

## Hands-On

Hands-On ini digunakan pada kegiatan Microcredential Associate Data Scientist 2021

### Tugas Mandiri Pertemuan 10

Pertemuan 10 (sepuluh) pada Microcredential Associate Data Scientist 2021 menyampaikan materi mengenai Membangun Model (Dasar Regresi dan Regresi Linier). silakan Anda kerjakan Latihan 1 s/d 20. Output yang anda lihat merupakan panduan yang dapat Anda ikuti dalam penulisan code :)

#### Latihan (1)

Melakukan import library yang dibutuhkan

```
In [1]: 1 # import library pandas
2 import pandas as pd
3
4 # Import Library numpy
5 import numpy as np
6
7 # Import Library matplotlib dan seaborn untuk visualisasi
8 import matplotlib.pyplot as plt
9 import seaborn as sns
10
11 # Import Module LinearRegression digunakan untuk memanggil algoritma Linear Regression.
12 from sklearn import linear_model
13
14 # Import Module train_test_split digunakan untuk membagi data kita menjadi training dan testing set.
15 from sklearn.model_selection import train_test_split
16
17 # import modul mean_absolute_error dari Library sklearn
18 from sklearn.metrics import mean_absolute_error
19
20 #import math agar program dapat menggunakan semua fungsi yang ada pada modul math.(ex:sqrt)
21 import math
22
23 # me-non aktifkan peringatan pada python
24 import warnings
25 warnings.filterwarnings('ignore')
```

Load Dataset

```
In [2]: 1 #Panggil file (Load file bernama CarPrice_Assignment.csv) dan simpan dalam dataframe Lalu tampilkan 10 baris awal dataset dengan head(10)
2 data = pd.read_csv('CarPrice_Assignment.csv')
3 dataset = pd.DataFrame(data=data)
4 dataset.head(10)
```

```
Out[2]: car_ID symboling CarName fuelytype aspiration doornumber carbody drivewheel enginelocation wheelbase ... enginesize fuelsystem boreration st
0 1 3 alfa-romero giulia gas std two convertible rwd front 88.6 ... 130 mpfi 3.47
1 2 3 alfa-romero stelvio gas std two convertible rwd front 88.6 ... 130 mpfi 3.47
2 3 1 alfa-romero Quadrifoglio gas std two hatchback rwd front 94.5 ... 152 mpfi 2.68
3 4 2 audi 100 ls gas std four sedan fwd front 99.8 ... 109 mpfi 3.19
4 5 2 audi 100ls gas std four sedan 4wd front 99.4 ... 136 mpfi 3.19
5 6 2 audi fox gas std two sedan fwd front 99.8 ... 136 mpfi 3.19
6 7 1 audi 100ls gas std four sedan fwd front 105.8 ... 136 mpfi 3.19
7 8 1 audi 5000 gas std four wagon fwd front 105.8 ... 136 mpfi 3.19
8 9 1 audi 4000 gas turbo four sedan fwd front 105.8 ... 131 mpfi 3.13
9 10 0 audi 5000s (diesel) gas turbo two hatchback 4wd front 99.5 ... 131 mpfi 3.13
```

10 rows x 26 columns

#### Latihan (2)

Review Dataset

```
In [3]: 1 # melihat jumlah baris dan jumlah kolom (bentuk data) pada data df dengan fungsi .shape
2 dataset.shape
```

```
Out[3]: (205, 26)
```

Data kita mempunyai 26 kolom dengan 205 baris.

```
In [4]: 1 # Melihat Informasi Lebih detail mengenai struktur DataFrame dapat dilihat menggunakan fungsi info()
2 dataset.info()
```

	car_ID	symboling	CarName	fuelytype	aspiration	doornumber	carbody	drivewheel	enginelocation	wheelbase	...	enginesize	fuelsystem	boration	st
0	1	3	alfa-romero giulia	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
1	2	3	alfa-romero stelvio	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	
2	3	1	alfa-romero Quadrifoglio	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	
3	4	2	audi 100 ls	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	
4	5	2	audi 100ls	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	
5	6	2	audi fox	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	
6	7	1	audi 100ls	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	
7	8	1	audi 5000	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	
8	9	1	audi 4000	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	
9	10	0	audi 5000s (diesel)	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	

```

18 borgeratio    205 non-null   float64
19 stroke       205 non-null   float64
20 compressionratio 205 non-null   float64
21 horsepower     205 non-null   int64
22 peakrpm        205 non-null   int64
23 citympg        205 non-null   int64
24 highwaympg      205 non-null   int64
25 price          205 non-null   float64
dtypes: float64(8), int64(8), object(10)
memory usage: 41.8+ KB

