```
# Latihan 1
# import library pandas
import pandas as pd

# Import library numpy
import numpy as np

# Import library matplotlib dan seaborn untuk visualisasi
import matplotlib.pyplot as plt
import seaborn as sns

# Import Module LinearRegression digunakan untuk memanggil algoritma
Linear Regression.
from sklearn.linear_model import LinearRegression

# import Module train_test_split digunakan untuk membagi data kita
menjadi training dan testing set.
from sklearn.model_selection import train_test_split

# import modul mean_absolute_error dari library sklearn
from sklearn.metrics import mean_absolute_error

#import math agar program dapat menggunakan semua fungsi yang ada pada
modul math.(ex:sqrt)
import math

# me-non aktifkan peringatan pada python
import warnings
warnings.filterwarnings('ignore')

#Panggil file (load file bernama CarPrice_Assignment.csv) dan simpan
dalam dataframe Lalu tampilkan 10 baris awal dataset dengan function
head()
data =pd.read_csv("CarPrice_Assignment.csv")
dataset = pd.DataFrame(data)
dataset.head(10)
```

```
   car_ID  symboling                   CarName  ... citympg highwaympg
price
0       1          3        alfa-romero giulia  ...      21         27
13495.000
1       2          3       alfa-romero stelvio  ...      21         27
16500.000
2       3          1  alfa-romero Quadrifoglio  ...      19         26
16500.000
3       4          2               audi 100 ls  ...      24         30
13950.000
4       5          2                audi 100ls  ...      18         22
17450.000
5       6          2                  audi fox  ...      19         25
```

```
15250.000
6       7          1                audi 100ls  ...      19         25
17710.000
7       8          1                 audi 5000  ...      19         25
18920.000
8       9          1                 audi 4000  ...      17         20
23875.000
9      10          0       audi 5000s (diesel)  ...      16         22
17859.167

[10 rows x 26 columns]
```

```
# Latihan 2
# melihat jumlah baris dan jumlah kolom (bentuk data) pada data df
dengan fungsi .shape
dataset.shape
```

```
(205, 26)
```

```
# Melihat Informasi lebih detail mengenai struktur DataFrame dapat
dilihat menggunakan fungsi info()
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   car_ID            205 non-null    int64
 1   symboling         205 non-null    int64
 2   CarName           205 non-null    object
 3   fueltype          205 non-null    object
 4   aspiration        205 non-null    object
 5   doornumber        205 non-null    object
 6   carbody           205 non-null    object
 7   drivewheel        205 non-null    object
 8   enginelocation    205 non-null    object
 9   wheelbase         205 non-null    float64
 10  carlength         205 non-null    float64
 11  carwidth          205 non-null    float64
 12  carheight         205 non-null    float64
 13  curbweight        205 non-null    int64
 14  enginetype        205 non-null    object
 15  cylindernumber    205 non-null    object
 16  enginesize        205 non-null    int64
 17  fuelsystem        205 non-null    object
 18  boreratio         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
```

```
# melihat statistik data untuk data numeric seperti count, mean,
standard deviation, maximum, mininum, dan quartile.
