Statistical Learning (5454) - Assignment 2

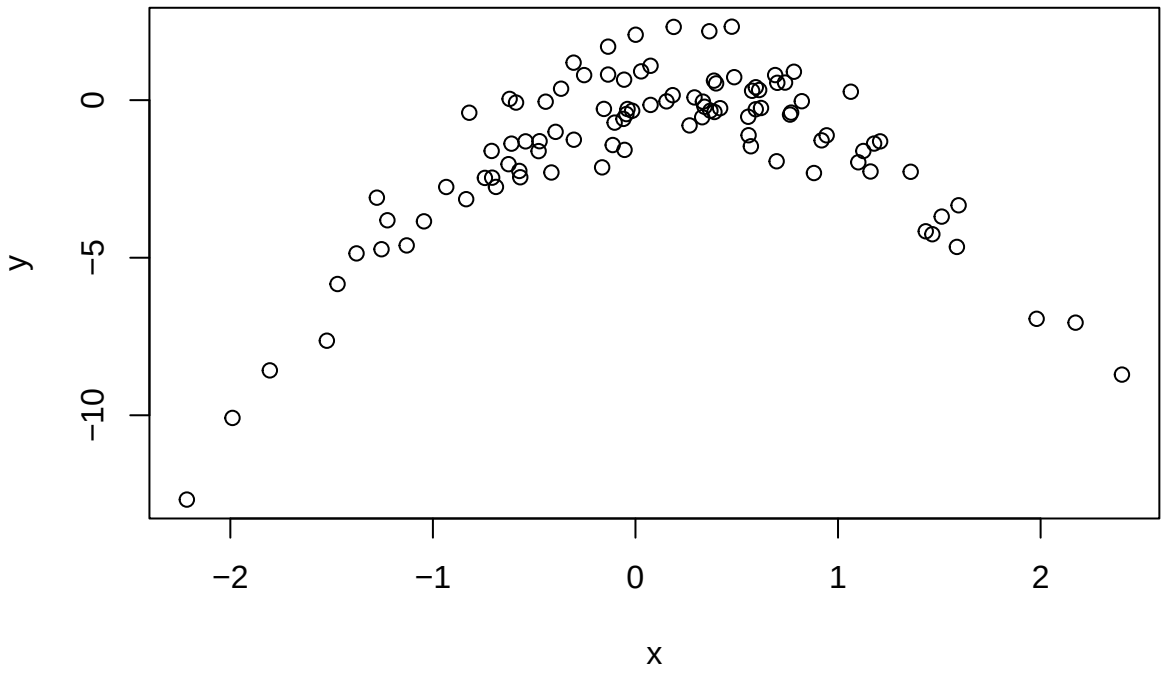
Matthias Hochholzer, Lukas Pirnbacher, Anne Valder

Due: 2024-04-22

Exercise 1

After generating the simulated data we fit four models using least squares, we calculate the AIC values and determine the in-sample error estimates by drawing suitable test data from the data generating process using twice the negative log-likelihood as loss function. Plotting the data we see a clearly non-linear relationship. In line with this we observe that the most preferable model in terms of AIC value is the second model. In terms of negative log-likelihood the fourth model appears to be the best. However, closely followed by model 2. The linear model (model 1) performs the worst out of all models measured by AIC and the negative log-likelihood.

```
#(a)
set.seed(1)
x <- rnorm(100)
y <- x - 2*x^2 + rnorm(100)
plot(x,y)
```



```r
#(b) model specifications:
lm1 <- lm(y ~ x) # alternative: lm1 <- glm(y ~ x, data, family = gaussian())
lm2 <- lm(y ~ x +I(x^2))
lm3 <- lm(y ~ x + I(x^2) + I(x^3))
lm4 <- lm(y ~ x + I(x^2) + I(x^3) + I(x^4))
```

```
## [1] 478.8804 280.1670 282.0886 282.2963
```

```
## [[1]]
## 'log Lik.' 526.7693 (df=3)
##
## [[2]]
## 'log Lik.' 279.7447 (df=4)
##
## [[3]]
## 'log Lik.' 522.8529 (df=4)
##
## [[4]]
## 'log Lik.' 278.7059 (df=6)
```

Exercise 2

In exercise 2, we perform leave-one-out cross-validation (LOOCV), k-fold cross-validation (kCV) and empirical bootstrapping based on the mean squared error loss that result from fitting the four models using least squares. The simulated data and specifications are the same as in task 1. The results of task (b) across the models are shown in table 1, incidcated by "seed1" in the last column.

