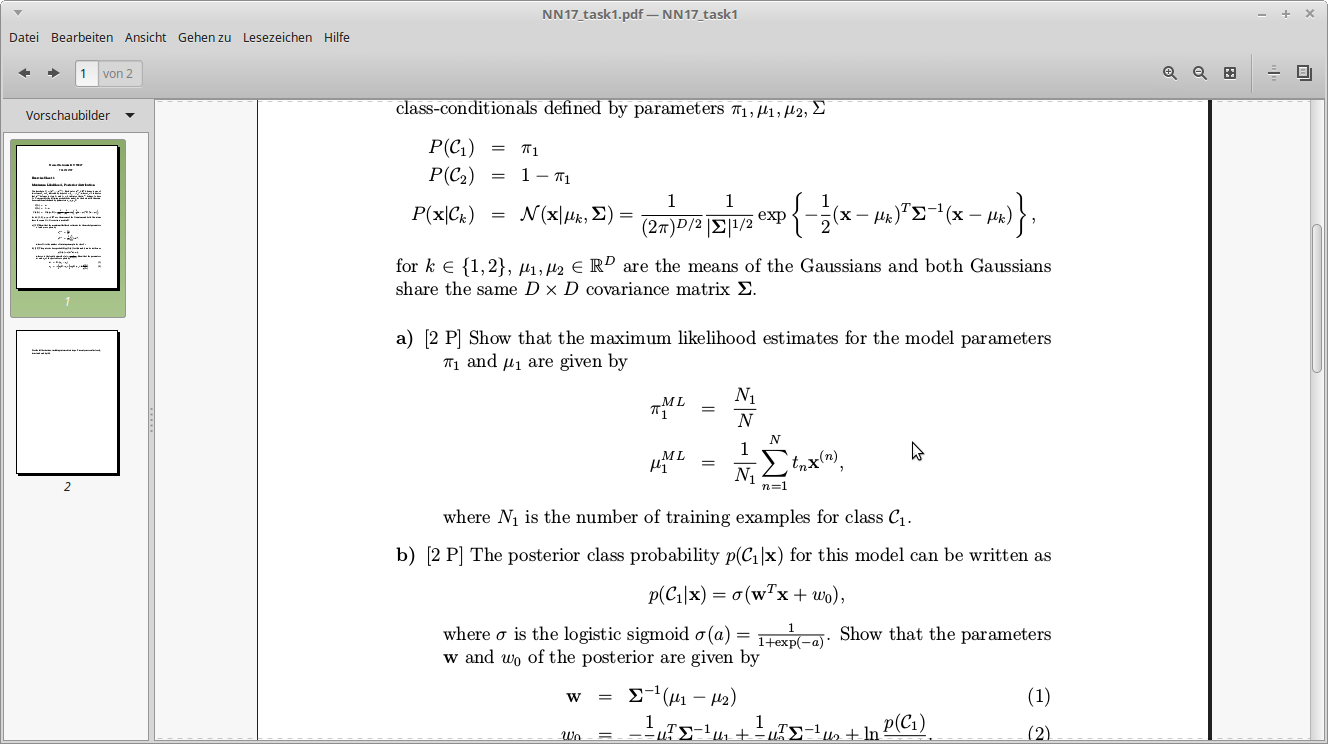
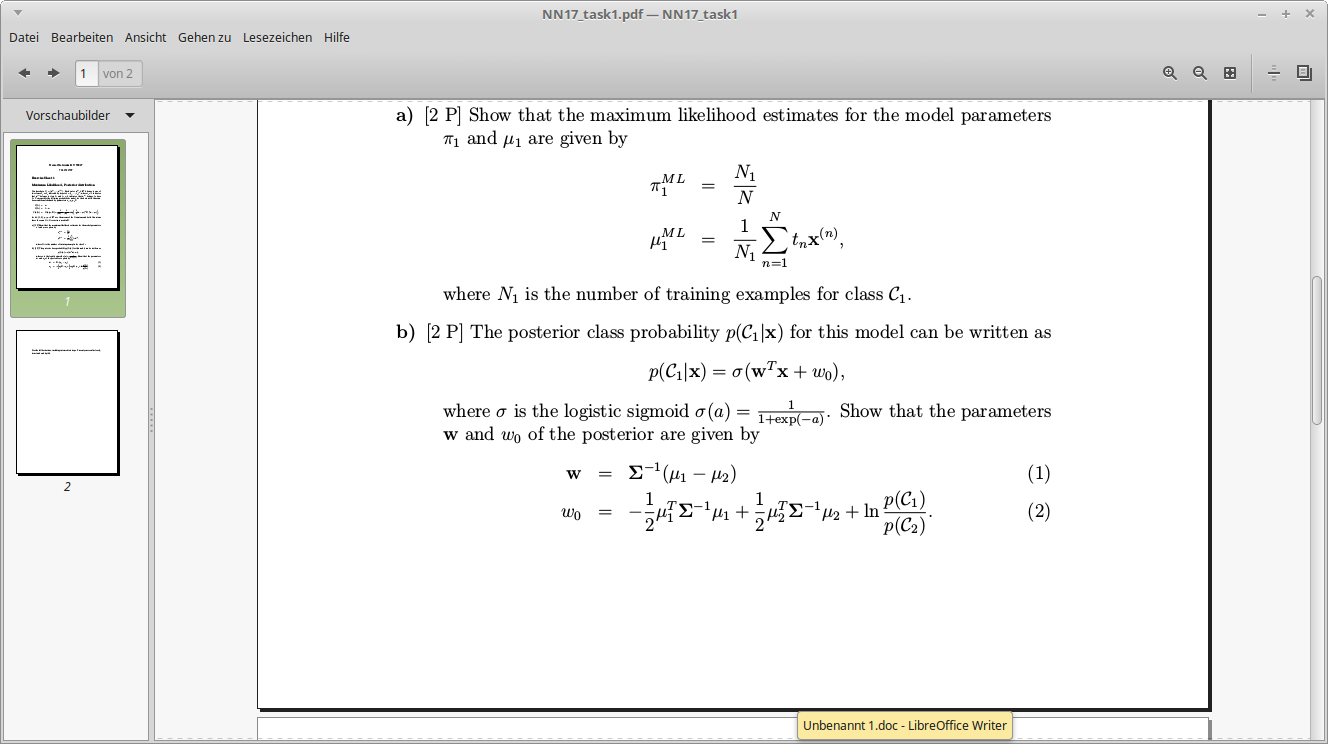
# Task a1