dataset.describe()
```

```
           car_ID   symboling   wheelbase  ...     citympg  highwaympg
price
count  205.000000  205.000000  205.000000  ...  205.000000  205.000000
205.000000
mean   103.000000    0.834146   98.756585  ...   25.219512   30.751220
13276.710571
std     59.322565    1.245307    6.021776  ...    6.542142    6.886443
7988.852332
min      1.000000   -2.000000   86.600000  ...   13.000000   16.000000
5118.000000
25%     52.000000    0.000000   94.500000  ...   19.000000   25.000000
7788.000000
50%    103.000000    1.000000   97.000000  ...   24.000000   30.000000
10295.000000
75%    154.000000    2.000000  102.400000  ...   30.000000   34.000000
16503.000000
max    205.000000    3.000000  120.900000  ...   49.000000   54.000000
45400.000000

[8 rows x 16 columns]
```

```
# cek nilai yang hilang / missing values di dalam data
dataset.isnull().sum()
```

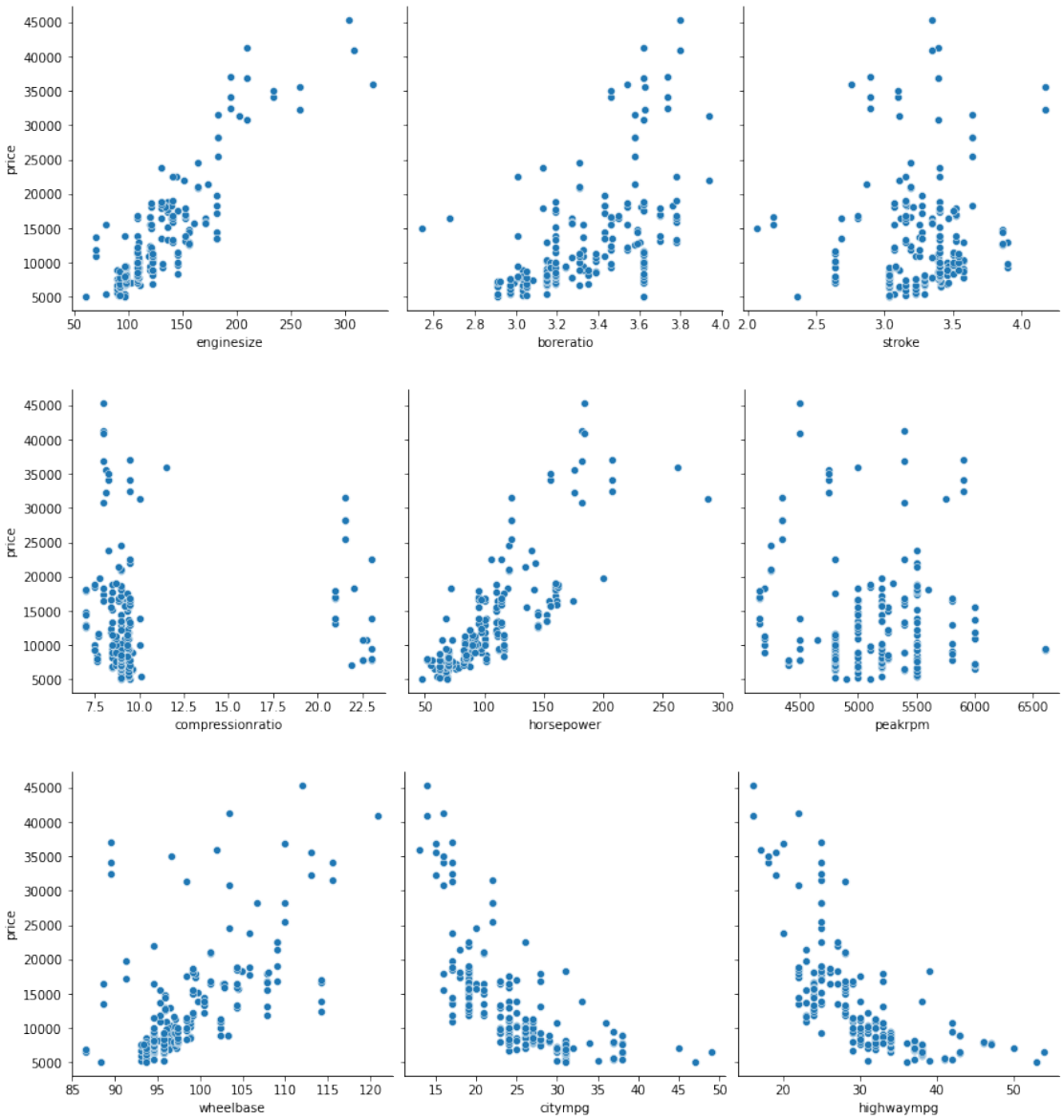
```
car_ID              0
symboling           0
CarName             0
fueltype            0
aspiration          0
doornumber          0
carbody             0
drivewheel          0
enginelocation      0
wheelbase           0
carlength           0
carwidth            0
carheight           0
curbweight          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
```

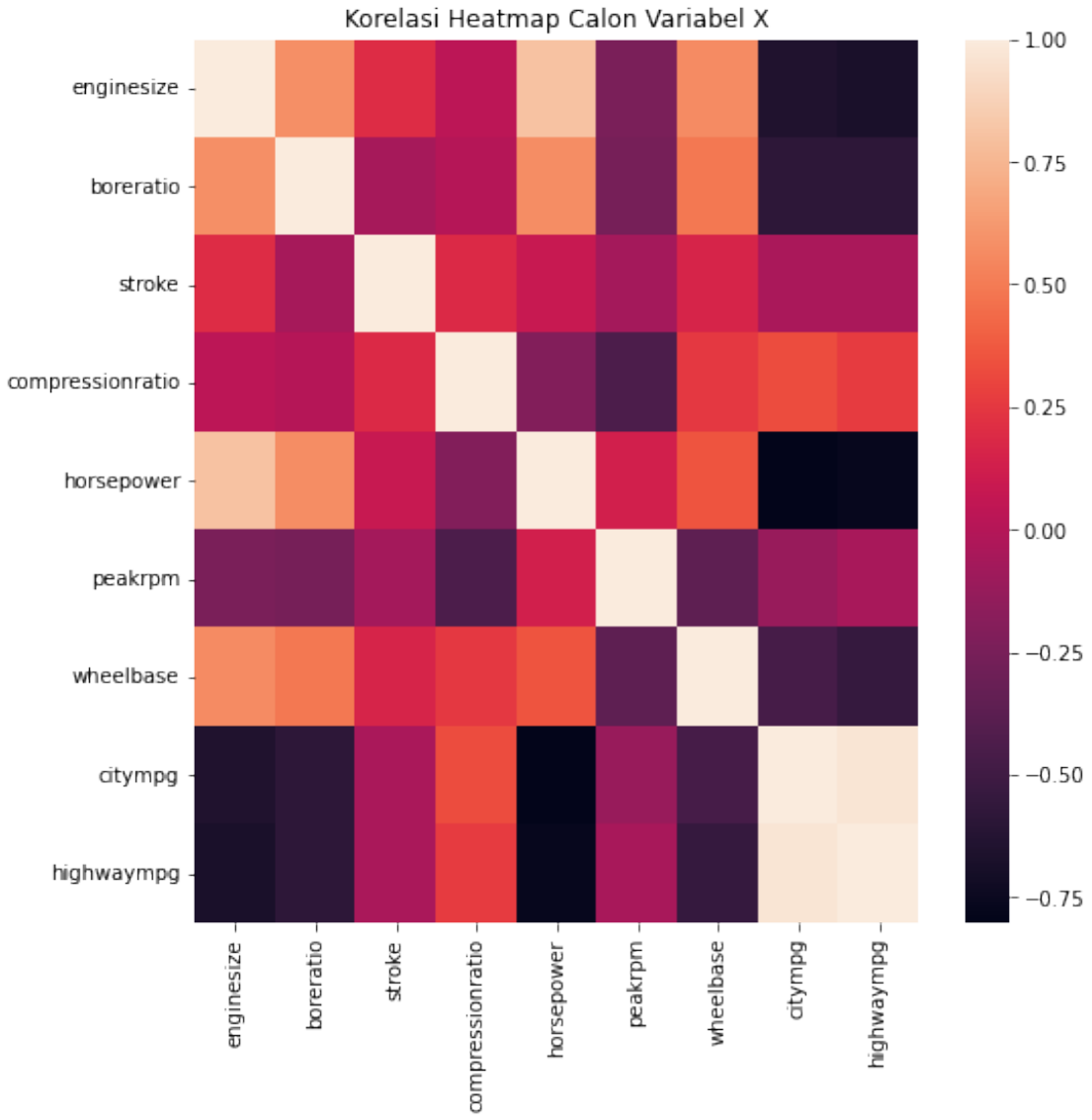
```
# Latihan 3
dataset.corr()
```

```
