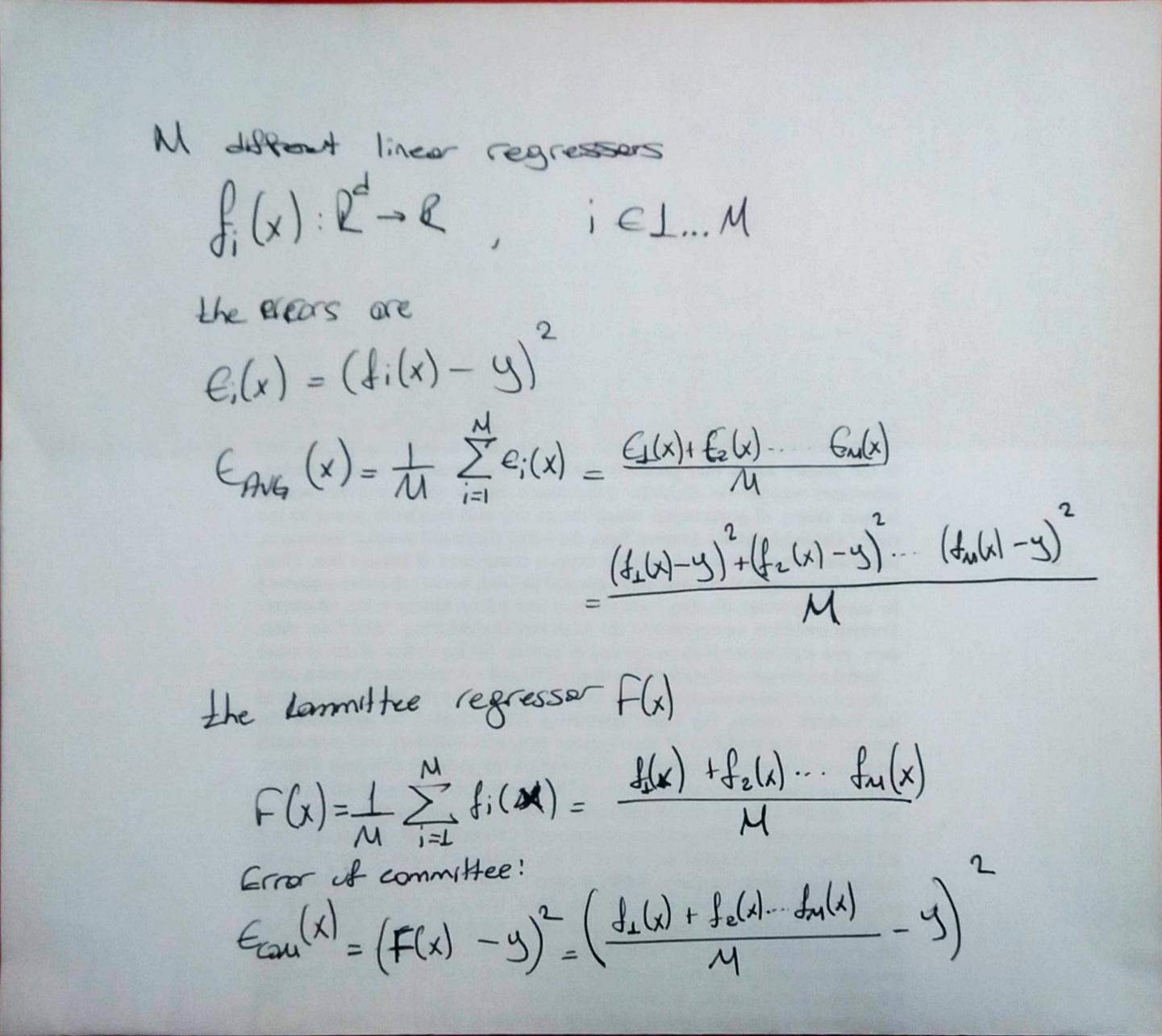
