Machine Learning: Simple Linear Regression

Setelah memahami konsep regresi, langkah selanjutnya adalah membuat model ML untuk SLR (*simple linear regression*).

Pada contoh kali ini, kita ingin membuat sebuah model regresi, yaitu fungsi antara lamanya bekerja terhadap besarnya gaji yang didapat. Nantinya output model kita ini (prediksi gaji) akan kita bandingkan dengan gaji yang sebenarnya, sehingga dapat dilihat apakah model kita sudah cukup baik (prediksi sangat mendekati kenyataan) atau sebaliknya. Data dapat diunduh dari https://github.com/kungfumas/mesin-belajar/blob/master/Salary_Data.csv

Bahasa Python

```
# Impor library yang dibutuhkan
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Impor dataset
dataset = pd.read_csv('/kaggle/input/salary-data-simple-linear-regression/Salary_Data.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values

# Membagi data menjadi Training Set dan Test Set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)

# Fitting Simple Linear Regression terhadap Training set
from sklearn.linear_model import LinearRegression
```

```
19 regressor = LinearRegression()
20 regressor.fit(X_train, y_train)
21
22 # Memprediksi hasil Test Set
23 y_pred = regressor.predict(X_test)
24
25 # Visualisasi hasil Training Set
26 plt.scatter(X_train, y_train, color = 'red')
27 plt.plot(X_train, regressor.predict(X_train), color = 'blue')
28 plt.title('Gaji vs Pengalaman (Training set)')
29 plt.xlabel('Tahun bekerja')
30 plt.ylabel('Gaji')
31 plt.show()
32
33 # Visualisasi hasil Test Set
34 plt.scatter(X_test, y_test, color = 'red')
35 plt.plot(X_train, regressor.predict(X_train), color = 'blue')
36 plt.title('Gaji vs Pengalaman (Test set)')
37 plt.xlabel('Tahun bekerja')
   plt.ylabel('Gaji')
   plt.show()
```

Penjelasan:

- Line 2 sampai 4 adalah mengimpor library yang diperlukan.
- Line 7 mengimpor data ke python.
- Line 8 menentukan variabel independen (sumbu x) yaitu kolom ke 1 (Tahun_bekerja). Perlu diperhatikan, ketika melakukan slicing kita menggunakan method *.values,* hal ini hanya akan memotong data tanpa headernya. Jika tanpa *.values* maka header dari slicing akan diikutkan.
- Line 9 menentukan variabel dependen (xumbu y) yaitu kolom ke 2 (Gaji).
- Line 12 mengimpor library untuk memisahkan menjadi test dan train set.
- Line 13 membagi menjadi test dan train set, di mana train set adalah 2/3 dari dataset yang ada. Anda bisa klik bagian *Variable explorer* di spyder untuk melihat hasil pembagian train dan test set nya.
- Line 16 mengimpor class LinearRegression dari library sklearn.linear_model yang diperlukan untuk membuat model regresi.

- Line 17 membuat objek regressor sebagai fungsi dari LinearRegression. Cukup menulis LinearRegression() maka model regresi sudah disiapkan. Jika Anda arahkan kursor pada LinearRegression dan klik CTRL+i maka akan menampilkan object inspector untuk LinearRegression. Di sini Anda bisa melihat parameter apa saja yang diperlukan.
- Line 18 membuat model regresi untuk train set dengan menuliskan regressor.fit(X_train, y_train). Untuk melihat parameter apa saja yang diperlukan untuk metode fit(), cukup arahkan kursor pada .fit() kemudian klik CTRL+i.

