Support vector machines (Assignment)

Name: Afraz Salim *r-number: r0439731*

In this section, we will use standard support vector machine model to classify the points which are generated randomly. Based on the distribution of the model, we will use a model that classifies the points correctly.
The distribution of the datapoints is shown in 1.

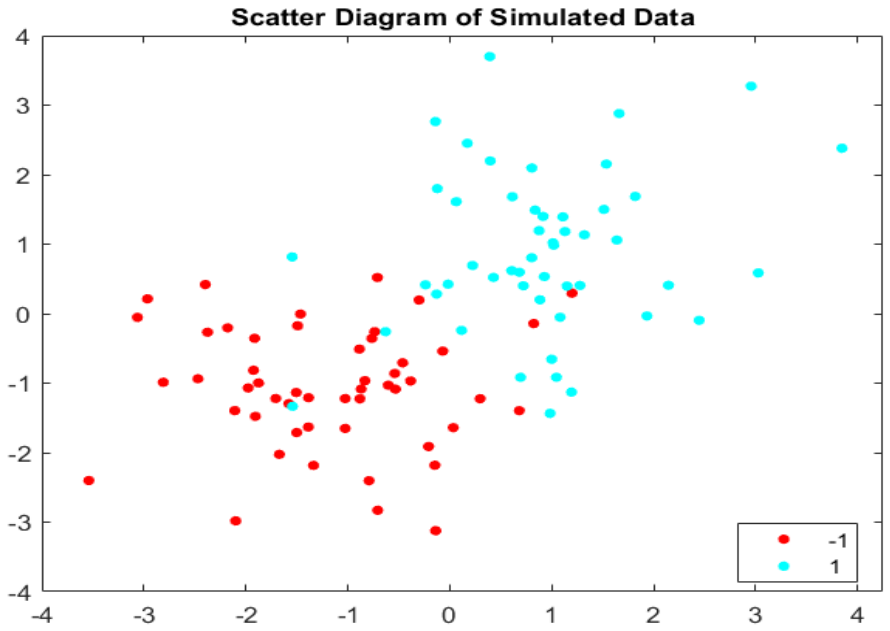


Figura 1: Data generated using a fixed random generator

- Obtain a line to classify the data by using what you know about the distributions of the data. In which sense is it optimal?

The datapoints overlap with each other and can not be separated with a linear decision boundary so we need to use kernel trick. We will try to look for an optimized solution by using maltab bayeopt tool.
The objective function that we are trying to minimize has the following form:

$$min_{w,b,\varepsilon}(w,\varepsilon) = \frac{1}{2}w^T.w + C\sum_{k=1}^{N}\varepsilon_k$$

The constraint section is not important to discuss the optimal results so we leave that part out. The first part focuses on the maximizing the margin while the second part tells us how important a misclassification is. Lower values of C make misclassification less important. Hence, the optimal solution will be one with high value of C and low value of $\frac{1}{2}w^T.w$. Following options were used to obtain the result shown in figure 2.

```
c = cvpartition(101,'KFold',10);
opts = struct('Optimizer','bayesopt','ShowPlots',true,'CVPartition',c,...
              'AcquisitionFunctionName','expected-improvement-plus');
svmmod = fitcsvm(X,Y,'KernelFunction','rbf','OptimizeHyperparameters','auto',...
        'KernelScale','auto' ,'ClassNames',[-1,1],'Solver','ISDA','HyperparameterOptimizationOptions',opts)
```

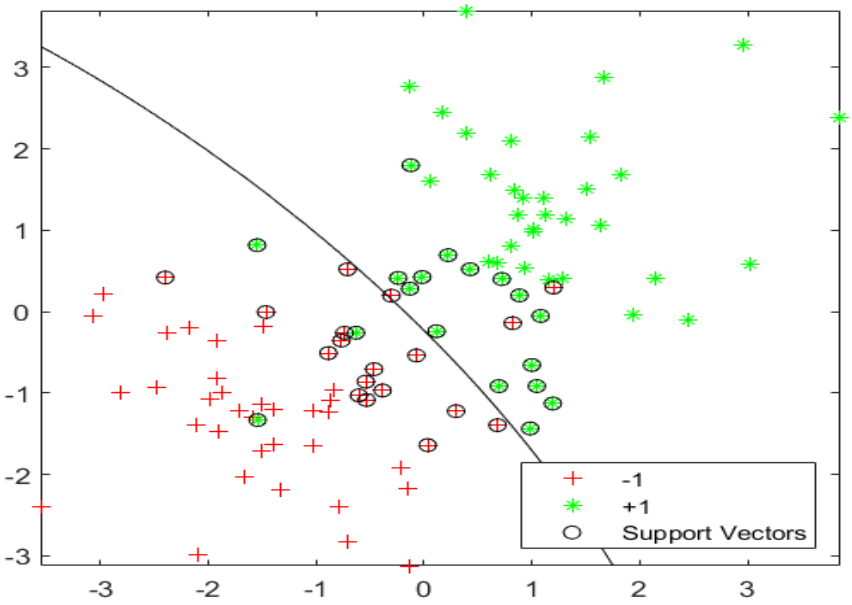


Figura 2: Classification using RBF kernel

We obtain the following results:

Objective function value	KernelScale	Boxconstraint
0.073526	5.0636	1.023

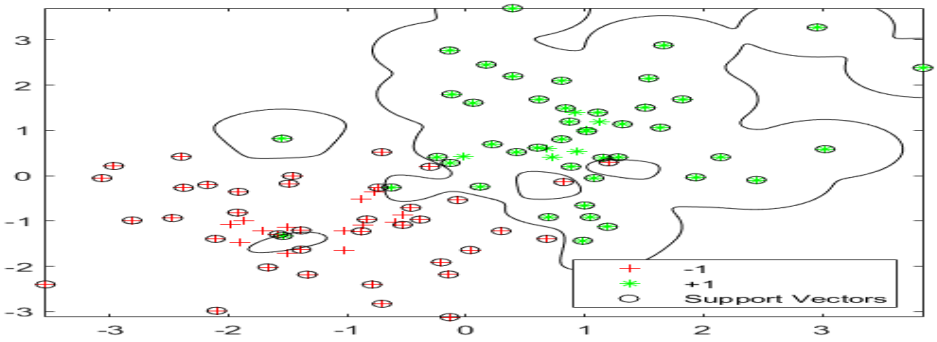


Figura 3: Classification using kernelscale= 0,3 and C= 50

The box constraint is C value which defines a bounds from α in lagrangian. One can use more sophisticated techniques to improve results.

We can also choose different values for *BoxConstraint* and *KernelScale* to fit highly non-linear boundary. One such example is shown in figure 3. One disadvantage of this model is that almost every datapoint is a support vector.

- What is a support vector? When does a particular datapoint become a support vector? When does the importance of the support vector change? Illustrate visually. Note that a support vector is indicated by a large circle with bold-lined circumference and that its importance is proportional to the size in the online application.

A datapoint becomes a support vector for which α value from lagrangian support is not zero. From the the KKTconditions, we know that $\lambda_i c_i(x_i) = 0$ where $i \in C_e$. For all datapoints, for which $\lambda \neq 0$ are support vectors.
The importance of the datapoint is controlled with the sigma value in rbf kernel. When we increase the sigma value then the distribution will shrik(shape of the gaussian curve in 2D) and only points which lie on the boundary(points belonging to two different classes) will create create a decision boundary. However, if we decrease the sigma value then more datapoints will contribute in the creation of decision boundary. In short, increasing the sigma value will decrease the range of a datapoint and decreasing the sigma value will increase it's range. One such example is show in figure 4.

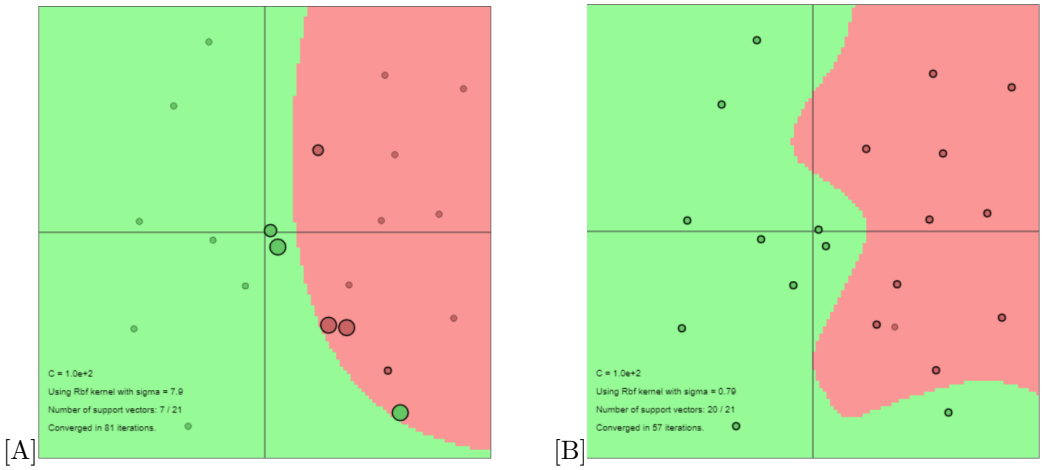


Figura 4: A : Important points with sigma=5. B: important points with sigma=0,79

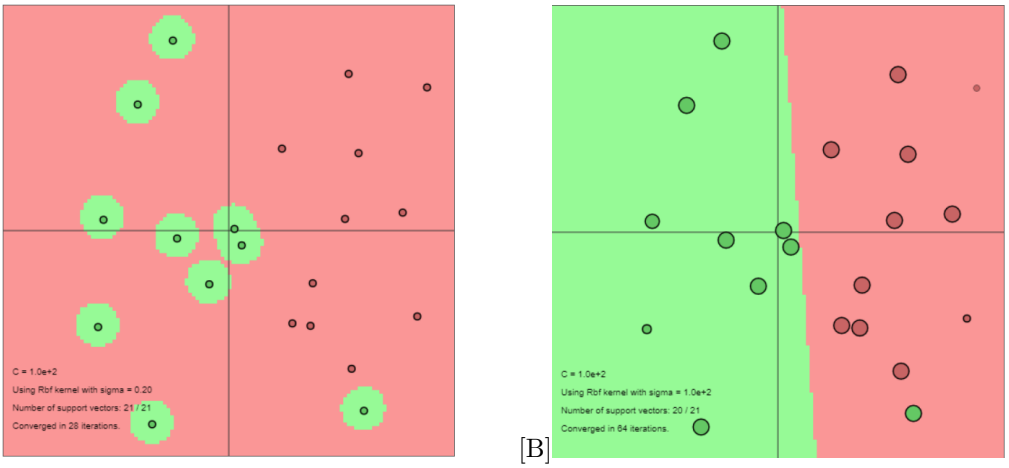
- What is the role of parameters C and sigma? What happens to the classification boundary if you change these parameters. Illustrate visually.

