# CSE 4020 - MACHINE LEARNING

# Lab 29+30

# Support Vector Machine(SVM)

# Submitted by: Alokam Nikhitha(19BCE2555)

Ques: Train SVM classifier using sklearn digits dataset (i.e. from sklearn.datasets import load\_digits) and then

1. Measure accuracy of your model using different kernels such as rbf, poly and linear.

2. Tune your model further using regularization and gamma parameters and try to come up with highest accuracy score.

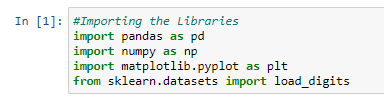
3. Use 80% of samples as training data size.

Dataset Used: load\_digits dataset from sklearn

**Procedure:**

* Using pandas, we first import the dataset into our workspace.
* The next step is to choose the independent and dependent variables that will be used in our regression model.
* After that, we divided our data into two sets: training and test.
* Then, using the 'rbf' kernel, we must initialise our Support Vector Machine classifier and fit it to the X\_train and y\_train attributes.
* Use the ‘linear' and ‘polynomial' kernels to repeat the previous process.
* Then, using the results predicted by X\_test on the 'rbf', 'linear', and 'polynomial' kernels, we establish three variables to store the X\_test result
* Finally, we compute evaluation metrics for each of the three kernels.

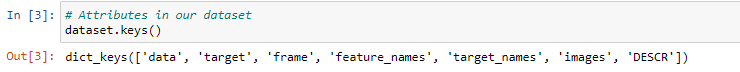
**CODE SNIPPETS AND EXPLAINATION**



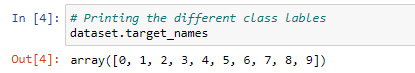
Importing the required Libraries



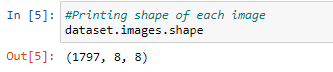
Importing the digits Dataset



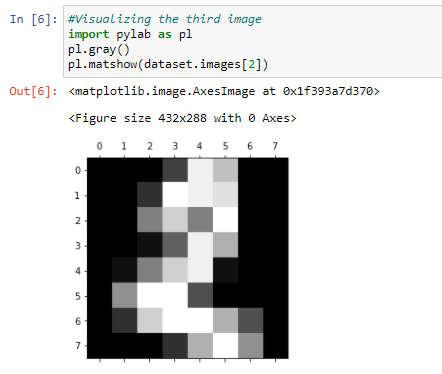
Listing the Attributes in our Dataset



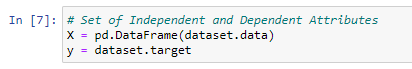
Printing the Different Class Labels



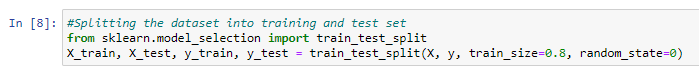
Print the image shape



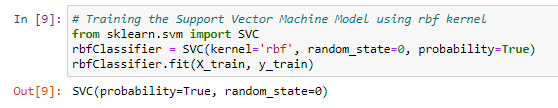
Visualizing the Third Image in the Dataset



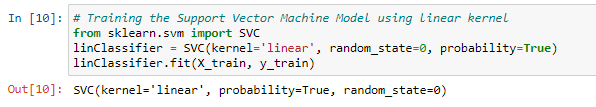
Taking the Independent and Depending Attributes



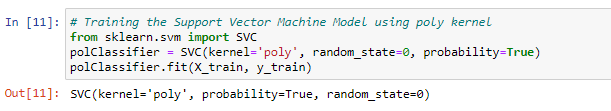
Splitting the dataset into Training set and Test set



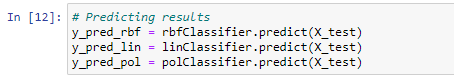
Training theSVM Model using rbf kernel



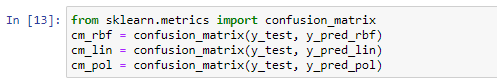
Training SVM model using Linear kernel



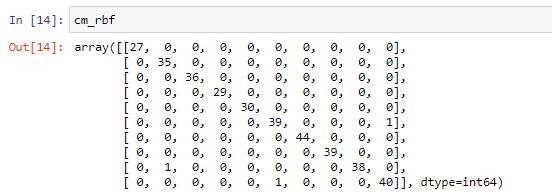
Training SVM model using poly kernel Model



