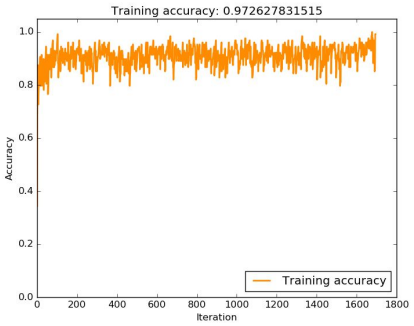
AI : Assignment 3

Anshuman Suri : 2014021

1. The dataset used was MNIST. The effect of varying the four parameters described in the assignment was tried and has been reported below:

    a. Varying weight initialization functions (single layer):
        i. Fanin_1 : 91.75

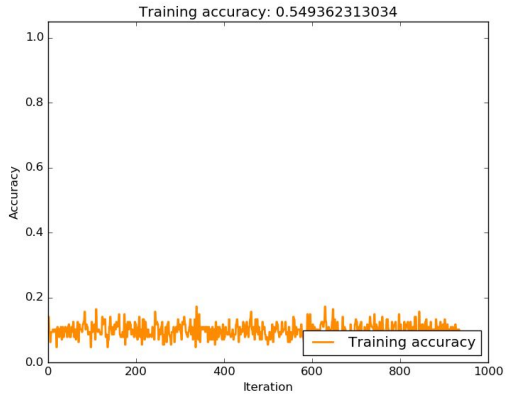


        ii. Fanout_1 : 91.74

        iii. Faninout_1: 91.74

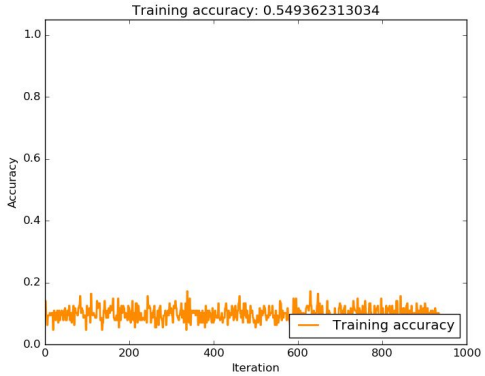
    b. Varying number of neurons in a layer (two layer):

i. Faninout_100_relu : 9.8



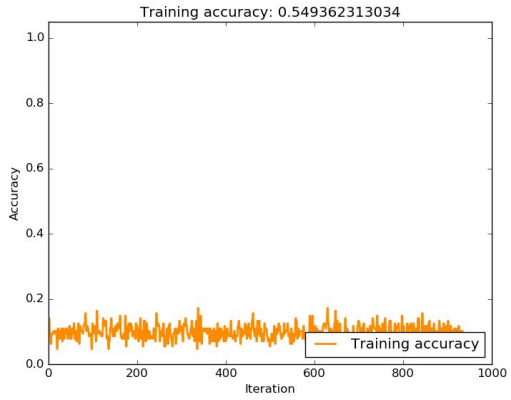
ii. Faninout_400_relu: 9.8

iii. Faninout_700_relu: 9.8



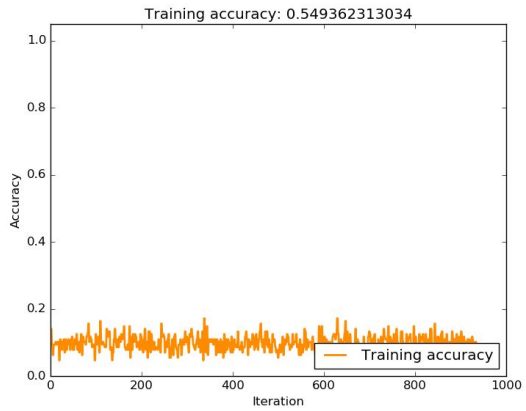
c. Varying activation function (three layer):