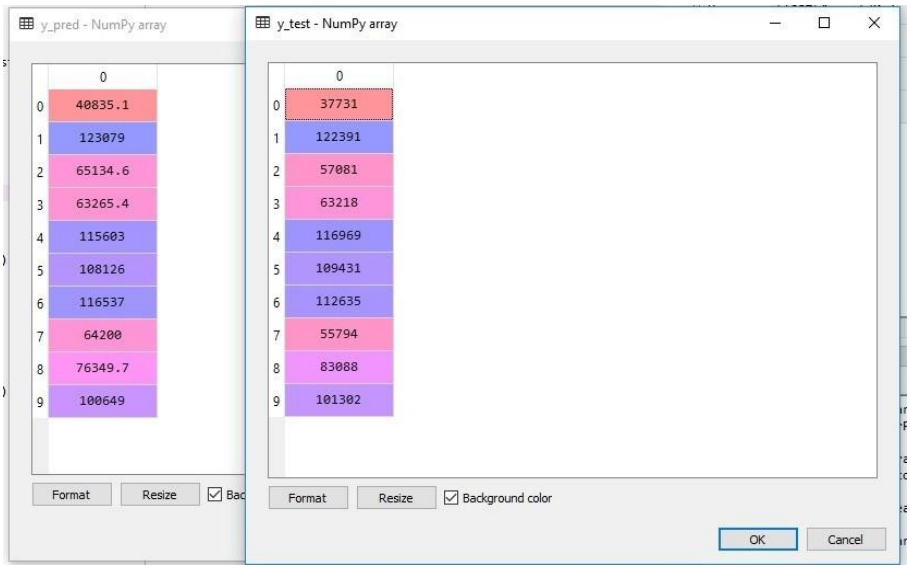
Line ke 18 adalah proses di mana kita membuat model machine learning regresi. Artinya, model kita sedang belajar untuk mencari hubungan antara X_train dan y_train.

Output di python akan tampak sebagai berikut Out[12]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False).

Setelah machine learning kita belajar mencari tahu hubungan antara X_train dan y_train, maka langkah selanjutnya adalah kita mencoba membuat prediksi ke depan, di mana prediksi ini menggunakan hubungan yang sudah dipelajari oleh model kita. Perintah untuk melakukan prediksi ini dilakukan di line 21.

- Line 21 membuat prediksi dengan menggunakan metode .predict. Untuk menegtahui parameter apa saja, kita bisa melihatnya melalui object inspector. Parameter yang diperlukan adalah variabel independen, dalam hal ini adalah X_test dan bukan X_train. Mengapa demikian? Karena kita ingin memprediksi data baru. Jika menggunakan X_train maka kita membuat prediksi berdasarkan pemahaman X_train, padahal pemahaman itu sendiri dibuat berdasarkan X_train. Oleh karena itu, kita menggunakan X_test. Nantinya prediksi ini kita bandingkan dengan y_test. Jika hasilnya mendekati (jaraknya tidak terlalu jauh), maka model kita sudah baik.

Jika sudah, mari kita bandingkan hasil y_pred dengan y_test, di mana y_pred adalah prediksi model kita dan y_test adalah data yang sesungguhnya.