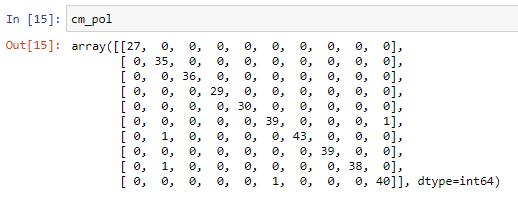
Predicting Results for various Kernels



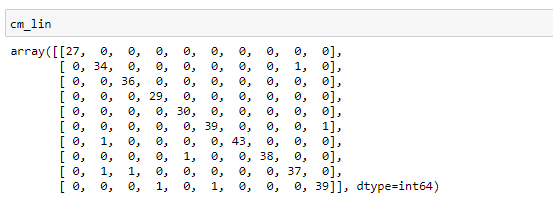
Confusion Matrix for various Kernels



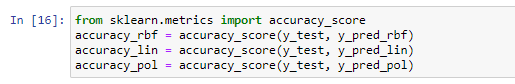
Confusion Matrix for rbf Kernel



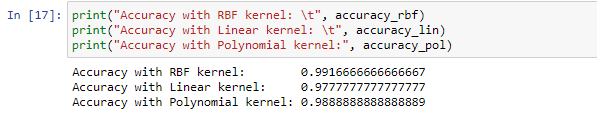
Confusion Matrix for poly Kernel



Confusion Matrix for linear Kernel



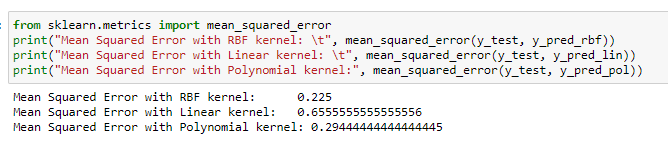
Calculating Accuracy for various Kernels



Printing Accuracy for Different Kernels

We can see here that the accuracy with rbf kernel is max and thus it is most suitable.

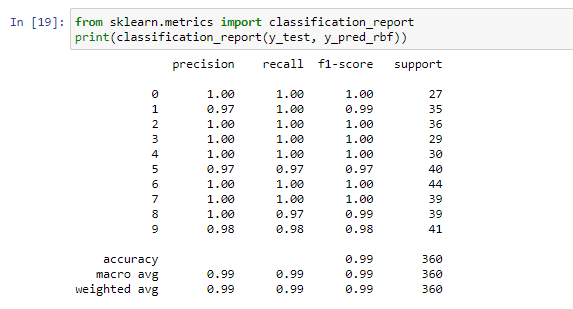
Accuracy(Linear) < Accuracy(Poly) < Accuracy(rbf)

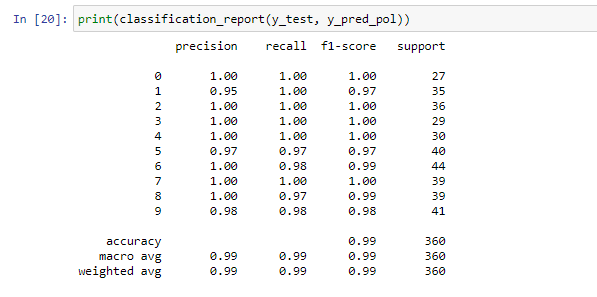


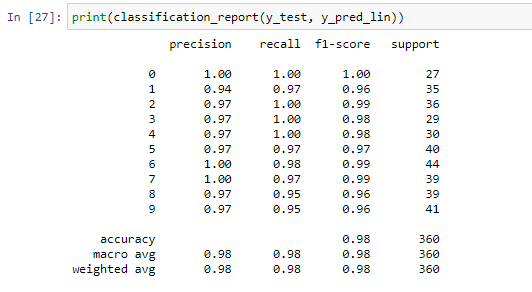
Mean Squared Errors with various Kernels

From here we can again infer that MSE in least for rbf kernel and hence it is the most suitable kernel for our dataset in Support Vector Classifier.