```

```
In [5]: 1 # melihat statistik data untuk data numeric seperti count, mean, standard deviation, maximum, minimum, dan quartile.
2 data.describe()
```

```
Out[5]:
car_ID symboling wheelbase carlength carwidth carheight curbweight enginesize boreroatio stroke compressionratio horsepower
count 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000 205.000000
mean 103.000000 0.834146 98.756585 174.049268 65.907805 53.724878 2555.565854 126.907317 3.329756 3.255415 10.142537 104.117073
std 59.322565 1.245307 6.021776 12.337289 2.145204 2.443522 520.680204 41.642693 0.270844 0.313597 3.972040 39.544167
min 1.000000 -2.000000 86.600000 141.100000 60.300000 47.800000 1488.000000 61.000000 2.540000 2.070000 7.000000 48.000000
25% 52.000000 0.000000 94.500000 166.300000 64.100000 52.000000 2145.000000 97.000000 3.150000 3.110000 8.600000 70.000000
50% 103.000000 1.000000 97.000000 173.200000 65.500000 54.100000 2414.000000 120.000000 3.310000 3.290000 9.000000 95.000000
75% 154.000000 2.000000 102.400000 183.100000 66.900000 55.500000 2935.000000 141.000000 3.580000 3.410000 9.400000 116.000000
max 205.000000 3.000000 120.900000 208.100000 72.300000 59.800000 4068.000000 326.000000 3.940000 4.170000 23.000000 288.000000
```

```
In [6]: 1 # cek nilai yang hilang / missing values di dalam data
2 data.isnull().sum()
```

```
Out[6]:
car_ID          0
symboling       0
CarName         0
fueltype        0
aspiration      0
doornumber      0
carbody         0
driveWheel      0
engineLocation  0
wheelbase        0
carlength        0
carwidth         0
carheight        0
curbweight        0
engineType       0
cylinderNumber  0
engineSize        0
fuelSystem       0
boreratio        0
stroke           0
compressionRatio 0
horsepower       0
peakRpm          0
citympg          0
highwaympg       0
price            0
dtype: int64
```

Ternyata data kita tidak ada missing values.

Simple linear regression atau regresi linear sederhana merupakan jenis regresi yang paling sederhana karena hanya melibatkan satu variabel bebas atau variabel independen X.

## Visualisasi data untuk pemilihan fitur / variabel independen X

1. Variabel y atau variabel dependent adalah 'price'
2. Lakukan Visualisasi dalam penerapannya agar dapat terlihat jelas / mempermudah dalam membaca data tsb
3. Untuk dapat menentukan variabel X yaitu dapat melihat korelasi antar variabel dengan variabel y / kolom 'price'

