Instituto Superior Técnico

Departamento de Engenharia Electrotécnica e de Computadores

Machine Learning

6th Lab Assignment

Shift sexta 14 Group number 1

Number 84053 Name Francisco Raposo de Melo

Number 89213 Name Rodrigo Tavares Rego

1 Dataset 1

1.1 Naive Bayes Classifier

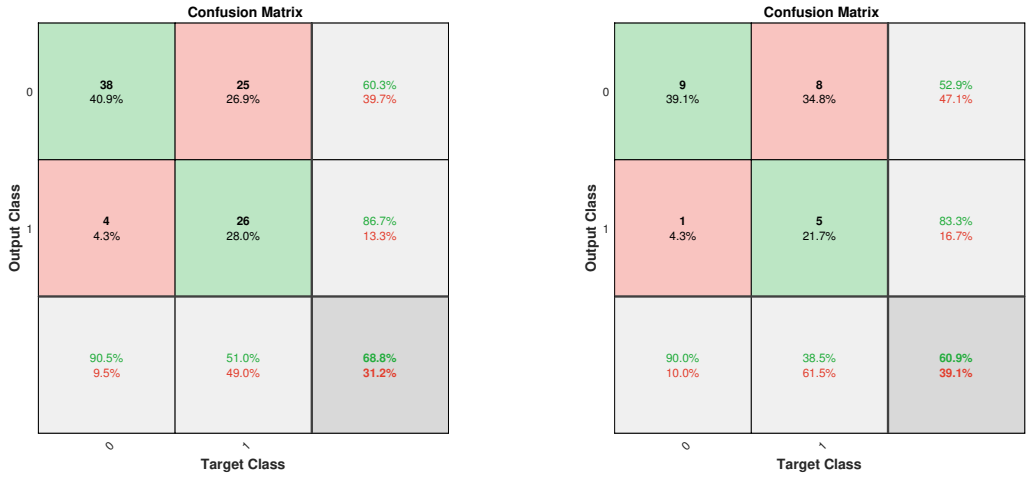
1.1.1 Choice of Hyperparameters

The Naive Bayes classifier uses estimations of probability densities of each class for each feature. The MATLAB function *fitcnb*, used to train the Naive Bayes classifier, computes these estimations of probability densities, variance and mean values.

Training Data

- Accuracy = 68,8%
- Error = 31,1%

1.1.2 Performance Evaluation



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 1: Confusion matrices for testing and training data for Naive Bayes Classifier

Testing Data

- Accuracy = 60,9 %
- Error = 39,1 %

- Sensitivity = 38,5 %
- Specificity = 90 %
- Precision = 0,83
- Recall = 0,38
- F-Measure = 0,53
- AUC = 0,642
- ROC curve

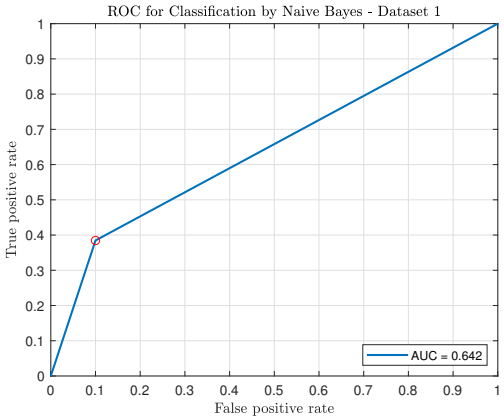


Figure 2: ROC curve for classification by Naive Bayes

1.1.3 Comments

The Naive Bayes classifier didn't yield a reasonably good result for this dataset. The ROC curve, which shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity), shows that the classification test was not very accurate (low AUC, should be closer to 1).

This is an indication that there is indeed a considerable dependency between each feature in this dataset, which violates the naive assumption of independent probabilities and translates to an inaccurate classification, otherwise the Naive Bayes classifier should perform fairly well.

However, this result could be improved with cross-validation, considering that this dataset is small.

1.2 Support Vector Machine (Polynomial Kernel)

1.2.1 Choice of Hyperparameters

Method:

- Start with a set of possible polynomial orders (and fixed box constraint = 10);
- Choose the best polynomial order (low training error);
- Test for a set of possible box constraints with fixed polynomial order chosen in the last step.