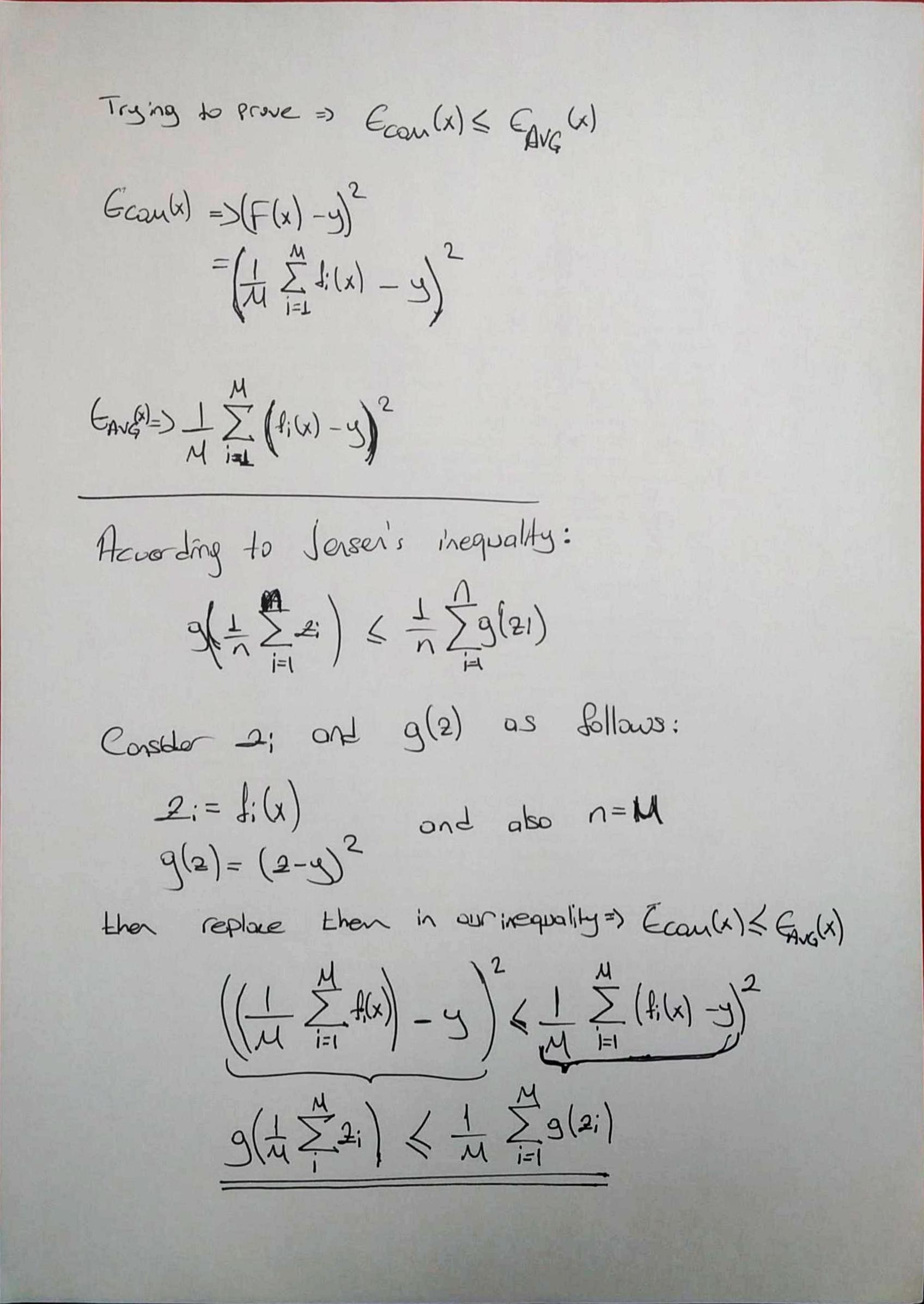
Machine Learning HW2

