praktikum-js3-elis-nurhidayati

September 11, 2024

Nama : Elis Nurhidayati

NIM : 2241720035

Kelas : TI - 3C

Link Google Colab : https://colab.research.google.com/drive/1wRgBOKz_RQU44snD4srxVpjAYm2qtAWC?us

#**Jobsheet 3: Regresi**##

Target Capaian Pembelajaran Mahasiswa mampu membuat model regresi dengan baik menggunakan metode regresi linier (sederhana dan berganda), polinomial, dan Support Vector Regression (SVR)

##Praktikum 1

0.0.1 Langkah 1: Persiapan Data

Download dan letakkan file data yang akan digunakan pada direktori yang sama. Pastikan data telah disimpan dalam format CSV.

0.0.2 Langkah 2: Import Library

Import library NumPy dan Pandas yang digunakan untuk manipulasi data.

[ ]:
```
# import package
import numpy as np
import pandas as pd
```

0.0.3 Langkah 3: Baca Data

Baca data dari file CSV dengan menggunakan Pandas.

[ ]:
```
data = pd.read_csv('/content/drive/MyDrive/Elis - 3C/SMT 5/ML/Jobsheet 3/
↪dataset.csv')
data.head()
```

[ ]:
```
                          Email  \
0      mstephenson@fernandez.com
1              hduke@hotmail.com
2               pallen@yahoo.com
3        riverarebecca@gmail.com
```

```
4  mstephens@davidson-herman.com

                                              Address            Avatar  \
0     835 Frank Tunnel\r\nWrightmouth, MI 82180-9605            Violet
1   4547 Archer Common\r\nDiazchester, CA 06566-8576         DarkGreen
2  24645 Valerie Unions Suite 582\r\nCobbborough,…           Bisque
3  1414 David Throughway\r\nPort Jason, OH 22070-…       SaddleBrown
4  14023 Rodriguez Passage\r\nPort Jacobville, PR…  MediumAquaMarine

   Avg. Session Length  Time on App  Time on Website  Length of Membership  \
0            34.497268    12.655651        39.577668              4.082621
1            31.926272    11.109461        37.268959              2.664034
2            33.000915    11.330278        37.110597              4.104543
3            34.305557    13.717514        36.721283              3.120179
4            33.330673    12.795189        37.536653              4.446308

   Yearly Amount Spent
0           587.951054
1           392.204933
2           487.547505
3           581.852344
4           599.406092
```

0.0.4 Langkah 4: Pemahaman Terhadap Data

Tampilkan beberapa data awal, ukuran data, informasi data, dan deskripsi statistik data untuk memahami karakteristik data.

[ ]:
```
# ukuran data
data.shape

# info data
data.info()

# deskripsi data
data.describe()
```

```
<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  \
count           500.000000   500.000000       500.000000
mean             33.053194    12.052488        37.060445
std               0.992563     0.994216         1.010489
min              29.532429     8.508152        33.913847
25%              32.341822    11.388153        36.349257
50%              33.082008    11.983231        37.069367
75%              33.711985    12.753850        37.716432
max              36.139662    15.126994        40.005182

       Length of Membership  Yearly Amount Spent
count            500.000000           500.000000
mean               3.533462           499.314038
std                0.999278            79.314782
min                0.269901           256.670582
25%                2.930450           445.038277
50%                3.533975           498.887875
75%                4.126502           549.313828
max                6.922689           765.518462
```

0.0.5 Langkah 5: Visualisasi Data

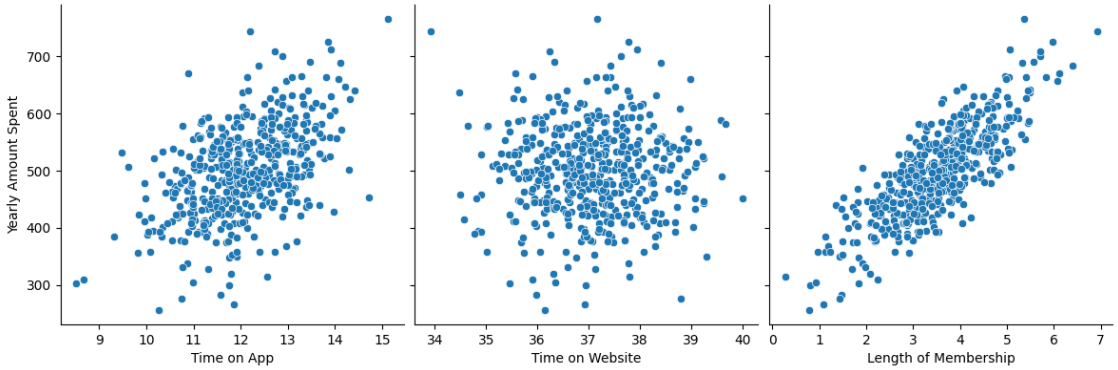
- Import library Matplotlib dan Seaborn untuk visualisasi data.