### Latihan (3)

untuk dapat menentukan lebih detail / akurat dalam pemilihan fitur dapat dilihat dari hubungan korelasinya dengan function corr()

```
In [7]: 1 dataset.corr()
Out[7]:
car_ID symboling wheelbase carlength carwidth carheight curbweight enginesize boreroatio stroke compressionratio horsepower
car_ID 1.000000 -0.151621 0.129729 0.170636 0.052387 0.255960 0.071962 -0.033930 0.260054 -0.160824 0.150276 -0.015006
symboling -0.151621 1.000000 -0.531954 -0.357612 -0.232919 -0.541038 -0.227691 -0.105790 -0.130051 -0.008735 -0.178515 0.070873
wheelbase 0.129729 -0.531954 1.000000 0.874587 0.795144 0.589435 0.778388 0.569329 0.488750 0.160959 0.249788 0.353294
carlength 0.170636 -0.357612 0.874587 1.000000 0.841118 0.491028 0.877728 0.683360 0.605454 0.129533 0.158414 0.552623
carwidth 0.052387 -0.323919 0.795144 0.841118 1.000000 0.279210 0.887032 0.735433 0.559150 0.182942 0.181129 0.640732
carheight 0.255960 -0.541038 0.589435 0.491029 0.279210 1.000000 0.295572 0.067149 0.171071 -0.055307 0.261214 -0.108802
curbweight 0.071962 -0.227691 0.776386 0.877728 0.867032 0.295572 1.000000 0.850594 0.648480 0.168790 0.151362 0.750739
enginesize 0.033930 -0.105790 0.569329 0.683360 0.735433 0.067149 0.850594 1.000000 0.583774 0.203129 0.028971 0.809769
boreratio 0.260054 -0.130051 0.488750 0.606454 0.559150 0.171071 0.648480 0.583774 1.000000 -0.055909 0.005197 0.573877
stroke -0.160824 -0.008735 0.160959 0.129533 0.182942 -0.055307 0.168790 0.203129 -0.055909 1.000000 0.186110 0.080940
compressionratio 0.150276 -0.178515 0.249788 0.158414 0.181129 0.261214 0.151362 0.028971 0.005197 0.186110 1.000000 -0.204326
horsepower 0.070873 0.353294 0.552623 0.640732 -0.108802 0.750739 0.809769 0.573877 0.080940 -0.204326 1.000000
peakRpm 0.203789 0.273806 -0.360469 -0.287242 -0.220012 -0.320411 -0.262643 -0.244660 -0.254976 -0.067964 -0.435741 0.131073
citympg 0.015940 -0.035023 -0.470414 -0.670099 -0.642704 -0.048640 -0.757414 -0.653585 -0.504532 -0.042145 0.324701 -0.001456
highwaympg 0.011255 0.034606 -0.544082 -0.704662 -0.677218 -0.107358 -0.797485 -0.677470 -0.587012 -0.043931 0.265201 -0.770544
price -0.109093 -0.079978 0.577816 0.682920 0.759325 0.119336 0.835305 0.07874145 0.553173 0.079443 0.067984 0.808139
```

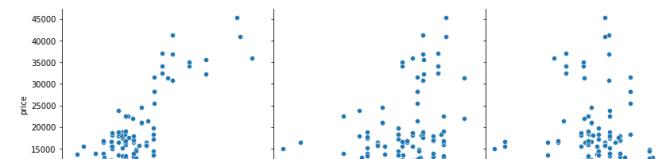
tampaknya enginesize, boreroatio, horsepower, wheelbase memiliki korelasi yang signifikan dengan harga/price.

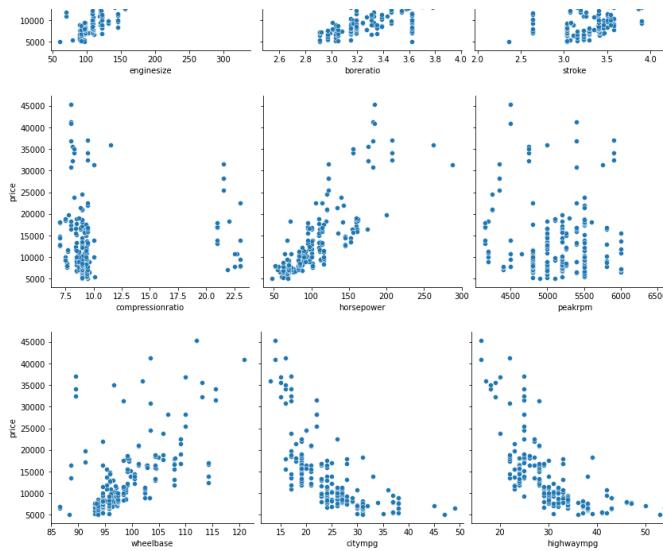
### Latihan (4)

Buat Visualisasi scatter plot dari kolom:

'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakRpm', 'wheelbase', 'citympg', 'highwaympg'

```
In [8]: 1 def pp(x,y,z):
2     sns.pairplot(dataset, x_vars=[x,y,z], y_vars='price',size=4, aspect=1, kind='scatter')
3     plt.show()
4
5 pp('enginesize', 'boreratio', 'stroke')
6 pp('compressionratio', 'horsepower', 'peakRpm')
7 pp('wheelbase', 'citympg', 'highwaympg')
```



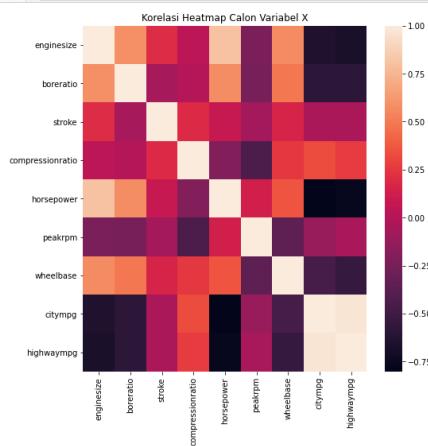


## Latihan (5)

Buat Visualisasi Heatmap dari kolom:

'enginesize', 'boreratio', 'stroke', 'compressionratio', 'horsepower', 'peakrpm', 'wheelbase', 'citympg', 'highwaympg'

```
In [9]: 1 plt.figure(figsize = (8,8))
2 data_fitur = dataset[['enginesize', 'boreratio', 'stroke','compressionratio', 'horsepower', 'peakrpm', 'wheelbase', 'citympg',
3 sns.heatmap(data_fitur.corr(),annot=False,fmt=".f").set_title("Korelasi Heatmap Calon Variabel X")
4 plt.show()
```



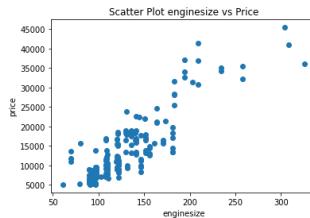
Dari hasil visualisasi diatas bahwa fitur/kolom enginesize memiliki korelasi yang tinggi terhadap kolom price / variabel dependent sehingga kita mengambil fitur/kolom enginesize untuk di training

- Independent variabel(x) adalah enginesize.
- Dependent variabel(y) adalah price.

## Latihan (6)

Buat Visualisasi Scatter Plot antara calon variabel X(engine size) dan y(price):

```
In [10]: 1 plt.scatter(dataset['enginesize'], dataset['price'])
2 plt.xlabel('enginesize')
3 plt.ylabel('price')
4 plt.title('Scatter Plot enginesize vs Price')
5 plt.show()
```



Scatter plot menunjukkan dengan jelas hubungan antarvariabel serta sebarannya di dataset. Selain itu, dengan scatter plot juga kita dapat mengindikasikan bahwa variabel enginesize dan price memiliki hubungan linear.

Catatan : korelasi 0.874145 adalah nilai yang cukup tinggi, artinya nilai price benar-benar sangat dipengaruhi oleh nilai i enginesize, karena korelasi tinggi maka algoritma Regresi Linier ini cocok digunakan untuk data tersebut.

## Latihan (7)

definisi variabel X(engine size) dan y(price):

```
In [11]: 1 # Prepare data
2 # Pertama, buat variabel x dan y.
3 x = dataset['enginesize'].values.reshape(-1,1)
4 y = dataset['price'].values.reshape(-1,1)
```

$$\hat{y} = \theta_0 + \theta_1 x_1$$

"Jika kita melihat formula regresi linear di atas, kita pasti ingat rumus persamaan garis yang pernah dipelajari di bangku sekolah, yaitu  $y = mx + c$ , dimana m merupakan gradien atau kemiringan garis dan c merupakan konstanta."