## **Question 1 - Ensembles**

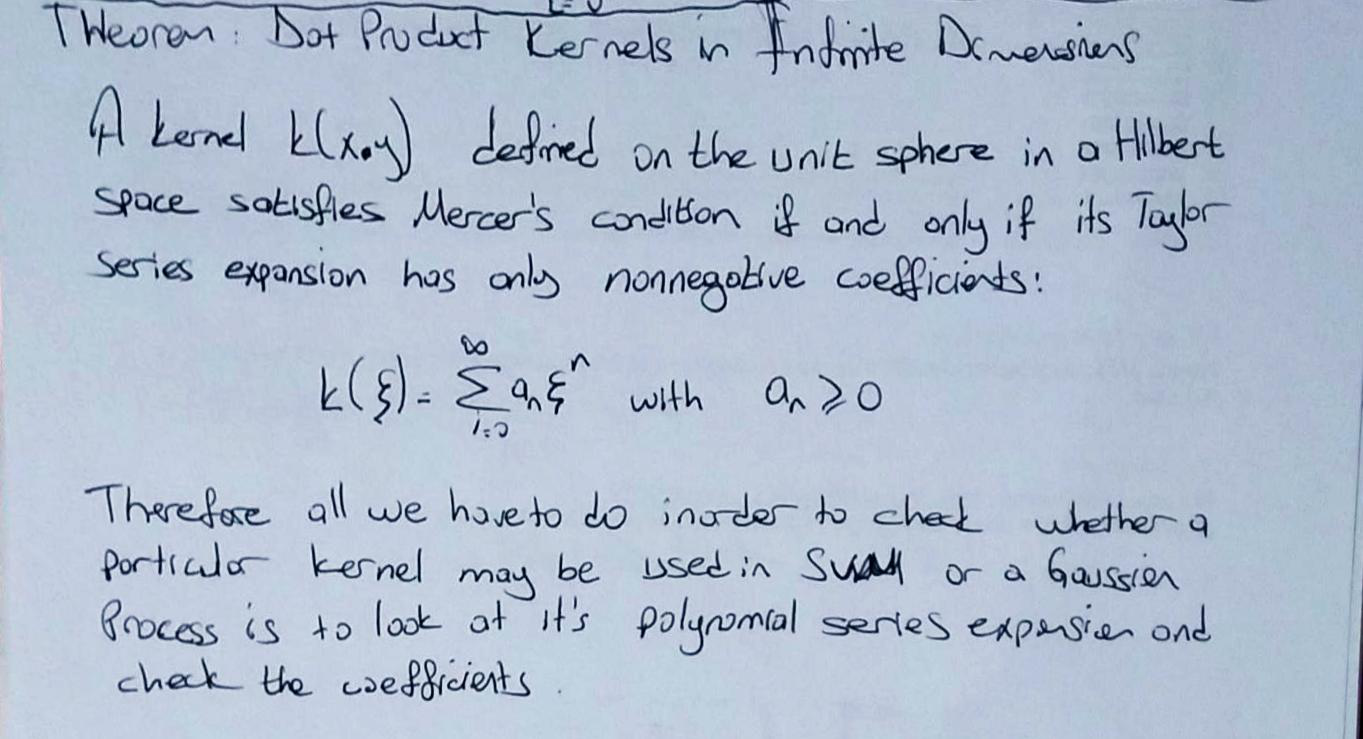
I made my calculations on paper, so that I am putting the scanned versions.





## **Question 2 - Kernels**

Even though I didn’t get exactly what we are supposed to do in this question, I read about it and according to my findings, Schoenberg proved his theorem on Dot Product Kernels in Infinite Dimensions which is as follows and relates to our topic:



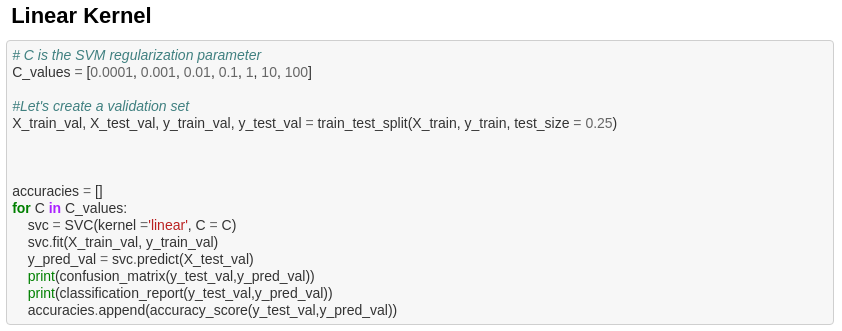
Explicitly not computing the dot product of vectors in this case is actually related to kernel trick, which helps us to avoid it.