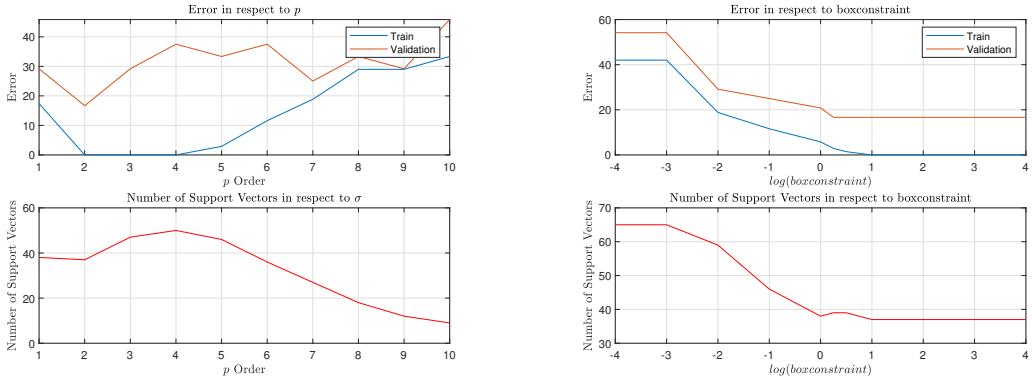
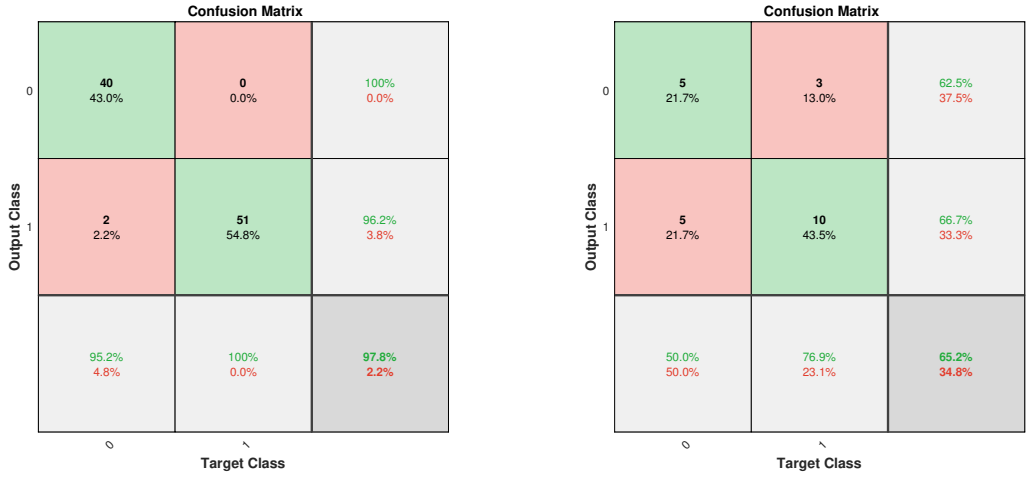


Figure 3: Test results for different polynomial orders and for different box constraints.

Training Data

- p = 2
- Box Constraint = 10
- Accuracy = 97,8 %
- Error = 2,15 %

1.2.2 Performance Evaluation



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 4: Confusion matrices for testing and training data for SVM (polynomial kernel)

Testing Data

- Accuracy = 65,2 %
- Error = 34,8 %
- Sensitivity = 76,9 %
- Specificity = 50 %
- Precision = 0,67
- Recall = 0,77
- F-Measure = 0,71
- AUC = 0,635
- ROC curve

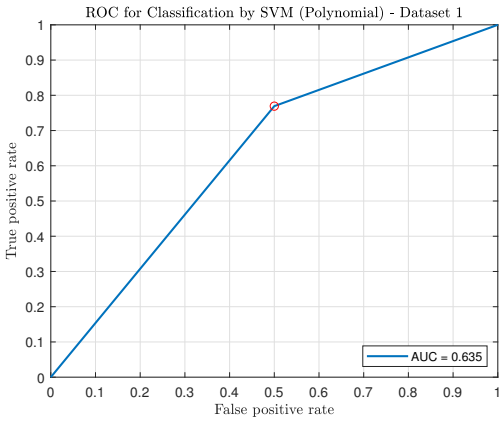


Figure 5: ROC curve for classification by SVM (polynomial kernel)

1.2.3 Comments

This SVM classifier with polynomial kernel performed just a little bit better than the Naive Bayes. This slight improvement is not enough to lead to a conclusion of whether we should choose one over the other, for this dataset. This result was obtained using a validation set.

The low value of boxconstraint indicates that there is a preference for a softer margin, and a lower polynomial order indicates that there wasn't too much overfitting, however it translates to a higher number of support vectors.

Using a validation set with this dataset is not recommended, since the dataset is very small, a better solution would be cross validation.

1.3 Support Vector Machine (Gaussian Kernel)

1.3.1 Choice of Hyperparameters

Method:

- Start with a set of possible values of σ (and fixed box constraint = 10);
- Choose the best value of σ (low training error);
- Test for a set of possible box constraints with a fixed value of σ chosen in the last step.

