

# MACHINE LEARNING



2023

Nama : Marsa Mawaddah Herawati

Kelas : TI-3D

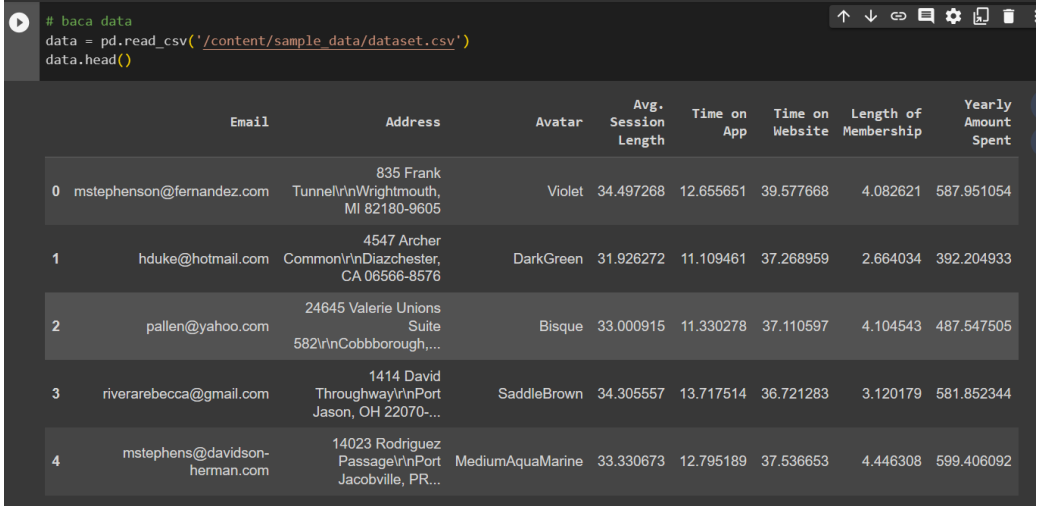
Nim/No : 2141720257/16



## Daftar Isi

Praktikum 1.....	3
Praktikum 2.....	9
Tugas Praktikum .....	11

## Praktikum 1

Langkah	Keterangan
1.	Langkah 1: Persiapan Data Download dan letakkan file data yang akan digunakan pada direktori yang sama. Pastikan data telah disimpan dalam format CSV.
2.	Langkah 2: Import Library <pre>[1] # import package import numpy as np import pandas as pd</pre> <p>Import library NumPy dan Pandas yang digunakan untuk manipulasi data.</p>
3.	Langkah 3: Baca Data  <p>Baca data dari file CSV dengan menggunakan Pandas.</p>
4.	Langkah 4: Pemahaman Terhadap Data <pre># pemahaman terhadap data # ukuran data data.shape  # info data data.info()  # deskripsi data data.describe()</pre> <p>Tampilkan beberapa data awal, ukuran data, informasi data, dan deskripsi statistik data untuk memahami karakteristik data.</p>