If the dot product is defined in an infinite dimensional space Taylor series expansion of kernel function k as a function of <x1, x2> should have no negative coefficients.

## **Question 3 - Letter recognition using SVM**

**a) Linear Kernel**

First I split my data to to %20 test, %80 train datasets. Then again split the train set to %25 validation, %75 train set. After that, for the hyperparameter optimization part, with the C values = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100], I trained my SVM classifier and found the optimum C value. The code below shows this cross validation process.



The accuracy results are as follows:

0.9554140127388535,

0.9681528662420382,

0.9745222929936306,

0.9840764331210191,

0.9808917197452229,

0.9777070063694268,

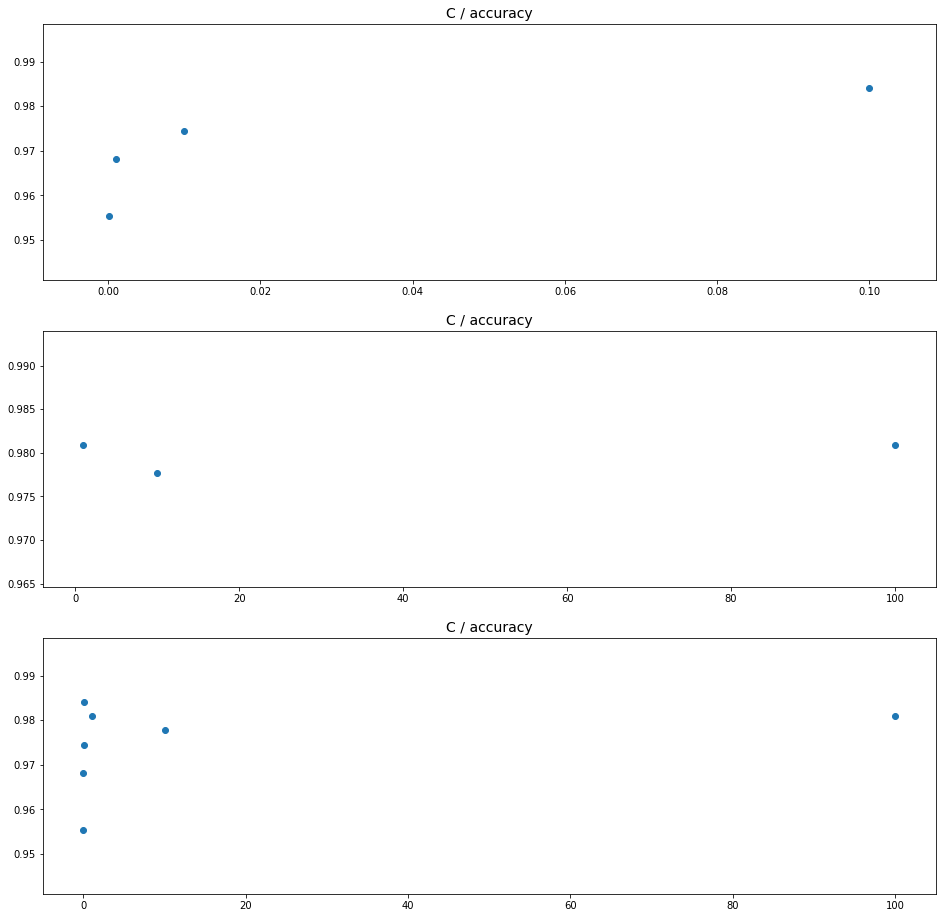
0.9808917197452229

I plotted the accuracies with respect to C values. For the sake of simplicity I have 3 graphs, since first 4 values are too close to each other, it is hard to interpret the overall graph.