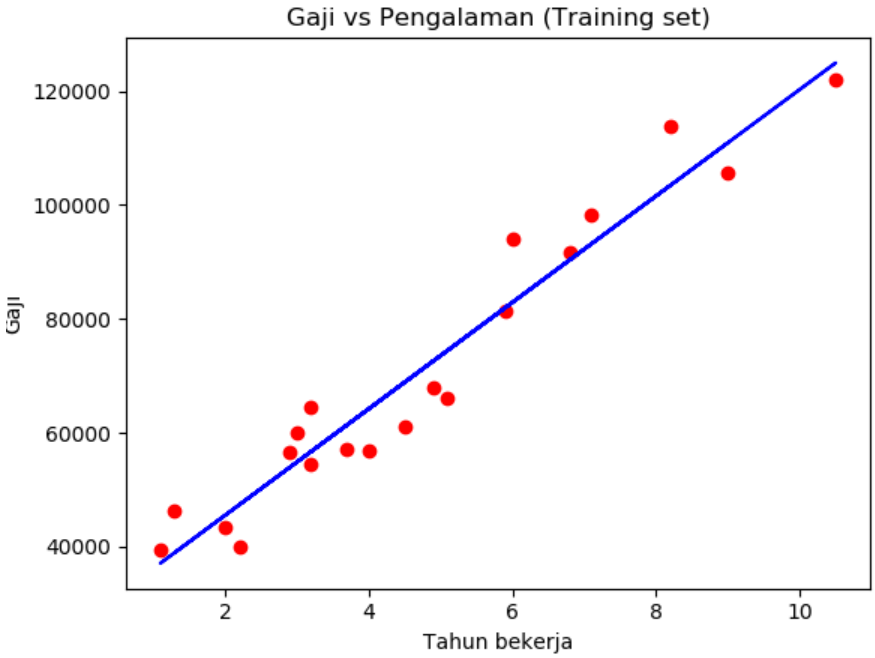


Hasil y_pred dan y_test

Untuk bisa membandingkan hasil antara y_pred dan y_test, akan sangat baik jika dibuat ilustrasi visualnya. Perlu diperhatikan, agar menampilkan visual yang baik di python, maka block semua line dari line 24 sampai line 29 kemudian klik CTRL+ENTER.

- Line 24 membuat scatter plot X_train dan y_train dengan warna merah untuk data poinnya. Tentu saja warnanya bisa dirubah sesuka hati.
- Line 25 membuat line plot nya (garis regresi) dengan warna biru. metode plot() membutuhkan parameter pertama yaitu data poin untuk sumbu x, dan parameter kedua adalah data poin untuk sumbu y. Data poin sumbu x adalah X_train karena kita ingin melihat model regresi dari training set, sementara data poin sumbu y adalah prediksi dari X_train dengan perintah regressor.predict(X_train). Perlu diingat, data poin sumbu y bukan y_predict atau regressor.predict(X_test), karena tujuan kita kali ini membuat plot regresi untuk X_train, bukan X_test. Semisal Anda salah, menggunakan regressor.predict(X_test), maka garis regresinya tidak akan muncul. Mengapa? Karena jumlah data poin X_train (20 baris) dan X_test (10 baris) sudah berbeda, maka tidak mungkin bisa dibuat garis regresinya.
- Line 26 membuat judul yang akan ditampilkan di bagian paling atas grafik.
- Line 27 membuat label untuk sumbu x.
- Line 28 membuat label untuk sumbu y.
- Line 29 mengeksekusi dan menampilkan hasil dari semua perintah dari line 24 sampai line 28.

Jika sudah, maka hasilnya akan nampak sebagai berikut:

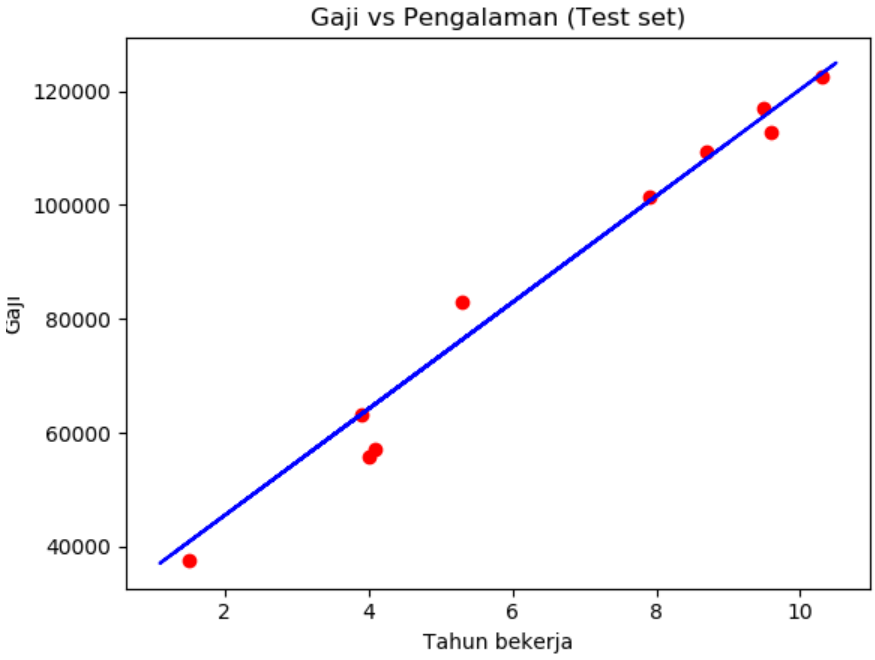


Ilustrasi simple regression

Titik merah adalah nilai gaji sesungguhnya, sementara garis biru adalah prediksi gaji dari model regresi kita.