- from scratch
- $y = ax + b$  atau  $y = w_1x + w_0$  atau  $y = mx + c$
- $x$  = input
- $y$  = output
- $b$  atau  $w_0$  = intercept / bias
- $a$  atau  $w_1$  = slope / gradient / coefficient

## Latihan (8)

definisi variabel nilai mean/rata-rata X(engineysize) dan nilai mean/rata-rata y(price):

```
In [12]: 1 x_mean = np.mean(dataset['enginesize'])
2 y_mean = np.mean(dataset['price'])
3 print('nilai mean var x: ', x_mean, '\n')
4     'nilai mean var y: ', y_mean)

nilai mean var x: 126.90731707317073
nilai mean var y: 13276.710570731706
```

## Latihan (9)

carilah nilai koefisien korelasi nya dengan rumus dibawah:

$$\text{Correlation\_Coefficient} \quad r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

```
In [13]: 1 atas = sum((x - x_mean)*(y - y_mean))
2 bawah = math.sqrt((sum((x - x_mean)**2)) * (sum((y - y_mean)**2)))
3 correlation = atas/bawah
4 print('Nilai Correlation Coefficient: ', correlation)

Nilai Correlation Coefficient: [0.8741448]
```

carilah nilai parameter theta 1 dan theta 0 dengan rumus dibawah:

$$\theta_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

```
theta_1 = ((111-104.11) * (13495-13276.71)) + ... + ((114-104.11) * (22625-13276.71)) / ((111-104.11)^2 + ... + (114-104.11)^2)
```

## Latihan (10)

carilah nilai theta\_1 atau nilai slope

```
In [14]: 1 # slope
2 # Slope adalah tingkat kemiringan garis, intercept
3 # adalah jarak titik y pada garis dari titik 0
4 variance = sum((x - x_mean)**2)
5 covariance = sum((x - x_mean) * (y - y_mean))
6 theta_1 = covariance/variance
7 print('Nilai theta_1: ',theta_1)

Nilai theta_1: [167.69841639]
```

## Latihan (11)

carilah nilai theta\_0 atau nilai intercept

```
In [15]: 1 # intercept
2 theta_0 = y_mean - (theta_1 * x_mean)
3 print('Nilai theta_0: ',theta_0)

Nilai theta_0: [-8005.44553115]
```

Maka persamaan garis :

$$y = 167.69x - 8005.44$$

Jadi persamaan garis diatas dapat digunakan untuk melakukan prediksi apabila kita memiliki data enginesize yang baru, price dapat diperkirakan dengan rumus tersebut, masukkan nilai enginesize baru ke x, maka perkiraan nilai y (price) akan didapat.

## Latihan (12)

carilah nilai prediksi secara manual dan buatlah visualisasi scatter plot nya

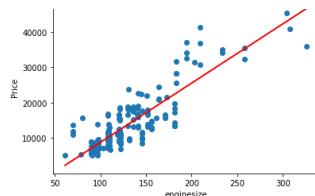
```
In [16]: 1 # prediction manual
2 y_pred = theta_0 + (theta_1 * 130)
3
4 print(y_pred)

[13795.34859997]
```

```
In [17]: 1 # visualisasi prediksi dengan scatter plot
2 y_pred = theta_0 + (theta_1 * x)
3
4 plt.scatter(dataset['enginesize'], dataset['price'])
5 plt.plot(x, y_pred, c='r')
6 plt.xlabel('enginesize')
7 plt.ylabel('Price')
8 plt.title('Plot enginesize vs Price')

Out[17]: Text(0.5, 1.0, 'Plot enginesize vs Price')
```

Plot enginesize vs Price



Linier Regression digunakan untuk Prediksi dengan mencari pola garis terbaik antara variable independent dan dependen

Pros:

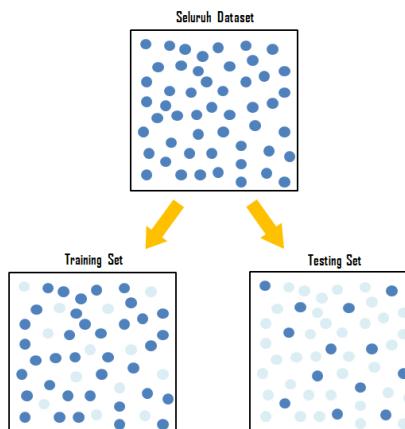
- Mudah diimplementasikan
- Digunakan untuk memprediksi nilai numerik/ continuous /data jenis interval dan ratio

Cons :

