Introduction to Data Science

Assignment 2

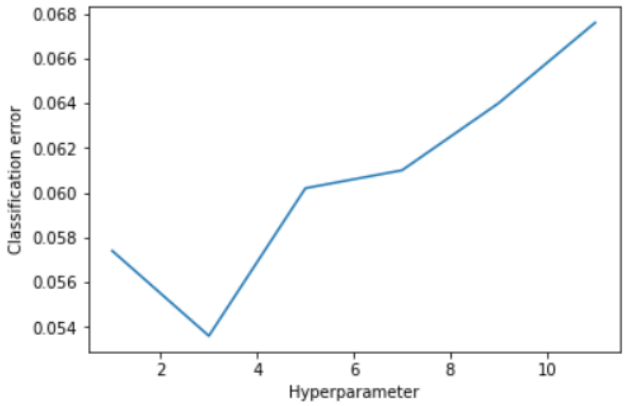
Péter Pölös

Exercise 1

The training accuracy of the 1-NN classifier turned out to be 1, while the testing accuracy is 0.96. This results are not surprising, since we know that a K-NN classifier with k=1 will always deliver 100% accuracy on the training data set. But this does not mean that it can classify anything correctly on any non-training data point, as we can see in this case as well. I have also compared the results of my own implementation with the built in one in scikit-learn, and they produce the same results. (but probably it is much slower)

Exercise 2

I am using cross-validation to determine a proper hyperparameter. First I split my training data set into 5 groups/folds (it is important that we should avoid using our testing dataset for this, since that would lead to overfitting). Each time I pick one of the 5 groups to be a validation set. The misclassification rate is then computed on the observations in the held-out fold. This procedure is repeated 5 times; each time, a different group of observations is treated as a validation set. This process results in 5 estimates of the test error which are then averaged out. I repeat this process for multiple hyperparameters, so in the end I will have average classification errors belonging to different values of hyperparameters. I will choose the hyperparameter with the least average classification error, as it can be seen from the plot, this case it turned out to be 3.



Exercise 3

The training accuracy of the model with the optimal number of neighbors (k=3) is 0.971, while the testing accuracy is a bit lower, around 0.949. Not surprising to see that there is a higher training accuracy than testing accuracy due to the big issue of testing on your training data, which can lead to overfitting. Overfitting means that your model has learnt rules specifically for the training set, those rules do not generalize well beyond the training set, as it can be see form our results here.

Exercise 4

The first version is the correct one, where we scale the training and testing data according to the training data set. As a general rule, we should not use the test data in the model building process at all (neither for training, data normalization, nor hyperparameter selection), because otherwise we may get a biased estimate of the generalization performance of the model. That is the reason why the other two methods are not good, since they use information from the testing data for scaling, what we should avoid doing.

Using cross-validation again I have looked for the best hyperparameter for the model, and it turned out to be 3 again. Our accuracy has improved in case of the training and testing accuracy as well. The training accuracy with k=3 is 0.972 and test accuracy with k=3 is around 0.9599.

Without normalization some features may have a large impact on the results depending on if their values move on larger scales then the values of other features. That's the idea behind the normalization, is to avoid the bias of these features and to sort of give an equal weight to all the features in predicting the labels of the data points. That's reason why we end with better estimates (higher accuracy) in case of normalization.