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

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

[ ]:
```
# operasi visualisasi data dengan pairplot
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()
```

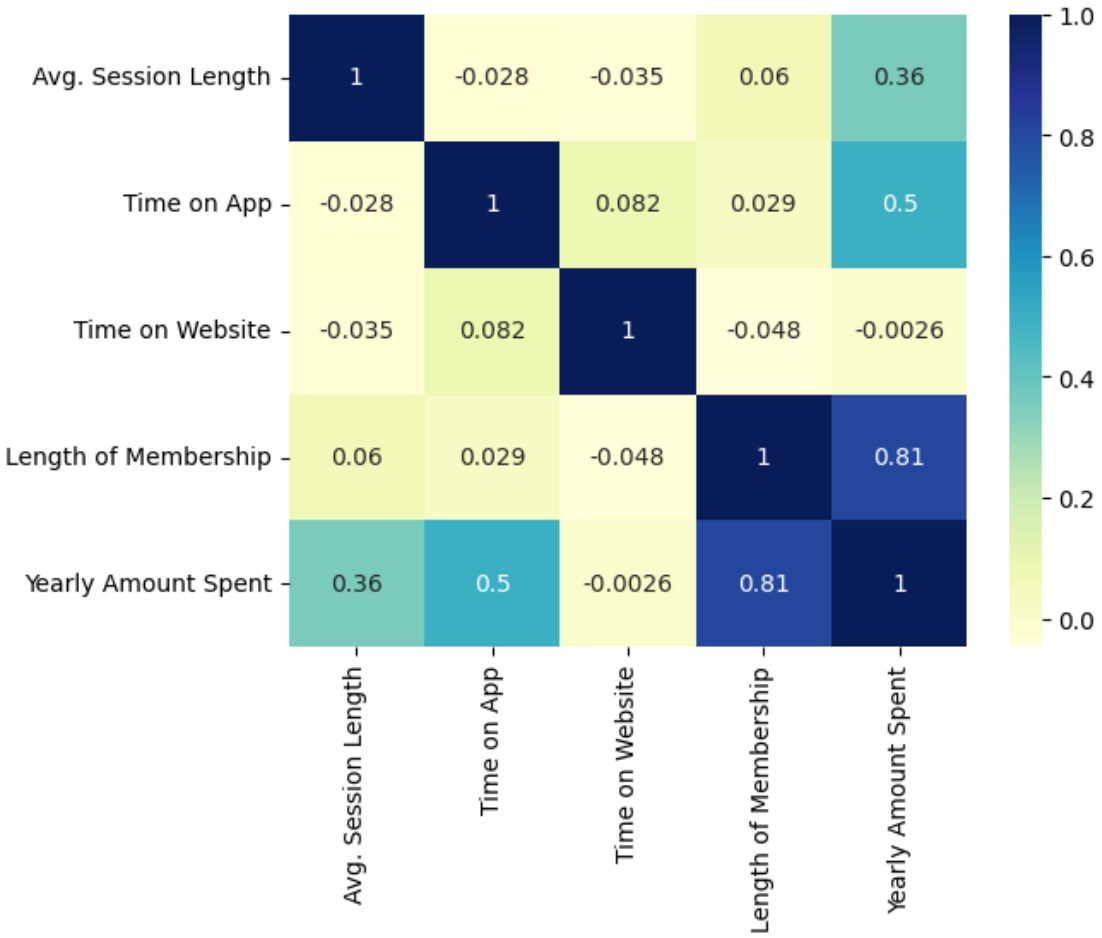
```
/usr/local/lib/python3.10/dist-packages/seaborn/axisgrid.py:2100: UserWarning:
The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```



output di atas kurang bisa menunjukkan korelasi antar data dalam x dengan data dalam y. Salah satu solusinya adalah menggunakan heatmap

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

[ ]:
```
# visualisasi korelasi dengan heatmap
numerical_data = data.select_dtypes(include=['number'])

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



dari bentuk visualisasi di atas terlihat bahwa Length of Membership memiliki korelasi yang paling kuat terhadap Yearly Amount Spent

0.0.6 Langkah 6: Regresi Linier

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

[ ]:
```
# Buat variabel bebas X dan Y, sebagai contoh ambil dari hasil analisis␣
↪korelasi dari kegaitan 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
```

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

[ ]:
```
# 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
```

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

[ ]:
```
# 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()
```

[ ]:

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, 10 Sep 2024	**Prob (F-statistic):**	1.59e-85
Time:	23:35:41	**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.