In our dataset of cars all cars which belong to class 2 (Saab) or class 3 (Bus) are extracted. Afterwards, the parameters of the Gaussian distribution are estimated by formulas below:



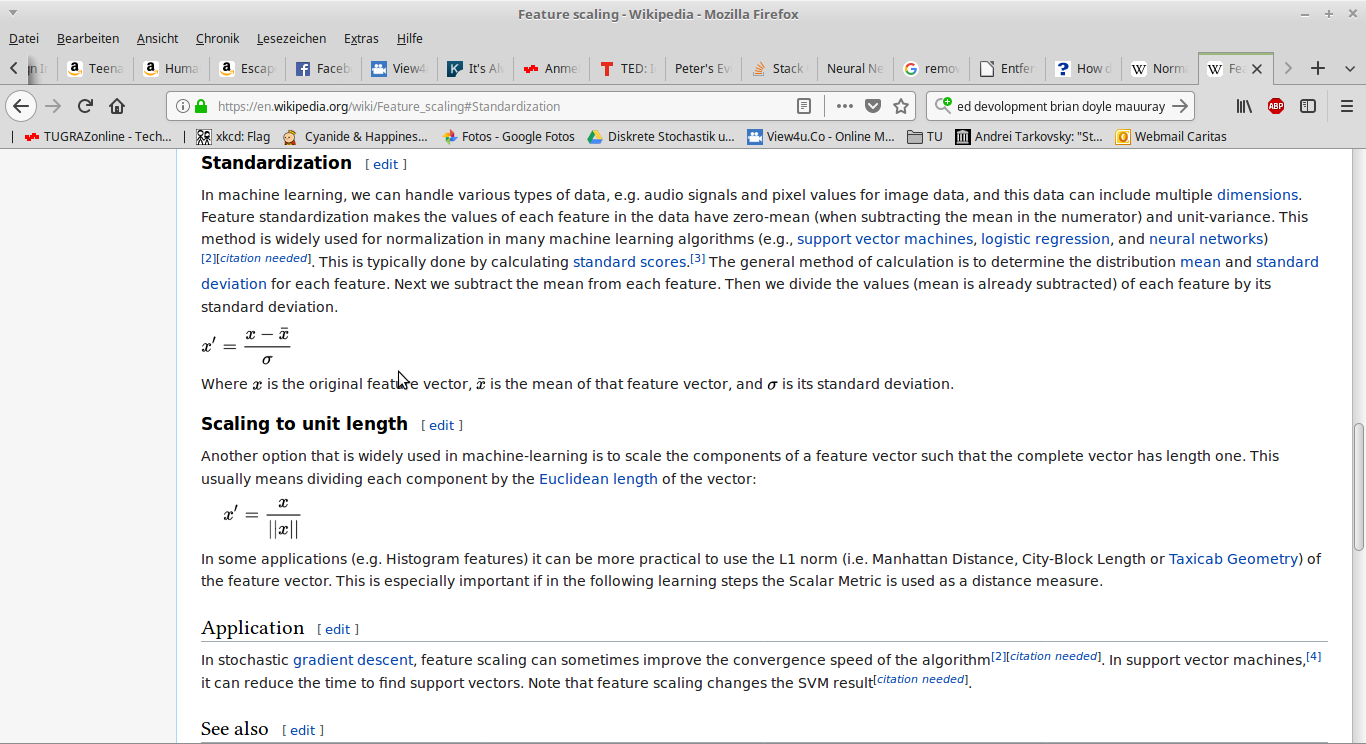
Then we apply classification by the posterior distribution over training and test set, using the formula