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
#   Column              Non-Null Count  Dtype
---  -
0   Email                500 non-null    object
1   Address              500 non-null    object
2   Avatar               500 non-null    object
3   Avg. Session Length  500 non-null    float64
4   Time on App          500 non-null    float64
5   Time on Website      500 non-null    float64
6   Length of Membership  500 non-null    float64
7   Yearly Amount Spent  500 non-null    float64
dtypes: float64(5), object(3)
memory usage: 31.4+ KB
```

	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
count	500.000000	500.000000	500.000000	500.000000	500.000000
mean	33.053194	12.052488	37.060445	3.533462	499.314038
std	0.992563	0.994216	1.010489	0.999278	79.314782
min	29.532429	8.508152	33.913847	0.269901	256.670582
25%	32.341822	11.388153	36.349257	2.930450	445.038277
50%	33.082008	11.983231	37.069367	3.533975	498.887875
75%	33.711985	12.753850	37.716432	4.126502	549.313828
max	36.139662	15.126994	40.005182	6.922689	765.518462

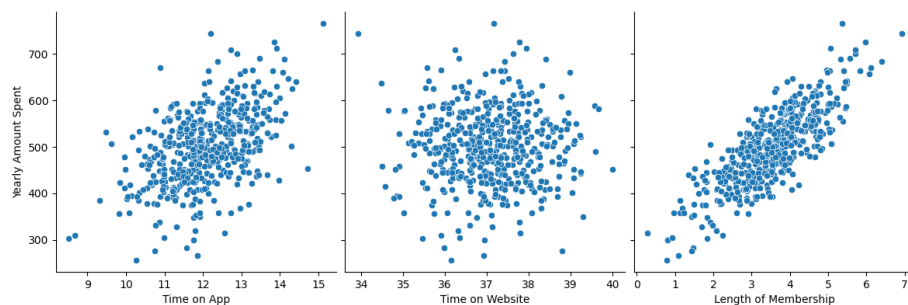
5.

## Langkah 5: Visualisasi Data

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

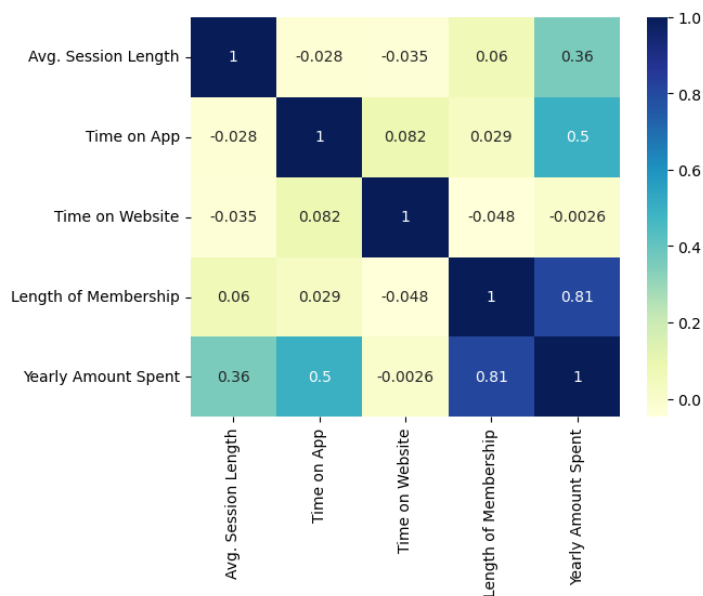
Import library Matplotlib dan Seaborn untuk visualisasi data.

```
# operasi visualisasi
sns.pairplot(data, x_vars=['Time on App', 'Time on Website', 'Length of Membership'],
              y_vars='Yearly Amount Spent', size=4, aspect=1, kind='scatter')
plt.show()
```



Gunakan pairplot untuk menampilkan hubungan antara variabel bebas dan variabel target dalam bentuk scatter plot.

```
sns.heatmap(data.corr(), cmap="YlGnBu", annot = True)
plt.show()
```



Gunakan heatmap untuk menampilkan matriks korelasi antara variabel-variabel dalam dataset. Semakin terang warna, semakin tinggi korelasinya.

6.

#### Langkah 6: Regresi Linier

```
# Buat variabel bebas X dan Y, sebagai contoh ambil dari hasil analisis korelasi dari kegiatan sebelumnya
X = data['Length of Membership']
y = data['Yearly Amount Spent']

X.head()
```

```
0    4.082621
1    2.664034
2    4.104543
3    3.120179
4    4.446308
Name: Length of Membership, dtype: float64
```

Pisahkan variabel bebas (X) dan variabel target (y).

```
# Buat pemisahan data uji dan data latih dengan proporsi 7:3
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7,
                                                    test_size = 0.3, random_state = 100)
```

```
# hasil training dtaset
X_train
y_train
```

```
153    657.019924
84     533.514935
310    479.614812
494    510.661792
126    516.831557
...
343    576.025244
359    561.874658
323    473.360496
280    511.979860
8      570.200409
Name: Yearly Amount Spent, Length: 350, dtype: float64
```

Bagi data menjadi data latih (70%) dan data uji (30%) menggunakan `train_test_split`.