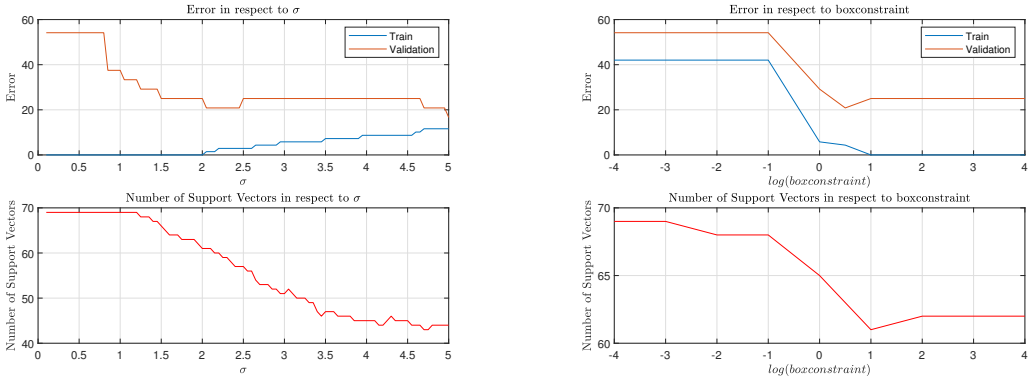
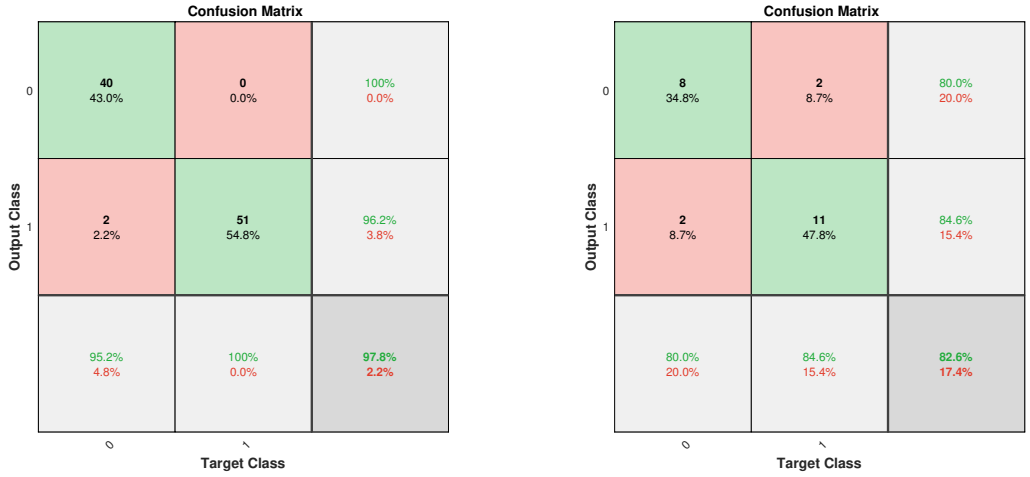


Figure 6: Test results for different values of σ and for different box constraints.

Training Data

- $\sigma = 2$
- Box Constraint = 10
- Accuracy = 97,8 %
- Error = 2,2 %

1.3.2 Performance Evaluation



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 7: Confusion matrices for testing and training data for SVM (Gaussian kernel)

Testing Data

- Accuracy = 82,6 %
- Error = 17,4 %
- Sensitivity = 84,6 %
- Specificity = 80 %
- Precision = 0,85
- Recall = 0,85
- F-Measure = 0,85
- AUC = 0,823
- ROC curve

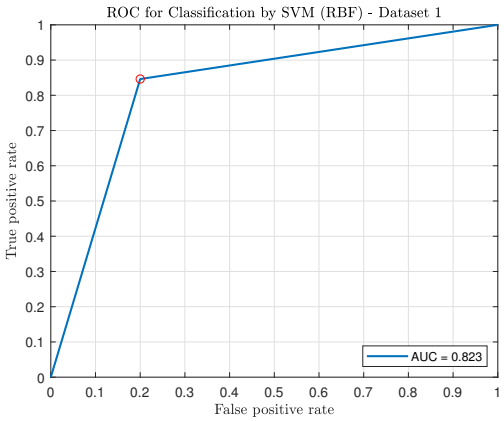


Figure 8: ROC curve for classification by SVM (Gaussian kernel)

1.3.3 Comments

This SVM classifier with Gaussian kernel performed better than the SVM classifier with polynomial kernel. Gaussian RBF has shown to be more flexible and better for data that can't be linearly separable, because it allows more possible parameters and classifiers for which we get proper classification, which is better for when we don't know exactly how the data is shaped beforehand.

This improvement (accuracy increased from 65,2% to 82,6% in testing data) is enough to lead to a conclusion that we should choose Gaussian Kernel over Polynomial Kernel, for this dataset. This result was obtained using a validation set.

The low value of boxconstraint indicates that there is a preference for a softer margin. The best classifier was obtained for $\sigma = 2$, for which the error was minimal, however it translates to a higher number of support vectors.

Using a validation set with this dataset is not recommended, since the dataset is very small, a better solution would be cross validation.

1.4 Decision Trees

1.4.1 Choice of Hyperparameters

It should be tested if the decision tree can be pruned, because too many levels tend to overfit the training data.

To check if it can be pruned, the training error was computed for each pruning (at each level).

