# Simple Linear Regression

---

Di notebook ini, kita akan membuat model *machine learning* untuk kasus regresi (yang memiliki target berupa nilai kontinue) dengan algoritma **Linear Regression**. Disini kita akan mempraktekkan **Simple Linear Regression** yang berarti hanya melibatkan satu variabel bebas.

## Table of Contents



---

**Catatan:** Untuk menjalankan kode program Python di Jupyter Notebook, klik pada *cell* yang ingin di-*run* lalu tekan `Shift` + `Enter`.

**Warning!:** Jika ada kode program yang *error* atau output yang dihasilkan tidak sesuai, silahkan **Restart & Run All** kernel pada bagian menu **Kernel** di menu bar Jupyter Notebook, atau **Restart & Clear Output** kernel kemudian jalankan satu per satu *cell* secara berurutan dari atas ke bawah.

---

## Simple Linear Regression

**Simple linear regression** atau regresi linear sederhana merupakan jenis regresi yang paling sederhana karena hanya melibatkan satu variabel bebas atau variabel independen X. Pada dasarnya konsep regresi linear berasal dari persamaan garis :

$$y = \theta_0 + \theta_1 x_1$$

Untuk menghasilkan garis yang tepat dengan dengan error seminimal mungkin, kita harus menentukan nilai $\theta_0$ dan $\theta_1$ yang digunakan sebagai parameter. $\theta_0$ merupakan sebuah intersep (*intercept*), sedangkan $\theta_1$ merupakan gradien atau kemiringan garis. $\theta_0$ dan $\theta_1$ dapat disebut juga koefisien persamaan linear.

---

## Dataset

Dataset yang akan digunakan pada praktek kali ini adalah dataset diabetes yang sudah disediakan oleh Scikit-Learn. Untuk dapat menggunakannya, kita harus mengimpor *package* `load_diabetes` terlebih dahulu dari `sklearn.datasets`.

Mari kita *import library* yang dibutuhkan.

```
from sklearn.datasets import load_diabetes
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Selanjutnya untuk me-*load* dataset ke dalam dataframe Pandas, kita dapat langsung menggunakan `load_diabetes()`. Argumen `as_frame = True` berfungsi agar kita dapat menggunakan `['frame']` untuk menampilkan data + target dalam satu dataframe.

```
 ['data'] ---> menampilkan data fitur
 ['target'] ---> menampilkan target
 ['frame'] ---> menampilkan data + target
 ['DESCR'] ---> menampilkan deskripsi dataset
 ['feature_names'] ---> menampilkan list nama kolom dataset
```

```
# load dataset diabetes
data_diabetes = load_diabetes(as_frame=True)