(c) Next, we repeat (b) using another random seed (indicated by "seed2" in table 1). For LOOCV we obtain exactly the same results for all four models regardless of the different seed. This is because in general the randomness lies in the data generation.This consistency is expected because LOOCV is deterministic in this context, not relying on random sampling. Each observation is used once as a test set while the rest are used for training, and this process is repeated for each observation in the dataset. Therefore, changing the seed does not affect LOOCV results. For kCV we observe slight differences in the errors between seed 1 and seed 2 across the models. This variation can be attributed to the random partitioning of the data into k folds. Each seed leads to a different random split, which can result in slight variations in the training and validation sets used in each fold, thus affecting the error estimates. Similar to kCV, the bootstrap error shows variations between the two seeds. Bootstrap resampling involves drawing samples with replacement from the original data set to create "new" data sets. The randomness introduced by the seed affects which observations are selected in each resample, leading to slight differences in the bootstrap error estimates between seed 1 and seed 2.

Table 1: Comparison of LOOCV, kCV, and Bootstrap Errors

	LOOCV	kCV	Bootstrap	Model	Seed
1	7.2882	6.1856	6.2931	Model 1	Seed1
5	7.2882	6.4165	6.2764	Model 1	Seed2
2	0.9374	0.9230	0.8702	Model 2	Seed1
6	0.9374	0.9012	0.8704	Model 2	Seed2
3	0.9566	0.9316	0.8554	Model 3	Seed1
7	0.9566	0.8854	0.8557	Model 3	Seed2
4	0.9539	0.9247	0.8393	Model 4	Seed1
8	0.9539	0.8962	0.8452	Model 4	Seed2

(d) The MSEs in table 1 suggest that according to LOOCV the second model is the best. With kCV the

third model is the best and for the empirical bootstrap error it is the fourth model. Also, we see that the empirical bootstrap error decreases further with model complexity, reflecting potential overfitting problems of this method. Remembering the plotted data in the beginning these results are in line with our expectations since higher order regression equations fit much better to the data than the linear case.

