **Mencari Kontribusi Setiap Parameter Terhadap Error**
Gunakan konsep turunan parsial untuk melihat sensitivitas error terhadap perubahan parameter m dan b:

$$\rightarrow \text{Gunakan konsep turunan: } \frac{\partial MSE}{\partial m} \text{ dan } \frac{\partial MSE}{\partial b}$$

- Turunan parsial menunjukkan seberapa besar m dan b harus diubah untuk meminimalkan MSE.
- Simbol $\oplus \rightarrow$ mengurangi parameter & $\ominus \rightarrow$ meningkatkan parameter (tergantung pada arah gradien).

④ **Update Parameter**
Perbarui nilai m dan b menggunakan nilai gradien yang telah dihitung.

$$m = m - \alpha\frac{\partial MSE}{\partial m}$$

$$b = b - \alpha\frac{\partial MSE}{\partial b}$$

dengan α adalah *learning rate* yang mengontrol ukuran langkah pembaruan.

⑤ **Iterasi**
Ulangi langkah 2 hingga 4 sampai mencapai nilai error paling minimum atau konvergen.

5 Langkah dalam Python

1. **Coba Exploratory Data Analysis (EDA)**

   Ketika melakukan EDA mengecek distribusi fitur target, diperoleh bahwa data terdistribusi normal $\rightarrow$ harga 1 jt – 1,5 jt jumlah banyak rumah semakin sedikit (menurun jumlahnya)

2. **Train Model**

   Pada modul sklearn sudah ada modul regresi linear

   $from\ sklearn.linear\_model\ import\ LinearRegression$

3. **Makna dari Koefisien dari Setiap Fitur**

   Anggap koefisiennya adalah β_1, maka diperoleh bahwa β_1 adalah laju perubahan rata-rata y terhadap x.

$$\beta_1 = \frac{\Delta y}{\Delta x}$$

   Jika x naik 1 satuan, maka nilai prediksi y berubah sebesar β_1 satuan dengan asumsi hubungan linear.

   > **Ingat konsep gradien** $y = mx + c$, sehingga $\beta_1 = m$ merupakan gradien atau kemiringan.

6 Program Python

Listing 1: Program Linear Regression Python

```
# Import modul atau libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

# Membaca data
df = pd.read_csv(r"E:\Perkuliahan\Karier\11-Linear-Regression\
   USA_Housing.csv")
df

# Menampilkan informasi data
df.head()
df.info()
df.describe()

# Exploratory Data Analysis

# Pairplot
sns.pairplot(df)

# Heatmap untuk cek korelasi data
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm'
   )
plt.tight_layout()

# PELATIHAN DATA
from sklearn.model_selection import train_test_split

# Pendefinisian X dan y
X = df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area
   Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
```

```
y = df['Price']

# Mekanisme pelatihan
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
    =0.4, random_state=42)

# Training model
from sklearn.linear_model import LinearRegression

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

# Mengecek parameter hasil pelatihan
print('Intercept:', lm.intercept_)
print('Coefficients:', lm.coef_)

cdf = pd.DataFrame(lm.coef_, X.columns, columns=['Coefficient']) #
    Untuk cek nama masing-masing kolom dan nilainya

# Mencari seberapa jauh hasil prediksi dengan data sebenarnya
prediksi = lm.predict(X_test)
plt.scatter(y_test, prediksi)

# Metrik evaluasi untuk Regresi Linear
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, prediksi))
print('MSE:', metrics.mean_squared_error(y_test, prediksi))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediksi)
    ))
```