# membuat dataframe
df = pd.DataFrame(data_diabetes['frame'])
df.head()
```

| | age | sex | bmi | bp | s1 | s2 | s3 | s4 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.038076 | 0.050680 | 0.061696 | 0.021872 | -0.044223 | -0.034821 | -0.043401 | -0.002592 | |
| **1** | -0.001882 | -0.044642 | -0.051474 | -0.026328 | -0.008449 | -0.019163 | 0.074412 | -0.039493 | |
| **2** | 0.085299 | 0.050680 | 0.044451 | -0.005670 | -0.045599 | -0.034194 | -0.032356 | -0.002592 | |
| **3** | -0.089063 | -0.044642 | -0.011595 | -0.036656 | 0.012191 | 0.024991 | -0.036038 | 0.034309 | |
| **4** | 0.005383 | -0.044642 | -0.036385 | 0.021872 | 0.003935 | 0.015596 | 0.008142 | -0.002592 | |

Kita dapat melihat deskripsi dataset dengan `['DESCR']`.

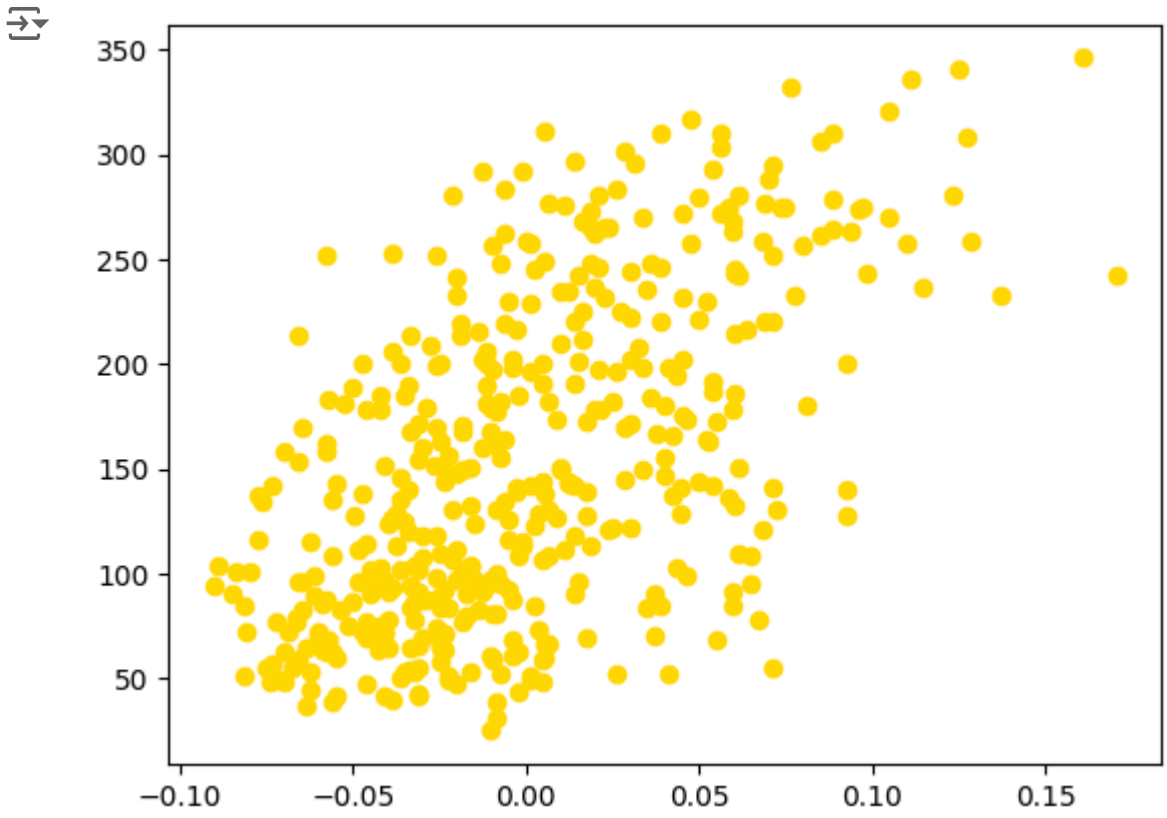
```
# Melihat deskripsi dataset diabetes

print(data_diabetes['DESCR'])
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values

  :Target: Column 11 is a quantitative measure of disease progression one year after

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, total serum cholesterol
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, total cholesterol / HDL
      - s5      ltg, possibly log of serum triglycerides level
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the st

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angl
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