- Cenderung mudah Overfitting
- Tidak dapat digunakan bila relasi antara variabel independen dan dependen tidak linier atau korelasi variabel rendah

## Linier Regression dengan menggunakan library sklearn

1. Pertama yang kita lakukan adalah split data. Train/test split adalah salah satu metode yang dapat digunakan untuk mengevaluasi performa model machine learning. Metode evaluasi model ini membagi dataset menjadi dua bagian yakni bagian yang digunakan untuk training data dan untuk testing data dengan proporsi tertentu. Train data digunakan untuk fit model machine learning, sedangkan test data digunakan untuk mengevaluasi hasil fit model tersebut.



Python memiliki library yang dapat mengimplementasikan train/test split dengan mudah yaitu Scikit-Learn. Untuk menggunakan, kita perlu mengimport Scikit-Learn terlebih dahulu, kemudian setelah itu kita dapat menggunakan fungsi `train_test_split()`.

## Latihan (13)

split data train dan test dengan function `train_test_split()` dengan `train_size=0.8, test_size=0.2` dan `random_state=100`

```
In [18]: 1 x_train,x_test,y_train,y_test = train_test_split(x, y, train_size = 0.8, test_size = 0.2, random_state = 100)
```

- `x_train`: Untuk menampung data source yang akan dilatih
- `x_test`: Untuk menampung data target yang akan dilatih
- `y_train`: Untuk menampung data source yang akan digunakan untuk testing.
- `y_test`: Untuk menampung data target yang akan digunakan untuk testing.

X dan y adalah nama variabel yang digunakan saat mendefinisikan data source dan data target. Parameter `test_size` digunakan untuk mendefinisikan ukuran data testing. Dalam contoh di atas, `test_size=0.2` berarti data yang digunakan sebagai data testing adalah sebesar 20% dari keseluruhan dataset.

Perlu diketahui bahwa metode ini akan membagi train set dan test set secara random atau acak. Jadi, jika kita mengulang proses running, maka tentunya hasil yang didapat akan berubah-ubah. Untuk mengatasinya, kita dapat menggunakan parameter `random_state`

## Latihan (14)

buat object variabel linier regression

```
In [19]: 1 regressor = linear_model.LinearRegression()
```

## Latihan (15)

training the model menggunakan training data yang sudah displit sebelumnya.

```
In [20]: 1 regressor.fit(x_train, y_train)
```

```
Out[20]: LinearRegression()
```

## Latihan (16)

cari tau nilai slope/koeffisien (m) dan intercept (b), dengan menggunakan function dari library sklearn -> LinierRegression

```
In [21]: 1 print(regressor.coef_)
2 print(regressor.intercept_)
```

```
[[168.17363122]]
[-8837.06049611]
```

Dari nilai m dan b diatas, kalau dimasukan ke dalam rumus persamaan menjadi:

$$y = 168.17x - 8037.06$$

## Latihan (17)

cari tahu accuracy score dari model kita menggunakan testing data yang sudah displit sebelumnya. Dan nilai korelasinya

```
In [22]: 1 regressor.score(x_test,y_test)
```

```
Out[22]: 0.8068161903454086
```

Model kita mendapatkan accuracy score sebesar 80.68%

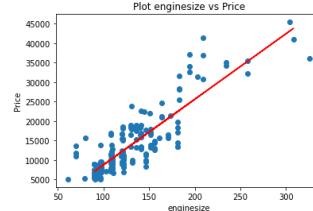
```
In [23]: 1 print('Correlation: ', math.sqrt(regressor.score(x_test,y_test)))  
Correlation:  0.8982294753265496
```

## Latihan (18)

visualisasi Regression Line menggunakan data testing.

```
In [24]: 1 y_prediksi = regressor.predict(x_test)  
2  
3 plt.scatter(dataset['enginesize'],dataset['price'])  
4 plt.plot(x_test, y_prediksi, c='r')  
5 plt.xlabel('enginesize')  
6 plt.ylabel('Price')  
7 plt.title('Plot enginesize vs Price')
```

```
Out[24]: Text(0.5, 1.0, 'Plot enginesize vs Price')
```



Garis merah merupakan Regression Line dari model yang telah dibuat sebelumnya.