                   car_ID  symboling  ...  highwaympg     price
car_ID           1.000000  -0.151621  ...    0.011255 -0.109093
symboling       -0.151621   1.000000  ...    0.034606 -0.079978
wheelbase        0.129729  -0.531954  ...   -0.544082  0.577816
carlength        0.170636  -0.357612  ...   -0.704662  0.682920
carwidth         0.052387  -0.232919  ...   -0.677218  0.759325
carheight        0.255960  -0.541038  ...   -0.107358  0.119336
curbweight       0.071962  -0.227691  ...   -0.797465  0.835305
enginesize      -0.033930  -0.105790  ...   -0.677470  0.874145
boreratio        0.260064  -0.130051  ...   -0.587012  0.553173
stroke          -0.160824  -0.008735  ...   -0.043931  0.079443
compressionratio  0.150276  -0.178515  ...    0.265201  0.067984
horsepower      -0.015006   0.070873  ...   -0.770544  0.808139
peakrpm         -0.203789   0.273606  ...   -0.054275 -0.085267
citympg          0.015940  -0.035823  ...    0.971337 -0.685751
highwaympg       0.011255   0.034606  ...    1.000000 -0.697599
price           -0.109093  -0.079978  ...   -0.697599  1.000000

[16 rows x 16 columns]
```

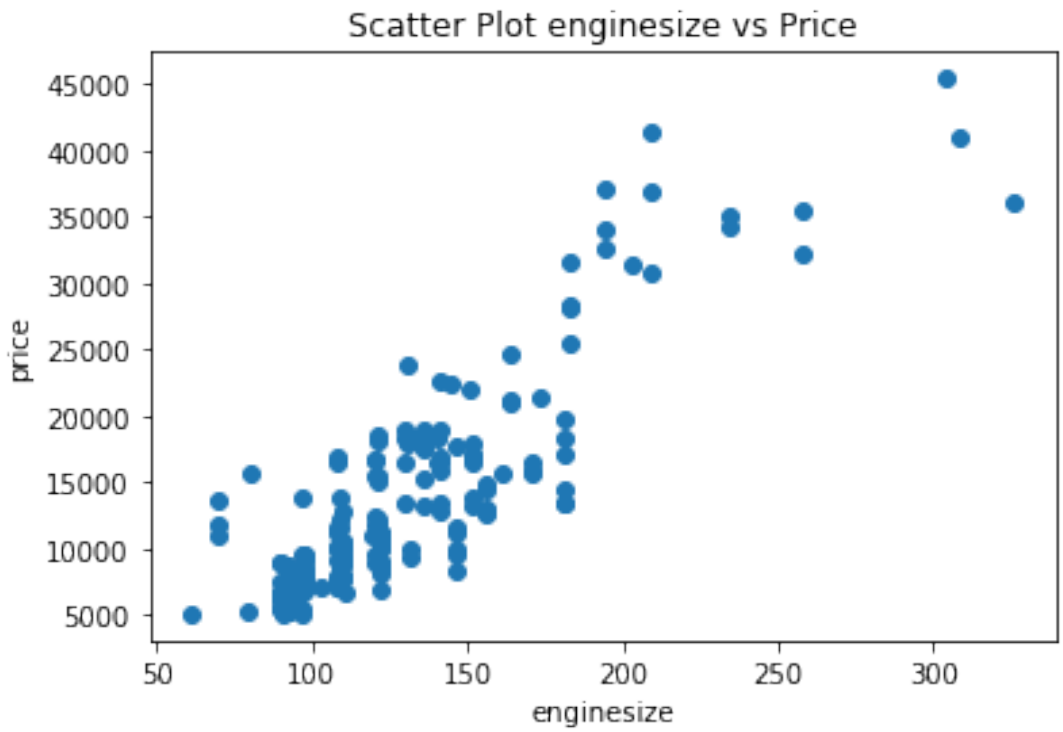
```
# Latihan 4
def pp(x,y,z):
    sns.pairplot(dataset, x_vars=[x,y,z], y_vars='price',size=4,
aspect=1, kind="scatter")
    plt.show()

pp('enginesize', 'boreratio', 'stroke')
pp('compressionratio', 'horsepower', 'peakrpm')
