

## Pokok Bahasan XII

# Support Vector Regression

**Kode Pokok Bahasan:** TIK.RPL03.001.009.01

**Deskripsi Pokok Bahasan:**

Membahas bagaimana melakukan proses penentuan keputusan prediksi dengan Support Vector Regression (SVR) .

No	Elemen Kompetensi	Indikator Kinerja	Jml Jam	Hal
1	Mengimplementasikan SVR dalam kasus yang diberikan	1.1 Menerapkan SVR untuk data beras yang tersimpan dalam database. 1.2 Membandingkan error metode SVR dan MLP	1	5

### TUGAS PENDAHULUAN

Hal yang harus dilakukan dan acuan yang harus dibaca sebelum praktikum :

1. Menginstal R pada PC masing-masing praktikan.
2. Menginstal R Studio pada PC masing-masing praktikan.

### DAFTAR PERTANYAAN

1. Apa yang dimaksud dengan Support Vector Regression (SVR)?

Support Vector Regression (SVR) adalah salah satu metode yang bisa digunakan dalam melakukan peramalan. Data yang digunakan adalah data penjualan roti manis, cake dan tawar dengan tipe data time series dan menggunakan 4 fitur.

2. Apa manfaat menggunakan Support Vector Regression (SVR)?

SVR memiliki kemampuan untuk mengatasi permasalahan overfitting, sehingga bisa mendapatkan suatu fungsi dengan tingkat kesalahan yang kecil dan menghasilkan prediksi yang bagus dengan cara menemukan Hyperplane (garis pemisah) terbaik.

3. Sebutkan kelebihan Support Vector Regression (SVR) dalam memprediksi data?

Dapat menggunakan beberapa classifier yang dilatih pada berbagai jenis data menggunakan aturan probabilitas. SVR melakukan komputasi yang lebih rendah dibandingkan dengan teknik regresi lainnya. Memiliki kemampuan generalisasi yang sangat baik, dengan akurasi prediksi yang tinggi.

### TEORI SINGKAT



Algoritma SVR (Support Vector Regression) adalah salah satu algoritma yang digunakan untuk pengambilan keputusan. Contoh yang dibahas kali ini adalah menentukan hasil jual tipe sepeda motor baru berdasarkan kelompok data yang sudah ada.

Algoritma ini dapat dikatakan sebagai perbaikan dari Regresi Linier / Analisis Regresi. Jika regresi linier menghasilkan sebuah fungsi dengan hasil linier / garis lurus, maka algoritma ini dapat menghasilkan sebuah fungsi dengan hasil yang bergelombang mengikuti jalur data yang terbentuk, sehingga prediksi yang didapatkan menjadi lebih akurat dibandingkan dengan regresi linier.

## LAB SETUP

Hal yang harus disiapkan dan dilakukan oleh praktikan untuk menjalankan praktikum modul ini.

1. Menginstal library yang dibutuhkan untuk mengerjakan modul.
2. Menjalankan R Studio.
3. XAMPP yang sudah dijalankan.
4. Membuat database pada phpmyadmin.

## ELEMEN KOMPETENSI I

### Deskripsi:

Mengimplementasikan SVR dalam kasus yang diberikan.

### Kompetensi Dasar:

- 1 Menerapkan SVR untuk data beras yang tersimpan dalam database.
- 2 Membandingkan error metode SVR dan MLP.

### Latihan 1.1.1

#### Penjelasan Singkat :

Pada latihan ini anda akan diminta untuk melakukan analisis regresi dengan SVR terhadap data beras yang diakses melalui database.

#### Langkah Praktikum :

1. Buatlah sebuah database pada phpmyadmin untuk menyimpan data beras.
2. Import data beras yang akan tersimpan dalam bentuk beberapa tabel.
3. Panggil library yang dibutuhkan

```
library(RMySQL)
library(nnfor)
library(dplyr)
library(hydroGOF)
library(e1071)
```

#### 4. Lakukan koneksi ke database

```
con = dbConnect(MySQL(), user = 'root', password = "", dbname = 'db_pibc_olap',
host = 'localhost')
dbListTables(con)
```

#### 5. Lakukan query “select” untuk mengambil data yang terpilih.

```
myQuery <- "select * from fact_harga;"
```

#### 6. Simpan data ke dalam dataframe.

```
df <- dbGetQuery(con, myQuery)
```



```
df1<-filter(df,SK_RICE_TYPE==9, SK_DATE>=20180101,  
SK_DATE<=20181231, SK_MARKET==0)  
View(df1)  
dim(df1)
```

### 7. Ubah dataframe menjadi time series.

```
myts=ts(df1$PRICE)
```

### 8. Lakukan prediksi SVM sebagai berikut:

```
X=c(1:243) #sesuaikan jumlah observasinya  
#Regression with SVM  
modelsvm = svm(df1$PRICE~X,data=df1)
```

```
#Predict using SVM regression  
predYsvm = predict(modelsvm, data=df1)
```

```
#Overlay SVM Predictions on Scatter Plot  
plot(X, predYsvm, col = "red", pch=16)  
points(X,df1$PRICE, col="blue", pch=16)
```