## Analisis dan visualisasi data

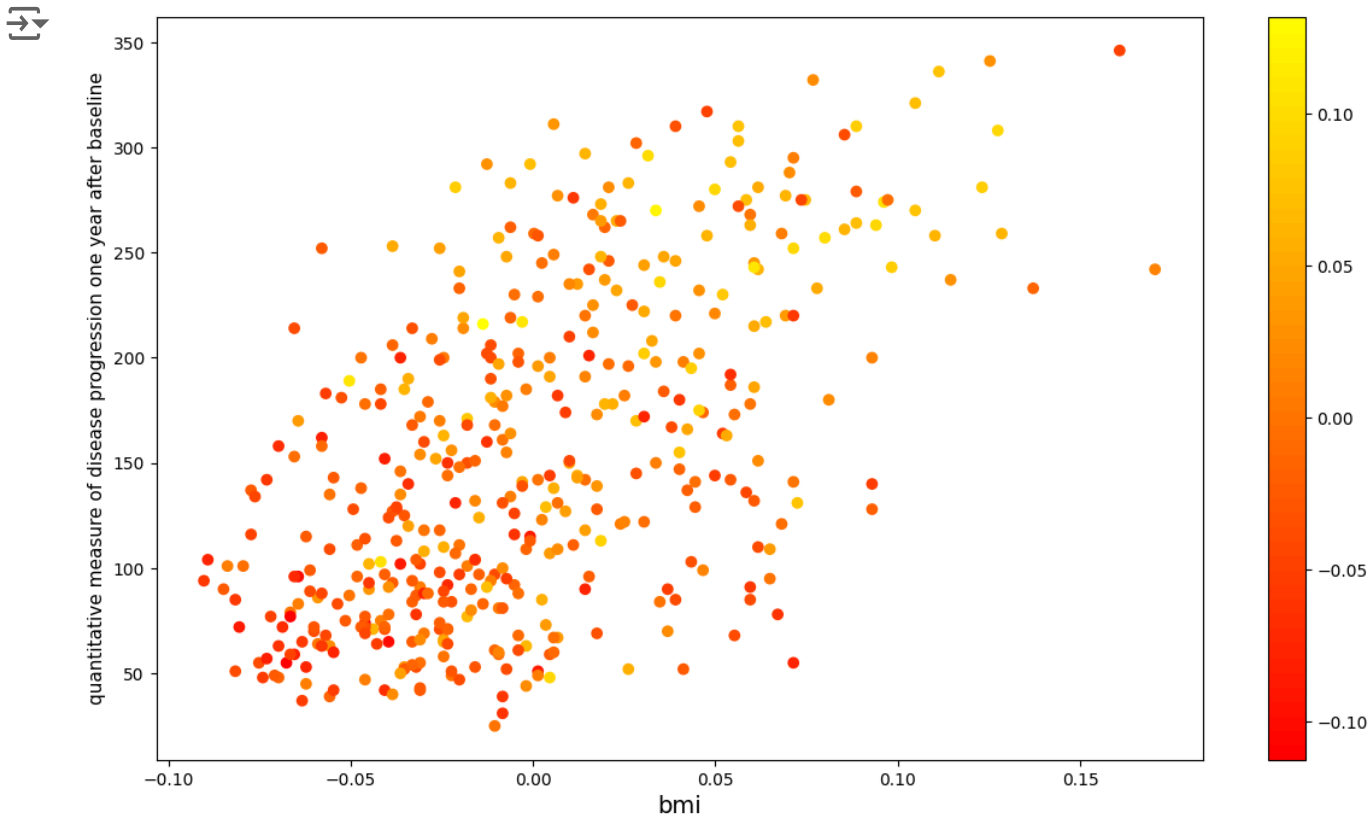
Mari kita lihat *scatter plot* antara kolom `bmi` dan `target`.

```
# Scatter plot kolom 'bmi' dan 'target'

plt.scatter(df['bmi'], df['target'], color='gold')
plt.show()
```




Kita juga dapat mengikutsertakan kolom `bp` dalam plot sebagai warnanya.