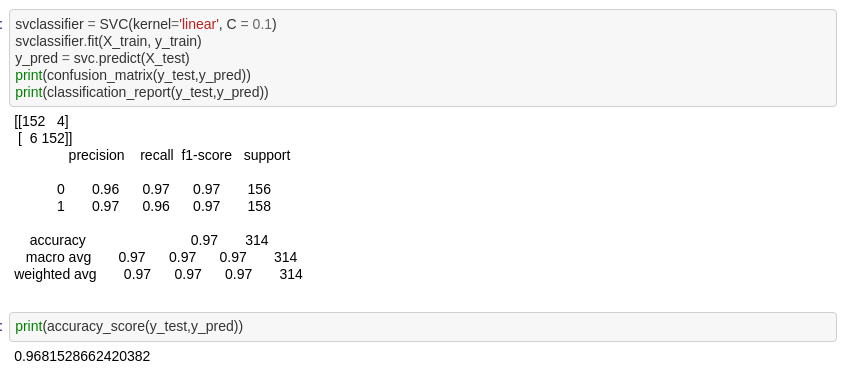
The first graph shows the C values: 0.0001, 0.001, 0.01, 0.1

The second graph shows the C values: 1, 10, 100

The last one shows all of them: 0.0001, 0.001, 0.01, 0.1, 1, 10, 100



I re-trained a model using the best C value which is 0.01 with accuracy 0.9840764331210191 and then run the model on test set with the code below. Results are as follows:

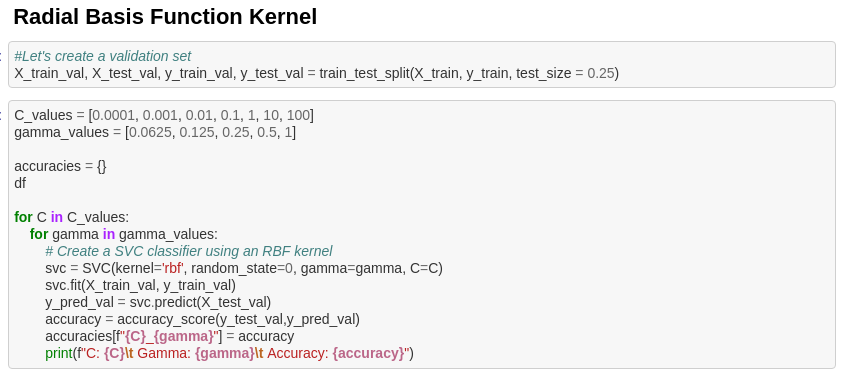


The accuracy is: 0.9681528662420382

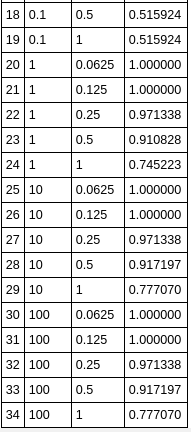
The decision values of test set is at the file named: **Decisions\_Linear\_Kernel.txt**.