```
# training model
import statsmodels.api as sm

X_train_sm = sm.add_constant(X_train)
```

```
# fitting garis regresi
lr = sm.OLS(y_train, X_train_sm).fit()
lr.params
```

```
const          265.248299
Length of Membership  66.301522
dtype: float64
```

```
# analisis statistika dari garis regresi
lr.summary()
```

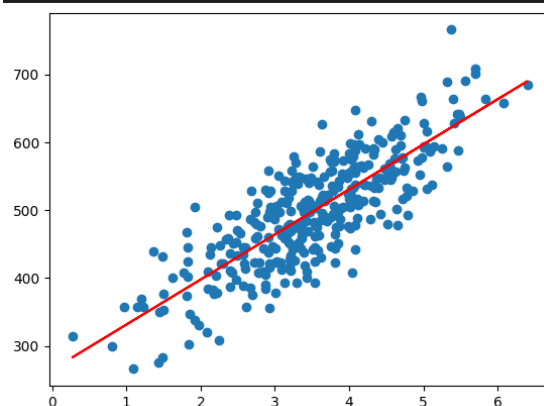
```
OLS Regression Results
Dep. Variable:  Yearly Amount Spent    R-squared:  0.669
Model:          OLS                    Adj. R-squared: 0.668
Method:        Least Squares          F-statistic: 702.9
Date:          Tue, 12 Sep 2023        Prob (F-statistic): 1.59e-85
Time:          02:55:59                Log-Likelihood: -1841.3
No. Observations: 350                  AIC:        3687.
Df Residuals:    348                  BIC:        3694.
Df Model:        1
Covariance Type: nonrobust

               coef  std err  t  P>|t| [0.025 0.975]
const         265.2483  9.120   29.083 0.000 247.311 283.186
Length of Membership 66.3015  2.501  26.512 0.000 61.383  71.220
Omnibus:       1.643   Durbin-Watson:  1.929
Prob(Omnibus): 0.440   Jarque-Bera (JB): 1.471
Skew:          -0.013   Prob(JB):   0.479
Kurtosis:      2.683    Cond. No.    14.2
```

Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Lakukan training model regresi linier menggunakan library StatsModels. Tambahkan konstanta (intercept) ke variabel bebas.

```
# visualisasi garis regresi
plt.scatter(X_train, y_train)
plt.plot(X_train, 265.2483 + 66.3015*X_train, 'r')
plt.show()
```



Visualisasikan garis regresi pada data latih.

7.

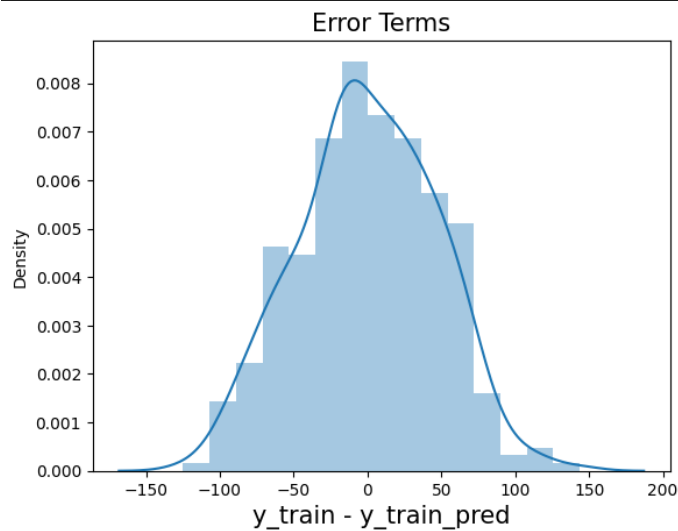
### Langkah 7: Analisis Residual