### 9. Hitung error RMSE pada model, dengan menggunakan fungsi rmse

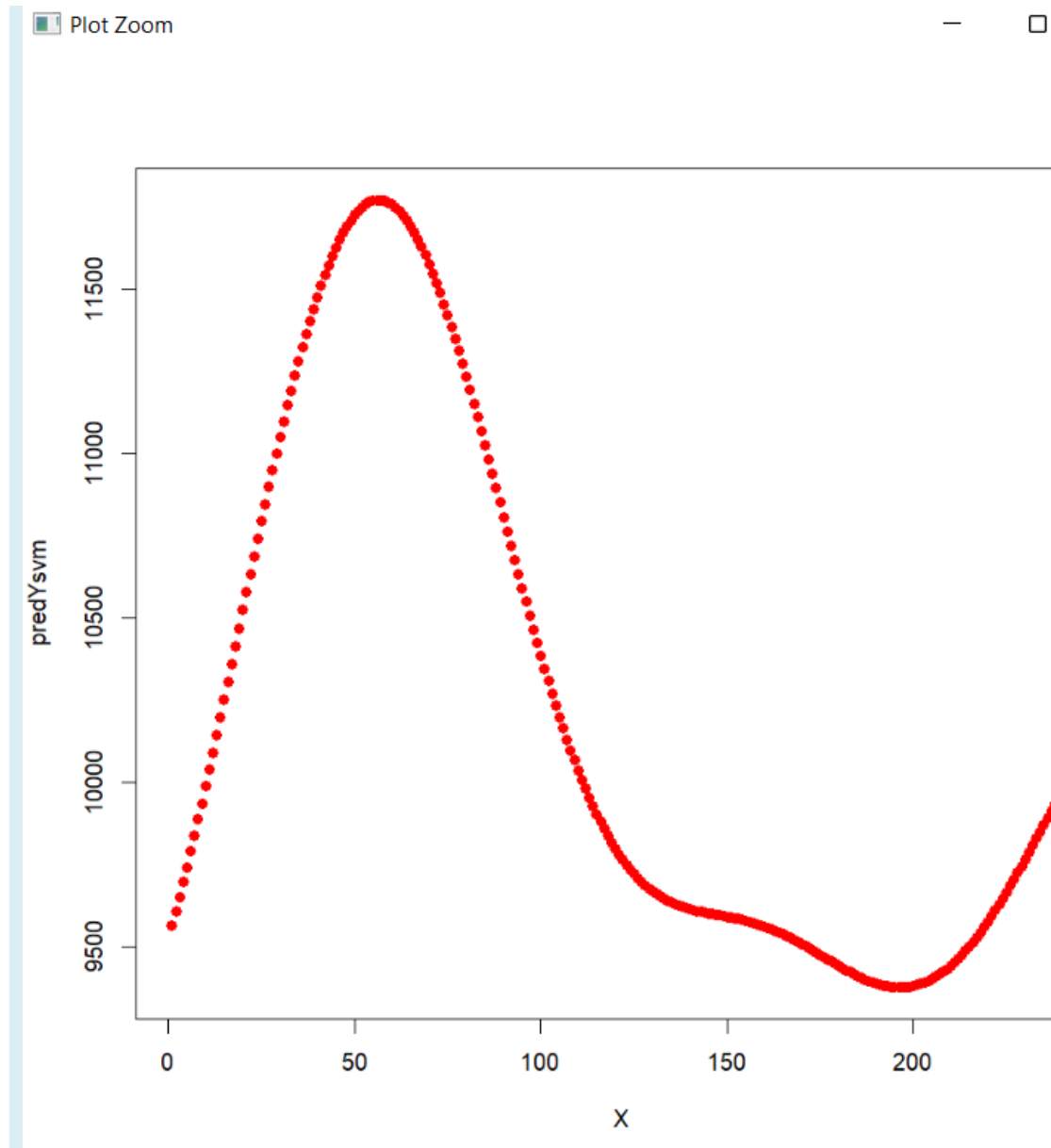
```
#Calculate RMSE  
RMSEsvm=rmse(predYsvm,df1$PRICE)  
RMSEsvm  
#Overlay SVM Predictions on Scatter Plot  
X=c(1:273)
```

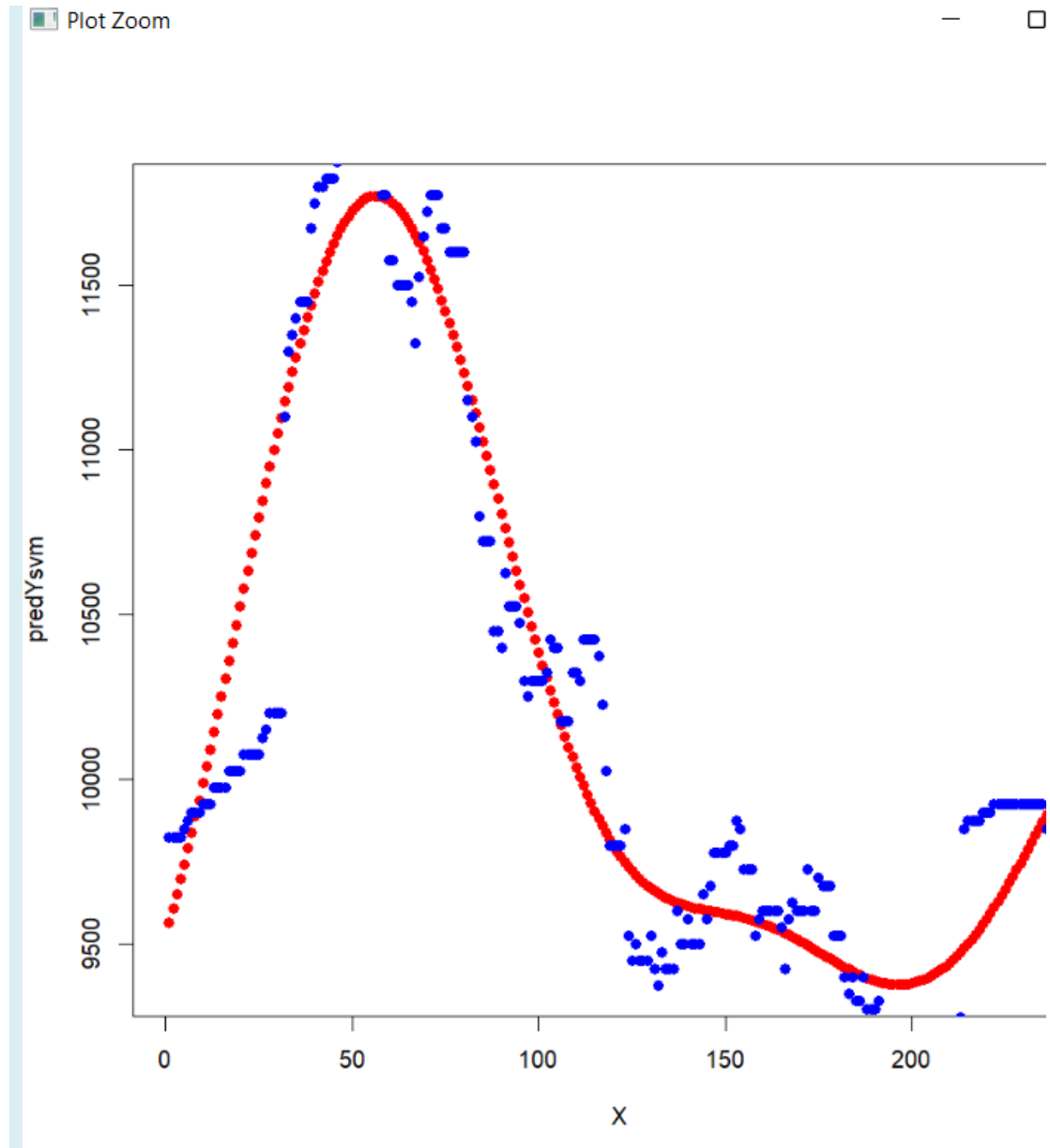
### 10. Membuat plot pada data model fit