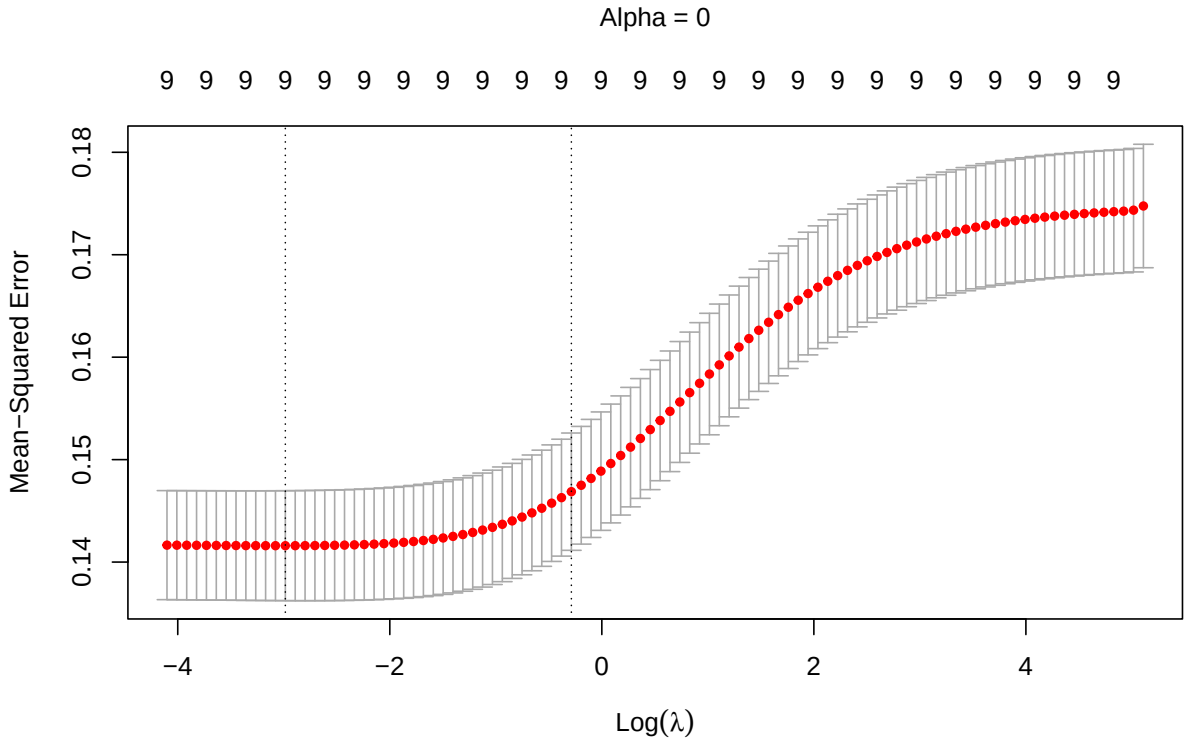
(e) Last, we have a look at the statistical significance of the coefficient estimates that result from fitting each of the models using least squares. The results here align with our previous conclusions, as the quadratic term has the lowest p-value.

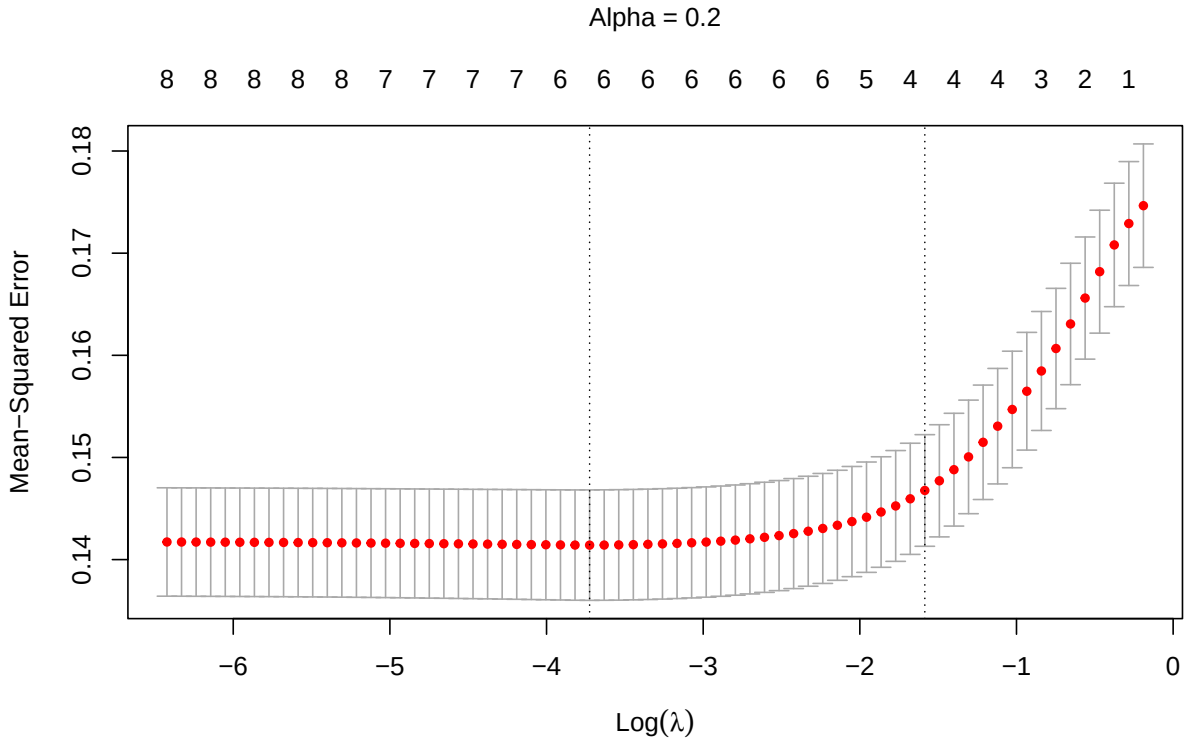
```
## [[1]]
##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -1.625427  0.2619366 -6.205420 1.309300e-08
## x            0.692497  0.2909418  2.380191 1.923846e-02
##
## [[2]]
##                 Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  0.05671501  0.1176555   0.482043 6.308613e-01
## x            1.01716087  0.1079827   9.419666 2.403287e-15
## I(x^2)      -2.11892120  0.0847657 -24.997388 4.584330e-44
##
## [[3]]
##                 Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  0.06150718 0.11950374   0.5146883 6.079538e-01
## x            0.97528027 0.18728149   5.2075636 1.089350e-06
## I(x^2)      -2.12379099 0.08700251 -24.4106856 5.873444e-43
## I(x^3)       0.01763858 0.06429037   0.2743580 7.843990e-01
##
## [[4]]
##                  Estimate Std. Error     t value     Pr(>|t|)
## (Intercept)  0.156702953 0.13946192   1.1236253 2.640034e-01
## x            1.030825643 0.19133655   5.3874999 5.174326e-07
## I(x^2)      -2.409898183 0.23485506 -10.2612148 4.575229e-17
## I(x^3)      -0.009132904 0.06722881  -0.1358481 8.922288e-01
## I(x^4)       0.069785421 0.05324006   1.3107691 1.930956e-01
```