```
# prediksi y_value dari data x yang telah dilatih
y_train_pred = lr.predict(X_train_sm)

res = (y_train - y_train_pred)
```

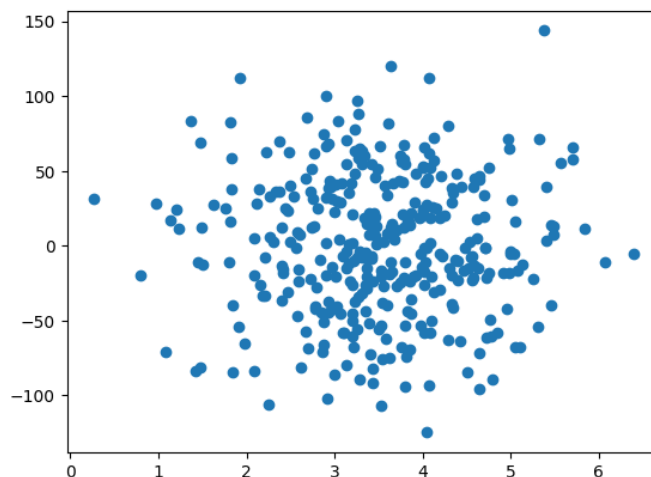
Lakukan prediksi nilai y dari data latih dan hitung residual (selisih antara nilai sebenarnya dan nilai prediksi).

```
# cek histogram apakah berdistribusi normal atau tidak
fig = plt.figure()
sns.distplot(res, bins = 15)
plt.title('Error Terms', fontsize = 15)
plt.xlabel('y_train - y_train_pred', fontsize = 15)
plt.show()
```



Visualisasikan residual dalam bentuk histogram dan scatter plot untuk mengevaluasi distribusi dan pola error.

```
plt.scatter(X_train, res)
plt.show()
```



8.

Langkah 8: Prediksi pada Data Uji dan Evaluasi Model

```
# prediksi pada data uji dan evaluasi model
X_test_sm = sm.add_constant(X_test)

# prediksi y value yang berkorelasi dengan X_test_sm
y_test_pred = lr.predict(X_test_sm)

# cetak 5 data terprediksi teratas
y_test_pred.head()
```

```
69      500.794385
29      579.688406
471     533.188991
344     446.066436
54      455.838449
dtype: float64
```

Lakukan prediksi pada data uji.

```
# hitung nilai r^2
from sklearn.metrics import r2_score

r_squared = r2_score(y_test, y_test_pred)
r_squared
```

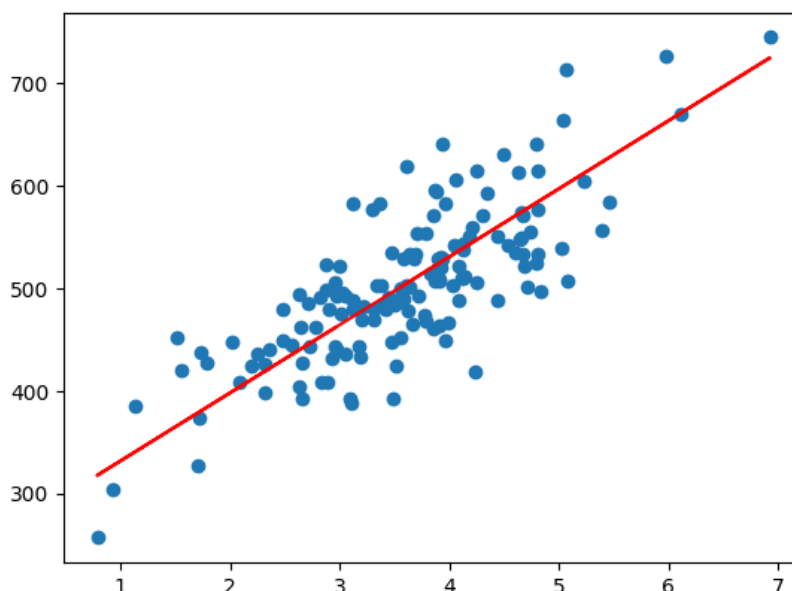
```
0.611948913768747
```

Hitung nilai R-squared untuk mengukur kinerja model pada data uji.

9.

Visualisasikan data uji dan hasil prediksi dalam bentuk scatter plot.