An indicator for measuring performance is the misclassification rate, which expresses the percentage of wrong classified cars. These performances are 3.5% on the training set and 2.68% on the test set, which is a quite low error rate. This result appears because the test dataset is quite big and the algorithm performs a good generalization.

# Task a2

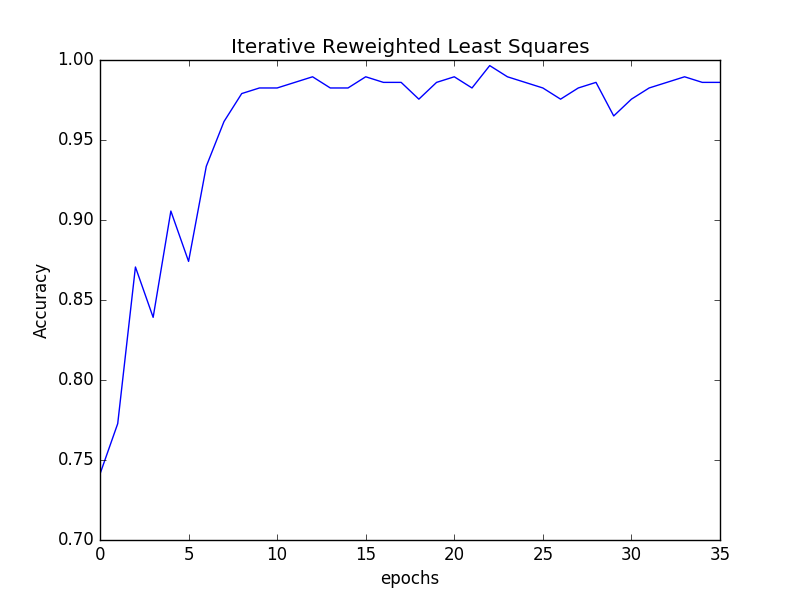
The question is the same as in a1, with the difference that the input data should be normalized before it gets processed using the formula



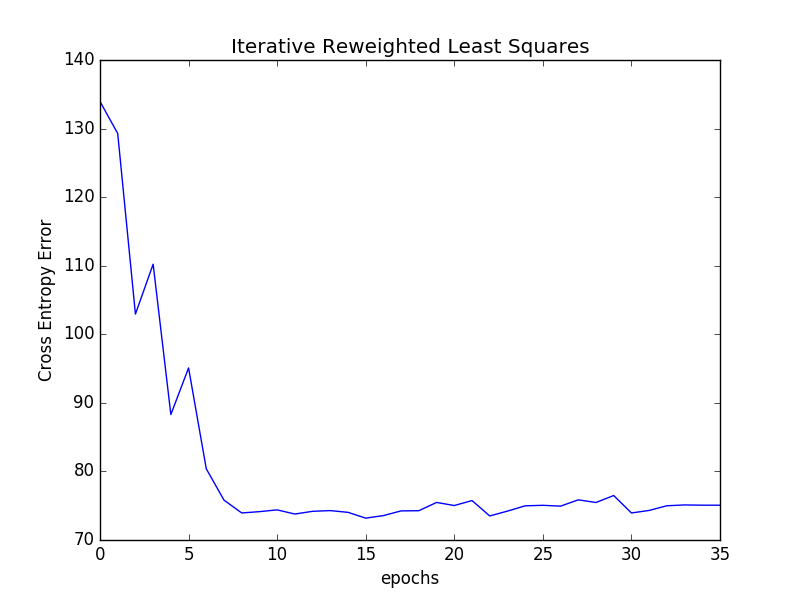
All test entries are normalized by using the parameters of the training data. Our misclassification rate on the test dataset increases to 4.03%, which means that the normalization does not increase the performance of the algorithm in this case. This results in the fact that normalization is not needed in this case in the algorithm. One special observation has been made by implementing the sigmoid function: if the values are too small, then Python reports a log overflow error.