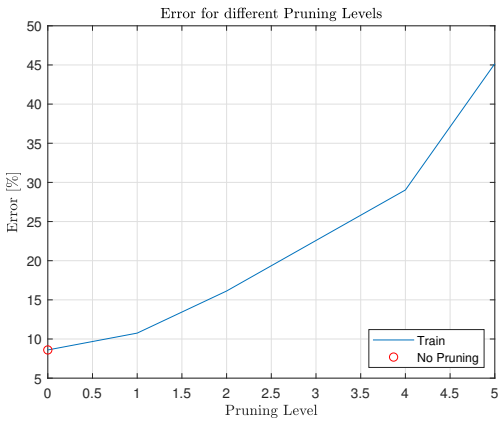


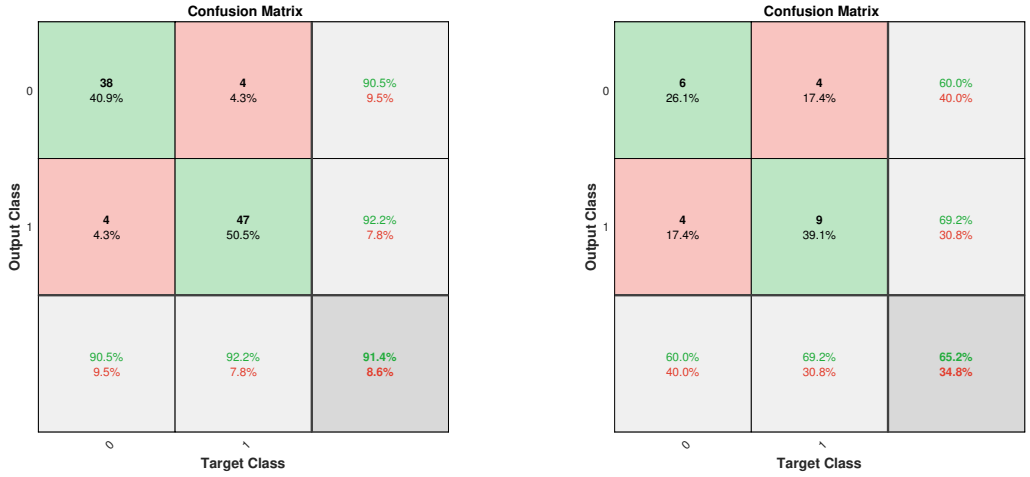
Figure 9: Error in respect to pruning level.

Since the error grows after the first prune, the tree should not be pruned.

Training Data

- Pruning Level = 0 (no pruning)
- Accuracy = 91,4 %
- Error = 8,6 %

1.4.2 Performance Evaluation



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 10: Confusion matrices for testing and training data for decision tree

Testing Data

- Accuracy = 65,2 %
- Error = 34,8 %
- Sensitivity = 69,2 %
- Specificity = 60 %
- Precision = 0,69
- Recall = 0,69
- F-Measure = 0,69
- AUC = 0,646
- ROC curve

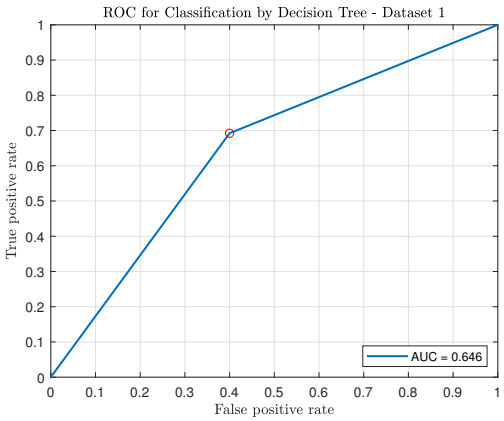


Figure 11: ROC curve for classification by Decision Trees

1.4.3 Comments

Decision trees are more intuitive to comprehend and tend to do feature selection, where the more important ones are found in the bottom of the tree. However, decision trees don't make any distributional assumptions about the data (especially the output), which can be either good or bad.

In case the probability distribution assumption is somehow weak for the dataset in question (which can happen with other classifiers), decision trees may hold a relative advantage. However, these assumptions might actually hold and, in this case, other methods may perform better.

With this dataset the classification with a decision tree didn't perform better than the SVM (RBF kernel) classifier, however it had an equivalent performance compared to Naive Bayes and SVM (polynomial kernel) classifiers.

1.5 Conclusions

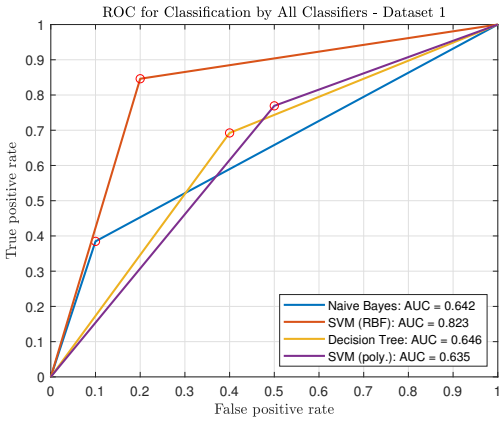


Figure 12: ROC curve for classification by Naive Bayes, SVM (polynomial kernel), SVM (gaussian Kernel) and Decision Trees

In order to obtain better results and thus better conclusions about the performance of the classifiers, feature selection and cross-validation should've been applied to this data.

As we can observe in figure 8, all of the classifiers have a ROC curve close to a 45º diagonal, except for the SVM with RBF kernel. This means that the SVM (RBF kernel) performs considerably better than all of the other classifiers for this first dataset. This is corroborated by the AUC values, where the AUC for the SVM (RBF kernel) classifier has a notably higher value of AUC (AUC should be considered as a general measure of predictive accuracy, may not be enough to decide which classifier performs better).

To sum up, the SVM (with RBF kernel) classifier generalizes more, however this conclusion may be dependent of the testing set in use. Cross-validation would have diminished bias with all these classifiers and different conclusions might have been made, however, in general it is acceptable to conclude that for `dataset1.mat` it is preferable to use a SVM with RBF kernel as a classification method.

2 Dataset 2

For `dataset2.mat` the training data was divided into a training set and a validation set. This is possible with this data set, since there is considerably more data than the previous dataset. The existence of a validation set allows the possibility of having a less biased choice of hyperparameters.

It was decided to have a validation set of approximately the same size as the testing set.