## Latihan (19)

Setelah kita yakin dengan model yang dibuat, selanjutnya adalah prediksi dari harga mobil dengan enginesize 100, 150, dan 200.

```
In [25]: 1 #Prediksi harga mobil dengan enginesize 130.  
2  
3 print('nilai prediksi harga dengan enginesize 100 : ', regressor.predict([[100]]))  
4 print('nilai prediksi harga dengan enginesize 150 : ', regressor.predict([[150]]))  
5 print('nilai prediksi harga dengan enginesize 200 : ', regressor.predict([[200]]))  
  
nilai prediksi harga dengan enginesize 100 : [[8790.30262568]]  
nilai prediksi harga dengan enginesize 150 : [[17188.98418658]]  
nilai prediksi harga dengan enginesize 200 : [[25597.66574748]]
```

```
In [26]: 1 np_table = np.concatenate((x_test,y_test,y_prediksi), axis=1)  
2 new_dataframe = pd.DataFrame(data=np_table, columns=['x_test','y_test','y_predict'])
```

```
In [27]: 1 new_dataframe
```

```
Out[27]:  
x_test y_test y_predict  
0 98.0 7738.0 8443.955363  
1 109.0 8495.0 10293.665307  
2 122.0 8845.0 12480.122512  
3 98.0 9298.0 8443.955363  
4 108.0 7603.0 10125.691675  
5 122.0 11245.0 12480.122512  
6 130.0 18420.0 13825.511562  
7 140.0 16503.0 15507.247874  
8 146.0 17669.0 16516.389662  
9 181.0 17199.0 22402.366754  
10 141.0 16845.0 15675.421506  
11 121.0 18150.0 12311.948881  
12 120.0 15580.0 12143.775250  
13 110.0 12945.0 10462.038938  
14 308.0 40960.0 43760.417919  
15 92.0 6855.0 7434.913576  
16 98.0 6938.0 8443.955363  
17 121.0 12170.0 12311.948881  
18 140.0 18280.0 15507.247874  
19 156.0 14869.0 18198.025974  
20 141.0 13415.0 15675.421506  
21 141.0 16515.0 15675.421506  
22 194.0 32528.0 24588.623960  
23 90.0 5572.0 7098.566314  
24 146.0 8449.0 16516.389662  
25 181.0 13499.0 22402.366754  
26 156.0 12784.0 18198.025974  
27 183.0 28176.0 22738.714017  
28 108.0 16925.0 10125.691675  
29 119.0 11048.0 11975.601619  
30 92.0 6189.0 7434.913576  
31 209.0 30760.0 27111.228428  
32 152.0 13880.0 17525.331449  
33 141.0 19045.0 15675.421506  
34 120.0 16630.0 12143.775250  
35 110.0 10698.0 10462.038938  
36 121.0 15040.0 12311.948881  
37 146.0 9989.0 16516.389662  
38 97.0 6849.0 8275.781732  
39 122.0 8948.0 12480.122512  
40 304.0 45400.0 43087.723394
```

Semakin tinggi nilai error, semakin besar errormya

## Latihan (20)

Cetak nilai Mean Absolute Error, Mean Squared Error, dan Root Mean Squared Error

```
In [28]: 1 from sklearn import metrics  
2 print('Mean Absolute Error: ', metrics.mean_absolute_error(y_test, y_prediksi))  
3 print('Mean Squared Error: ', metrics.mean_squared_error(y_test, y_prediksi))  
4 print('Root Mean Squared Error: ', np.sqrt(metrics.mean_squared_error(y_test, y_prediksi)))
```

```
Mean Absolute Error: 3123.611515387693
Mean Squared Error: 14882644.972928163
Root Mean Squared Error: 3857.8031278083854
```

```
In [29]: 1 plt.title('Comparison of Y values in test and the Predicted values')
2 plt.ylabel('Test Set')
3 plt.xlabel('Predicted values')
4 plt.plot(new_dataframe['y_predict'] , '.', new_dataframe['y_test'] , 'x')
5 plt.show()
```