[ ]:
```
# visualisasi garis regresi pada data latih
plt.scatter(X_train, y_train)
plt.plot(X_train, 265.2483 + 66.3015*X_train, 'r')
plt.show()
```

0.0.7 Langkah 7: Residual Analysis

Dipakai untuk mengetahui tingkat error dari variabel yang dipengaruhi (y)

Error = Actual y value - y predicted value

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

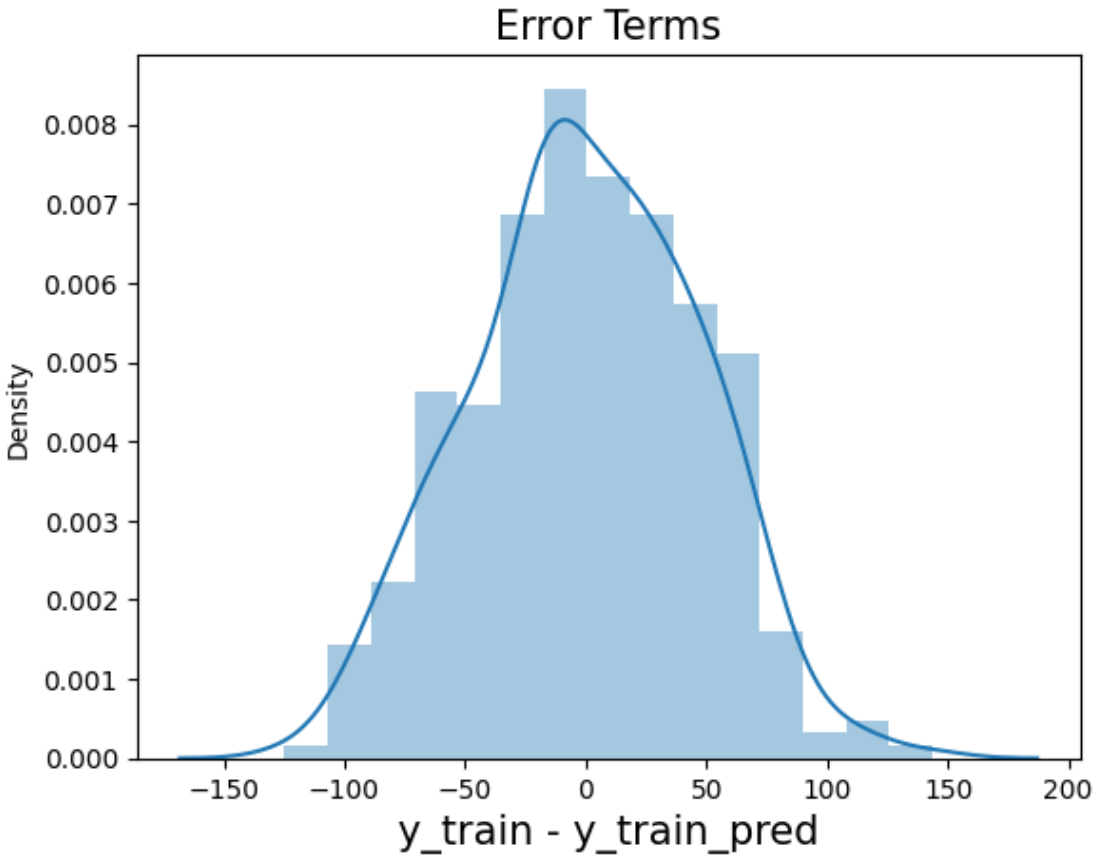
[ ]:
```
# prediksi y_value dari data x yang telah dilatih
y_train_pred = lr.predict(X_train_sm)

res = (y_train - y_train_pred)
```