2.1 Support Vector Machine (Polynomial Kernel)

2.1.1 Choice of Hyperparameters

Method:

- Start with a set of possible polynomial orders (and fixed box constraint = 10^4);
- Choose the best polynomial order (low training error considering the behaviour of the validation error);
- Test for a set of possible box constraints with fixed polynomial order chosen in the last step.

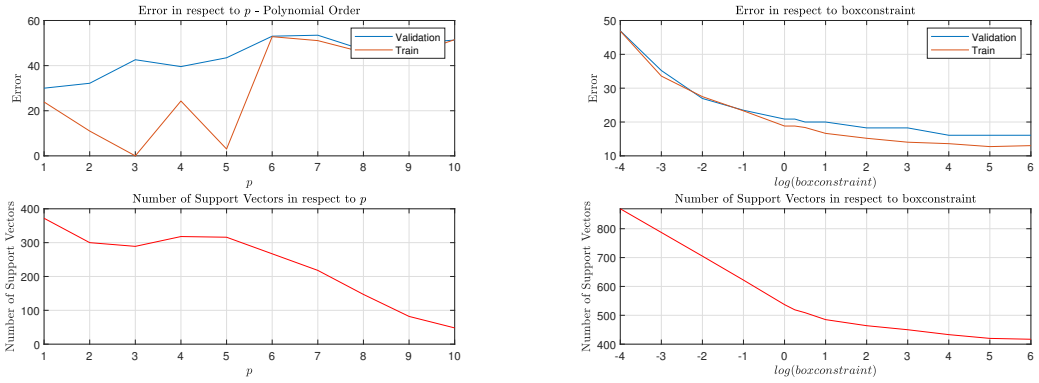


Figure 13: Test results for different polynomial orders and for different box constraints.

Training Data

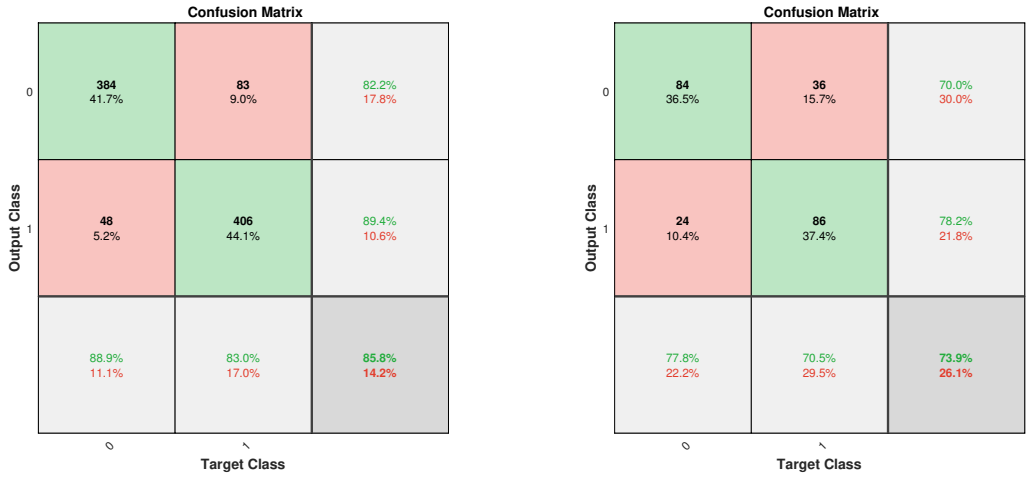
- p = 2
- Box Constraint = 10^4

- Accuracy = 85,8 %
- Error = 14,2 %

Validation Data

- Accuracy = 83,9 %
- Error = 16,1 %

2.1.2 Performance Evaluation



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 14: Confusion matrices for testing and training data for SVM (polynomial kernel)

Testing Data

- Accuracy = 73,9 %
- Error = 26,1 %
- Sensitivity = 70,5 %
- Specificity = 77,8 %
- Precision = 0,78

- Recall = 0,70
- F-Measure = 0,74
- AUC = 0,741
- ROC curve

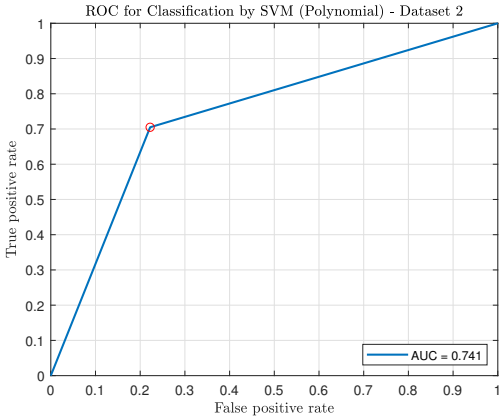


Figure 15: ROC curve for classification by SVM (polynomial kernel)

2.1.3 Comments

This SVM classifier performed fairly well considering the high number of dimensions of features.

The value of boxconstraint indicates that there is a preference for a harder margin, and a lower polynomial order indicates that there wasn't too much overfitting, however it translates to a higher number of support vectors.

2.2 Support Vector Machine (Gaussian Kernel)

2.2.1 Choice of Hyperparameters

Method:

- Start with a set of possible values of σ (and fixed box constraint = 10);
- Choose the best value of σ (low training error considering the behaviour of the validation error);
- Test for a set of possible box constraints with a fixed value of σ chosen in the last step.

