Pattern Recognition and Machine Learning

(Winter 2022)

Assignment 10: Support Vector Machine Classifier

Report

Question - 1:

Methodology:

Details of the SVM package used.

For this task, I imported the SVM package from the Scikit-learn library. There are 3 classes in the library SVM i.e., SVC, NuSVC and LinearSVC capable of performing binary and multi-class classification on the dataset.

For this assignment I used the SVC library of SVM package since, it accepts the parameter 'kernel'. And, I used LinearSVC for the linear kernel. Now, this will help us to determine the best value of generalization constant C of each kernel functions.

The dataset provided has 57 features with 4601 instances. The class of each instance was either 0 (Not Spam) or 1 (Spam).

While data pre-processing, I checked for nan/null values in the dataset and removed if any. I have scaled the feature values using the StandardScaler class of Scikit-Learn which standardizes features by subtracting the mean and scaling to unit variance for each column. This helps in reducing the training time for the SVM to train. Also, Support Vector Machine (SVM) optimization occurs by minimizing the decision vector w, the optimal hyperplane is influenced by the scale of the input features and it's therefore recommended that data be standardized (mean 0, var 1) prior to SVM model training. I then split the data into training and testing dataset.

<u>Task - 1:</u>

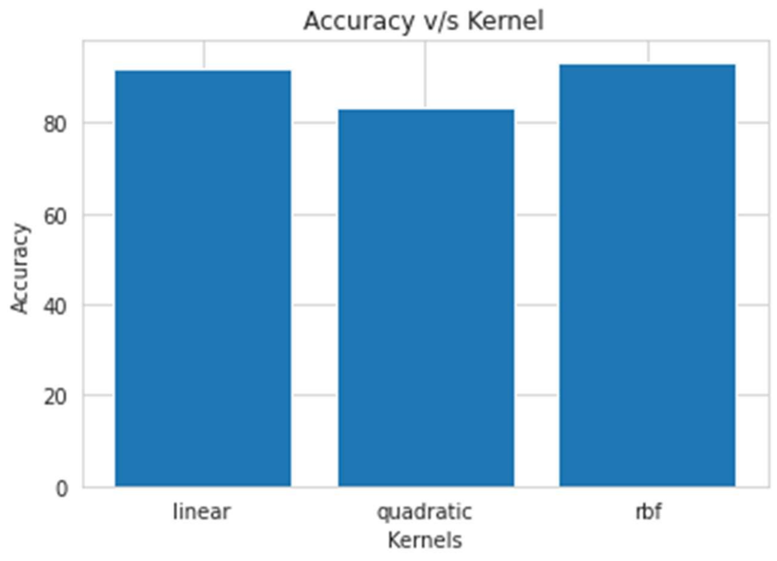
For this task, I used the LinearSVC for linear kernel. Then, I used the SVC library from SVM package for 'rbf' and 'quadratic' kernel. I used these libraries for each of the different types of clusters. The performance of different models was evaluated on the test dataset.

Support Vector Classifier with linear kernel 91.81752353367125%

Support Vector Classifier with quadratic kernel 83.2005792903693%

Support Vector Classifier with RBF kernel 93.33816075307749%

I also plotted the graph to check accuracy v/s kernel.



<u>Task - 2:</u>

For this task, I trained different models of SVM with different kernels over different values of the generalization constant C and evaluated the performance on the training and testing dataset.

I also plotted the plots to visualize the performance of the model.