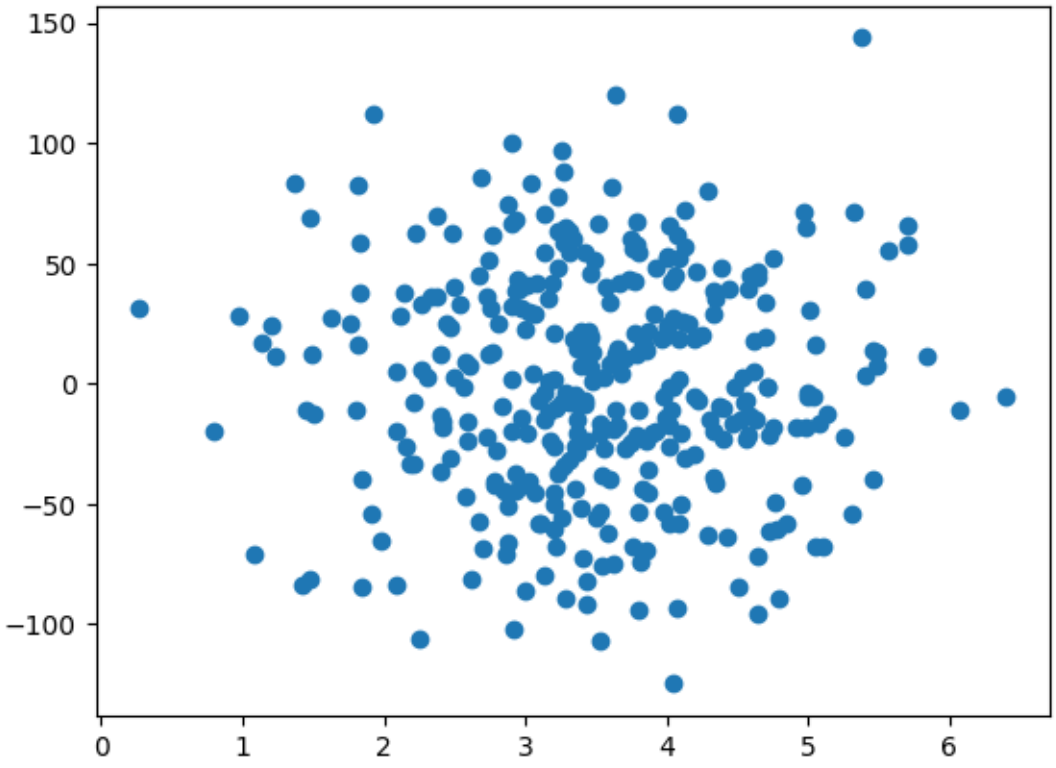
- Visualisasikan residual dalam bentuk histogram dan scatter plot untuk mengevaluasi distribusi dan pola error.

[ ]:
```
# 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()
```

```
# scatter plot residual
plt.scatter(X_train,res)
plt.show()
```

```
<ipython-input-24-cced1957cf38>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(res, bins = 15)
```





0.0.8 Langkah 8: Prediksi pada Data Uji dan Evaluasi Model

- Lakukan prediksi pada data uji.