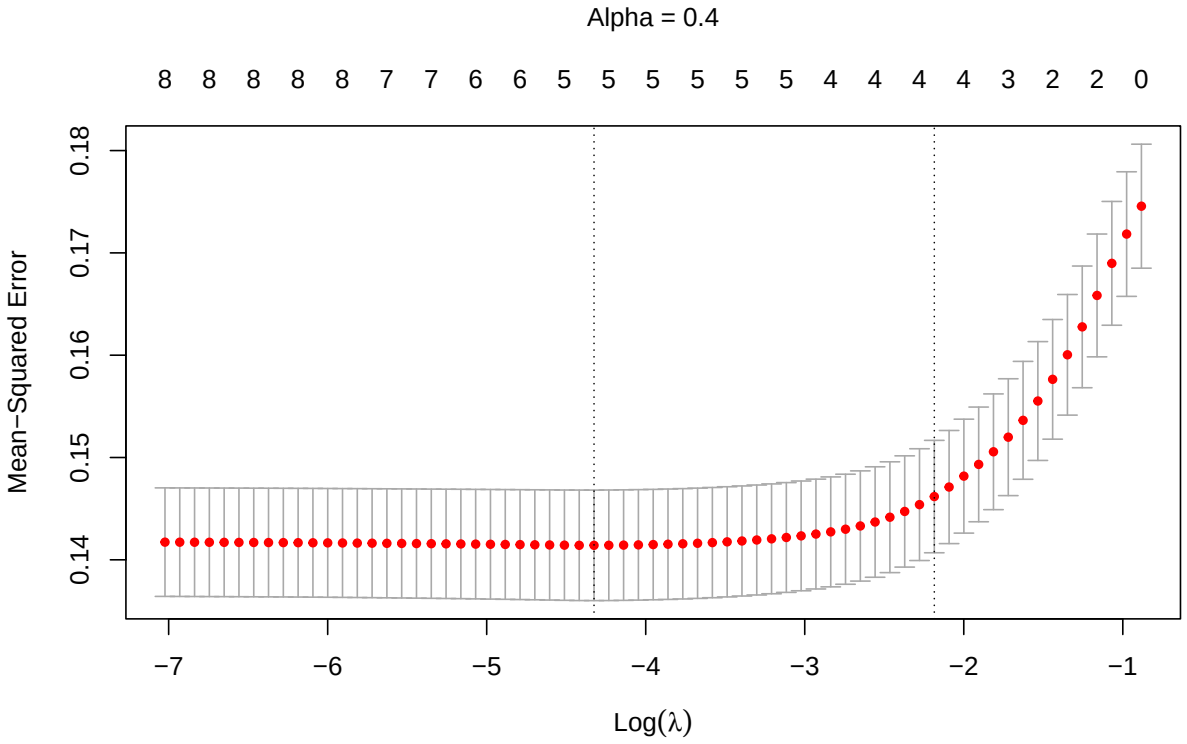
Exercise 3

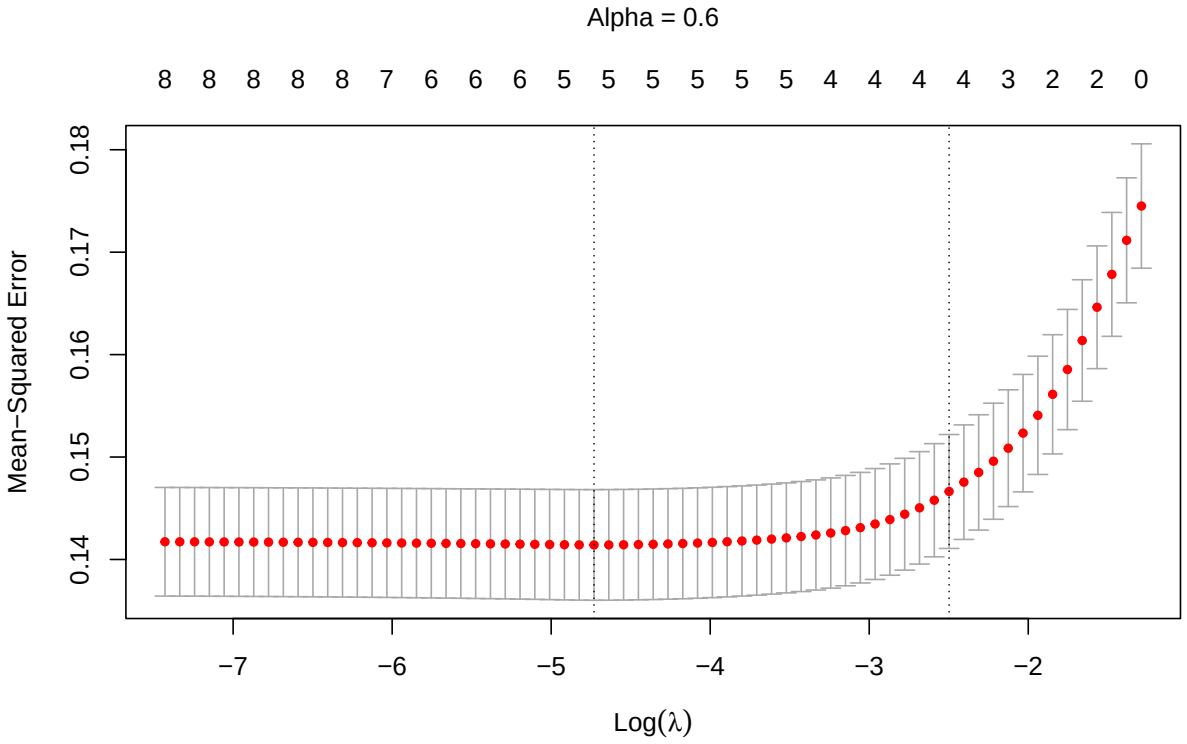
Here we are using the wage data set available as data object *Schooling* in package **Ecda**. First, we omit observations with missing values and the variable wage76, and mutate variable mar76 into a binary variable. Next, we fit regularized linear regression models with lwage76 as dependent variable using only linear effects for the covariates and varying the α parameter for the elastic from 0 to 1 in step sizes of 0.2. We do this using using cv.glmnet with 10-fold cross-validation considering the MSE for a range of penalty values of λ. The argument foldid ensures that the same data partitions are used for each model fitting, making the comparisons fair and consistent. The default plot method for cv.glmnet objects is used to visualize the cross-validation curves, which show the mean squared error (MSE) across a range of λ values for each α. The plot displays the λ values on the log scale along the bottom x-axis and the corresponding MSE on the y-axis. The vertical dotted line in each plot indicates the λ value that gives the minimum MSE. By varying α, we essentially move between different regularization methods: $\alpha = 0$ corresponds to ridge regression (penalty on the square of coefficients),$\alpha = 1$ corresponds to lasso regression (penalty on the absolute value of coefficients). The top x-axis shows these numbers, which correspond to the model's complexity at different levels of regularization. As lambda increases, the regularization penalty becomes more severe, leading to more coefficients being shrunk to zero.The dual x-axes in the plot serve to simultaneously convey how lambda affects both the model's predictive accuracy (through the MSE shown on the y-axis) and its complexity (in terms of the number of predictors used, shown on the bottom x-axis). Looking at the different plots given each value of α we observe with α closer to 1 more pronounced changes in the number of non-zero coefficients as lambda