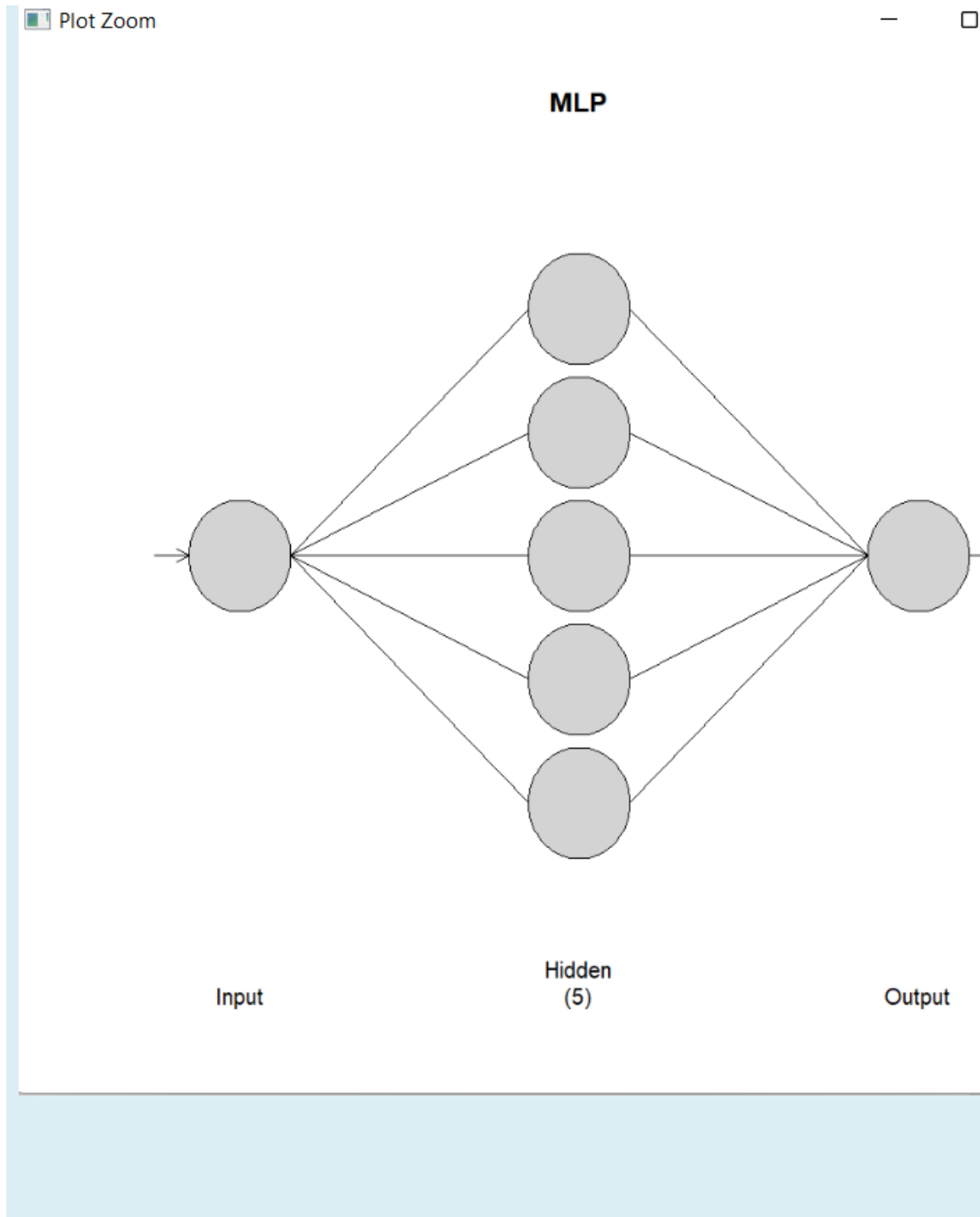
```
fit2 <- mlp(myts)  
plot(fit2, col = "red", pch=16)  
points(df1$PRICE, col="blue", pch=16)
```

Output :









Penjelasan :

Plot SVM memuat harga asli dan prediksi dari data Fact\_Harga, yang bergaris merah menunjukkan data prediksi, yang bergaris biru menunjukkan data asli. MLP mendapatkan data input kemudian diolah di Hidden kemudian jadi output yang diinginkan

### Latihan 1.1.2

#### Penjelasan Singkat :

Pada latihan ini anda akan diminta untuk melakukan peramalan dengan metode MLP terhadap data beras yang diakses melalui database. Lalu dibandingkan dengan metode SVR.

#### Tugas:

Gunakan data pada database “db\_pasokan\_beras”. Kemudian Lakukan Langkah Langkah seperti di Latihan 1.1.1. kemudian bandingkan errornya antara MLP dan SVM dengan metode SVR. Manakah hasilnya yang lebih akurat.