Reducing C means we make a misclassification less important while sigma determine the width of the distribution or we can say that it determines the reach of a datapoint. Higher value of sigma will create very few support vectors(datapoints) while lower value of sigma will create many support vectors as show in figure 4.

- What happens to the classification boundary when sigma is taken very large? Why?

When we take sigma too small then every point will try to create it's own bounday because even points belonging to same class will not reach each other. However, if we increase sigma to some extent such that all points belonging to same class can reach eachother then there will be only one decision boundary. Too small sigma causes overfitting and too big value of sigma will have some misclassifications as shown in figure 5

For linear kernel, there is no sigma however increasing the C value will decrease the with/margin as we penalize the misclassifications hardly and decreasing the C value will increase the margin and there will be more misclassifications.



[A] [B]

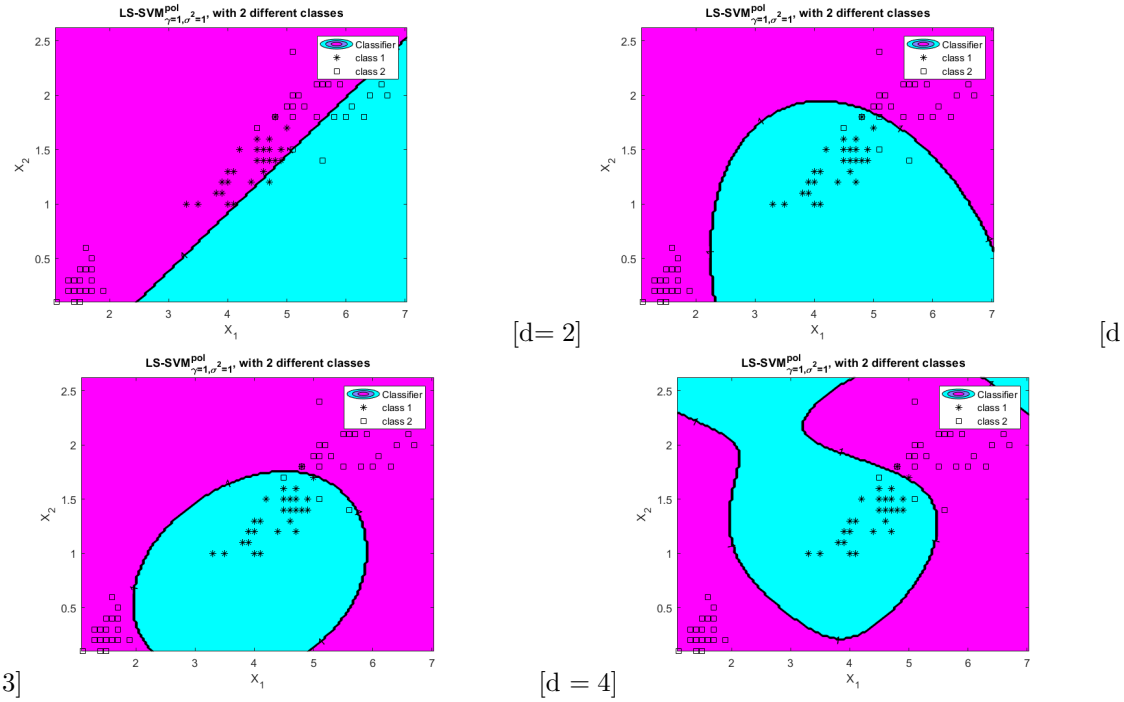
Figura 5: A : overfitting with sigma=0,79 & C=100. B: misclassifications with sigma=100 & C=100

LS-SVM

In this section, we will analyze build lssvm model and investigate the impact of different parameters.

- Try out a polynomial kernel with degree $= 1, 2, 3, ...$and $t = 1$ (fix gam $= 1$). Assess the performance on the test set. What happens when you change the degree of the polynomial kernel?

As we increase the degree of the polynomial, we get more complex curve. Since a polynomial simply defines a curve and polynomial with higher degree is capable of forming quite complex shape. Raising the degree of polynomial can be helpful to form the decision boundary for a very high dimensional space. Figure **??** illustrates the effect of increasing the degree of the polynomial.