pp('wheelbase', 'citympg', 'highwaympg')
```




```
# Latihan 5
plt.figure(figsize = (8,8))
data_fitur = dataset[['enginesize', 'boreratio',
'stroke','compressionratio', 'horsepower', 'peakrpm', 'wheelbase',
'citympg', 'highwaympg']]
sns.heatmap(data_fitur.corr(),annot=False,fmt="f").set_title("Korelasi
Heatmap Calon Variabel X")
plt.show()
```




```
# Latihan 6
plt.scatter(dataset['enginesize'], dataset['price'])
plt.xlabel('enginesize')
plt.ylabel('price')
plt.title('Scatter Plot enginesize vs Price')
plt.show()
```




```
# Latihan 7
# Prepare data
# Pertama, buat variabel x dan y.
x = dataset['enginesize'].values.reshape(-1,1)
y = dataset['price'].values.reshape(-1,1)

# Latihan 8
x_mean = np.mean(x)
y_mean = np.mean(y)
print('nilai mean var x: ', x_mean,'\n'
      'nilai mean var y: ', y_mean)

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

# Latihan 9
atas = sum((x - x_mean)*(y - y_mean))
bawah = math.sqrt((sum((x - x_mean)**2)) * (sum((y - y_mean)**2)))
correlation = atas/bawah
print('Nilai Correlation Coefficient: ', correlation)

Nilai Correlation Coefficient:  [0.8741448]

# Latihan 10
# slope
# Slope adalah tingkat kemiringan garis, intercept
# adalah jarak titik y pada garis dari titik 0
variance = sum((x - x_mean)**2)
```

```
covariance = sum((x - x_mean) * (y - y_mean))
theta_1 = covariance/variance
print('Nilai theta_1: ',theta_1)
```

```
Nilai theta_1:  [167.69841639]