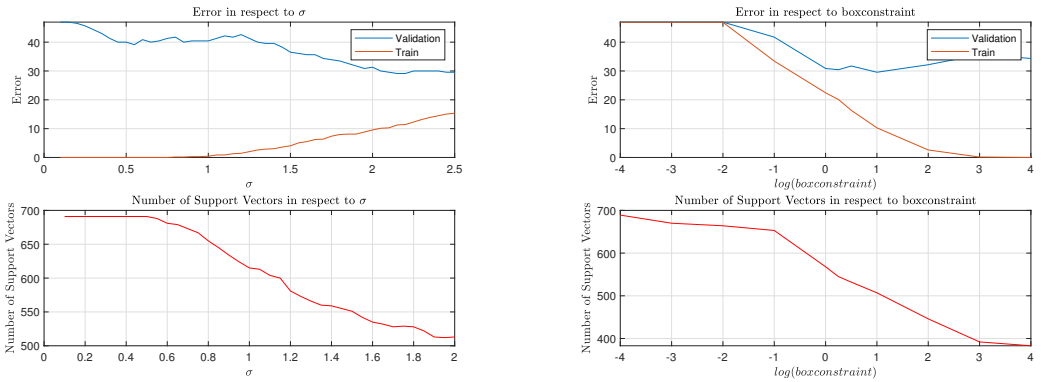


Figure 16: Test results for different values of σ and for different box constraints.

Training Data

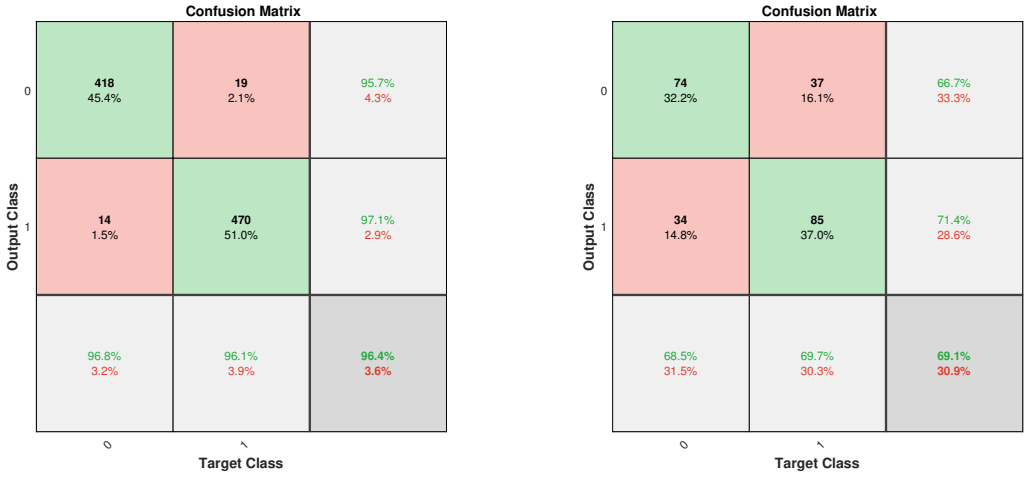
- σ = 2,1
- Box Constraint = 10^2
- Accuracy = 96,4 %
- Error = 3,6 %

Validation Data

- Accuracy = 96,5 %
- Error = 3,5 %

2.2.2 Performance Evaluation

Testing Data



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 17: Confusion matrices for testing and training data for SVM (Gaussian kernel)

- Accuracy =69,1 %
- Error = 30,9 %
- Sensitivity = 69,7 %
- Specificity = 68,5 %
- Precision = 0,71
- Recall = 0,69
- F-Measure = 0,71
- AUC = 0,69
- ROC curve

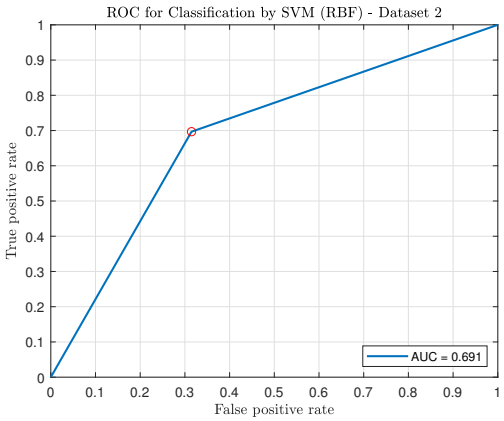


Figure 18: ROC curve for classification by SVM (gaussian kernel)

2.2.3 Comments

This SVM classifier with Gaussian kernel performed slightly worse than the SVM classifier with polynomial kernel. There are some situations where the RBF kernel is not suitable. In particular, when the number of features is very large. If the number of features is large, one may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance.

The value of boxconstraint indicates that there is a preference for a softer margin. The best classifier was obtained for $\sigma = 2{,}1$, for which the error and number of support vectors were minimal for the validation set, allowing some error with the training set which translates to less overfitting and more generalization.

2.3 Decision Trees

2.3.1 Choice of Hyperparameters

It should be tested if the decision tree can be pruned, because too many levels tend to overfit the training data.

To check if it can be pruned, the training error was computed for each pruning (at each level) along with a validation error for this dataset.