Catatan : Semua output dalam bentuk screenshot dimana nama object model peramalannya ditulis sesuai nama praktikan (contoh modelsvm diganti dengan svm\_namapraktikan). Berikan penjelasan atas output yang didapatkan.

```
#Latihan 1.1.2
library(statsr)
library(RMySQL)
library(dplyr)
con = dbConnect(MySQL(), user = 'root', password = '', dbname = 'dbpasokanberas', host = 'localhost')
dbListTables(con)
myQuery <- "select * from fact_price;"
df <- dbGetQuery(con, myQuery)
df1<-filter(df,id_rice_type==10, id_date>=20170101,id_date<=20171231)
df2<- df1[order(df1$id_date),]
View(df1)

myts=ts(df1$harga)
myts

X=c(1:365) #sesuaikan jumlah observasinya
#Regression with SVM
modelsvm = svm(df1$harga~X,data=df1)
modelsvm
```



```
#Predict using SVM regression
predYsvm = predict(modelsvm, data=df1)

#Overlay SVM Predictions on Scatter Plot
plot(X, predYsvm, col = "red", pch=16)
points(X,df1$harga, col="blue", pch=16)

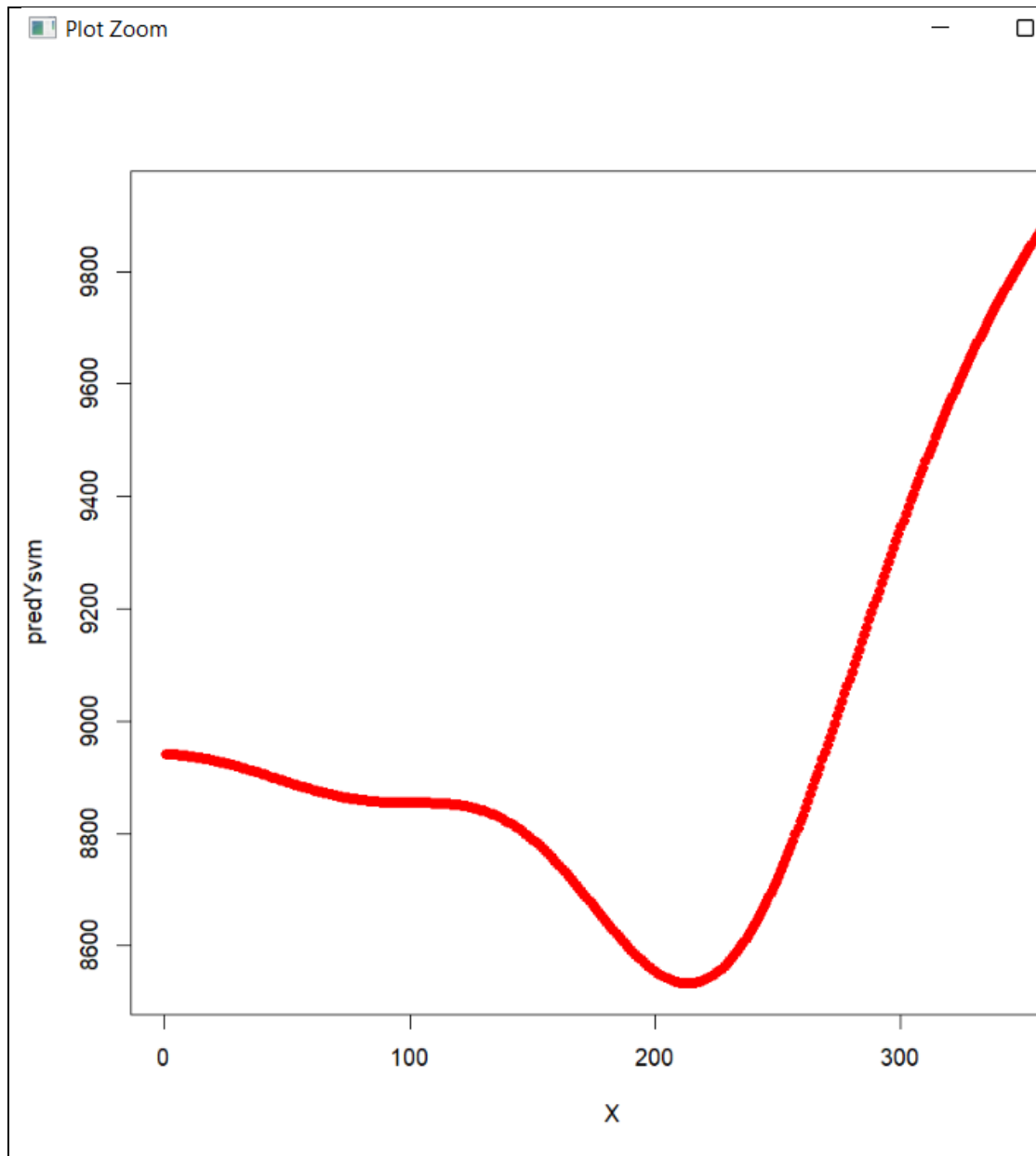
#Calculate RMSE
RMSEsvm=rmse(predYsvm,df1$harga)
RMSEsvm

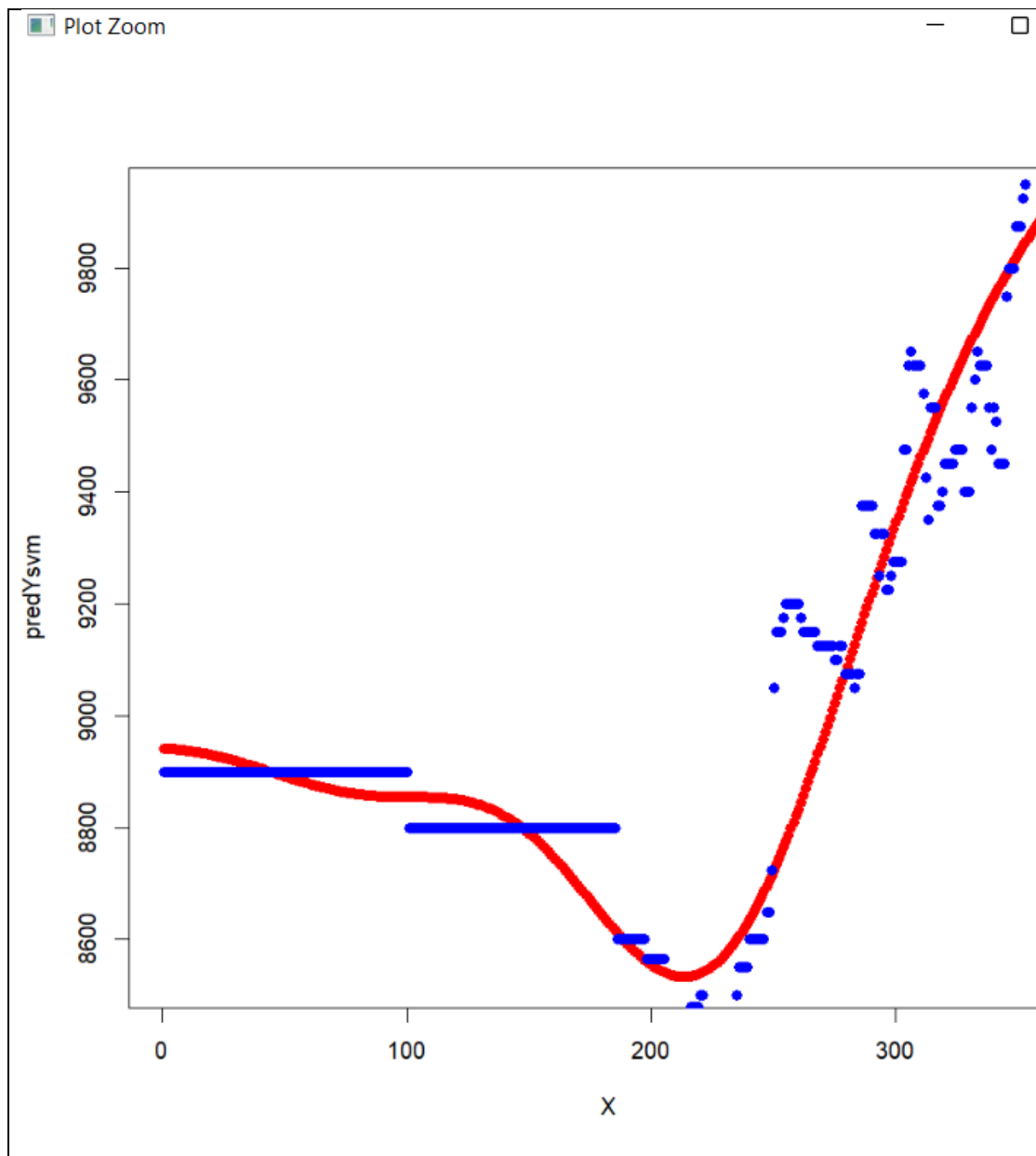
#Overlay SVM Predictions on Scatter Plot
X=c(1:273)

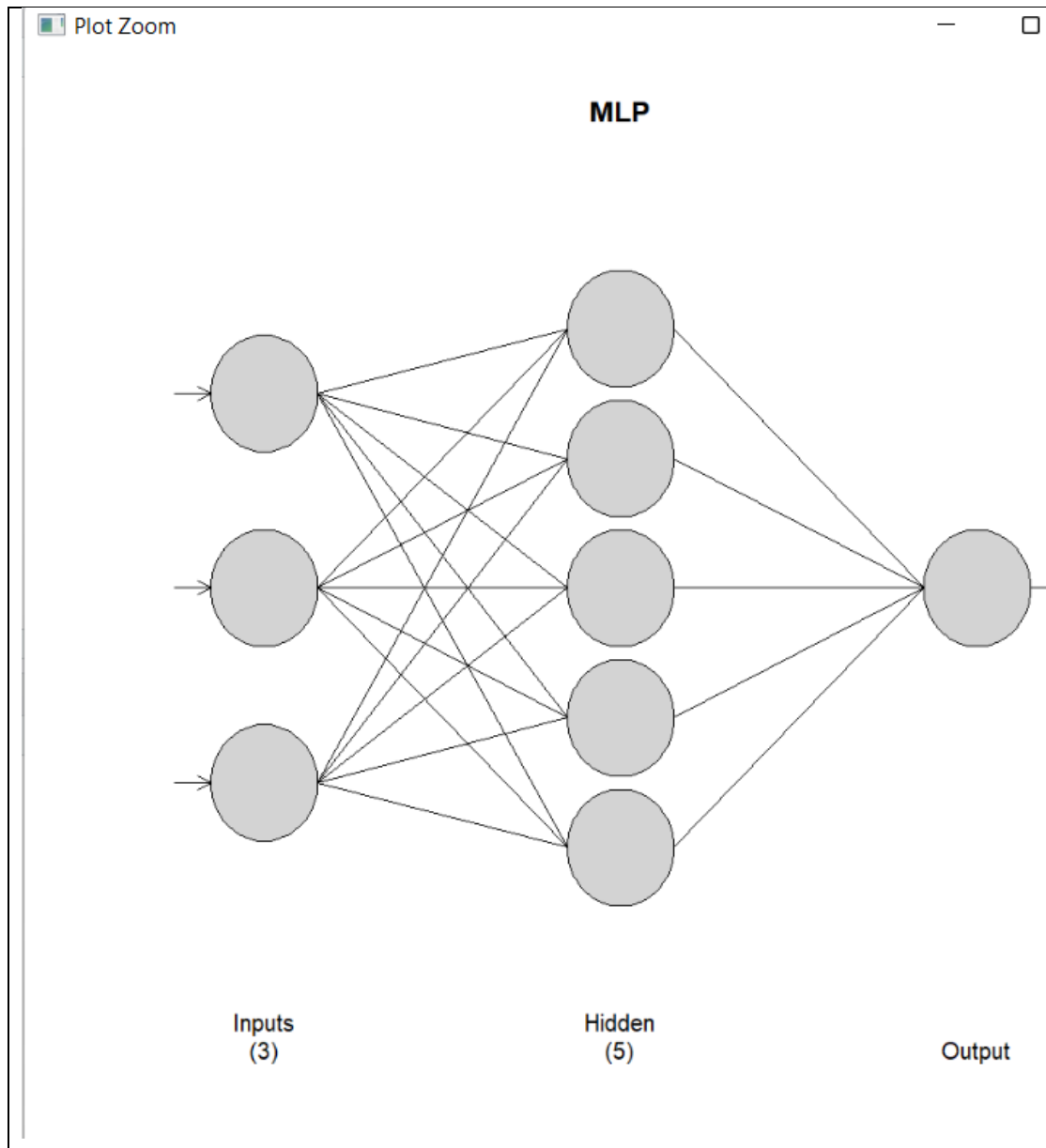
fit2 <- mlp(myts)
plot(fit2, col = "red", pch=16)
points(df1$harga, col="blue", pch=16)
```