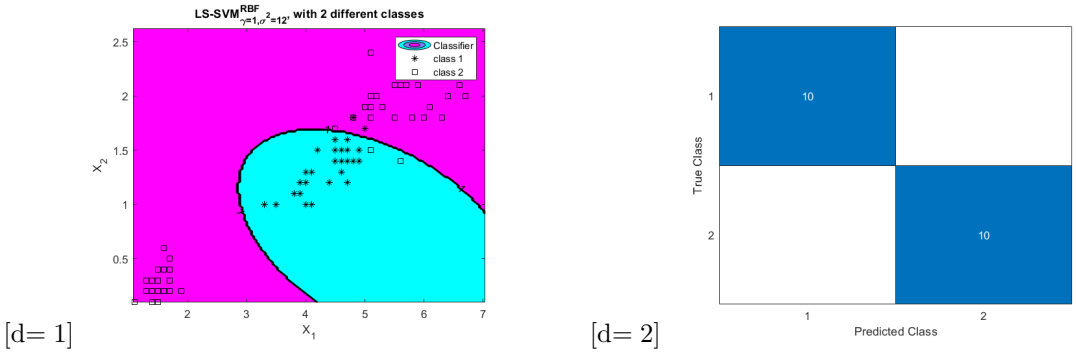


[d= 1] [d= 2] [d = 3] [d = 4]

Figura 6: Models with polynomial kernels and different degrees.

- Try out a good range of different sig2 values as kernel parameters (fix gam $= 1$).Assess the performance on the test set. What is a good range for sig2?

We tried different sigma values while the value of *gam* remains fixed. Best value for the sigma is 12. The resuls can be seen in figure 7.

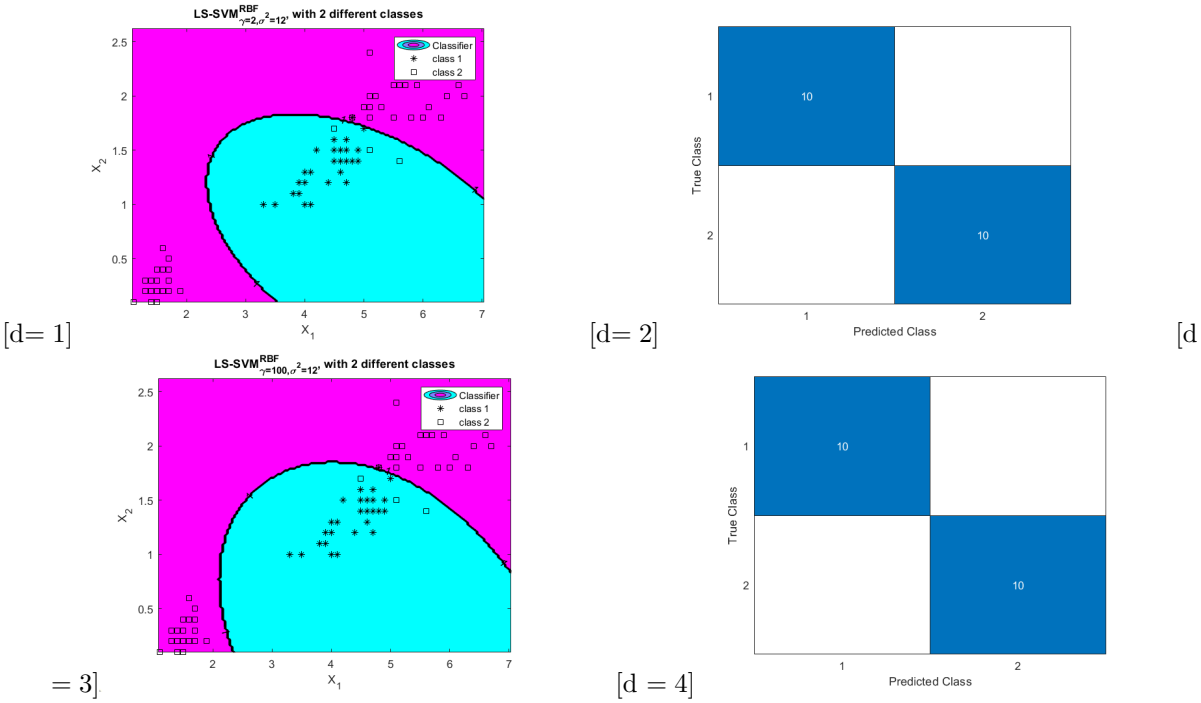


[d= 1] [d= 2]

Figura 7: Best value of sigma.

- Fix a reasonable choice for the sig2 parameter and compare the performance using a range of gam. What is a good range for gam?

We changed the values of gam starting from 0 to 100 and the result remains same but the decision boundary shifts a little bit. The results are shown in figure **??**.



[d= 1] [d= 2] [d = 3] [d = 4]