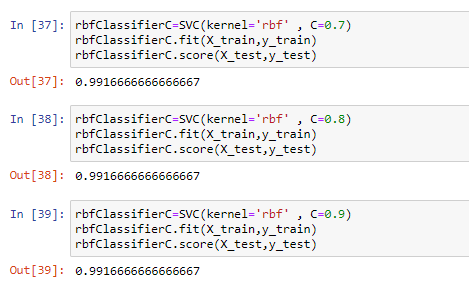
MSE(rbf) < MSE(Poly) < MSE(Linear)







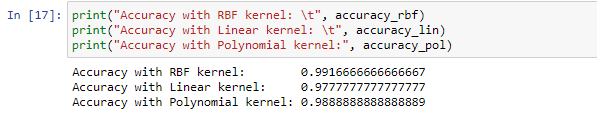
Here we have printed the classification report of Support Vector Classifier with all three kernels.

Here we have tried to tune in the C value for rbf kernel. Initially we have used 0.3 as we increase the C value and we can see that the accuracy increases till C=0.7, after that C remains constant and there is no significant increase in models accuracy and hence the C value can be taken as C=0.7.

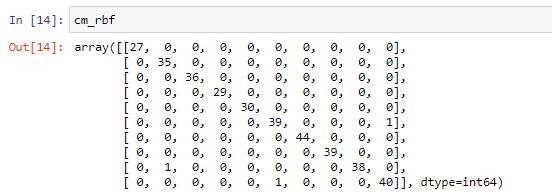
**Result and Conclusion:**

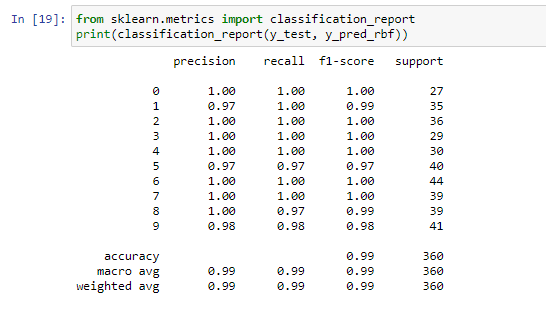
**Accuracy**

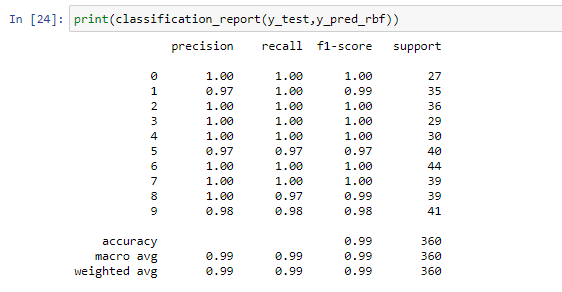


RBF Kernel-

• Model Accuracy = 99.1667%



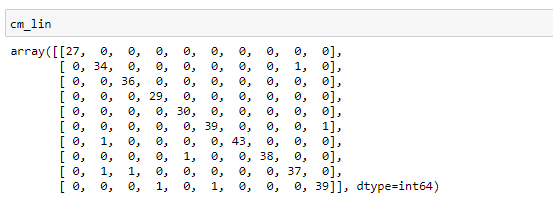


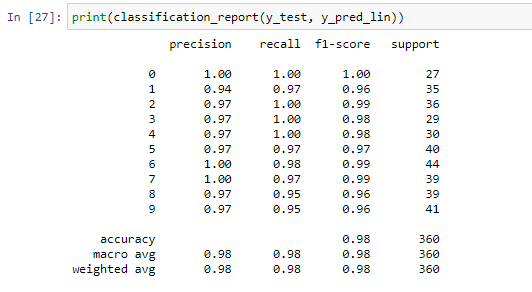


* Identified 0 = 27
* True 0 = 27
* Identified 1 = 34
* True 1 = 35
* Identified 2 = 36
* True 2 = 36
* Identified 3 = 29
* True 3 = 29
* Identified 4 = 30
* True 4 = 30
* Identified 5 = 39
* True 5 = 40
* Identified 6 = 44
* True 6 = 44
* Identified 7 = 39
* True 7 = 39
* Identified 8 = 39
* True 8 = 39
* Identified 9 = 40
* True 9 = 41
* Precision of 0 = 1.00
* Precision of 1 = 0.97
* Precision of 2 = 1.00
* Precision of 3 = 1.00
* Precision of 4 = 1.00
* Precision of 5 = 0.97
* Precision of 6 = 1.00
* Precision of 7 = 1.00
* Precision of 8 = 1.00
* Precision of 9 = 0.98
* Recall of 0 = 1.00
* Recall of 1 = 1.00
* Recall of 2 = 1.00
* Recall of 3 = 1.00
* Recall of 4 = 1.00
* Recall of 5 = 0.97
* Recall of 6 = 1.00
* Recall of 7 = 1.00
* Recall of 8 = 0.97
* Recall of 9 = 0.98

Linear Kernel:

• Model Accuracy = 97.77%

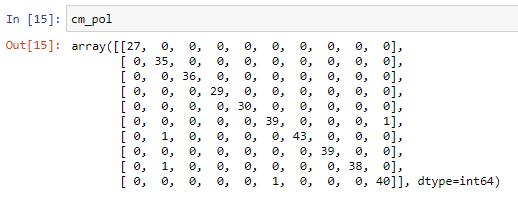