Output









### Penjelasan

Plot SVM memuat harga asli dan prediksi dari data Pasokan Beras, yang bergaris merah menunjukkan data prediksi, yang bergaris biru menunjukkan data asli. MLP mendapatkan data input kemudian diolah di Hidden kemudian jadi output yang diinginkan

### Latihan 1.2.1

#### Penjelasan Singkat :

Pada latihan ini akan diminta menggunakan SVR dengan python.

## Langkah Praktikum :

### 1. Import libraries

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)
import matplotlib.pyplot as plt # for data visualization
import seaborn as sns # for statistical data visualization
%matplotlib inline
```

```
import warnings
```

```
warnings.filterwarnings('ignore')
```

### 2. Import dataset

```
df = pd.read_csv("fact_harga.csv")
```

### 3. Explorasi data

```
# melihat dimensi dataset
df.shape
```

output:

```
[ ] # melihat dimensi dataset
df.shape

(33339, 5)
```

```
# melihat dataset
df.head()
```

output:

```
# melihat dataset
df.head()
```

	id	SK_DATE	SK_RICE_TYPE	SK_MARKET	PRICE
0	1	20170101	1	0	13500
1	2	20170101	2	0	12350
2	3	20170101	3	0	12100
3	4	20170101	4	0	11500
4	5	20170101	5	0	10400

```
# melihat nama kolom pada dataframe
```

```
col_names = df.columns
```

```
col_names
```

output:

```
[ ] # melihat nama kolom pada dataframe
col_names = df.columns
col_names

Index(['id', 'SK_DATE', 'SK_RICE_TYPE', 'SK_MARKET', 'PRICE'], dtype='object')
```

```
# melihat summary dari dataset
```

```
df.info()
```

output:

```
# melihat summary dari dataset
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33339 entries, 0 to 33338
Data columns (total 5 columns):
#   Column          Non-Null Count  Dtype
---  -
0   id              33339 non-null  int64
1   SK_DATE         33339 non-null  int64
2   SK_RICE_TYPE    33339 non-null  int64
3   SK_MARKET       33339 non-null  int64
4   PRICE           33339 non-null  int64
dtypes: int64(5)
memory usage: 1.3 MB
```