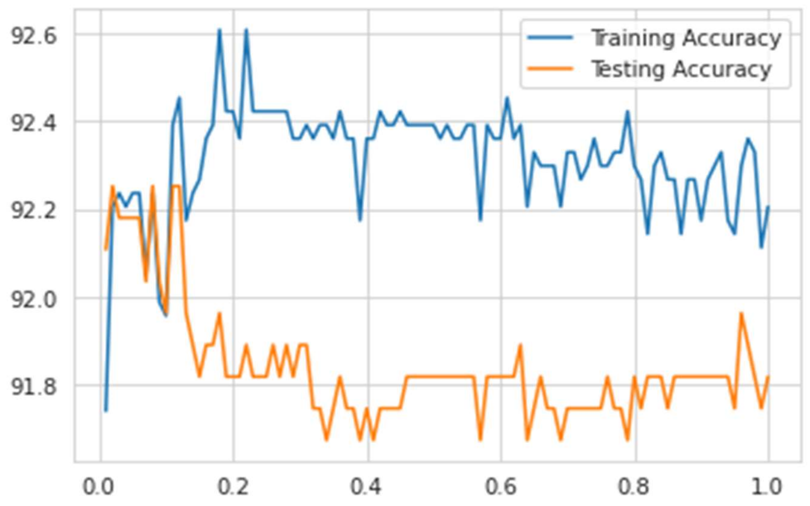
The best value of C obtained was the ones for which the models performed the best on the test dataset.

I also made the comparison table to compare the training and testing accuracies with the generalization constant C.

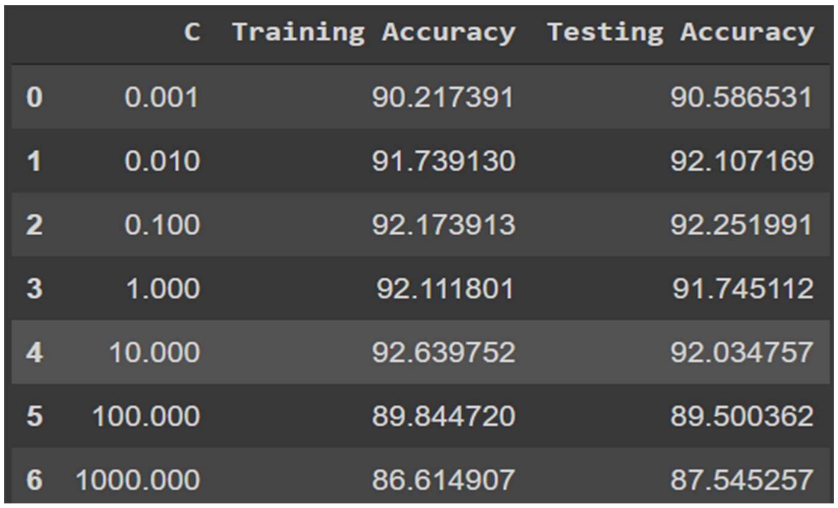
Experimental Results:

- **For 'linear' kernel.**

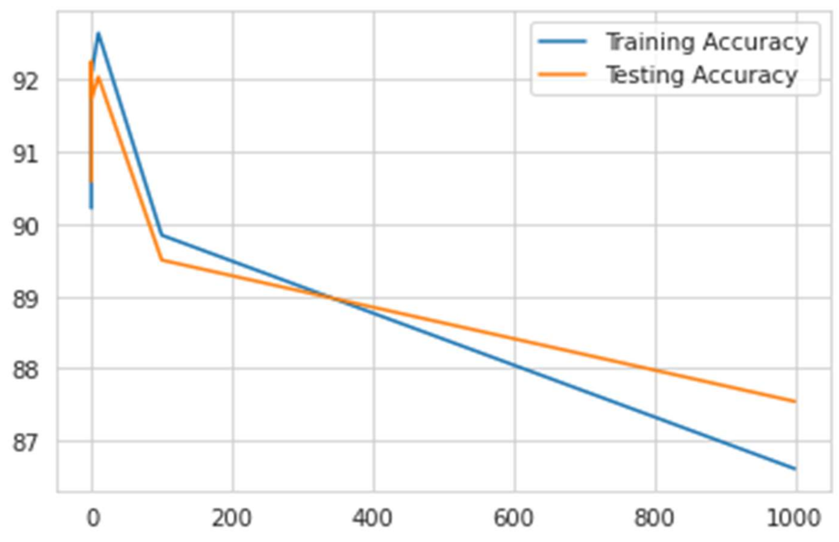
  - For values varying in (0.01, 0.02, ……., 0.99, 1.00) different models were trained and the following graph was obtained. Also, I made the comparison table in my colab file.