[ ]:
```
# 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
```

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

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

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

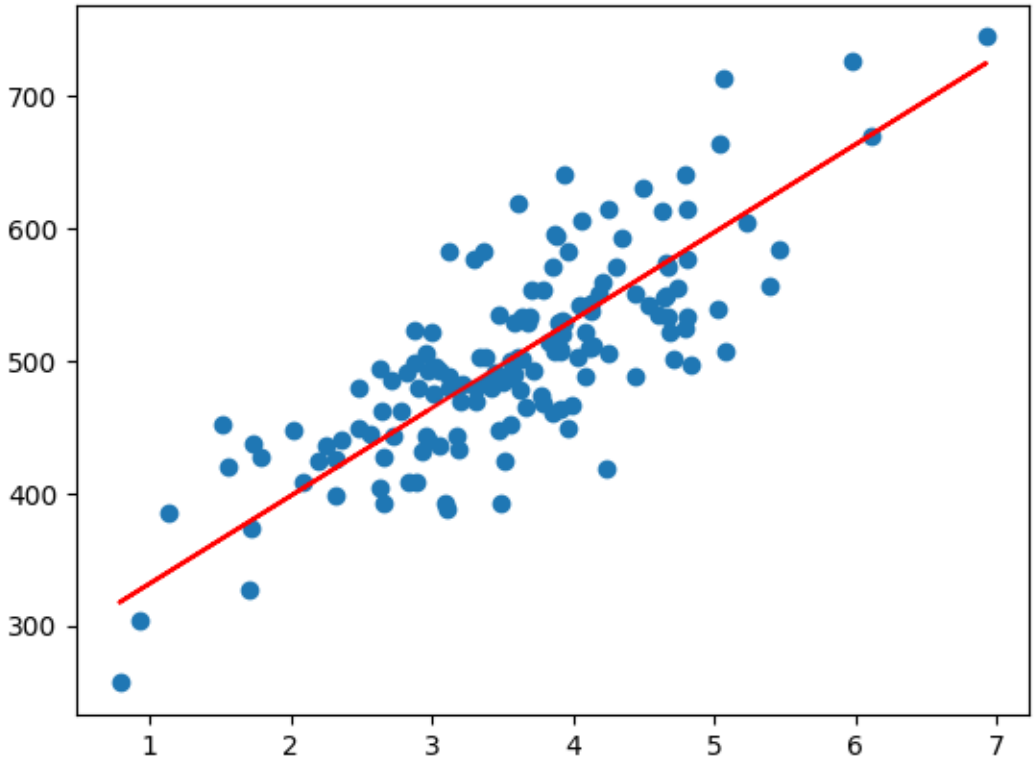
[ ]:
```
0.611948913768747
```

perlu diperhatikan bahwa r^2 dari data training adalah 0.669 sedangkan dari data testing adalah 0,612. Hal ini berarti model yang dibentuk cukup stabil (tidak berselisih jauh antara training dengan testing)

0.0.9 Langkah 9: Visualisasi Hasil

- Visualisasikan data uji dan hasil prediksi dalam bentuk scatter plot.

[ ]:
```
# visualisasi data uji dan hasil prediksi
plt.scatter(X_test, y_test)
plt.plot(X_test, y_test_pred, 'r')
plt.show()
```



1 Praktikum 2

1.0.1 Langkah 1: Mengimpor Library

Lakukan import library yang diperlukan terlebih dahulu, termasuk NumPy, Matplotlib, dan pandas.

[ ]:
```
# Mengimpor library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

1.0.2 Langkah 2: Mengimpor Database

- 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).

[ ]:
```
# Mengimpor dataset (Pastikan Anda memiliki file CSV 'Posisi_gaji.csv' dalam␣
↪direktori yang sama)
dataset = pd.read_csv('/content/drive/MyDrive/Elis - 3C/SMT 5/ML/Jobsheet 3/
↪Posisi_gaji.csv')
X = dataset.iloc[:, 1:2].values
y = dataset.iloc[:, 2].values  # Ubah menjadi satu kolom saja
```

1.0.3 Langkah 3: Feature scalling

Menggunakan StandardScaler untuk melakukan penskalaan fitur X dan target y. Ini diperlukan karena SVM sangat sensitif terhadap skala data.

[ ]:
```
# 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))
```

1.0.4 Langkah 4: Fitting SVR ke Database

Lakukan pembuatan model SVR dengan kernel RBF (Radial Basis Function) dan melatihnya dengan data yang telah di-scaled.

[ ]:
```
# Fitting SVR ke dataset
from sklearn.svm import SVR
regressor = SVR(kernel='rbf')
regressor.fit(X, y)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1183:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples, ), for example using
ravel().
  y = column_or_1d(y, warn=True)
```

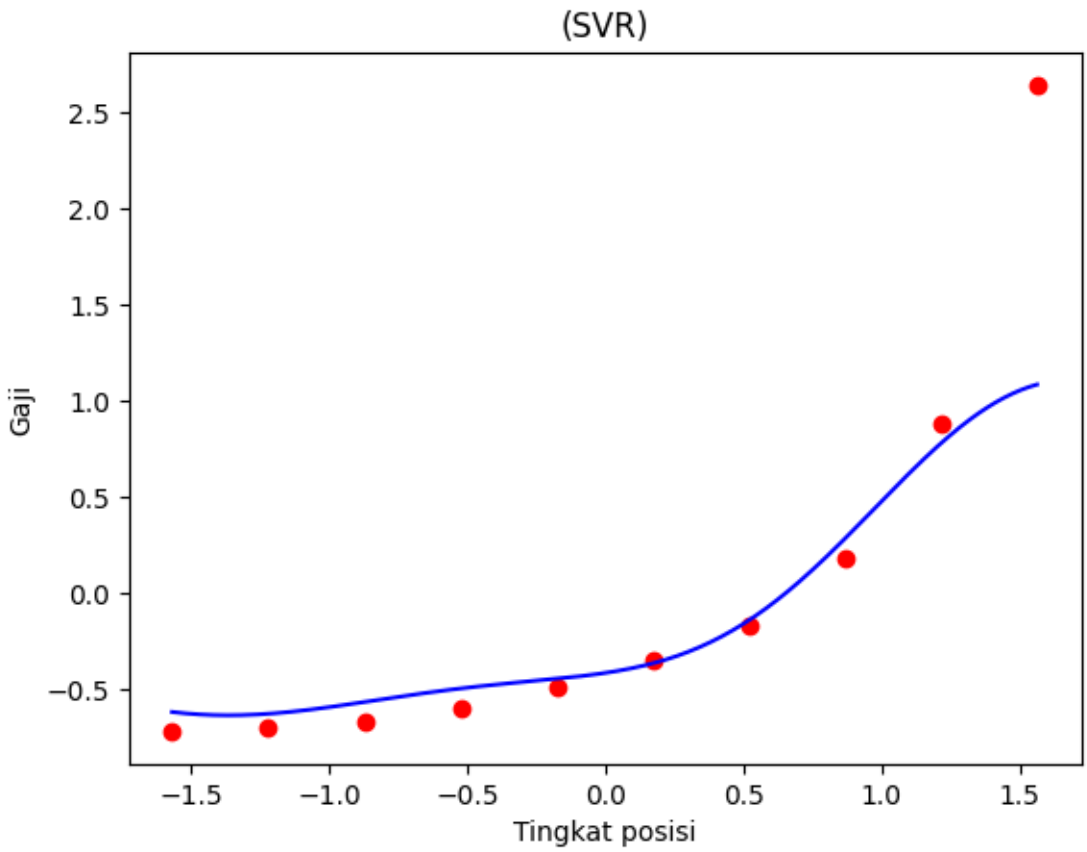
[ ]:
```
SVR()
```

1.0.5 Langkah 5: Visualisasi Hasil ke SVR

langkah selanjutkan, 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.

[ ]:
```
# Visualisasi hasil SVR (resolusi tinggi dan kurva yang lebih halus)
X_grid = np.arange(min(X), max(X), 0.01).reshape(-1, 1)
plt.scatter(X, y, color='red')
plt.plot(X_grid, regressor.predict(X_grid), color='blue')
plt.title('(SVR)')
plt.xlabel('Tingkat posisi')
plt.ylabel('Gaji')
plt.show()
```