- Line 32 sampai line 37 adalah perintah menampilkan grafik untuk test set. Perhatikan bahwa di line 33 plot line nya masih menggunakan X_train dan bukan X_test, karena model regresi kita berbasis pada training set. Walau demikian, sangat kebetulan sekali seandainya Anda salah menggunakan perintah ini misalnya: plt.plot(X_test, regressor.predict(X_test), color = 'blue'), maka hasilnya akan sama. Namun, saya ulang kembali Anda tetap perlu mengingat bahwa model regresinya adalah menggunakan X_train dan bukan X_test.

Jika sudah, maka hasil dari test set akan seperti berikut:



Pastilah tidak mungkin model regresi memiliki nilai 1, karena jika demikian maka model kita terlalu sempurna (ada yang salah, atau tidak beres). Model kita memiliki nilai 0.96 tentunya modelnya sangatlah baik (karena ini hanya data contoh untuk mempermudah).

Jika Anda melakukan pendekatan regresi, kemudian nilai sangat buruk, misal kurang dari 0.5 atau mendekati nol, maka bisa dipastikan hubungan variabel dependen dan independen tidaklah linear. Jika demikian, Anda bisa melakukan *fitting* (melihat hubungan variabel dependen dengan independen) dengan model regresi polinomial, atau non linear.

Simple Linear Regression Untuk Data Gaji

Draft saved

File Edit View Run Add-ons Help

Share

Run All

Draft Session (27m)

[1]:
```
# import all the lib
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
# read the dataset using pandas
data = pd.read_csv('/kaggle/input/salary-data-simple-linear-regression/Salary_Data.csv')
```

[3]:
```
# This displays the top 5 rows of the data
data.head()
```

Out[3]:

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

+ Code + Markdown

Simple Linear Regression Untuk Data Gaji

Draft saved

File Edit View Run Add-ons Help

Run All

Draft Session (28m)

[4]:
```
# Provides some information regarding the columns in the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
YearsExperience    30 non-null float64
Salary             30 non-null float64
dtypes: float64(2)
memory usage: 608.0 bytes
```

[ ]:
```
# this describes the basic stat behind the dataset used
data.describe()
```

[5]:
```
# These Plots help to explain the values and how they are scattered

plt.figure(figsize=(12,6))
sns.pairplot(data,x_vars=['YearsExperience'],y_vars=['Salary'],size=7,kind='scatter')
plt.xlabel('Years')
plt.ylabel('Salary')
plt.title('Salary Prediction')
plt.show()
```

```
/opt/conda/lib/python3.6/site-packages/seaborn/axisgrid.py:2079: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
```

Simple Linear Regression Untuk Data Gaji Draft saved

File Edit View Run Add-ons Help

Run All Draft Session (29m)

[6]:
```
# Cooking the data
X = data['YearsExperience']
X.head()
```

Out[6]:
```
0    1.1
1    1.3
2    1.5
3    2.0
4    2.2
Name: YearsExperience, dtype: float64
```

[7]:
```
# Cooking the data
y = data['Salary']
y.head()
```

Out[7]:
```
0    39343.0
1    46205.0
2    37731.0
3    43525.0
4    39891.0
Name: Salary, dtype: float64
```

Simple Linear Regression Untuk Data Gaji Draft saved

File Edit View Run Add-ons Help

Run All Draft S

[9]:
```
# Import Segregating data from scikit learn
from sklearn.model_selection import train_test_split
```

[20]:
```
# Split the data for train and test
X_train,X_test,y_train,y_test = train_test_split(X,y,train_size=0.7,random_state=0)
```

[21]:
```
# Create new axis for x column
X_train = X_train[:,np.newaxis]
X_test = X_test[:,np.newaxis]
```

[22]:
```
# Importing Linear Regression model from scikit learn
from sklearn.linear_model import LinearRegression
```