```
# visualisasi data
plt.scatter(X_test, y_test)
plt.plot(X_test, y_test_pred, 'r')
plt.show()
```





## Praktikum 2

Langkah	Keterangan
1.	<p>1. Mengimpor Library: Lakukan import library yang diperlukan terlebih dahulu, termasuk NumPy, Matplotlib, dan pandas.</p> <pre># Mengimpor library import numpy as np import matplotlib.pyplot as plt import pandas as pd</pre>
2.	<p>2. Mengimpor Dataset: Pastikan sudah mendownload file CSV 'Posisi_gaji.csv' dan letakkan dalam direktori yang sama. Ini adalah dataset yang akan digunakan dalam praktikum ini. Membaca dataset menggunakan pd.read_csv dan memilih fitur (variabel independen X) dan target (variabel dependen y).</p> <pre># Mengimpor dataset (Pastikan Anda memiliki file CSV 'Posisi_gaji.csv' dalam direktori yang sama) dataset = pd.read_csv('/content/sample_data/Posisi_gaji.csv') X = dataset.iloc[:, 1:2].values y = dataset.iloc[:, 2].values # Ubah menjadi satu kolom saja</pre>
3.	<p>3. Feature Scaling: Menggunakan StandardScaler untuk melakukan penskalaan fitur X dan target y. Ini diperlukan karena SVM sangat sensitif terhadap skala data.</p> <pre># Feature Scaling from sklearn.preprocessing import StandardScaler sc_X = StandardScaler() sc_y = StandardScaler() X = sc_X.fit_transform(X.reshape(-1, 1)) y = sc_y.fit_transform(y.reshape(-1, 1))</pre>
4.	<p>4. Fitting SVR ke Dataset: Lakukan pembuatan model SVR dengan kernel RBF (Radial Basis Function) dan melatihnya dengan data yang telah di-scaled.\</p> <pre># Fitting SVR ke dataset from sklearn.svm import SVR regressor = SVR(kernel='rbf') regressor.fit(X, y)</pre> <p>▼ SVR SVR()</p>
5.	<p>5. Visualisasi Hasil SVR: langkah selanjutnya, lakukan visualisasi Menggunakan grafik untuk memvisualisasikan hasil prediksi model SVR. Ini mencakup plotting data asli (titik-titik merah) dan kurva hasil prediksi (garis biru) untuk tingkat posisi yang bervariasi.</p>

	<p>(SVR)</p>
6.	<p><b>6. Prediksi Hasil:</b>  Membuat array 2D yang berisi tingkat posisi yang akan diprediksi. Dalam contoh ini, tingkat posisi 6.5.  Menskalakan fitur prediksi menggunakan <code>sc_X.transform</code>.  Melakukan prediksi menggunakan model SVR yang telah dilatih.  Mengembalikan hasil prediksi ke dalam skala aslinya menggunakan <code>sc_y.inverse transform</code>.</p> <pre># Prediksi hasil # Buat array 2D yang berisi tingkat posisi yang akan diprediksi tingkat_posisi_prediksi = np.array([[6.5]]) # Penskalaan fitur untuk data yang akan diprediksi tingkat_posisi_prediksi = sc_X.transform(tingkat_posisi_prediksi) # Melakukan prediksi menggunakan model SVR gaji_prediksi = regressor.predict(tingkat_posisi_prediksi) # Kembalikan hasil prediksi ke skala aslinya gaji_prediksi = sc_y.inverse_transform(gaji_prediksi.reshape(-1, 1))</pre>
7.	<p><b>7. Menampilkan Hasil:</b>  Menampilkan hasil prediksi gaji untuk tingkat posisi 6.5 dalam kode</p> <pre># Menampilkan hasil prediksi print("Prediksi Gaji untuk Tingkat Posisi 6.5:", gaji_prediksi[0])</pre> <p>Prediksi Gaji untuk Tingkat Posisi 6.5: [170370.0204065]</p>
8.	<p><b>8. Validasi Hasil:</b></p> <pre>Prediksi Gaji untuk Tingkat Posisi 6.5: [170370.0204065]</pre> <p>Hasil output Gambar 3.10 adalah grafik dari model Support Vector Regression (SVR) yang telah dilatih untuk memprediksi gaji berdasarkan tingkat posisi. Grafik tersebut merupakan visualisasi dari hubungan antara tingkat posisi (x-axis) dan gaji (y-axis) setelah menerapkan model SVR.  Grafik SVR:</p>

	<p>Pada grafik tersebut, titik-titik merah mewakili data pengamatan asli yang digunakan untuk melatih model.</p> <p>Garis biru adalah hasil dari prediksi model SVR. Garis ini mencoba untuk mengikuti pola data asli sebaik mungkin dan merupakan representasi dari hubungan non-linear antara tingkat posisi dan gaji.</p> <p>Hasil prediksi yang ditampilkan adalah prediksi gaji untuk tingkat posisi 6.5. Hasilnya adalah sekitar \$170,370.02. Ini berarti model SVR memperkirakan bahwa seseorang dengan tingkat posisi 6.5 akan memiliki gaji sekitar \$170,370.02 berdasarkan pola hubungan yang ditemukan dalam data latihan.</p> <p>Grafik ini memvisualisasikan bagaimana model SVR mencoba untuk menyesuaikan diri dengan data yang ada dan memberikan prediksi yang sesuai berdasarkan tingkat posisi yang diberikan (6.5 dalam hal ini). Dalam prakteknya, Anda dapat menggunakan model ini untuk membuat prediksi gaji berdasarkan tingkat posisi lainnya dengan mengganti nilai tingkat_posisi_prediksi.</p>
9.	<p>9. Evaluasi Model SVR</p> <p>Langkah terakhir adalah melakukan evaluasi model meliputi MAE, MSE dan R-squared</p> <pre> # Evaluasi model from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score  # Membalikkan penskalaan pada data target yang sudah diprediksi y_actual = sc_y.inverse_transform(y) y_pred = regressor.predict(x)  # Menghitung MAE mae = mean_absolute_error(y_actual, y_pred)  # Menghitung MSE mse = mean_squared_error(y_actual, y_pred)  # Menghitung RMSE rmse = np.sqrt(mse)  # Menghitung R-squared r2 = r2_score(y_actual, y_pred)  print("MAE:", mae) print("MSE:", mse) print("RMSE:", rmse) print("R-squared:", r2) </pre> <p>Output:</p> <pre> MAE: 249500.11150357974 MSE: 142912240625.2814 RMSE: 378037.3534788347 R-squared: -0.7717363528203269 </pre>

## Tugas Praktikum

Langkah	Keterangan
---------	------------

- Identifikasi variabel-variabel yang akan digunakan sebagai variabel bebas (fitur) dan variabel target (biaya medis personal).