```

```
# Latihan 11
# intercept
theta_0 = y_mean - (theta_1 * x_mean)
print('Nilai theta_1: ',theta_0)
```

```
Nilai theta_1:  [-8005.44553115]
```

```
# Latihan 12
# prediction manual
y_pred = theta_0 + (theta_1 * 130)

print(y_pred)
```

```
[13795.34859997]
```

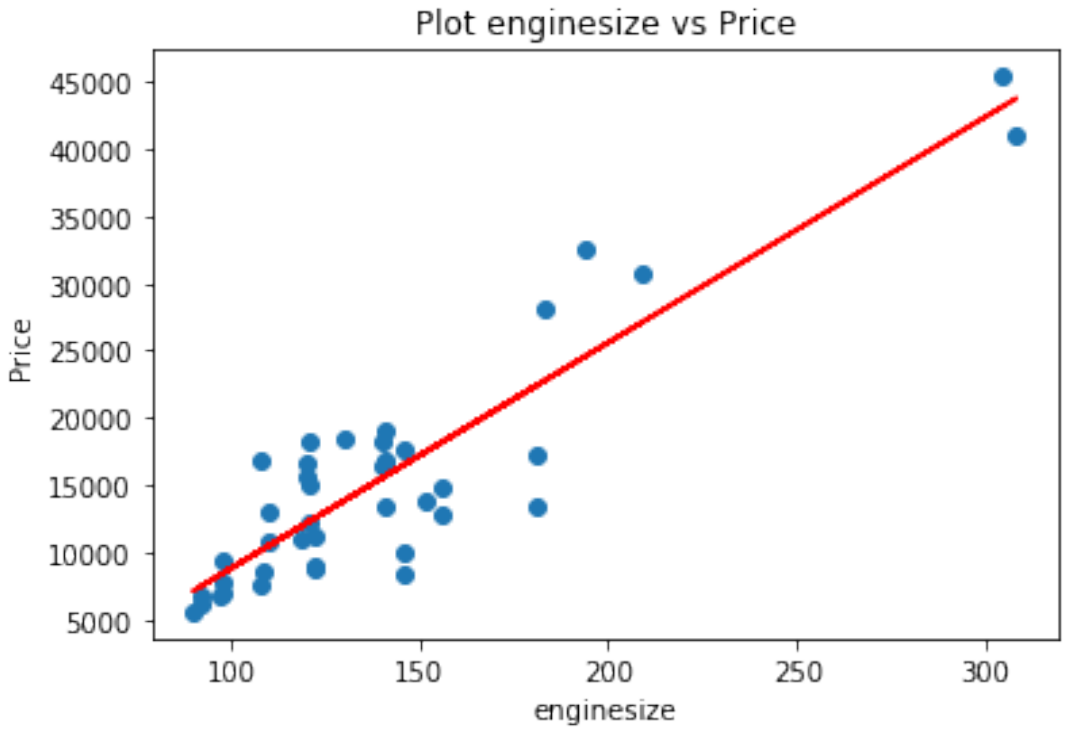
```
# visualisasi prediksi dengan scatter plot
y_pred = theta_0 + (theta_1 * x)

plt.scatter(x,y)
plt.plot(x, y_pred, c='r')
plt.xlabel('enginesize')
plt.ylabel('Price')
plt.title('Plot enginesize vs Price')
```

```
Text(0.5, 1.0, 'Plot enginesize vs Price')
```


```
# Latihan 13
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size =
0.8, test_size = 0.2, random_state = 100)

# lathihan 14
regressor = LinearRegression()

# Latihan 15
regressor.fit(x_train, y_train)

LinearRegression()

# Latihan 16
print(regressor.coef_)
print(regressor.intercept_)

[[168.17363122]]
[-8037.06049611]

# Latihan 17
regressor.score(x_test, y_test)

0.8068161903454086

# Latihan 18
y_prediksi = regressor.predict(x_test)

plt.scatter(x_test, y_test)
plt.plot(x_test, y_prediksi, c='r')
```

```
plt.xlabel('enginesize')
plt.ylabel('Price')
plt.title('Plot enginesize vs Price')
```

```
Text(0.5, 1.0, 'Plot enginesize vs Price')
```




```
# Latihan 19
#Prediksi harga mobil dengan enginesize 130.

print('nilai prediksi harga dengan enginesize 100 :
',regressor.predict([[100]]))
print('nilai prediksi harga dengan enginesize 150 :
',regressor.predict([[150]]))
print('nilai prediksi harga dengan enginesize 200 :
',regressor.predict([[200]]))
```

```
nilai prediksi harga dengan enginesize 100 :  [[8780.30262568]]
nilai prediksi harga dengan enginesize 150 :  [[17188.98418658]]
nilai prediksi harga dengan enginesize 200 :  [[25597.66574748]]
```

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

new_dataframe
```

```
    x_test   y_test     y_predict
0     98.0   7738.0   8443.955363
1    109.0   8495.0  10293.865307
```

```
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  16516.289662
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  16516.289662
25   181.0  13499.0  22402.366754
26   156.0  12764.0  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  13860.0  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  16516.289662
38    97.0   6849.0   8275.781732
39   122.0   8948.0  12480.122512
40   304.0  45400.0  43087.723394
```

```
# Latihan 20
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test,
y_prediksi))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_prediksi))
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
```

```
plt.title('Comparison of Y values in test and the Predicted values')
plt.ylabel('Test Set')
plt.xlabel('Predicted values')
plt.plot(y_prediksi, '.', y_test, 'x')
plt.show()
```