[23]:
```
# Fitting the model
lr = LinearRegression()
lr.fit(X_train,y_train)
```

Simple Linear Regression Untuk Data Gaji Draft saved

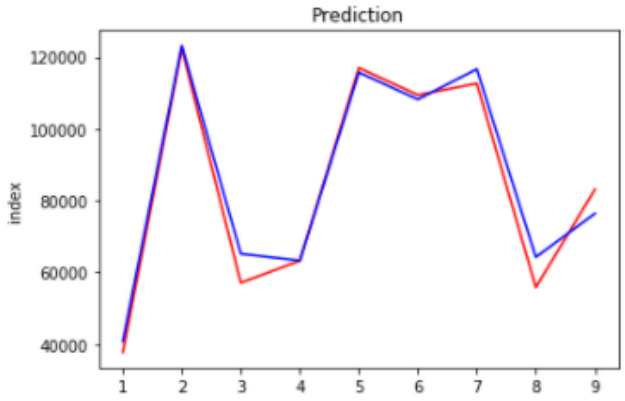
File Edit View Run Add-ons Help

Run All

[24]:
```
# Predicting the Salary for the Test values
y_pred = lr.predict(X_test)
```

[25]:
```
# Plotting the actual and predicted values

c = [i for i in range (1,len(y_test)+1,1)]
plt.plot(c,y_test,color='r',linestyle='-')
plt.plot(c,y_pred,color='b',linestyle='-')
plt.xlabel('Salary')
plt.ylabel('index')
plt.title('Prediction')
plt.show()
```

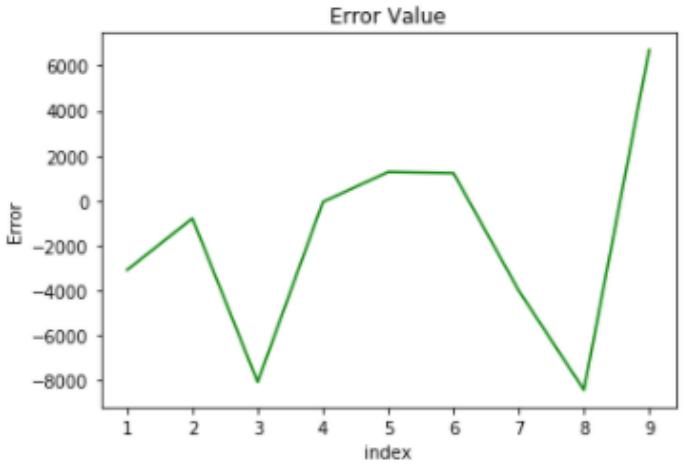


Simple Linear Regression Untuk Data Gaji Draft saved

File Edit View Run Add-ons Help

Run All

[26]:
```
# plotting the error
c = [i for i in range(1,len(y_test)+1,1)]
plt.plot(c,y_test-y_pred,color='green',linestyle='-')
plt.xlabel('index')
plt.ylabel('Error')
plt.title('Error Value')
plt.show()
```



[27]:
```
# Importing metrics for the evaluation of the model
from sklearn.metrics import r2_score,mean_squared_error
```

Simple Linear Regression Untuk Data Gaji Draft saved

File Edit View Run Add-ons Help

Run All

[28]:
```
# calculate Mean square error
mse = mean_squared_error(y_test,y_pred)
```

[29]:
```
# Calculate R square vale
rsq = r2_score(y_test,y_pred)
```

[30]:
```
print('mean squared error :',mse)
print('r square :',rsq)
```

```
mean squared error : 23370078.800832972
r square : 0.9740993407213511
```

[31]:
```
# Just plot actual and predicted values for more insights
plt.figure(figsize=(12,6))
plt.scatter(y_test,y_pred,color='r',linestyle='-')
plt.show()
```

[32]:
```
# Intecept and coeff of the line
print('Intercept of the model:',lr.intercept_)
print('Coefficient of the line:',lr.coef_)
```

```
Intercept of the model: 26777.391341197625
Coefficient of the line: [9360.26128619]
```

Then it is said to form a line with

$$y = 26777.391 + 9360.26x$$