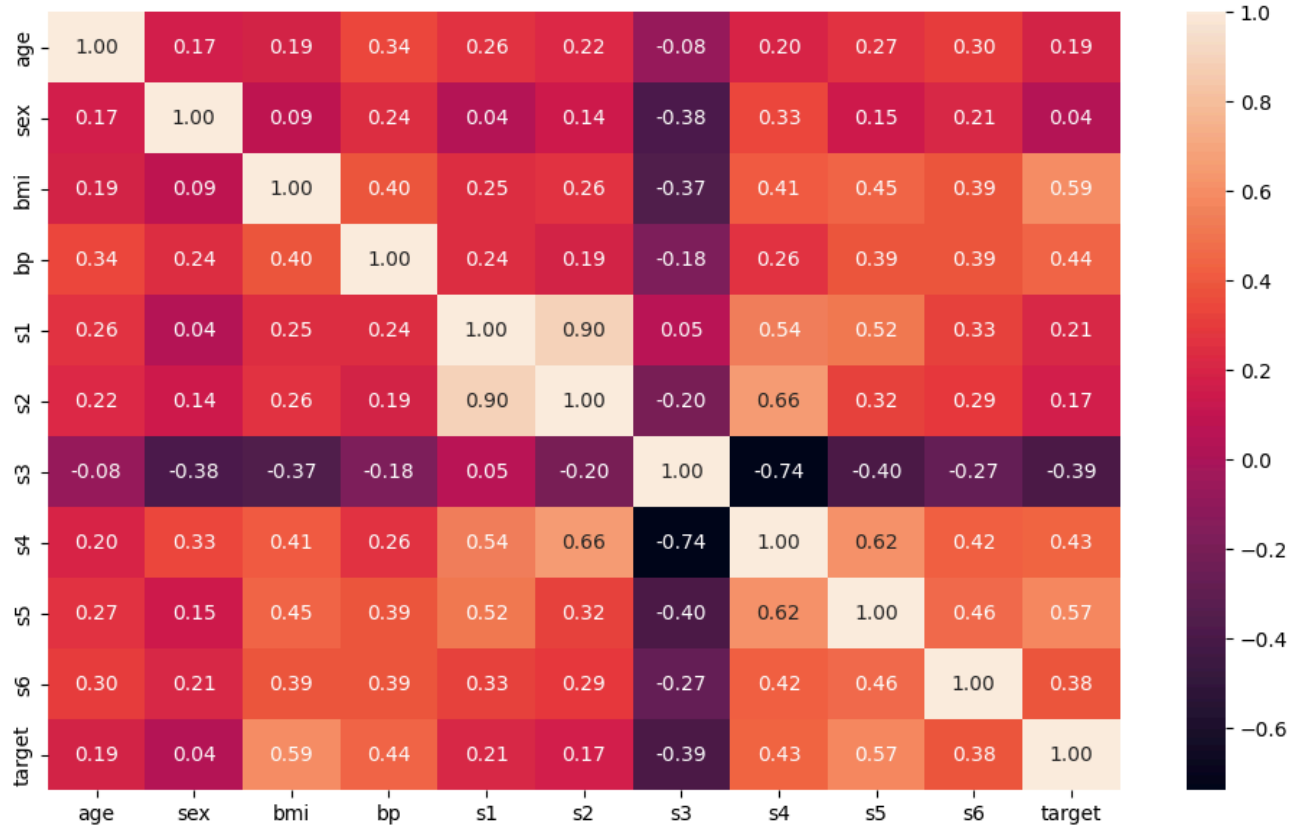
```
# Mengatur warna scatter plot dengan color map

fig, ax = plt.subplots(figsize=(14,8))
x = ax.scatter(df['bmi'], df['target'], c=df['bp'], cmap='autumn')

ax.set_xlabel('bmi', size=14)
ax.set_ylabel('quantitative measure of disease progression one year after baseline', size

# Menambakan color bar
fig.colorbar(x)

plt.show()
```




Sekarang mari kita lihat korelasi antarkolom dengan *heatmap*.

```
# Menampilkan korelasi antarkolom

plt.figure(figsize=(12,7))

corr = df.corr()
sns.heatmap(corr, annot=True, fmt='.2f')
```

<Axes: >




---

## Data Preparation

## Membagi data train dan test

Dataset yang sudah disediakan Scikit-Learn ini, datanya sudah dalam bentuk yang dinormalisasi sehingga kita di bagian *data preparation*, kita hanya perlu membagi data menjadi data *train* dan *test*.

```
# Membagi data train dan test

np.random.seed(42)
split = np.random.rand(len(df)) < 0.8
train = df[split]
test = df[~split]
```

Selanjutnya kita definisikan `X_train`, `y_train`, `X_test`, dan `y_test`.

Yang akan kita praktekkan sekarang adalah *simple linear regression* sehingga kita hanya menggunakan satu fitur. Disini kita akan ambil fitur `bmi`.

```
# Mendefinisikan X_train, y_train, X_test, dan y_test

X_train = np.asanyarray(train[['bmi']])
y_train = np.asanyarray(train[['target']])

X_test = np.asanyarray(test[['bmi']])
y_test = np.asanyarray(test[['target']])
```

---

# Modeling

## Linear regression dengan satu variabel bebas

Pemodelan *machine learning* untuk kasus regresi yang paling sederhana dapat menggunakan `LinearRegression()`.

Mari kita membuat model *linear regression* dan melatihnya dengan `.fit()`.

```
from sklearn.linear_model import LinearRegression

# Membuat dan melatih model
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

Kita dapat menampilkan *coefficient* dengan atribut `coef_` dan *intercept* dengan atribut `intercept_`.