```
# cek adanya missing value
df.isnull().sum()
```

output:

```
# cek adanya missing value
df.isnull().sum()

id          0
SK_DATE     0
SK_RICE_TYPE 0
SK_MARKET   0
PRICE       0
dtype: int64
```

```
# melihat statistik summary pada variabel numerik
round(df.describe(),2)
```

output:

```
# melihat statistik summary pada variabel numerik
round(df.describe(),2)
```

	id	SK_DATE	SK_RICE_TYPE	SK_MARKET	PRICE
count	33339.00	33339.00	33339.00	33339.00	33339.00
mean	16670.00	20178739.06	7.66	11.38	11628.06
std	9624.28	9309.65	3.34	12.39	2810.76
min	1.00	20160101.00	1.00	0.00	0.00
25%	8335.50	20170919.00	5.00	0.00	10000.00
50%	16670.00	20181121.00	8.00	7.00	11500.00
75%	25004.50	20181227.00	10.00	22.00	12725.00
max	33339.00	20190131.00	14.00	38.00	28000.00

#### 4. Visualisasi dengan boxplot dan histogram

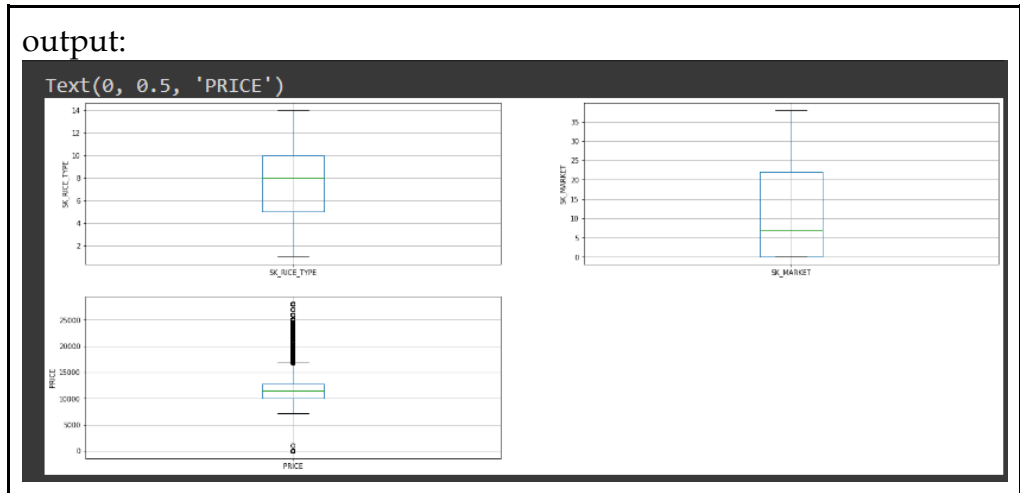
```
# menggambarkan boxplots untuk visualisasi outlier
plt.figure(figsize=(24,20))

plt.subplot(4, 2, 3)
fig = df.boxplot(column='SK_RICE_TYPE')
fig.set_title('')
fig.set_ylabel('SK_RICE_TYPE')
```

```
plt.subplot(4, 2, 4)
fig = df.boxplot(column='SK_MARKET')
fig.set_title('')
fig.set_ylabel('SK_MARKET')

plt.subplot(4, 2, 5)
fig = df.boxplot(column='PRICE')
fig.set_title('')
fig.set_ylabel('PRICE')
```

output:



```
# plot histogram untuk mengecek distribusi
plt.figure(figsize=(24,20))
```

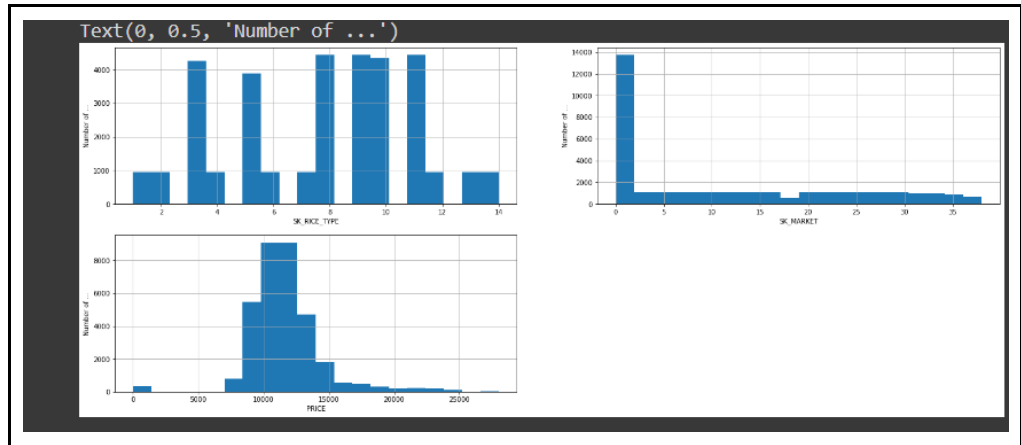
```
plt.subplot(4, 2, 3)
fig = df['SK_RICE_TYPE'].hist(bins=20)
fig.set_xlabel('SK_RICE_TYPE')
fig.set_ylabel('Number of ...')
```

```
plt.subplot(4, 2, 4)
fig = df['SK_MARKET'].hist(bins=20)
fig.set_xlabel('SK_MARKET')
fig.set_ylabel('Number of ...')
```

```
plt.subplot(4, 2, 5)
fig = df['PRICE'].hist(bins=20)
fig.set_xlabel('PRICE')
fig.set_ylabel('Number of ...')
```

output:





## 5. Split data

```
X = df.drop(['PRICE'], axis=1)
y = df['PRICE']
```

```
# split X and y into training and testing sets
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 0)
```

```
# check the shape of X_train and X_test
X_train.shape, X_test.shape
```

output:

```
X = df.drop(['PRICE'], axis=1)
y = df['PRICE']

# split X and y into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 0)

# check the shape of X_train and X_test
X_train.shape, X_test.shape

((26671, 4), (6668, 4))
```

## 6. Feature Scaling

```
cols = X_train.columns
```



```

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train = pd.DataFrame(X_train, columns=[cols])
X_test = pd.DataFrame(X_test, columns=[cols])