Figura 8: sigma fixed while gam changes

- Compare your results with SampleScript iris.m, which is available on Toledo

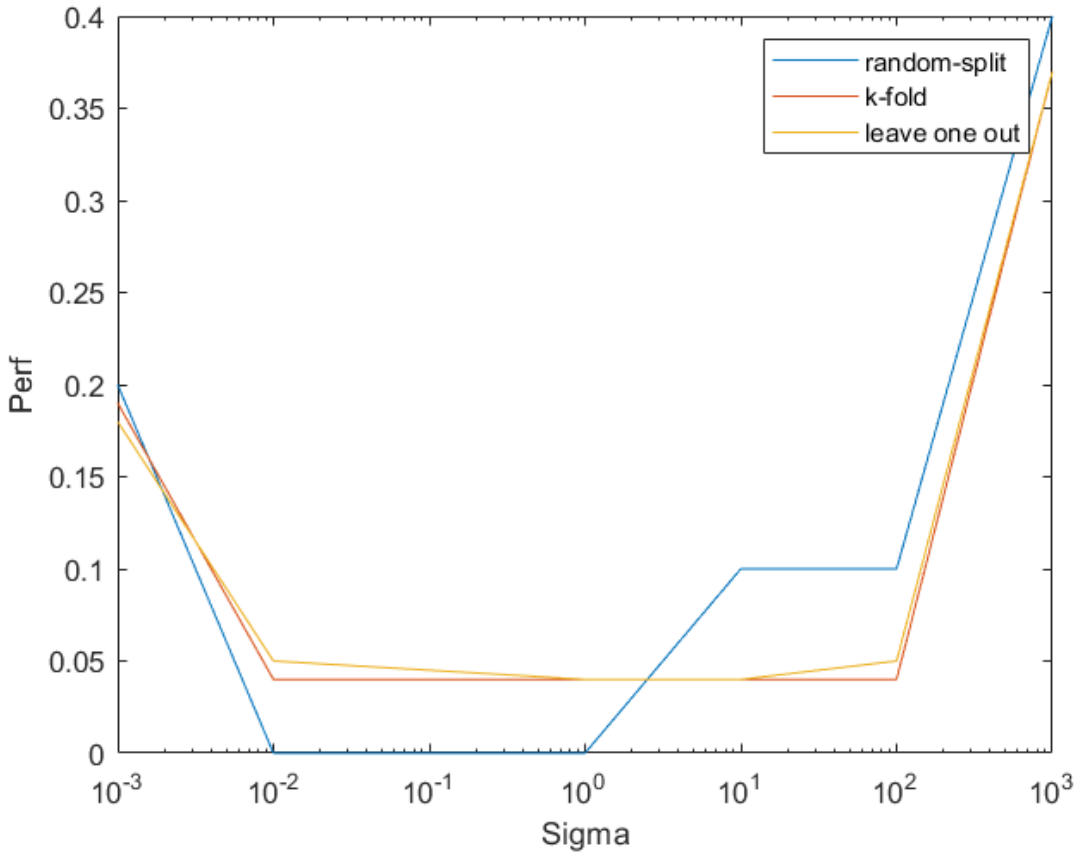
While comparing the results, we see that we have same results except that in SampleScript-iris.m, the value for σ is also chosen very small. From the picture one can see that it can cause overfitting as the function is fitted very tightly.

Validation techniques

- Compute the performance for a range of gam and sig2 values ($e.g.,, \sigma = 10^{-3}, ..., 10^{3}$). Use the random split method, 10-fold crossvalidation and leave-one-out validation. Visualize the results of

each method: do you observe differences? Interpret the results: which values of gam and sig2 would you choose?

We fixed the value for $\gamma = 100$ and iterated over different values of σ in range $10^{-3}, .., 10^{3}$. The reason to fix the value of γ is to get $2D$ array so that we can visualize the performance. We know that random



[A]

Figura 9:

split has biased selection. We will get different results every time we generate different random numbers. In figure 9 one can see that for $\sigma = 10^3$, all validation techniques show high performance but for random split, the result will be different when we generate new random numbers. Such biased selection is removed by k-fold cross validation. Leave-one-out is probably better of all these two but can be computationally expensive.

- Why should one prefer crossvalidation over simple validation (random split)? How to choose the value of k in k-fold crossvalidation?

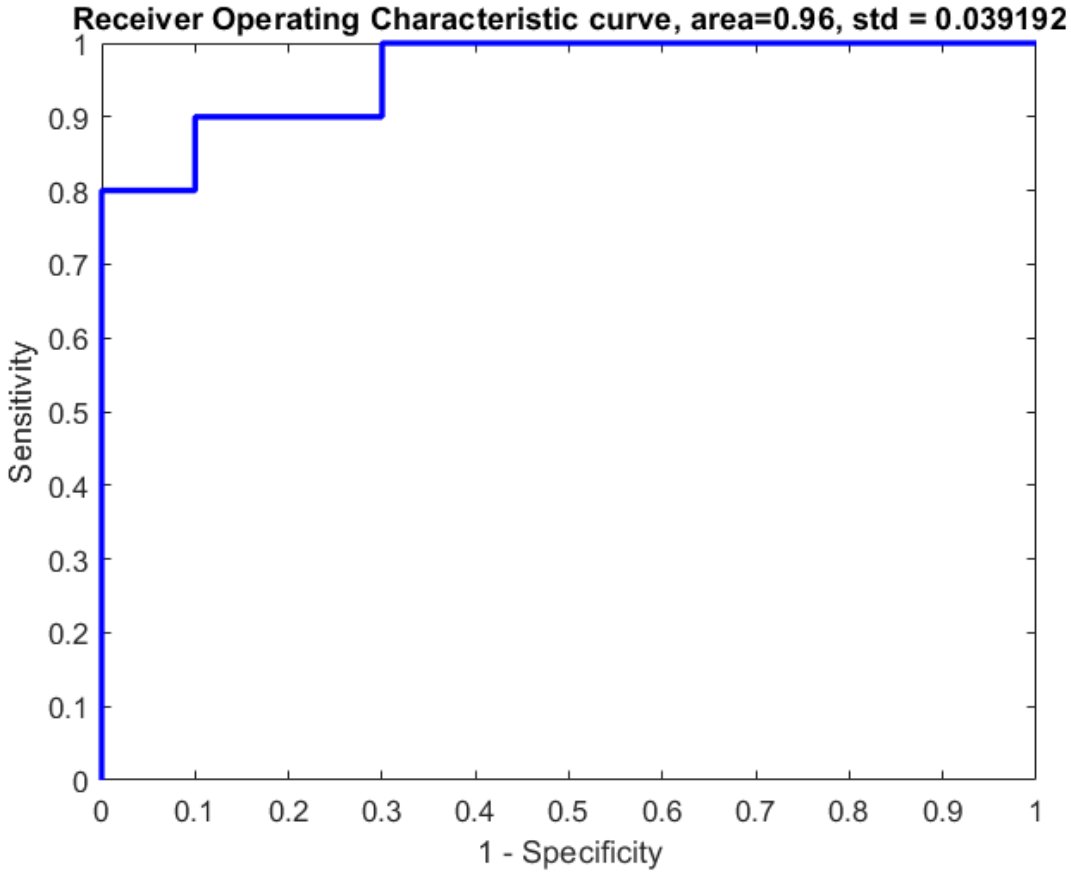
As dicussed in previous section, random split has biased selection. The performance or error that we get is based on what training data we have chosen for training a model and what data we have chosen for validation. Such biased selection is removed by k-fold crossvalidation and that is the reason k-fold cross validation is chosen.
There is no general rule to select the K value in k-fold crossvalidation but selecting too large k value results in few models that we can build and test. A good value of K depends on the size of available data.