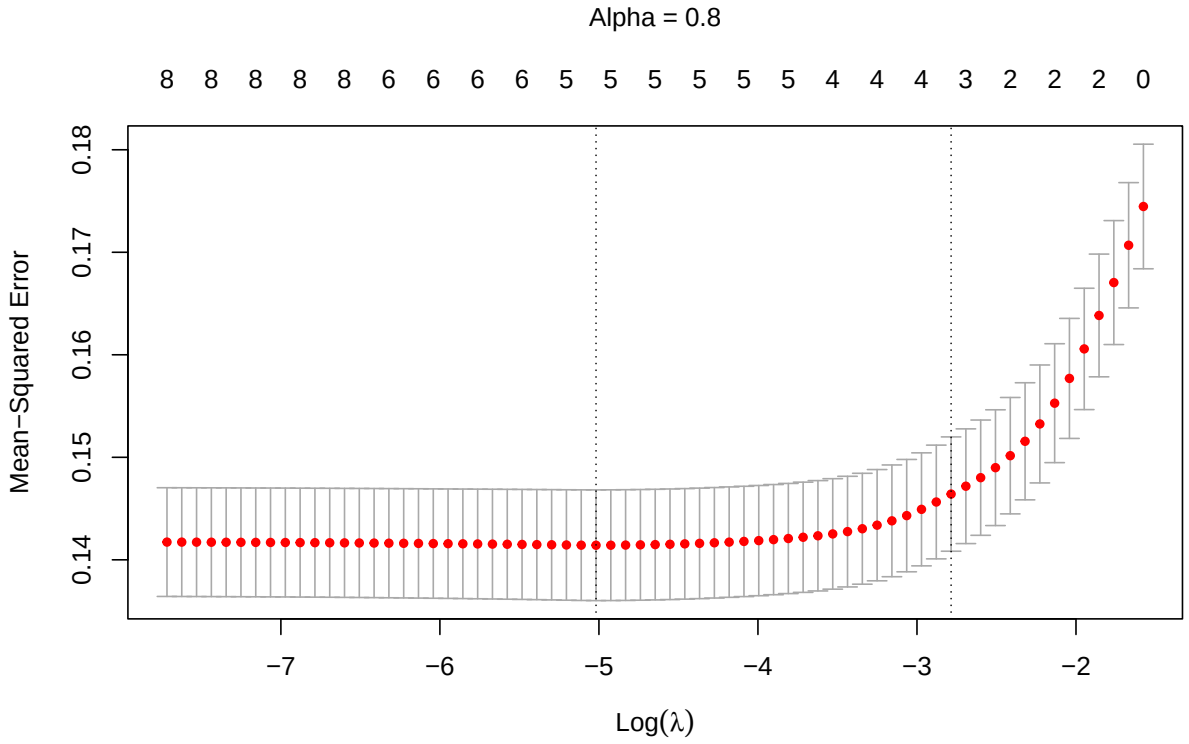
changes, reflecting the lasso's variable selection property. For ridge regression ($\alpha = 0$), the coefficients are shrunk towards zero but not exactly to zero, so the model complexity in terms of the non-zero coefficient count may not reduce as dramatically as with lasso or elastic net.