# Task b

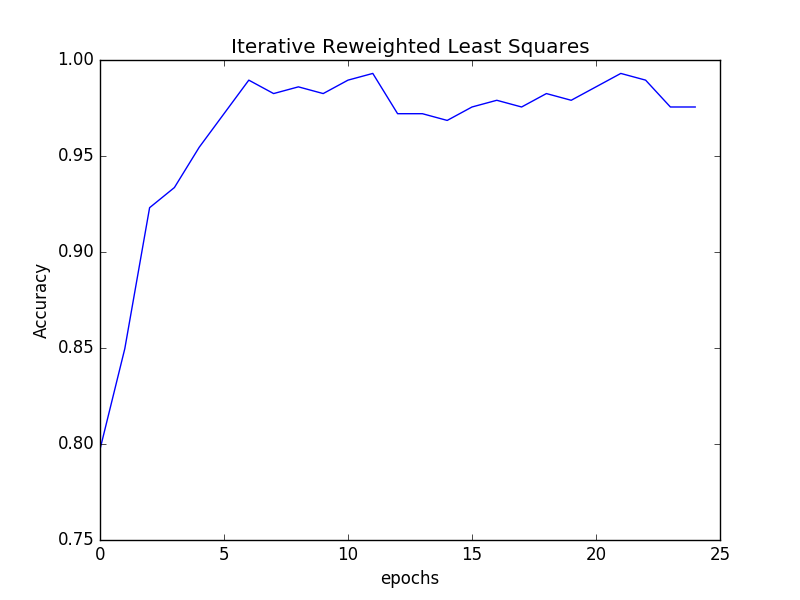
The task was to implement the iterative reweighted least squares (IRLS) algorithm and plot the cross entropy error and the misclassification rate. These plots are displayed on the next pages and show us, that the algorithm performs also quite good with a misclassification rate of about 2% and a cross entropy error of about 80 for both, normalized and unnormalized data. A difference between normalized and unnormalized data is visible in performance is visible in the plots. The normalized curves look more like a continuous function, whereas the other ones have some small jumps in between, especially at the first few epochs. All results are quite similar to those of task a, but the IRLS algorithm converges very fast. If the output does not change for 3 iterations we stop the algorithm (stopping criterion). The initial weights must not be too big, as this would result in a singular hessian matrix.



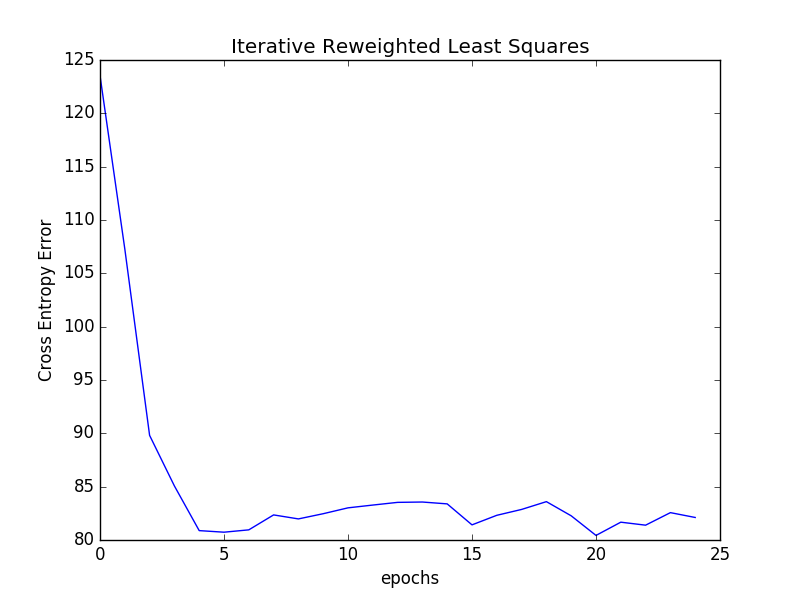
IRLS with unnormalized input data



Cross entropy error with unnormalized input data



IRLS with normalized input data



Cross entropy error with normalized input data