- Try out the different 'algorithm'. What differences do you observe? Why do the obtained hyper-parameters differ a lot in different runs? What about the cost? Computational speed? Explain the results.

We tried first simplex algorithm. The obtained cost does not change but the values of γ and σ change. This is common behaviour because the same cost can be obtained by either penalizing the errors with high γ value or by using low γ value but high σ value. In the second case, the model can overfit as it will create highly non-linear decision boundary. Simplex algorithm takes more iterations as compared to gridsearch. The computation speed is not very much noticeable as the training data is not large enough.

- In practice, we compute the ROC curve on the test set, rather than on the training set. Why?
- Generate the ROC curve for the iris.mat dataset (use tuned γ and σ values).Interpret the result.

The roc-curve generated on test data is shown in figure 10. ROC curve shows false positive rate on x-axis while true positive rate on y-axis. The more points lie on the upper left corner, the better results we have. We want to maximize the area under the curve. If the curve the lies on the diagonal line then the classifier is a random classifier. Points below the diagonal line indicate that classifier is predicting a positive instance negative and a negative instance positive.



[A]

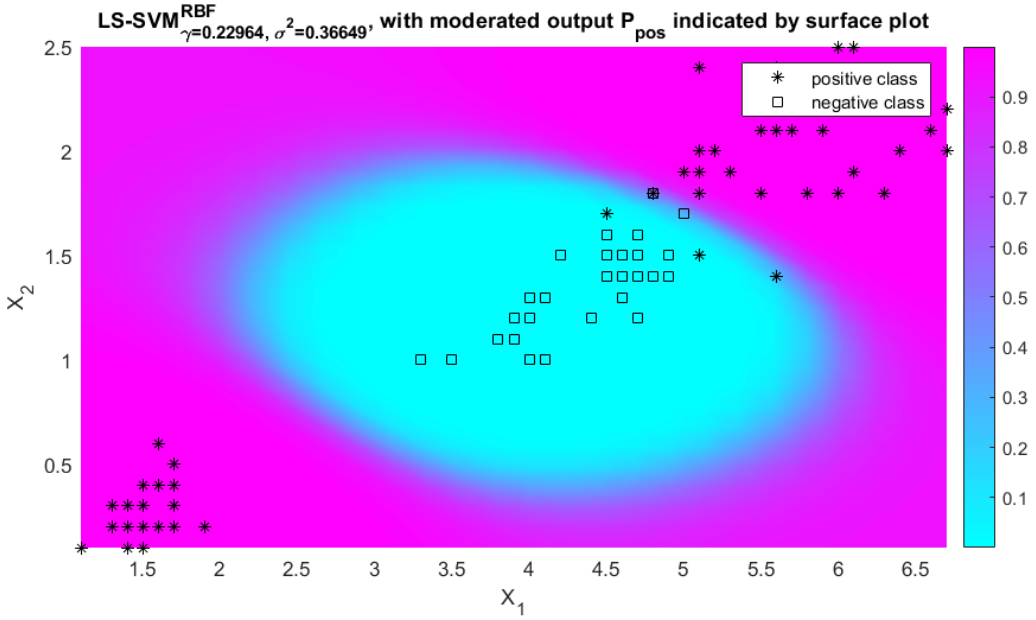
Figura 10:

Bayesian framework

- Using the Bayesian framework, it is possible to get probability estimates. Use a tuned pair of parameters (γ,σ).

- How do you interpret the colors of the plot? Hint: activate the colorbar of the figure.

Using the tuned parameters, we obtained a figure as shown in 18. We can choose a proability threshold to classfify two datapoints belonging to two different classes. Points that that belong to negative class, have probability $\leq 0{,}4$.