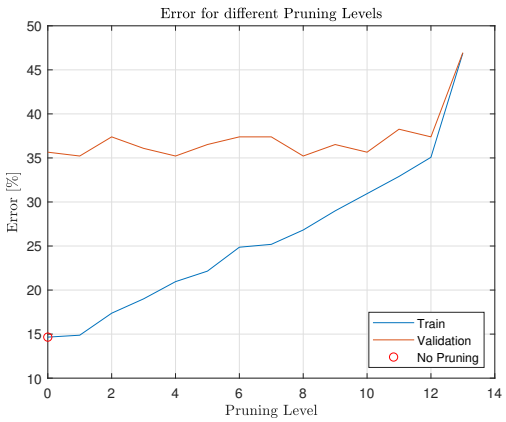


Figure 19: Error in respect to pruning level.

Since the training error grows insignificantly in pruning level 1 and the validation set error diminishes, then the tree should be pruned to level 1 (reduces overfitting).

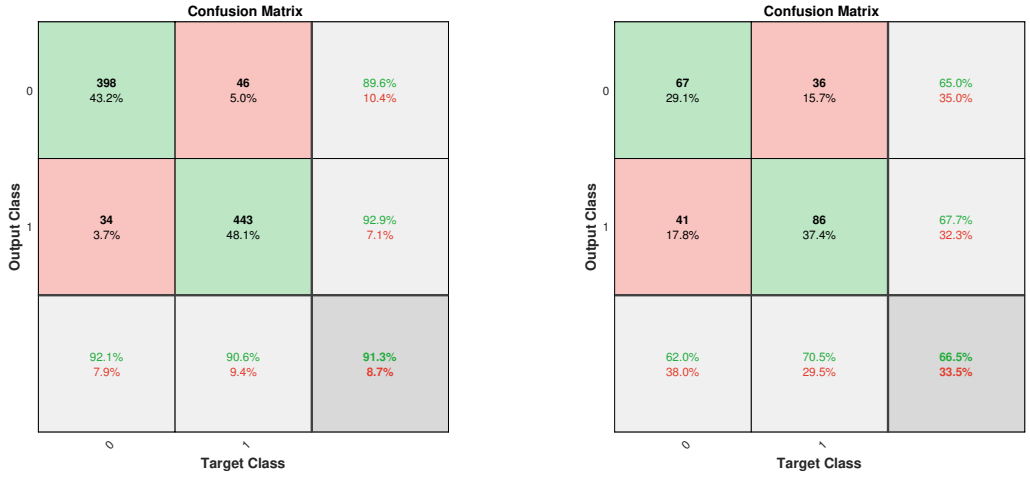
Training Data

- Pruning Level = 1
- Accuracy = 91,4 %
- Error = 8,6 %

Validation

- Accuracy = 91,3 %
- Error = 8,7 %

2.3.2 Performance Evaluation



(a) Confusion Matrix for training data

(b) Confusion Matrix for testing data

Figure 20: Confusion matrices for testing and training data for decision tree

Testing Data

- Accuracy = 66,5 %
- Error = 33,5 %
- Sensitivity = 70,5 %
- Specificity = 62,0 %
- Precision = 0,68
- Recall = 0,70
- F-Measure = 0,69
- AUC = 0,663
- ROC curve

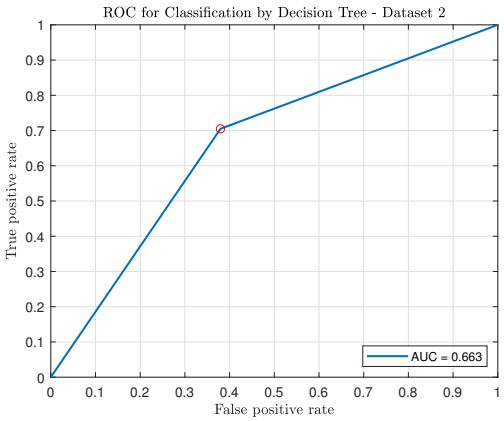


Figure 21: ROC curve for classification by Decision Trees

2.3.3 Comments

Considering what has been said about decision trees in the previous dataset, with this dataset the classification with a decision tree didn't perform better than the SVM (polynomial kernel) classifier, however it had an equivalent performance compared to SVM (RBF kernel) classifiers.

In this case, the validation set was used to evaluate whether or not the tree should be pruned, in order to lessen the overfitting.

2.4 Neural Network

2.4.1 Choice of Hyperparameters

Learning rate and momentum constant were determined by trial and error in order to minimize number of epochs.

Training Data

- Number of units in hidden layer = 15
- Performance Function: Mean squared normalized error
- Activation Function: hyperbolic tangent (layer 1 - hidden)
- Activation Function: softmax (layer 2 - output)
- Network Training Function: Gradient descent with momentum and adaptive learning rate backpropagation
- Learning Rate = 0,5
- Momentum Constant = 0.6
- Max Epochs = 10000
- Training Goal = 0.05
- MSE = 0,2070
- Accuracy = 68 %
- Error = 32 %
- Epochs = 55