```
<ipython-input-32-e94124bddd16>:2: DeprecationWarning: Conversion of an array
with ndim > 0 to a scalar is deprecated, and will error in future. Ensure you
extract a single element from your array before performing this operation.
(Deprecated NumPy 1.25.)
  X_grid = np.arange(min(X), max(X), 0.01).reshape(-1, 1)
```



1.0.6 Langkah 6: Prediksi Hasil

- Membuat array 2D yang berisi tingkat posisi yang akan diprediksi. Dalam contoh ini, tingkat posisi 6.5.
- Menskalakan fitur prediksi menggunakan sc_X.transform.
- Melakukan prediksi menggunakan model SVR yang telah dilatih.
- Mengembalikan hasil prediksi ke dalam skala aslinya menggunakan sc_y.inverse_transform.

[ ]:
```
# 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))
```

1.0.7 Langkah 7: Menampilkan Hasil

Menampilkan hasil prediksi gaji untuk tingkat posisi 6.5 dalam kode

[ ]:
```
print("Prediksi Gaji untuk Tingkat Posisi 6.5:", gaji_prediksi[0])
```

```
Prediksi Gaji untuk Tingkat Posisi 6.5: [170370.0204065]
```

1.0.8 Langkah 8: Validasi Hasil

- Dari hasil langkah 5 menampilkan visualisasi hasil prediksi model Support Vector Regression (SVR) untuk memprediksi gaji berdasarkan tingkat posisi. Titik merah: mewakili data asli yang digunakan untuk melatih model. Garis biru: menunjukkan prediksi dari model SVR, menggambarkan hubungan non-linear antara tingkat posisi dan gaji.
- Sedangkan dari hasil langkah 6 dan 7 yang menampilkan output dari prediksi gaji untuk tingkat posisi 6.5 adalah sekitar 170,370.02.

Berdasarkan hasil diatas, model SVR menyesuaikan diri dengan data dan memberikan prediksi sesuai berdasarkan posisi yang dimasukkan.

1.0.9 Langkah 9: evaluasi Model SVR

Langkah terakhir adalah melakukan evaluasi model meliputi MAE, MSE dan R-squared

[ ]:
```
# Evaluasi model
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

y_actual = 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)
```

```
MAE: 0.22299274095734414
MSE: 0.24839989293792014
```

```
RMSE: 0.4983973243687411
R-squared: 0.7516001070620798
```