[A]

Figura 11:

Home work

- Visualize the data. Inspect the data structure: what seems to be important properties of the data? Which classification model do you think you need, based on the complexity of the data?

We visualize the ripley.mat. diabetes.mat and breast.mat has greater than 2 dimensions and can not be displayed. Figure 12 shows ripley.mat dataset. We can see that the data points in this plot are not

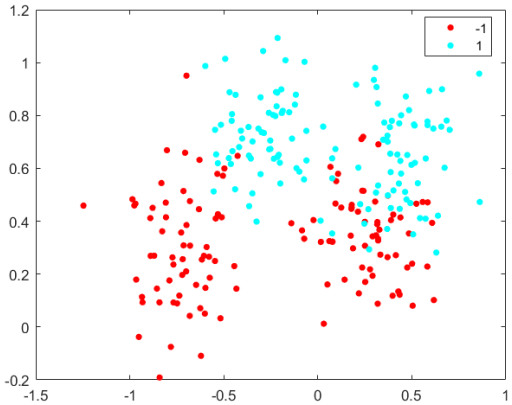
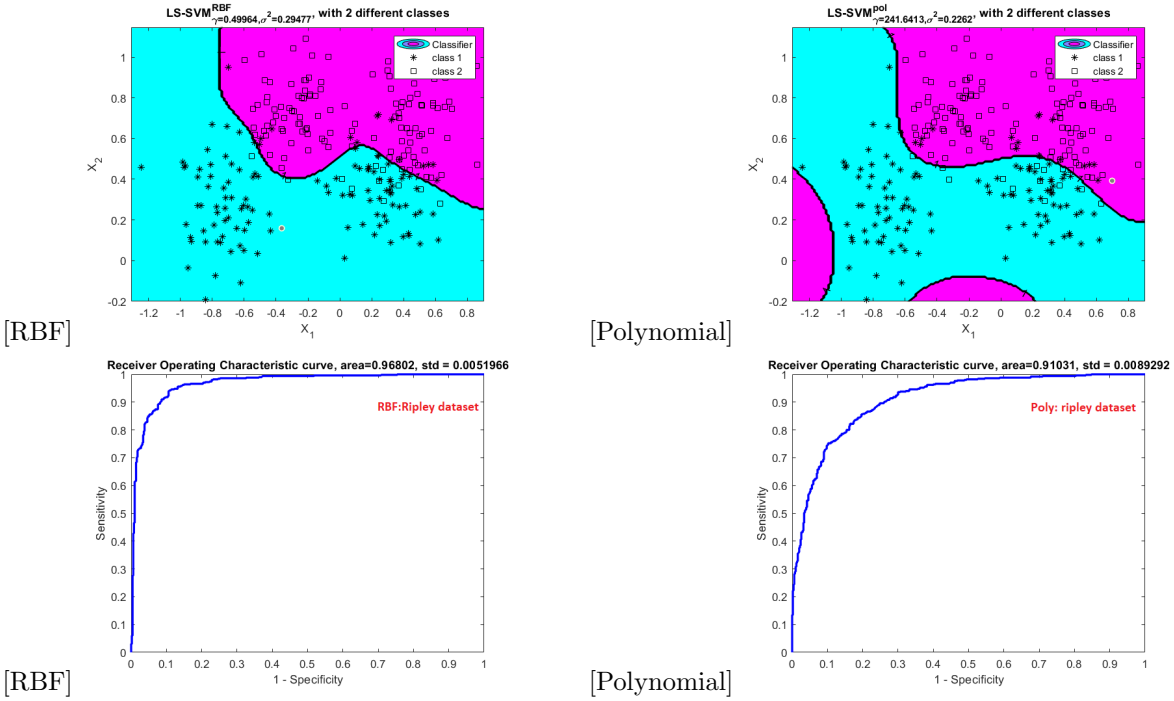


Figura 12: Riplye dataset

linearly separable. Hence, we need to use kernel trick.

- Try out different models (linear, polynomial, RBF kernel) with tuned hyperparameter and kernel parameters. Compute the ROC curves. Which model performs best? Which model would you choose?

We try different kernels for ripley dataset. All of these results are obtained by using automatic tuning with 10 fold cross validation. Since the data in breast.mat and diabetes.mat has higher dimensions. We can not visualize the data but it is possible to compute ROC curves which shows us the results that we

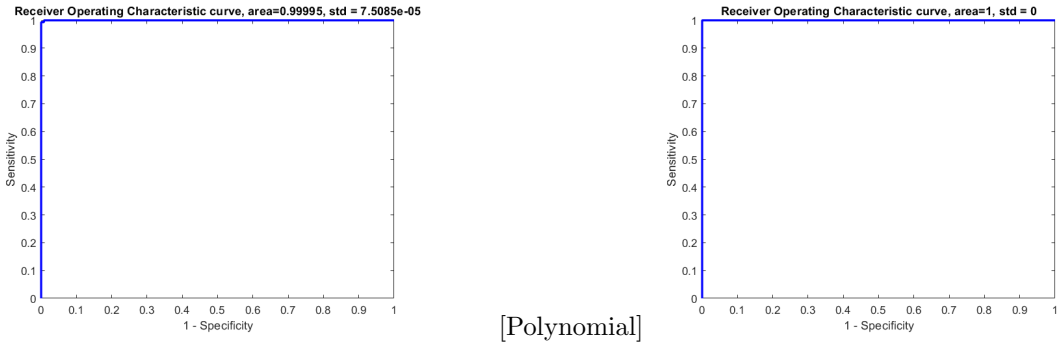


[RBF] [Polynomial]

[RBF] [Polynomial]

Figura 13: Ripley dataset: RBF vs Polynomial kernel

obtained using different kernels.
we can now plot the ROC curves for diabetes.mat and breast.mat. Roc curves for breast dataset are shown in figure 14 and 15.