```
[78] # mengidentifikasi variabel independen dan dependen
```

```
import seaborn as sns
sns.pairplot(datas, x_vars=['age', 'sex', 'bmi', 'children', 'smoker', 'region'],
              y_vars='charges', height=5, aspect=1, kind='scatter')
```
- Bagi dataset menjadi data latih (train) dan data uji (test) dengan proporsi yang sesuai.

```
# merubah tipe data sex, smoker, dan region
```

```
datas['sex'] = datas['sex'].astype('category')
datas['sex'] = datas['sex'].cat.codes

datas['smoker'] = datas['smoker'].astype('category')
datas['smoker'] = datas['smoker'].cat.codes

datas['region'] = datas['region'].astype('category')
datas['region'] = datas['region'].cat.codes
```

```
datas
```

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	1	3	16884.92400
1	18	1	33.770	1	0	2	1725.55230
2	28	1	33.000	3	0	2	4449.46200
3	33	1	22.705	0	0	1	21984.47061
4	32	1	28.880	0	0	1	3866.85520
...	...	...	...	...	...	...	...
1333	50	1	30.970	3	0	1	10600.54830
1334	18	0	31.920	0	0	0	2205.98080
1335	18	0	36.850	0	0	2	1629.83350
1336	21	0	25.800	0	0	3	2007.94500
1337	61	0	29.070	0	1	1	29141.36030

1338 rows × 7 columns

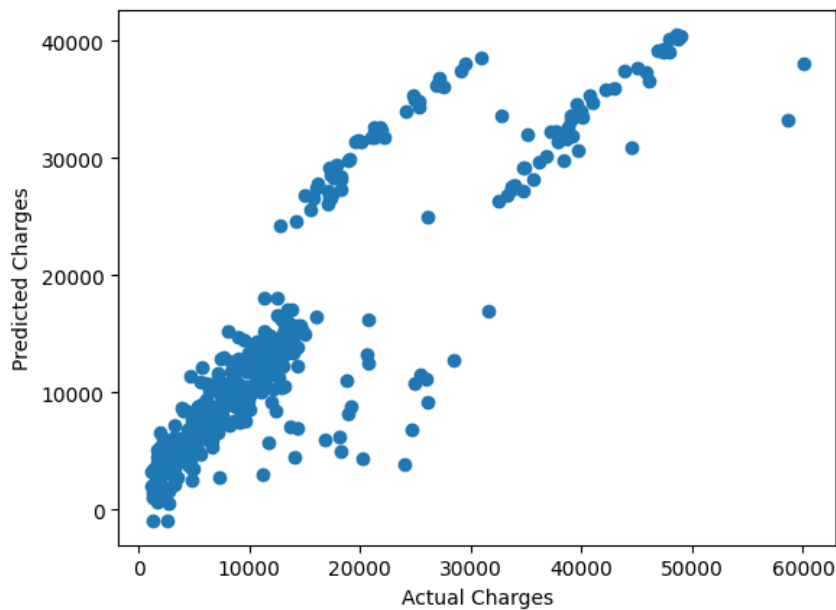
	<pre># menentukan variabel independen dan dependen X = datas.drop(columns = 'charges') y = datas['charges'] X</pre> <table><thead><tr><th></th><th>age</th><th>sex</th><th>bmi</th><th>children</th><th>smoker</th><th>region</th></tr></thead><tbody><tr><td>0</td><td>19</td><td>0</td><td>27.900</td><td>0</td><td>1</td><td>3</td></tr><tr><td>1</td><td>18</td><td>1</td><td>33.770</td><td>1</td><td>0</td><td>2</td></tr><tr><td>2</td><td>28</td><td>1</td><td>33.000</td><td>3</td><td>0</td><td>2</td></tr><tr><td>3</td><td>33</td><td>1</td><td>22.705</td><td>0</td><td>0</td><td>1</td></tr><tr><td>4</td><td>32</td><td>1</td><td>28.880</td><td>0</td><td>0</td><td>1</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr><tr><td>1333</td><td>50</td><td>1</td><td>30.970</td><td>3</td><td>0</td><td>1</td></tr><tr><td>1334</td><td>18</td><td>0</td><td>31.920</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1335</td><td>18</td><td>0</td><td>36.850</td><td>0</td><td>0</td><td>2</td></tr><tr><td>1336</td><td>21</td><td>0</td><td>25.800</td><td>0</td><td>0</td><td>3</td></tr><tr><td>1337</td><td>61</td><td>0</td><td>29.070</td><td>0</td><td>1</td><td>1</td></tr></tbody></table> <p>1338 rows x 6 columns</p>		age	sex	bmi	children	smoker	region	0	19	0	27.900	0	1	3	1	18	1	33.770	1	0	2	2	28	1	33.000	3	0	2	3	33	1	22.705	0	0	1	4	32	1	28.880	0	0	1	...	...	...	...	...	...	...	1333	50	1	30.970	3	0	1	1334	18	0	31.920	0	0	0	1335	18	0	36.850	0	0	2	1336	21	0	25.800	0	0	3	1337	61	0	29.070	0	1	1