```
# Coefficient dan Intercept
print ('Coefficients: ', lr_model.coef_)
print ('Intercept: ', lr_model.intercept_)
```

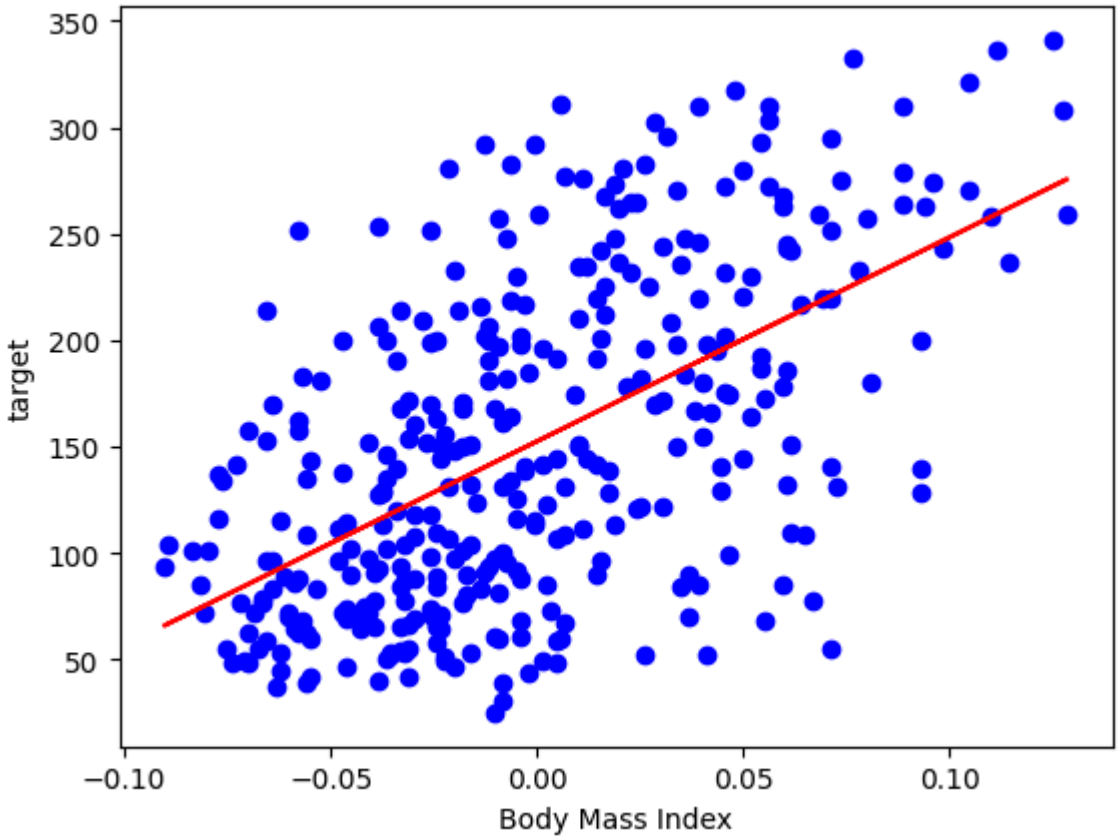
```
Coefficients:  [[958.28967126]]
Intercept:  [152.46395235]
```

## Visualisasi linear regression

Sekarang mari kita visualisasikan *linear regression* dengan *scatter plot*.

```
# Visualisasi dengan scatter plot

plt.scatter(X_train, y_train,  color='blue')
plt.plot(X_train, lr_model.coef_[0][0]*X_train + lr_model.intercept_[0], '-r')
plt.xlabel('Body Mass Index')
plt.ylabel('target')
```

```
Text(0, 0.5, 'target')
```




## Prediksi

Mari kita menguji model dengan menggunakan data `X_test`.

```
# Menguji model dengan X_test

y_pred = lr_model.predict(X_test)

print('Data asli: \n', y_test[0:10])
print('\n')
print('Hasil prediksi: \n', y_pred[0:10])
```

```
Data asli:
 [[ 75.]
 [ 63.]
 [ 69.]
 [179.]
 [ 87.]
 [ 65.]
 [102.]
 [ 92.]
 [155.]
 [ 59.]]


Hasil prediksi:
 [[103.13689113]
 [150.64827531]
 [169.23968652]
 [124.82687086]
 [104.1697473 ]
 [ 91.77547317]
 [122.76115851]
 [129.99115175]
 [145.48399442]
 [143.41828207]]
```

## Evaluasi

Selanjutnya evaluasi kinerja model dengan `mean_absolute_error` dan `mean_squared_error`.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Menampilkan MAE dan MSE
print('Mean Absolute Error (MAE): %.2f' % mean_absolute_error(y_pred, y_test))
print('Mean Squared Error (MSE): %.2f' % mean_squared_error(y_pred, y_test))
```

```
Mean Absolute Error (MAE): 51.84
Mean Squared Error (MSE): 3694.70
```

---