[RBF] [Polynomial]

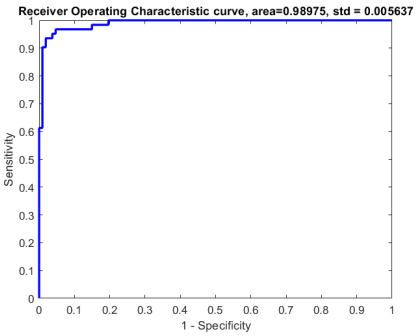
Figura 14: RBF vs Polykernel: Breast dataset: ROC on training

Next, we plot ROC curves for diabetes dataset in figure 16.We also included the ROC graph computed on the test dataset for linear kernel in figure 18.

- Are you satisfied with the performance of your model? Would you advise another methodology?

The models show good performance for either radial basis function kernel or for polynomial kernel. It can be seen from figure 14 and 16 that polynomial kernel outperforms radial basis function kernel if the dimension of the input data is high. 14 shows the breast dataset which has 30 features. We see that polynomial kernel outperforms rbf kernel. 16 shows diabetes dataset which has 8 features and we can see that there also polynomial kernel outperforms rbf kernel.
It seems that models with polynomial kernels are overfitting. The performance of linear kernel is as good as that of rbf kernel. We conclude that in some cases, data is in high dimensions but stil linear kernel can perform better. It would be a wrong decision to pick polynomial or rbf kernel without trying linear kernel.

[RBF]



[Polynomial]

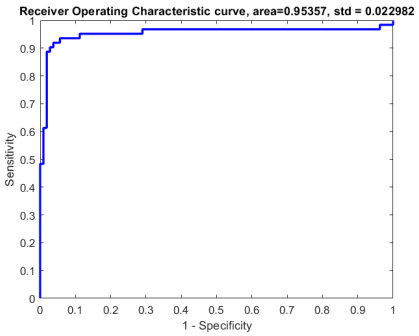
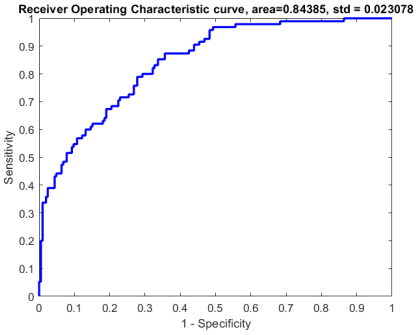


Figura 15: RBF vs Polykernel: Breast dataset: ROC on Testdata

[RBF]



[Polynomial]

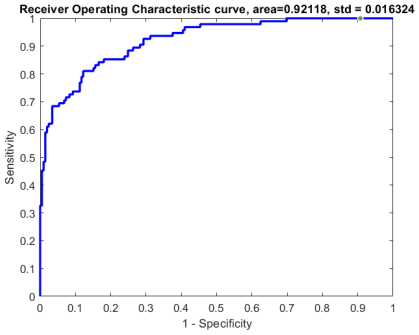
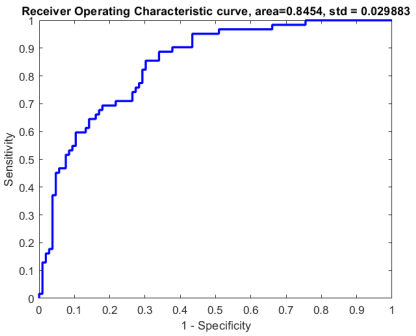


Figura 16: RBF vs Polykernel: Diabetes dataset: ROC on training

[RBF]



[Polynomial]

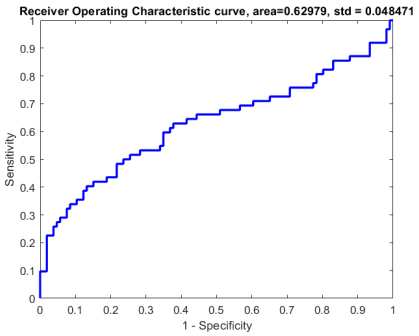


Figura 17: RBF vs Polykernel: Diabetes dataset: ROC on Testset

[Linear]

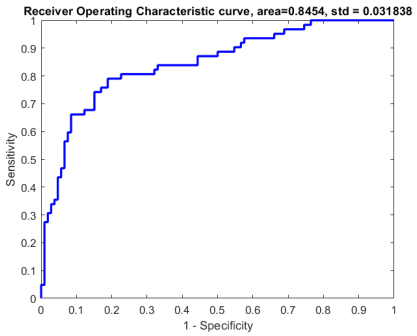


Figura 18: Linear kernel: Diabetes dataset: ROC on Testset