Chapter 2 : Section 4

**Support Vector Regression (SVR)**

**2.4.1 Implementation of SVR in Python**

* Data Preparation: We continue with our previous problem of "Bluff Detection"

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib**.**pyplot **as** plt

#*----------- data preprocessing ------------------*

#*Importing the dataset. previous problem of "Bluff Detection"*

datASet = pd**.read\_csv**("Position\_Salaries.csv")

X = datASet**.**iloc[:, 1:2]**.**values

y = datASet**.**iloc[:, 2]**.**values

* Fitting SVR to the dataset:

We'll just import ***SVR*** class from the sikatLearn ***svm*** library because ***SVR*** is actually a support vector machine SVM for regression.

#*Fitting SVR to the dataset*

**from** sklearn**.**svm **import** SVR

regressor = **SVR**(kernel='rbf')

regressor**.fit**(X, y)

* Choosing kernel:
* We have many parameters for many ML models. But the most important parameter that we need to focus on is the ***kernel***. The kernel is whether you want a linear SVR or a polynomial SVR or Gaussian SVR.***'linear'***, ***'poly'***, ***'rbf'***, ***'sigmoid'***, ***'precomputed'*** are the most common kernels.
* The one we want right now is the **'rbf'** kernel. And why is that? Because we know our problem is non-linear. The *linear kernel* would make a *linear machine model* that would *not* therefore be *appropriate* for a *nonlinear problem*.
* And then we have the choice between ***'poly'*** and ***'rbf'***, *both* these kernels *could* *work* for our problem. But we're going to take the most common one which is the Gaussian kernel and therefore ***'rbf'*** here.
* Prediction without scaling:

In ***y\_pred***, we used ***[[6.5]]***, since Parameter must be ***2-D array***.

y\_prd = regressor**.predict**([[6.5]])  #*[[6.5]], since Parmeter must be 2-D array*

**print**(y\_prd)

***Before scaling is applied***

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib**.**pyplot **as** plt

#*----------- data preprocessing ------------------*

#*Importing the dataset. previous problem of "Bluff Detection"*

datASet = pd**.read\_csv**("Position\_Salaries.csv")

X = datASet**.**iloc[:, 1:2]**.**values

y = datASet**.**iloc[:, 2]**.**values

#*Splitting the dataset into the Training set and Test set: No need here*

#*Feature Scaling*

#*Fitting SVR to the dataset*

**from** sklearn**.**svm **import** SVR

regressor = **SVR**(kernel='rbf')

regressor**.fit**(X, y)

#*---------------- Visualising the SVR results ----------------*

plt**.scatter**(X, y, color = "red")

plt**.plot**(X, regressor**.predict**(X), color = "blue")

#*------------ prediction -----------*

"""

Here we will predict the output for level 6.5

because the candidate has 4+ years' experience as a regional manager,

so he must be somewhere between levels 7 and 6.

"""

y\_prd = regressor**.predict**([[6.5]])  #*[[6.5]], since Parmeter must be 2-D array*

**print**(y\_prd)

#*python prctc\_SVR.py*

* Feature scaling needed: SVR does not apply automatic Feature-Scaling as Simple linear and Multiple linear Model. SVR is not a common class as Linearregression.

Feature Scaling applied:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib**.**pyplot **as** plt

#*----------- data preprocessing ------------------*

#*Importing the dataset. previous problem of "Bluff Detection"*

datASet = pd**.read\_csv**("Position\_Salaries.csv")

X = datASet**.**iloc[:, 1:2]**.**values

y = datASet**.**iloc[:, 2]**.**values

#*Splitting the dataset into the Training set and Test set: No need here*

#*Feature Scaling*

**from** sklearn**.**preprocessing **import** StandardScaler

sc\_X = **StandardScaler**()

sc\_y = **StandardScaler**()

X\_scaled = sc\_X**.fit\_transform**(X)

y\_scaled = sc\_y**.fit\_transform**(y**.reshape**(-1, 1))

* New prediction:
* No fit\_transform(6.5) only transform(): We don’t use ***fit\_transfrom()*** because model is already fitted. We only use ***transform()***.

sc\_X**.transform**(np**.array**([[6.5]]))

* Transform 6.5 to array: Since expecting ***2-d-array*** we need to pass ***6.5*** as array the trick is given below:

np**.array**([[6.5]])

* Reverse Scaling (inverse scale transformation): Since is applied and we passed 6.5 as scaled array. The result y\_pred is also is scaled output. So we need to Reverse scale y\_pred.

y\_inv\_sc = sc\_y**.inverse\_transform**(y\_prd)

y\_prd = regressor**.predict**(sc\_X**.transform**(np**.array**([[6.5]])))

**print**("prediction under scaled data : ", y\_prd)

y\_inv\_sc = sc\_y**.inverse\_transform**(y\_prd)

**print**("Reverse scaled prediction : ", y\_inv\_sc)

* Notice in "Visualizing the SVR results" all scaled-dada is used to plot the model.
* SVR fitted to the place where most observations appear.