i. Faninout_250_100_relu: 9.8



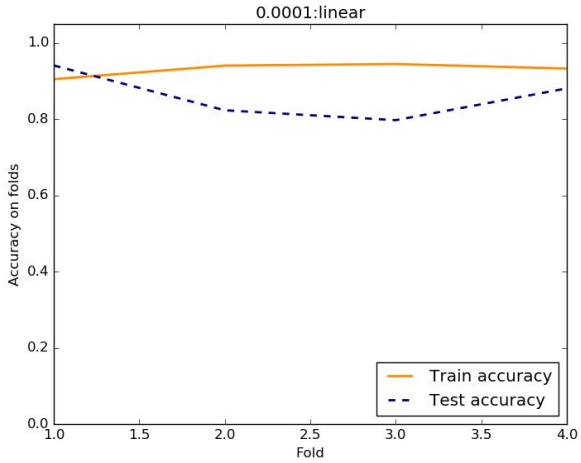
ii. Faninout_250_100_sigmoid: 97.6

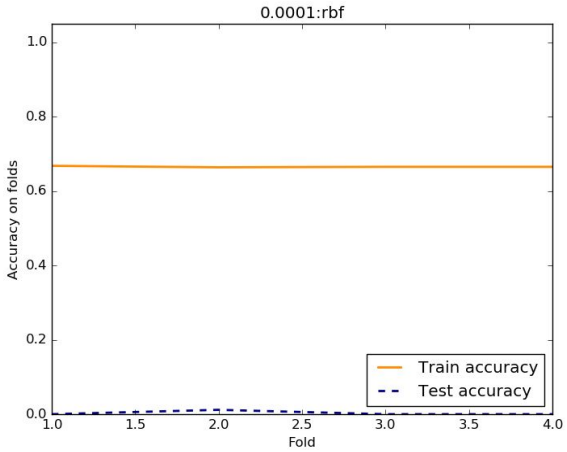
iii. Faninout_250_100_linear: 9.8

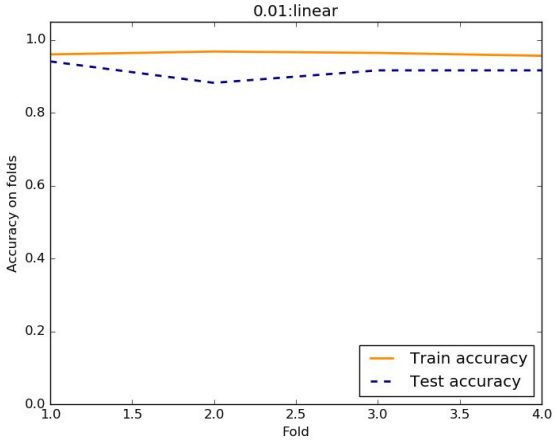


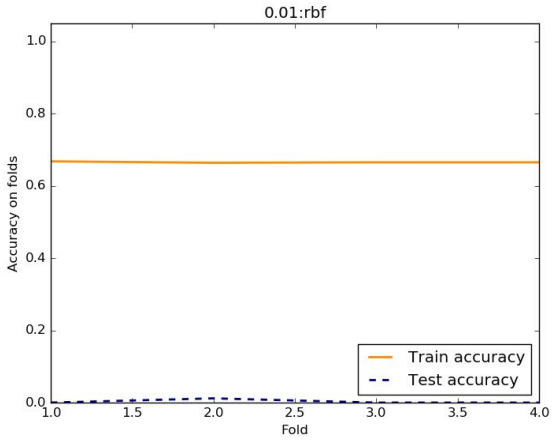
Linear and ReLU activation functions do not work well, yielding accuracies below random. Sigmoid activation works best.
Increasing the width of layer does not have significant increase in the model's accuracy. However, increasing the depth of the network does increase the accuracy from ~90% to ~97%.
Weight initialization does not affect the final accuracy (when the random seed is fixed). However, analysis of the training accuracy/iteration graph reveals that it does decrease training time.

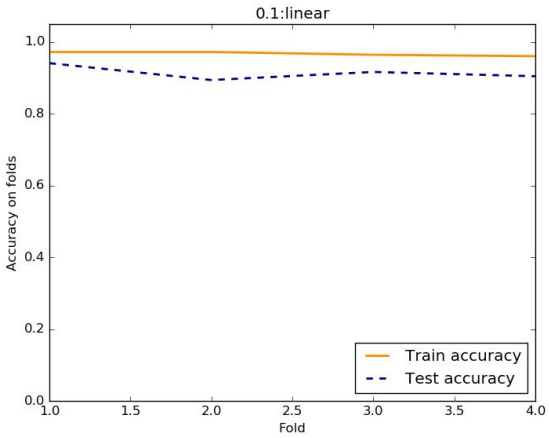
2. Grid search was used across 2 kernels (linear and rbf) and for various values of C (C = infinity implies hard margin, while all other values imply soft margin).
   Data was divided into train and test sets using class balancing, so that distributions remain the same between the two sets (80:20 split for train:test)
   Accuracy plots (for different folds) for all configurations are given below:

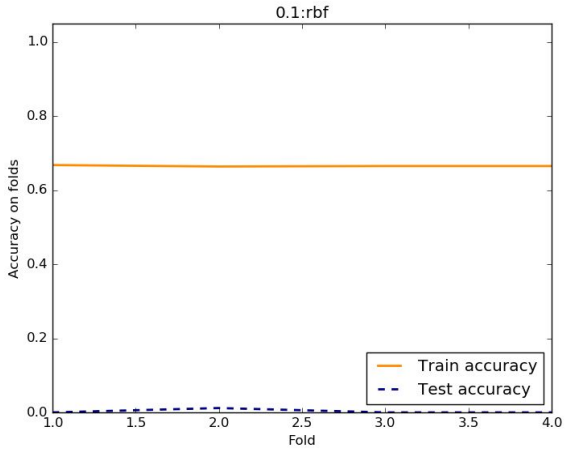


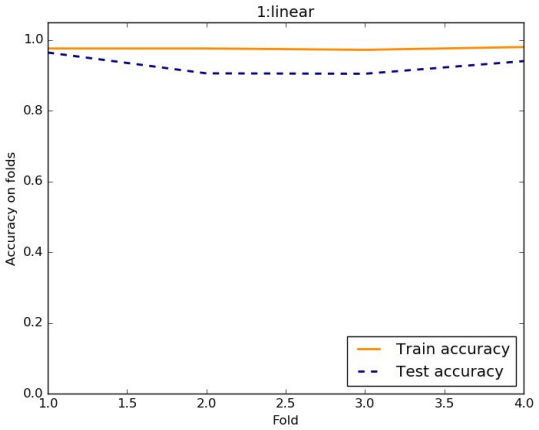


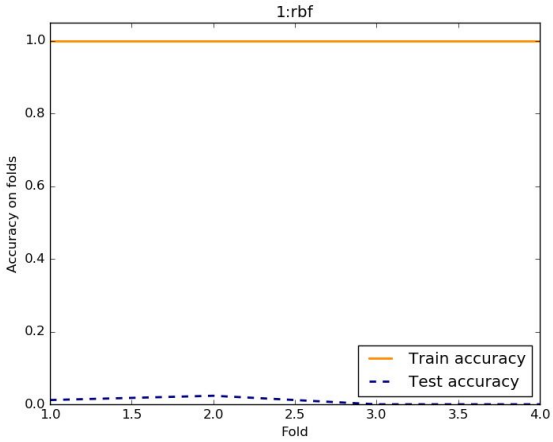


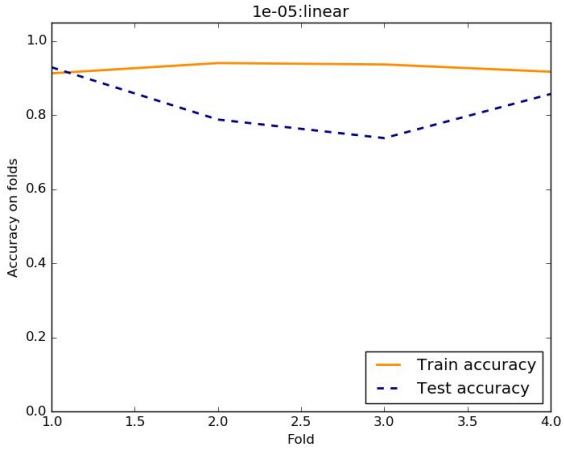


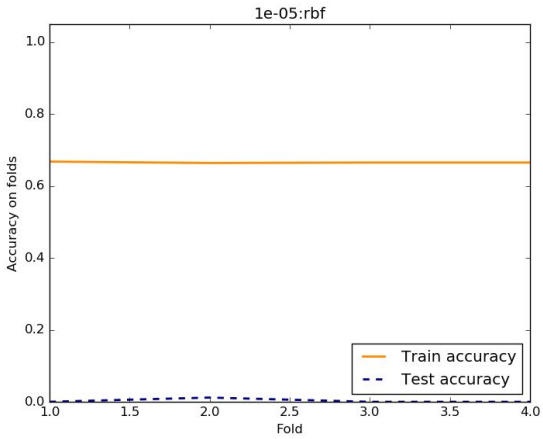

0.1:linear
Accuracy on folds
Fold
Train accuracy
Test accuracy

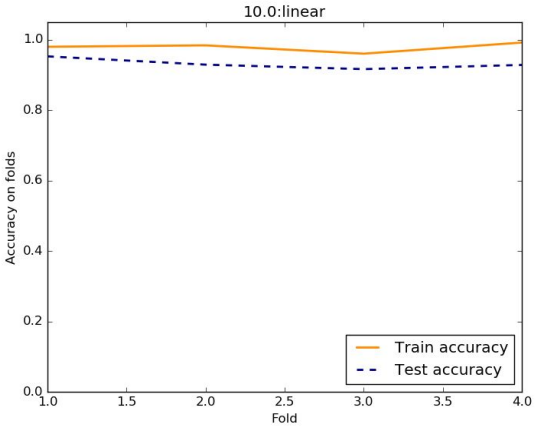

0.1:rbf
Accuracy on folds
Fold
Train accuracy
Test accuracy

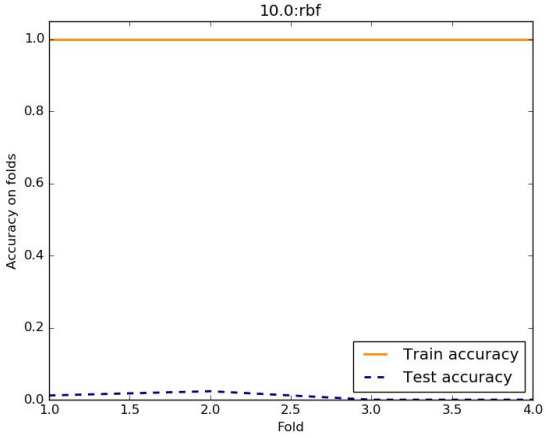

1:linear
Accuracy on folds
Fold
Train accuracy
Test accuracy

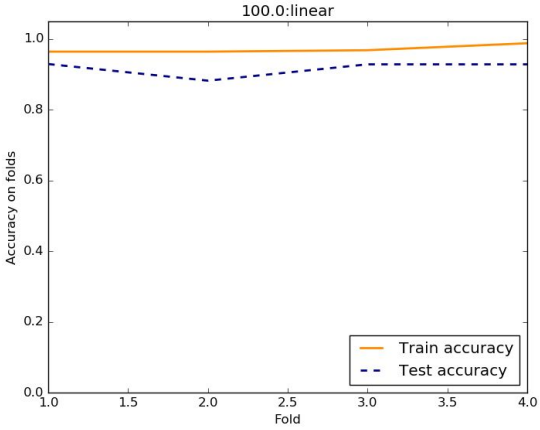

1:rbf
Accuracy on folds
Fold
Train accuracy
Test accuracy