X_train.describe()

```

output:

	id	SK_DATE	SK_RICE_TYPE	SK_MARKET
count	2.667100e+04	2.667100e+04	2.667100e+04	2.667100e+04
mean	-1.031257e-16	1.310739e-13	-3.355354e-16	3.666637e-16
std	1.000019e+00	1.000019e+00	1.000019e+00	1.000019e+00
min	-1.730502e+00	-2.002152e+00	-1.999671e+00	-9.204392e-01
25%	-8.627302e-01	-8.402062e-01	-8.003697e-01	-9.204392e-01
50%	4.716332e-04	2.557851e-01	9.910614e-02	-3.559564e-01
75%	8.649199e-01	2.671714e-01	6.987567e-01	8.536496e-01
max	1.732069e+00	1.223626e+00	1.898058e+00	2.143896e+00

## 7. Menjalankan SVM dengan default hyperparameters

### SVM dengan C = 1.0

```

# import SVC classifier
from sklearn.svm import SVC

# import metrics to compute accuracy
from sklearn.metrics import accuracy_score

# instantiate classifier with default hyperparameters
svc=SVC()

# fit classifier to training set
svc.fit(X_train,y_train)

# make predictions on test set
y_pred=svc.predict(X_test)

```



```
# compute and print accuracy score
print('Model accuracy score with default hyperparameters:
{0:0.4f}'.format(accuracy_score(y_test, y_pred)))
```

output:

```
Model accuracy score with default hyperparameters: 0.2469
```

### SVM dengan C = 1000.0

```
# instantiate classifier with rbf kernel and C=1000
svc=SVC(C=1000.0)

# fit classifier to training set
svc.fit(X_train,y_train)

# make predictions on test set
y_pred=svc.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with rbf kernel and C=1000.0 :
{0:0.4f}'.format(accuracy_score(y_test, y_pred)))
```

output:

```
Model accuracy score with rbf kernel and C=1000.0 : 0.4676
```

## Latihan 1.2.2

### Penjelasan singkat :

Pada latihan ini akan diminta menjalankan SVM dengan polynomial kernel, sigmoid kernel, dan linear kernel menggunakan python.

### 8. Menjalankan SVM dengan polynomial kernel

#### SVM dengan C = 1.0

```
# instantiate classifier with polynomial kernel and C=1.0
poly_svc=SVC(kernel='poly', C=1.0)

# fit classifier to training set
poly_svc.fit(X_train,y_train)
```



```
# make predictions on test set
y_pred=poly_svc.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with polynomial kernel and C=1.0
: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

output:

```
[18] # no 8 instantiate classifier with polynomial kernel and C=1.0
poly_svc=SVC(kernel='poly', C=1.0)

# fit classifier to training set
poly_svc.fit(X_train,y_train)

# make predictions on test set
y_pred=poly_svc.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with polynomial kernel and C=1.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

Model accuracy score with polynomial kernel and C=1.0 : 0.2005
```

SVM dengan C = 100.0

```
# instantiate classifier with polynomial kernel and C=100.0
poly_svc100=SVC(kernel='poly', C=100.0)

# fit classifier to training set
poly_svc100.fit(X_train, y_train)

# make predictions on test set
y_pred=poly_svc100.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with polynomial kernel and C=1.0
: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

output:

```
✓ # instantiate classifier with polynomial kernel and C=100.0
poly_svc100=SVC(kernel='poly', C=100.0)

# fit classifier to training set
poly_svc100.fit(X_train, y_train)

# make predictions on test set
y_pred=poly_svc100.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with polynomial kernel and C=1.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

Model accuracy score with polynomial kernel and C=1.0 : 0.2867
```

## 9. Menjalankan SVM dengan sigmoid kernel SVM dengan C = 1.0



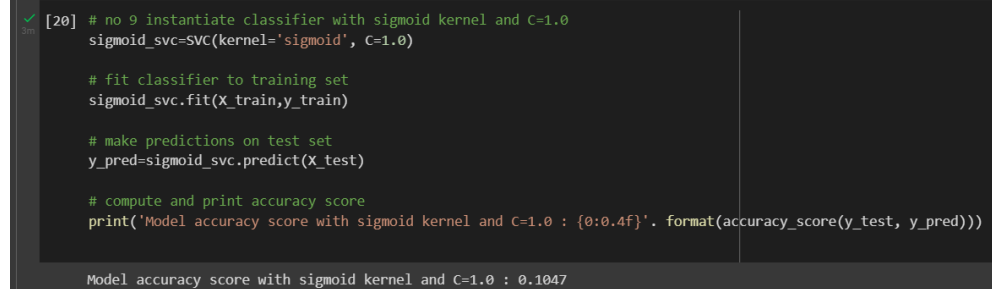
```
# instantiate classifier with sigmoid kernel and C=1.0
sigmoid_svc=SVC(kernel='sigmoid', C=1.0)

# fit classifier to training set
sigmoid_svc.fit(X_train,y_train)

# make predictions on test set
y_pred=sigmoid_svc.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with sigmoid kernel and C=1.0 :
{0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

output:



```
[20] # no 9 instantiate classifier with sigmoid kernel and C=1.0
      sigmoid_svc=SVC(kernel='sigmoid', C=1.0)

      # fit classifier to training set
      sigmoid_svc.fit(X_train,y_train)

      # make predictions on test set
      y_pred=sigmoid_svc.predict(X_test)

      # compute and print accuracy score
      print('Model accuracy score with sigmoid kernel and C=1.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

      Model accuracy score with sigmoid kernel and C=1.0 : 0.1047
```

### SVM dengan C = 100.0

```
# instantiate classifier with sigmoid kernel and C=100.0
sigmoid_svc100=SVC(kernel='sigmoid', C=100.0)

# fit classifier to training set
sigmoid_svc100.fit(X_train,y_train)

# make predictions on test set
y_pred=sigmoid_svc100.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with sigmoid kernel and C=100.0
: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

output:

```
[21] # instantiate classifier with sigmoid kernel and C=100.0
      sigmoid_svc100=SVC(kernel='sigmoid', C=100.0)

      # fit classifier to training set
      sigmoid_svc100.fit(X_train,y_train)

      # make predictions on test set
      y_pred=sigmoid_svc100.predict(X_test)

      # compute and print accuracy score
      print('Model accuracy score with sigmoid kernel and C=100.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

Model accuracy score with sigmoid kernel and C=100.0 : 0.1254
```

## 10. Menjalankan SVM dengan Linear Kernel

### SVM dengan C = 1.0

```
# instantiate classifier with linear kernel and C=1.0
linear_svc=SVC(kernel='linear', C=1.0)

# fit classifier to training set
linear_svc.fit(X_train,y_train)

# make predictions on test set
y_pred_test=linear_svc.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with linear kernel and C=1.0 :
{0:0.4f}'. format(accuracy_score(y_test, y_pred_test)))
```

output:

```
# no 10 instantiate classifier with linear kernel and C=1.0
linear_svc=SVC(kernel='linear', C=1.0)

# fit classifier to training set
linear_svc.fit(X_train,y_train)

# make predictions on test set
y_pred_test=linear_svc.predict(X_test)

# compute and print accuracy score
print('Model accuracy score with linear kernel and C=1.0 : {0:0.4f}'. format(accuracy_score(y_test, y_pred_test)))

Model accuracy score with linear kernel and C=1.0 : 0.1540
```

### SVM dengan C = 100.0

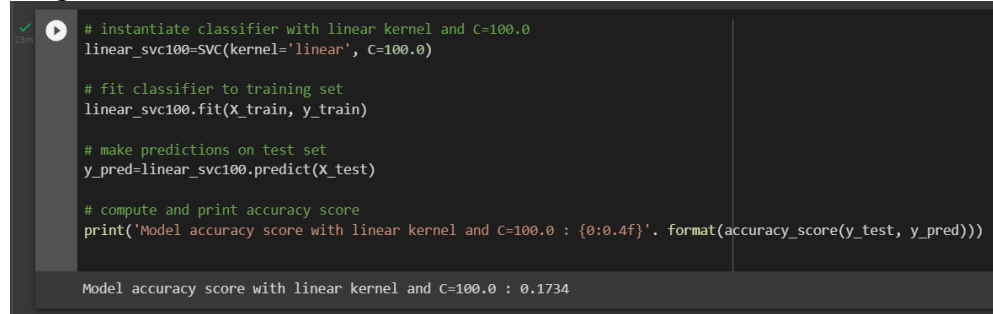
```
# instantiate classifier with linear kernel and C=100.0
linear_svc100=SVC(kernel='linear', C=100.0)

# fit classifier to training set
linear_svc100.fit(X_train, y_train)

# make predictions on test set
y_pred=linear_svc100.predict(X_test)
```

```
# compute and print accuracy score
print('Model accuracy score with linear kernel and C=100.0 :
{0:0.4f}'.format(accuracy_score(y_test, y_pred)))
```

output:



```
# instantiate classifier with linear kernel and C=100.0
linear_svc100=SVC(kernel='linear', C=100.0)

# fit classifier to training set
linear_svc100.fit(X_train, y_train)

# make predictions on test set
y_pred=linear_svc100.predict(X_test)

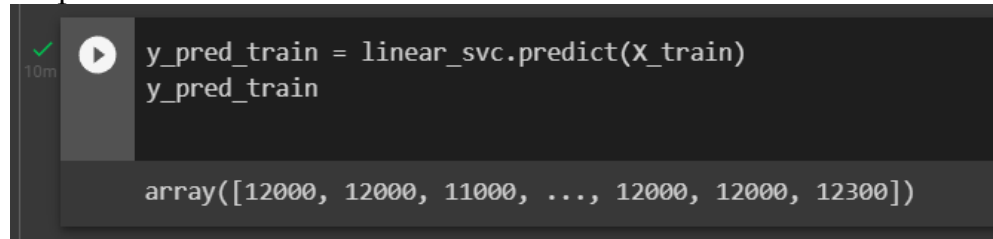
# compute and print accuracy score
print('Model accuracy score with linear kernel and C=100.0 : {0:0.4f}'.format(accuracy_score(y_test, y_pred)))

Model accuracy score with linear kernel and C=100.0 : 0.1734
```

Membandingkan akurasi train-set dan test\_set

```
y_pred_train = linear_svc.predict(X_train)
y_pred_train
```

output:

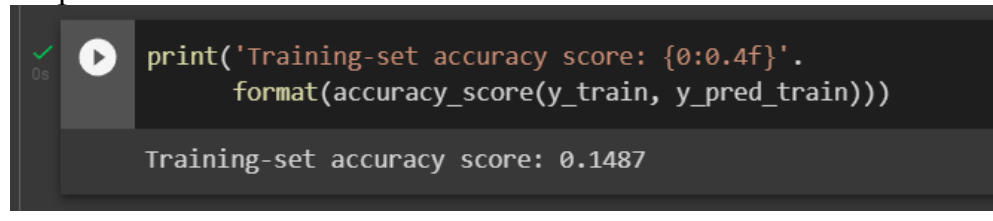


```
y_pred_train = linear_svc.predict(X_train)
y_pred_train

array([12000, 12000, 11000, ..., 12000, 12000, 12300])
```

```
print('Training-set accuracy score: {0:0.4f}'.
format(accuracy_score(y_train, y_pred_train)))
```

output:



```
print('Training-set accuracy score: {0:0.4f}'.
      format(accuracy_score(y_train, y_pred_train)))

Training-set accuracy score: 0.1487
```

Mengecek adanya overfitting dan underfitting

```
# print the scores on training and test set
print('Training set score:
{:.4f}'.format(linear_svc.score(X_train, y_train)))

print('Test set score:
{:.4f}'.format(linear_svc.score(X_test, y_test)))
```

output:

```
[26] # print the scores on training and test set
      print('Training set score: {:.4f}'.format(linear_svc.score(X_train, y_train)))

      print('Test set score: {:.4f}'.format(linear_svc.score(X_test, y_test)))

Training set score: 0.1487
Test set score: 0.1540
```

Membandingkan akurasi model dan akurasi null

```
# check null accuracy score
null_accuracy = (3306/(3306+274))

print('Null accuracy score: {0:0.4f}'.
      format(null_accuracy))
```

output:

```
# check null accuracy score
null_accuracy = (3306/(3306+274))

print('Null accuracy score: {0:0.4f}'. format(null_accuracy))

Null accuracy score: 0.9235
```

## CEK LIST

Elemen Kompetensi	No Latihan	Penyelesaian	
		Selesai	Tidak selesai
1	1.1.1	✓	
	1.1.2	✓	
	1.2.1	✓	
	1.2.2	✓	

## FORM UMPAN BALIK

Elemen Kompetensi	Tingkat Kesulitan	Tingkat Ketertarikan	Waktu Penyelesaian dalam menit
Mengimplementasikan SVR dalam kasus yang	<input type="checkbox"/> Sangat Mudah	<input type="checkbox"/> Tidak Tertarik	





diberikan	<input type="checkbox"/> Mudah	<input type="checkbox"/> Cukup Tertarik	70
	<input type="checkbox"/> ✓ Biasa	<input type="checkbox"/> Tertarik	
	<input type="checkbox"/> Sulit	<input type="checkbox"/> ✓ Sangat Tertarik	
	<input type="checkbox"/> Sangat Sulit		