Practiced version

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib**.**pyplot **as** plt

#*----------- data preprocessing ------------------*

#*Importing the dataset. previous problem of "Bluff Detection"*

datASet = pd**.read\_csv**("Position\_Salaries.csv")

X = datASet**.**iloc[:, 1:2]**.**values

y = datASet**.**iloc[:, 2]**.**values

#*Splitting the dataset into the Training set and Test set: No need here*

#*Feature Scaling*

**from** sklearn**.**preprocessing **import** StandardScaler

sc\_X = **StandardScaler**()

sc\_y = **StandardScaler**()

X\_scaled = sc\_X**.fit\_transform**(X)

y\_scaled = sc\_y**.fit\_transform**(y**.reshape**(-1, 1))

#*Fitting SVR to the dataset*

**from** sklearn**.**svm **import** SVR

regressor = **SVR**(kernel='rbf')

#*regressor.fit(X, y)*

regressor**.fit**(X\_scaled, y\_scaled)

#*---------------- Visualising the SVR results ----------------*

#*Feature scaling is needed*

#*plt.scatter(X, y, color = 'red')*

#*plt.plot(X, regressor.predict(X), color = 'blue')*

plt**.scatter**(X\_scaled, y\_scaled, color = "red")

plt**.plot**(X\_scaled, regressor**.predict**(X\_scaled), color = "blue")

plt**.title**('Truth or Bluff (SVR)')

plt**.xlabel**('Position level')

plt**.ylabel**('Salary')

plt**.show**()

#*Visualizing the SVR results (for higher resolution and smoother curve)*

X\_grid = np**.arange**(min(X\_scaled), max(X\_scaled), 0.1)

X\_grid = X\_grid**.reshape**((len(X\_grid), 1))

plt**.scatter**(X\_scaled, y\_scaled, color = "red")

plt**.plot**(X\_grid, regressor**.predict**(X\_grid), color = 'blue')

plt**.title**('Truth or Bluff (SVR)')

plt**.xlabel**('Position level')

plt**.ylabel**('Salary')

plt**.show**()

#*------------ prediction -----------*

"""

Here we will predict the output for level 6.5

because the candidate has 4+ years' experience as a regional manager,

so he must be somewhere between levels 7 and 6.

"""

#*y\_prd = regressor.predict([[6.5]])  # [[6.5]], since Parmeter must be 2-D array*

#*We need to transfom 6.5 in our scaling*

#*y\_prd = regressor.predict(sc\_X.transform([[6.5]])) # alternative*

y\_prd = regressor**.predict**(sc\_X**.transform**(np**.array**([[6.5]])))

**print**("prediction under scaled data : ", y\_prd)

y\_inv\_sc = sc\_y**.inverse\_transform**(y\_prd)

**print**("Reverse scaled prediction : ", y\_inv\_sc)

|  |  |
| --- | --- |
|  |  |

***Instructor version: 6.5 is not transformed***

#*Support Vector Regression (SVR)*

#*Importing the libraries*

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

**import** pandas **as** pd

#*Importing the dataset*

dataset = pd**.read\_csv**('Position\_Salaries.csv')

X = dataset**.**iloc[:, 1:-1]**.**values

y = dataset**.**iloc[:, -1]**.**values

#*Feature Scaling*

**from** sklearn**.**preprocessing **import** StandardScaler

sc\_X = **StandardScaler**()

sc\_y = **StandardScaler**()

X = sc\_X**.fit\_transform**(X)

y = sc\_y**.fit\_transform**(y**.reshape**(-1,1))

#*Training the SVR model on the whole dataset*

**from** sklearn**.**svm **import** SVR

regressor = **SVR**(kernel = 'rbf')

regressor**.fit**(X, y)

#*Predicting a new result*

y\_pred = regressor**.predict**([[6.5]])

y\_pred = sc\_y**.inverse\_transform**(y\_pred)

#*Visualising the SVR results*

plt**.scatter**(X, y, color = 'red')

plt**.plot**(X, regressor**.predict**(X), color = 'blue')

plt**.title**('Truth or Bluff (SVR)')

plt**.xlabel**('Position level')

plt**.ylabel**('Salary')

plt**.show**()

#*Visualising the SVR results (for higher resolution and smoother curve)*

X\_grid = np**.arange**(min(X), max(X), 0.01) #*choice of 0.01 instead of 0.1 step because the data is feature scaled*

X\_grid = X\_grid**.reshape**((len(X\_grid), 1))

plt**.scatter**(X, y, color = 'red')

plt**.plot**(X\_grid, regressor**.predict**(X\_grid), color = 'blue')

plt**.title**('Truth or Bluff (SVR)')

plt**.xlabel**('Position level')

plt**.ylabel**('Salary')

plt**.show**()