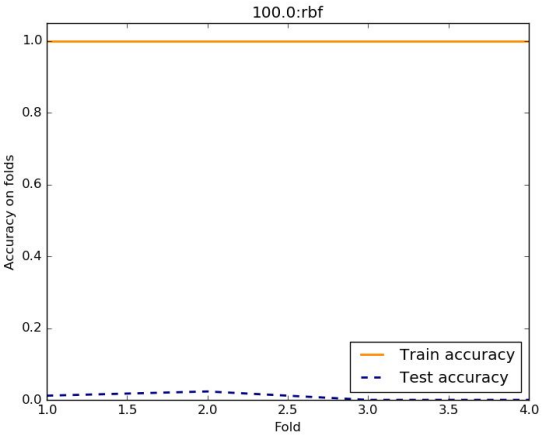

1e-05:linear
Accuracy on folds
Fold
Train accuracy
Test accuracy

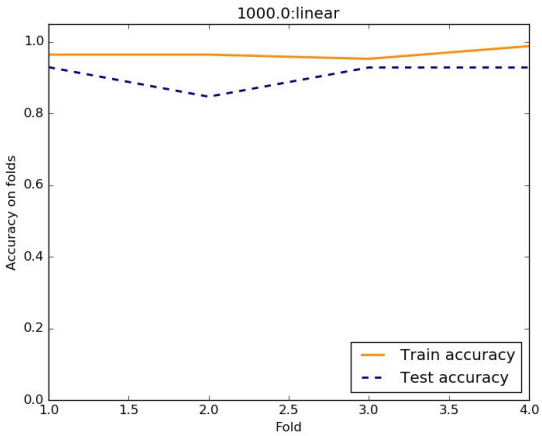

1e-05:rbf
Accuracy on folds
Fold
Train accuracy
Test accuracy

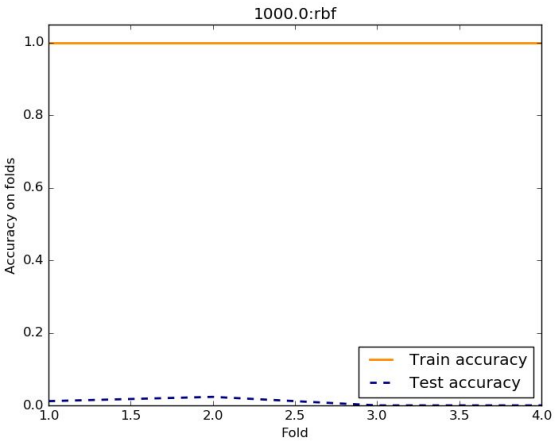

10.0:linear
Accuracy on folds
Fold
Train accuracy
Test accuracy

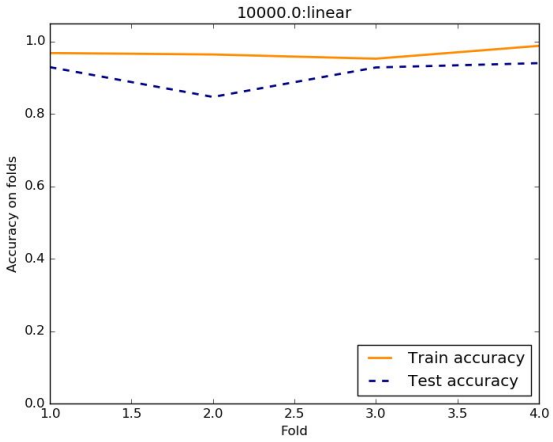

10.0:rbf
Accuracy on folds
Fold
Train accuracy
Test accuracy

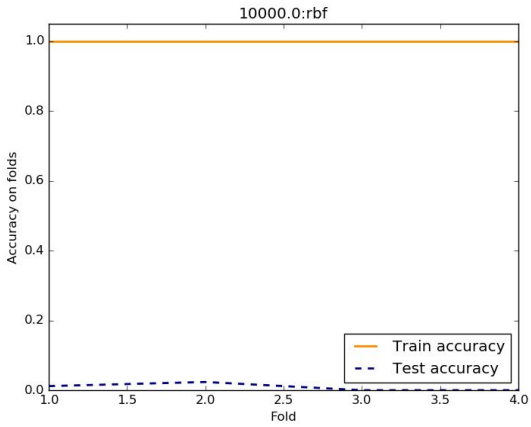

100.0:linear
Accuracy on folds
Fold
Train accuracy
Test accuracy

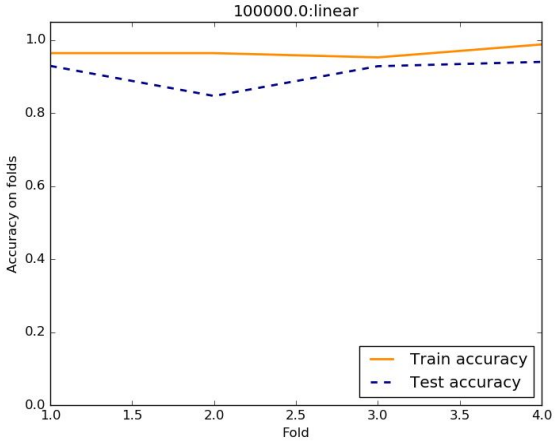

100.0:rbf
Accuracy on folds
Fold
Train accuracy