2.4.2 Performance Evaluation

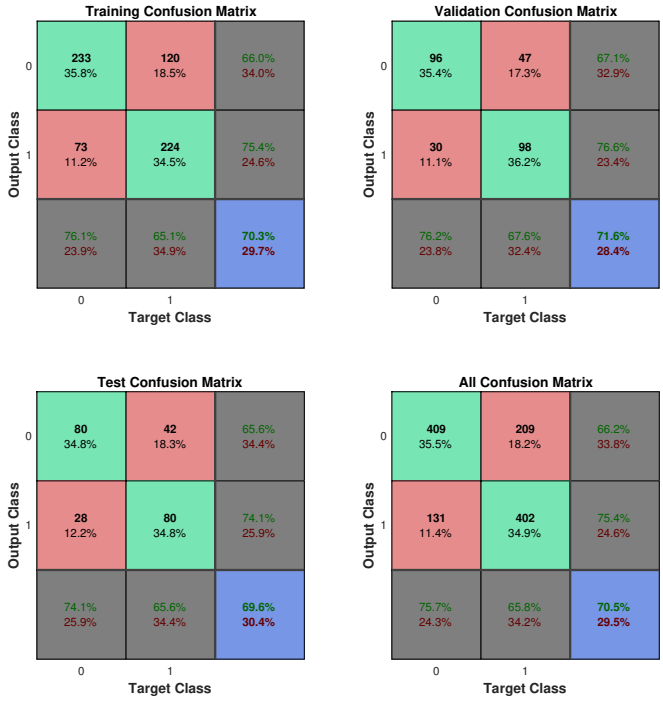


Figure 22: Confusion matrices of training, validation and testing for neural network

Note: These confusion matrices were obtained from another run of the neural network.

Testing Data

- MSE = 0,2139
- Accuracy = 67 %
- Error = 33 %
- AUC = 0,72
- ROC curve

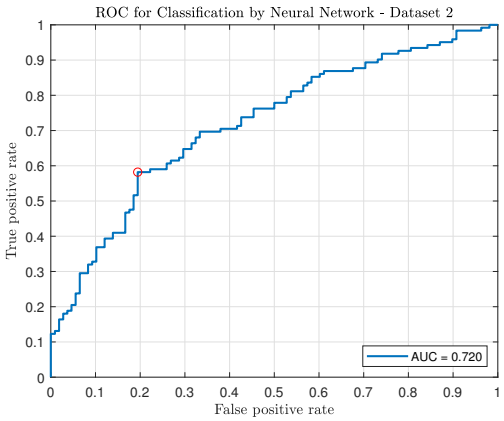


Figure 23: ROC curve for classification by neural networks

2.4.3 Comments

Multiple tests were made in order to choose the best parameters to train the neural network.

The stopping criterion was the number of validation checks.

The quality of networks is best evaluated with the global percentage of correct classifications.

The MSE of the testing error and validation error will accordingly be minimized, however it will end up a little deviated from the original goal. The only way of knowing how much that deviation impacts the classification is by checking the global percentage of correct classifications of the testing set (unbiased).

With this dataset the classification with neural network, didn't perform better than the SVM (polynomial kernel) classifier, however it had an equivalent performance compared to SVM (RBF kernel) and decision trees classifiers.

2.5 Conclusions

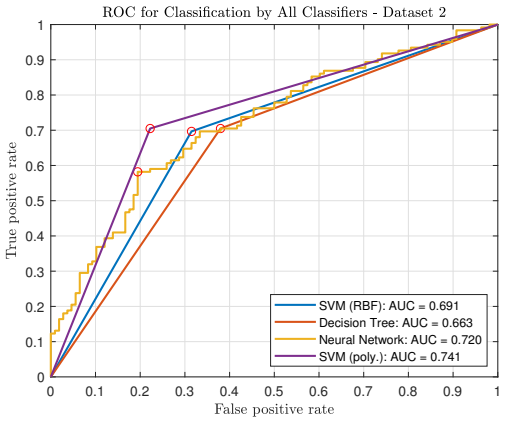


Figure 24: ROC curve for classification by SVM (polynomial kernel), SVM (gaussian kernel), Decision Trees and Neural Network.

In order to obtain better results and thus better conclusions about the performance of the classifiers, feature selection should've been applied to this data, especially considering that this dataset has 17 features.

As we can observe in figure 16, the SVM (polynomial kernel) and the Neural Network had the best performances, however the choice is not evident since the relative performance is close between each classifier. Overall, and accordingly to the AUC as well, the best choice of classifier and the one that generalizes more is the SVM (polynomial kernel).

The use of a validation set allowed for a better choice of the hyperparameters in this dataset. The idea was to get a more generalized model in expense of getting some training error and gaining less overfitting.

Considering that the best classifier was a SVM, it is important to acknowledge that with high dimensions it's preferable to start with linear kernel functions (as explained before). At the end we tried a SVM with linear kernel, however the results were basically the same as the ones we get with a polynomial kernel, except when we consider the gain of training error.

Personally we think that the Neural Network could have been better trained with a better choice of the hyperparameters and ideally it would yield the best results given that it would take more time to train. However, since the dataset in question is still considered small when the matter is the training of a neural network, that might not be the case.

To sum up, the choice of a classifier is dependent of the size of the datasets (neural networks need a lot of training examples), the distributional assumptions of the data (decision trees don't consider distributional assumptions), the number of features (the choice of the kernel function for a SVM depends on the dimensions of the dataset), the type of response variable (in this case the classification is binary). In order to choose the best classifier one should consider all of these points and also try to define whether or not the features are dependent of each other, or if the data is noisy, considering as well validation methods and feature selection.