**EXP NO : 6**

**Support Vector Machine model usage for classification and regression**

**AIM:**

To demonstrate the use of Support Vector Machine (SVM) for regression using the scikit-learn library. The experiment includes generating synthetic data, splitting it into training and testing sets, standardizing features, training an SVM regression model, making predictions, evaluating the model, and visualizing the results.

**INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) REQUIRED:**

JUPYTER NOTEBOOK

**REQUIRED LIBRARIES FOR PYTHON:**

∙ Tensorflow

∙ Numpy

∙ Matplotlib

∙ scikit-learn

**PROCEDURE:**

1. **Import Libraries:**

∙ Import the necessary libraries including scikit-learn, NumPy, and Matplotlib.

2. **Generate Example Data:**

∙ Use scikit-learn's **datasets.make\_regression** function to generate synthetic data

for regression.

3. **Split Data:**

∙ Split the generated data into training and testing sets using **train\_test\_split** from

scikit-learn.

4. **Standardize Features:**

∙ Standardize the features using **StandardScaler** from scikit-learn.

5. **Create SVM Regression Model:**

∙ Create an SVM Regression model with a linear kernel using **SVR** from scikit

learn.

6. **Train the Model:**

∙ Train the SVM Regression model using the training data.

**Support Vector Machine model usage for classification and regression**

**Exp no: 9**7. **Make Predictions:**

∙ Use the trained model to make predictions on the test set.

8. **Evaluate Model:**

∙ Evaluate the performance of the model using Mean Squared Error.

9. **Visualize Results:**

∙ Visualize the regression line and compare it with the actual data using Matplotlib.

**PROGRAM:**

#support vector machine in scikit learn library

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error

import numpy as np

# Generate some example data

X, y = datasets.make\_regression(n\_samples=100, n\_features=1, noise=10,

random\_state=42)

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2,

random\_state=42)

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create SVM Regression model

svm\_regressor = SVR(kernel='linear', C=1.0)

# Train the SVM Regression model

svm\_regressor.fit(X\_train, y\_train)

# Make predictions on the test set

y\_pred = svm\_regressor.predict(X\_test)

# Evaluate the performance using Mean Squared Error

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse}')

# Visualize the regression line

import matplotlib.pyplot as plt

plt.scatter(X\_test, y\_test, color='black', label='Actual Data')

plt.plot(X\_test,

Regression')

y\_pred, color='blue', linewidth=3, label='SVR

Linear

plt.xlabel('Feature')

plt.ylabel('Target')

plt.title('Support Vector Machine Regression')

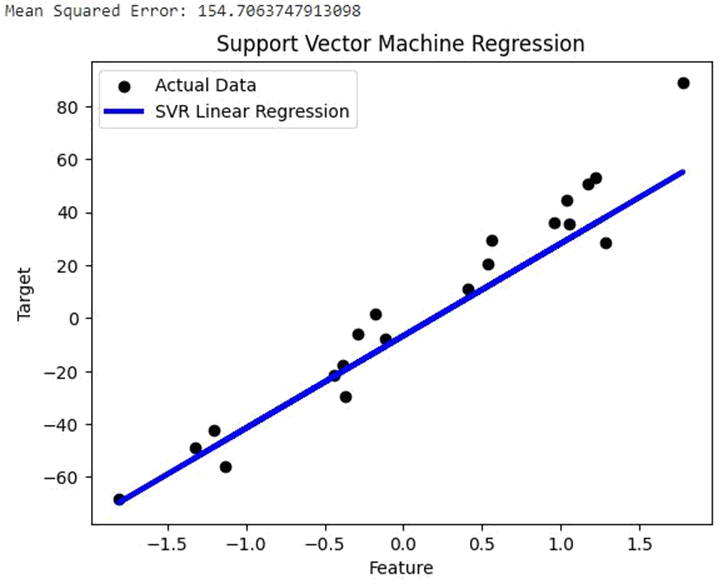
plt.legend()

plt.show()

**OUTPUT:**

The output of this code includes the Mean Squared Error and a visualization of the regression

line compared to the actual data.



**RESULT:**

The visualization shows the SVM Regression model captures the underlying pattern in the data.

The blue line represents the regression line predicted by the model, and the black dots represent

the actual data points.