	age	sex	bmi	children	smoker	region																																																																															
0	19	0	27.900	0	1	3																																																																															
1	18	1	33.770	1	0	2																																																																															
2	28	1	33.000	3	0	2																																																																															
3	33	1	22.705	0	0	1																																																																															
4	32	1	28.880	0	0	1																																																																															
...	...	...	...	...	...	...																																																																															
1333	50	1	30.970	3	0	1																																																																															
1334	18	0	31.920	0	0	0																																																																															
1335	18	0	36.850	0	0	2																																																																															
1336	21	0	25.800	0	0	3																																																																															
1337	61	0	29.070	0	1	1																																																																															
3.	<p>Lakukan feature scaling jika diperlukan.</p> <pre>[66] # Feature Scaling X = datas.iloc[:, 0:5].values y = datas.iloc[:, 6].values from sklearn.preprocessing import StandardScaler sc_X = StandardScaler() sc_y = StandardScaler() X = sc_X.fit_transform(X.reshape(-1, 1)) y = sc_y.fit_transform(y.reshape(-1, 1))</pre> <div><div></div><pre># Fitting SVR ke dataset from sklearn.svm import SVR regressor = SVR(kernel='rbf') regressor.fit(X, y)</pre><div><div></div><div><div>▼ SVR</div><div>SVR()</div></div></div></div>																																																																																				
4.	Buat model multiple linear regression menggunakan Scikit-Learn.																																																																																				

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, y_train)
c = model.intercept_
print("Konstanta = ", c)
```

```
m = model.coef_
print("Koefisien Regresi = ", m)
```

```
Konstanta = -10428.119803691543
Koefisien Regresi = [ 2.59634761e+02 -5.43235302e-02  2.93390832e+02  4.67684029e+02
 2.40111697e+04 -4.99424941e+02]
```

```
y_pred = model.predict(X_test)
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Charges")
plt.ylabel("Predicted Charges")
plt.show()
```