Test accuracy

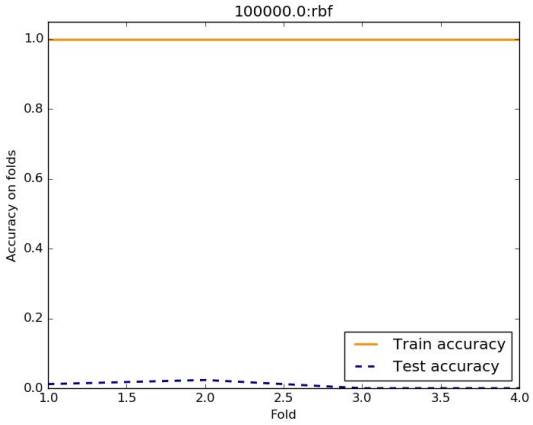

1000.0:linear
Accuracy on folds
Fold
Train accuracy
Test accuracy

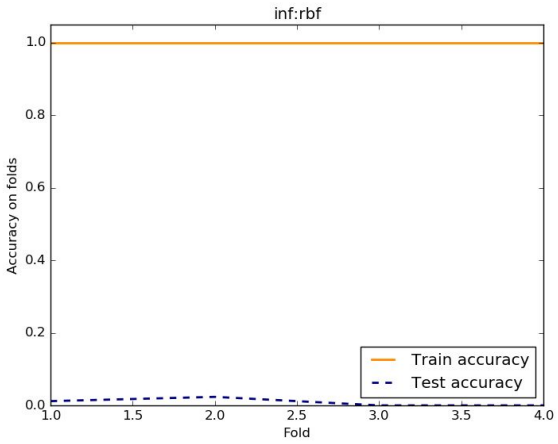

1000.0:rbf
Accuracy on folds
Fold
Train accuracy
Test accuracy

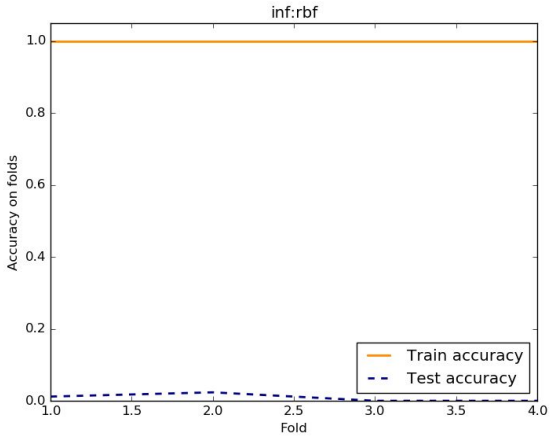










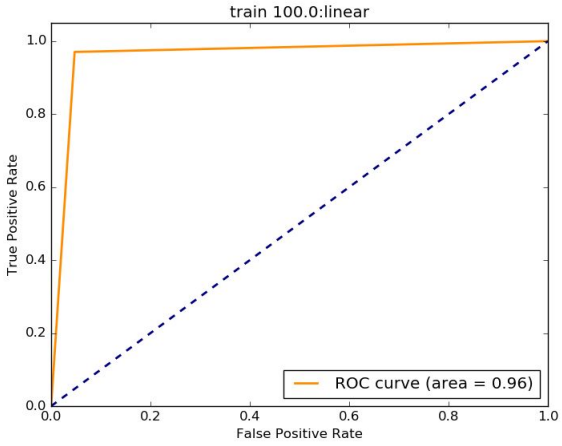


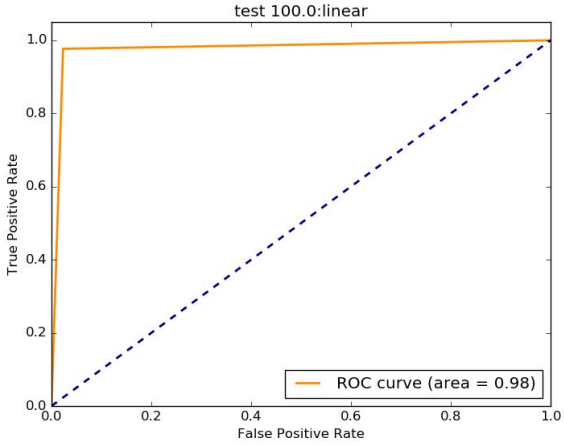
The configuration that worked best amongst these is : a linear kernel with C = 100, gave a testing accuracy of 97.67%.

Apart from the case of hard margin (where both kernels perform badly), a linear kernels perform much better than the rbf kernel. This possibly indicates that the data is already linearly separable, and using a complex kernel makes things worse. Changing the value of C does not have much

effect, unless it is set to infinity. This shows that the data is not perfectly linearly separable, but the number of outliers is less enough for it to be able to correctly classify, even for large values of C.

RoC curves (train and test) for this configuration are given below:





The shape of the curve (an almost perfect inverted L) shows that the classifier is performing really great, and not just in terms of accuracy.