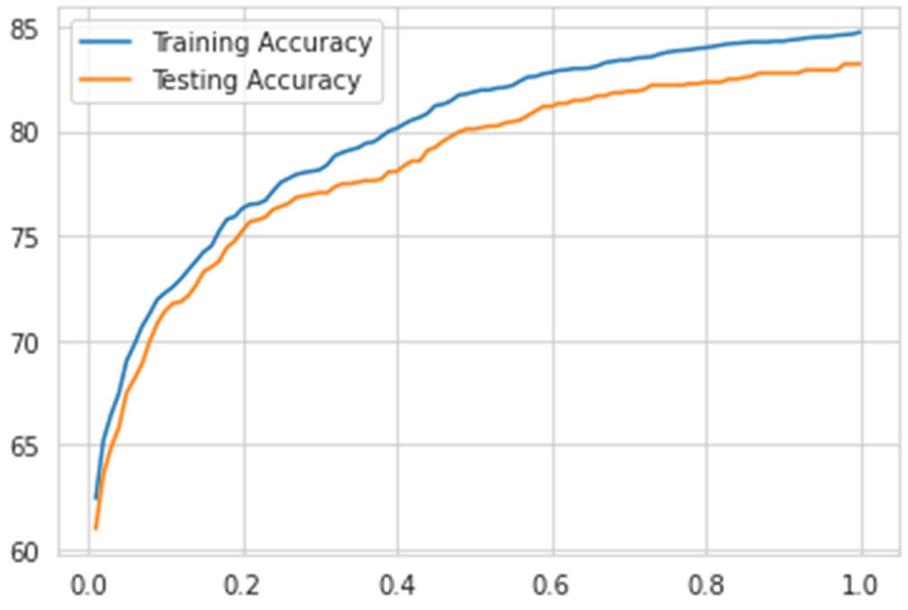
  - For values varying in (0.001, 0.01, 0.1, 1, 10, 100, 1000) different models were trained and the following comparison table and graph was obtained.



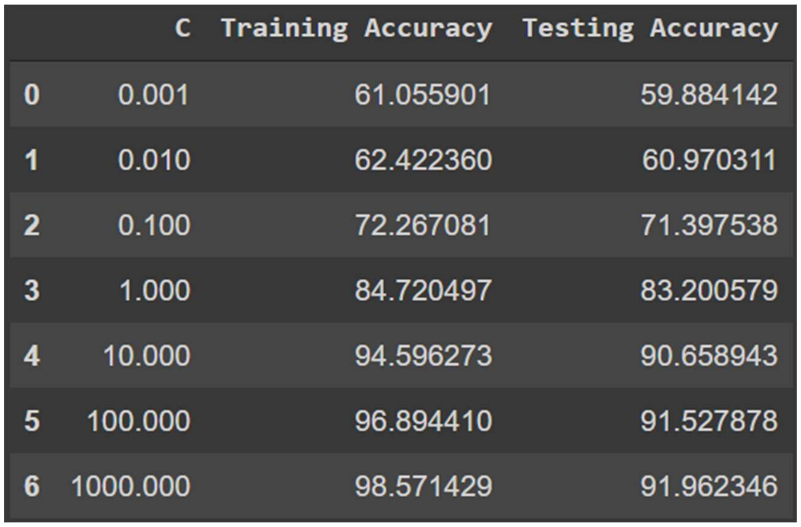
	C	Training Accuracy	Testing Accuracy
0	0.001	90.217391	90.586531
1	0.010	91.739130	92.107169
2	0.100	92.173913	92.251991
3	1.000	92.111801	91.745112
4	10.000	92.639752	92.034757
5	100.000	89.844720	89.500362
6	1000.000	86.614907	87.545257