**a) Radial Basis Function Kernel**

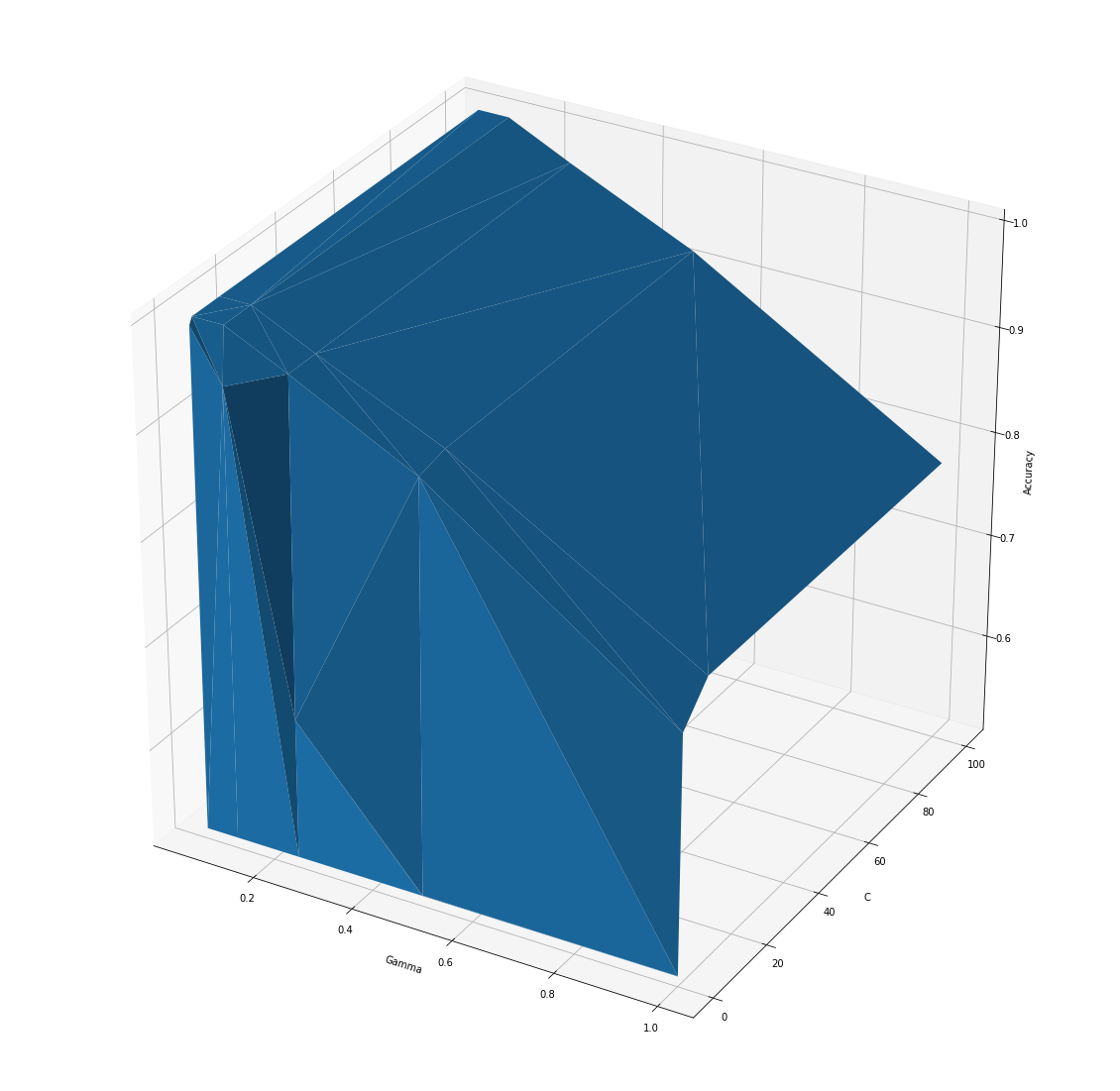
I again split my data to to %20 test, %80 train datasets. Then again split the train set to %25 validation, %75 train set. After that, for the hyperparameter optimization part, with the C values = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100] and gamma values = [0.0625, 0.125, 0.25, 0.5, 1], I applied cross validation on my SVM classifier and found the optimum C value. The code below shows this cross validation process.



Accuracy results are as follows:

I plotted the accuracies with respect to C values and gamma values.



Then found the best accuracy values and corresponding C and gamma values.



### So, After tuning, we have several optimum C and gamma values

C: 1, Gamma: 0.0625, Accuracy: 1.0

C: 1, Gamma: 0.125, Accuracy: 1.0

C: 10, Gamma: 0.0625, Accuracy: 1.0

C: 10, Gamma: 0.125, Accuracy: 1.0

C: 100, Gamma: 0.0625, Accuracy: 1.0

C: 100, Gamma: 0.125, Accuracy: 1.0

So Let's pick one of them and test our data

Picked:

C: 10, Gamma: 0.0625



The decision values of test set is at the file named: **Decisions\_RBF\_Kernel.txt**.

The accuracy for Linear Kernel was: 0.9681528662420382 and the accuracy for Radial Basis Function Kernel is: 1.0 which clearly states the SVM with RBF kernel is better.

This means our prediction gets better when we use non-linear model which is SVM with RBF kernel. Since SVM with Linear kernel is linear and tries to separate the labels with linear line and RBF on the contrary uses curves to separate it, it is likely to have a better performance with RBF. The kernel defines the function class that we work with. The squared exponential kernel such as RBF defines a function space that is a lot larger than that of the linear kernel. So that it is more flexible and with the risk of overfitting, it fits the data better than linear kernel. That’s why prediction of RBF is better than Linear kernel.