5.

Latih model pada data latih dan lakukan prediksi pada data uji.

```
X_train
y_train
```

```
966    23967.38305
522     9866.30485
155     6948.70080
671     3943.59540
1173    6457.84340
...
802     2103.08000
53      37742.57570
350     11830.60720
79       6571.02435
792      2731.91220
Name: charges, Length: 936, dtype: float64
```

```
#training model
X_train_sm = sm.add_constant(X_train)
lr = sm.OLS(y_train, X_train_sm).fit()
lr.params
```

```
const      -10428.119804
age         259.634761
sex         -0.054324
bmi         293.390832
children    467.684029
smoker      24011.169706
region     -499.424941
dtype: float64
```

```
lr.summary()
```

```

              OLS Regression Results
Dep. Variable: charges      R-squared:  0.738
Model: OLS                  Adj. R-squared: 0.736
Method: Least Squares      F-statistic: 435.6
Date: Sun, 17 Sep 2023     Prob (F-statistic): 5.63e-266
Time: 23:42:01             Log-Likelihood: -9503.1
No. Observations: 936      AIC: 1.902e+04
Df Residuals: 929          BIC: 1.905e+04
Df Model: 6
Covariance Type: nonrobust

   coef    std err   t    P>|t|  [0.025   0.975]
const -1.043e+04 1150.477 -9.064  0.000 -1.27e+04 -8170.286
age    259.6348   14.598  17.786  0.000  230.987  288.283
sex    -0.0543    408.970  -0.000  1.000 -802.666  802.558
bmi    293.3908   33.625   8.725  0.000  227.402  359.380
children 467.6840  169.422   2.760  0.006  135.190  800.178
smoker  2.401e+04  516.044  46.529  0.000  2.3e+04  2.5e+04
region -499.4249  189.415  -2.637  0.009 -871.156 -127.694
Omnibus: 224.452   Durbin-Watson: 2.024
Prob(Omnibus): 0.000   Jarque-Bera (JB): 538.473
Skew: 1.272         Prob(JB): 1.18e-117
Kurtosis: 5.708      Cond. No.    290.
```

6. Evaluasi model dengan menghitung metrik seperti R-squared, MSE, dan MAE. Tampilkan hasil evaluasi.

```
regressor = LinearRegression()
regressor.fit(X, y)

# Make predictions
y_pred = regressor.predict(X)

# Calculate MAE
mae = mean_absolute_error(y, y_pred)

# Calculate MSE
mse = mean_squared_error(y, y_pred)

# Calculate RMSE
rmse = np.sqrt(mse)

# Calculate R-squared
r2 = r2_score(y, y_pred)

print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R-squared:", r2)
```

```
MAE: 4172.48711494405
MSE: 36527659.88568238
RMSE: 6043.811701706331
R-squared: 0.7507372027994937
```