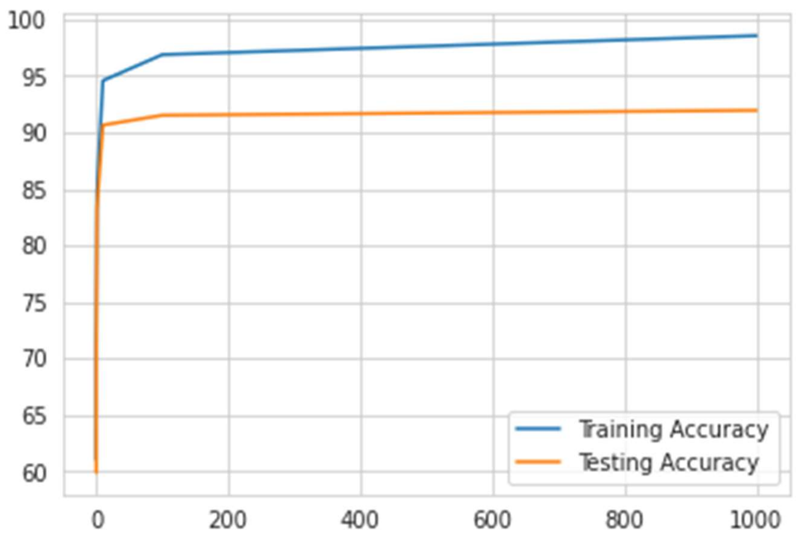
- **For 'quadratic' kernel.**
    - For values varying in (0.01, 0.02, ……., 0.99, 1.00) different models were trained and the following graph was obtained. Also, I made the comparison table in my colab file.



- For values varying in (0.001, 0.01, 0.1, 1, 10, 100, 1000) different models were trained and the following comparison table and graph was obtained.

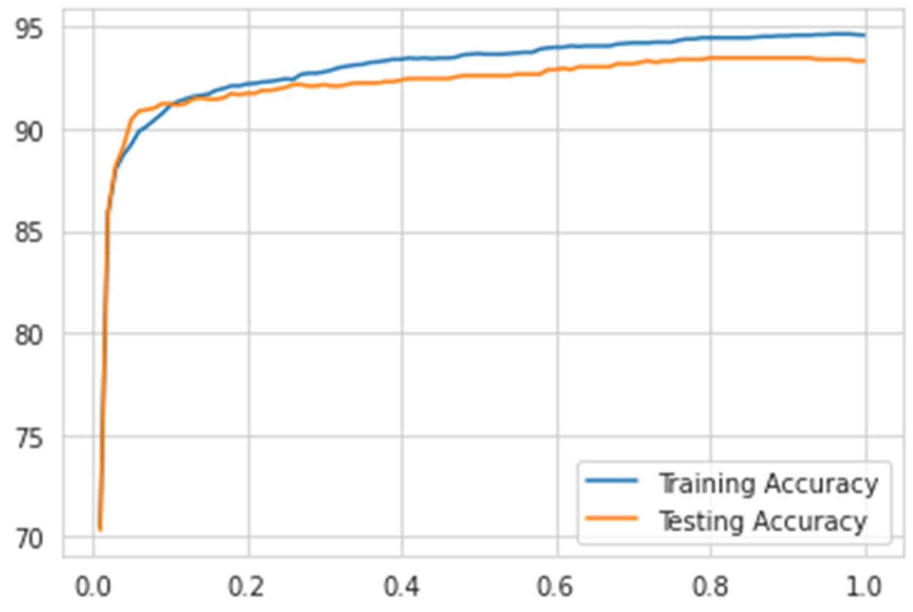


	C	Training Accuracy	Testing Accuracy
0	0.001	61.055901	59.884142
1	0.010	62.422360	60.970311
2	0.100	72.267081	71.397538
3	1.000	84.720497	83.200579
4	10.000	94.596273	90.658943
5	100.000	96.894410	91.527878
6	1000.000	98.571429	91.962346