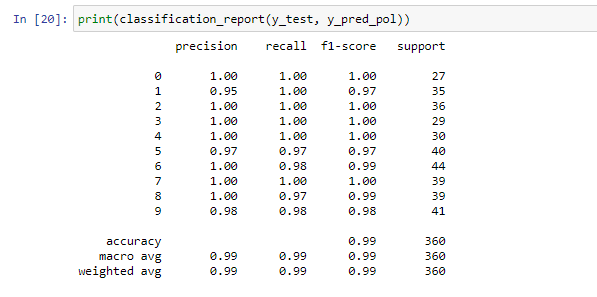


* Identified 0 = 27
* True 0 = 27
* Identified 1 = 33
* True 1 = 35
* Identified 2 = 35
* True 2 = 36
* Identified 3 = 28
* True 3 = 29
* Identified 4 = 29
* True 4 = 30
* Identified 5 = 39
* True 5 = 40
* Identified 6 = 44
* True 6 = 44
* Identified 7 = 39
* True 7 = 39
* Identified 8 = 38
* True 8 = 39
* Identified 9 = 40
* True 9 = 41
* Precision of 0 = 1.00
* Precision of 1 = 0.97
* Precision of 2 = 1.00
* Precision of 3 = 1.00
* Precision of 4 = 1.00
* Precision of 5 = 0.97
* Precision of 6 = 0.98
* Precision of 7 = 0.97
* Precision of 8 = 0.95
* Precision of 9 = 0.95
* Recall of 0 = 1.00
* Recall of 1 = 0.96
* Recall of 2 = 0.99
* Recall of 3 = 0.98
* Recall of 4 = 0.98
* Recall of 5 = 0.97
* Recall of 6 = 0.99
* Recall of 7 = 0.99
* Recall of 8 = 0.96
* Recall of 9 = 0.96

Poly Kernel:

• Model Accuracy = 98.88%





* Identified 0 = 27
* True 0 = 27
* Identified 1 = 35
* True 1 = 35
* Identified 2 = 36
* True 2 = 36
* Identified 3 = 29
* True 3 = 29
* Identified 4 = 30
* True 4 = 30
* Identified 5 = 39
* True 5 = 40
* Identified 6 = 44
* True 6 = 44
* Identified 7 = 39
* True 7 = 39
* Identified 8 = 39
* True 8 = 39
* Identified 9 = 40
* True 9 = 41
* Precision of 0 = 1.00
* Precision of 1 = 0.95
* Precision of 2 = 1.00
* Precision of 3 = 1.00
* Precision of 4 = 1.00
* Precision of 5 = 0.97
* Precision of 6 = 1.00
* Precision of 7 = 1.00
* Precision of 8 = 1.00
* Precision of 9 = 0.98
* Recall of 0 = 1.00
* Recall of 1 = 1.00
* Recall of 2 = 1.00
* Recall of 3 = 1.00
* Recall of 4 = 1.00
* Recall of 5 = 0.97
* Recall of 6 = 0.98
* Recall of 7 = 1.00
* Recall of 8 = 0.97
* Recall of 9 = 0.98