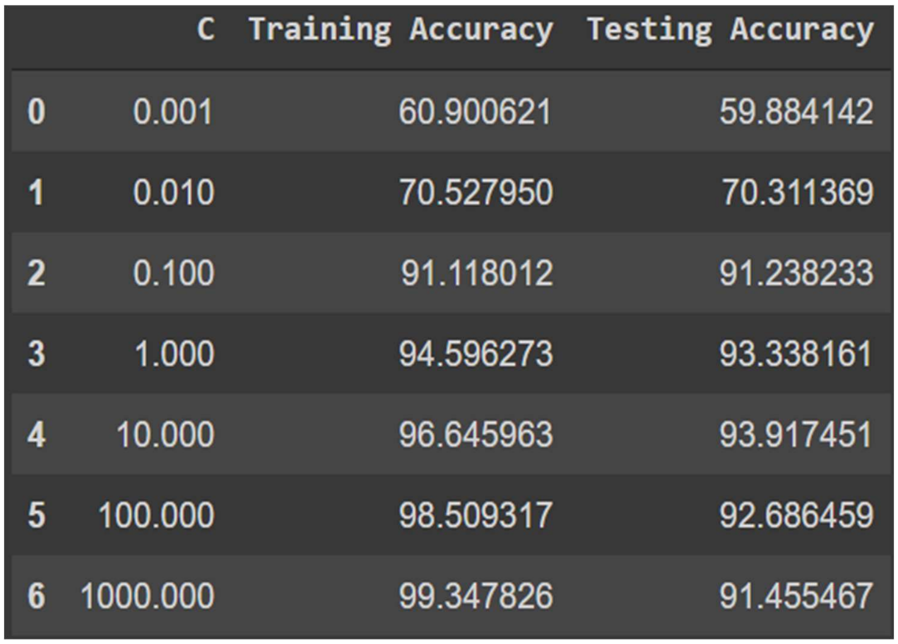


- **For 'rbf' kernel.**

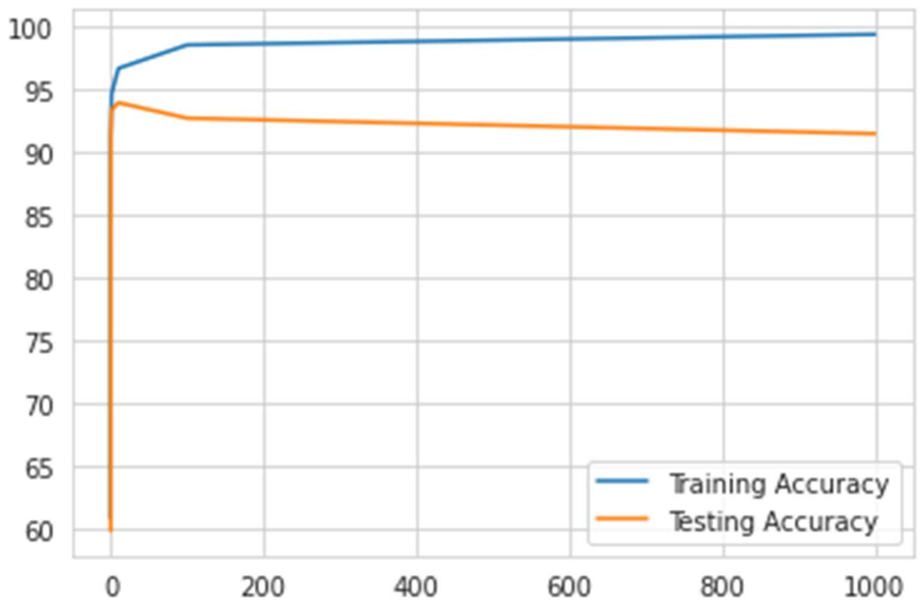
    - For values varying in (0.01, 0.02, ……., 0.99, 1.00) different models were trained and the following graph was obtained. Also, I made the comparison table in my colab file.



    - For values varying in (0.001, 0.01, 0.1, 1, 10, 100, 1000) different models were trained and the following comparison table and graph was obtained.



	C	Training Accuracy	Testing Accuracy
0	0.001	60.900621	59.884142
1	0.010	70.527950	70.311369
2	0.100	91.118012	91.238233
3	1.000	94.596273	93.338161
4	10.000	96.645963	93.917451
5	100.000	98.509317	92.686459
6	1000.000	99.347826	91.455467



Observations:

The parameter generalization constant C is large, it is expected that very less number of training examples will get misclassified as high C will try to place the hyperplane in such a way to minimise the error in training set. However, this setting is very prone to overfitting in the training set.

For small C, it get low classification accuracies for both training and test set.