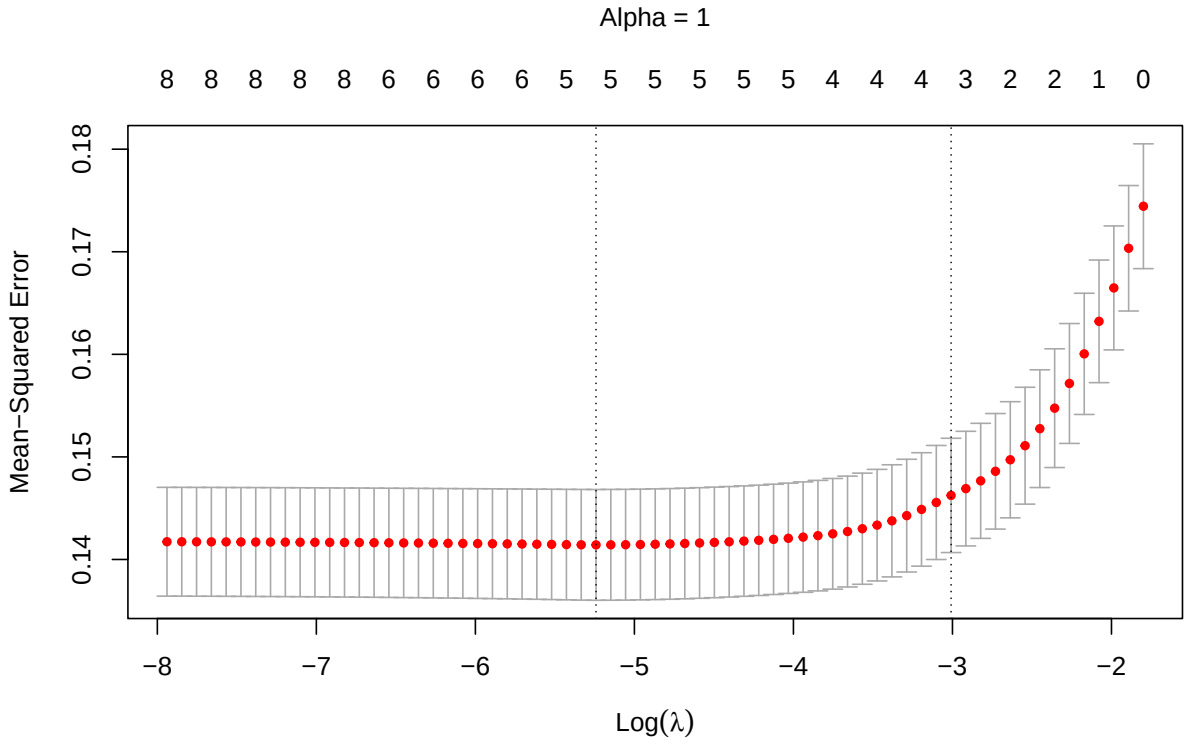





Alpha = 0.4
8 8 8 8 8 7 7 6 6 5 5 5 5 5 5 4 4 4 4 3 2 2 0
Mean−Squared Error
0.14
0.15
0.16
0.17
0.18
−7
−6
−5
−4
−3
−2
−1
Log(λ)


Alpha = 0.6
8 8 8 8 8 7 6 6 6 5 5 5 5 5 5 4 4 4 4 3 2 2 0
Mean−Squared Error
0.14
0.15
0.16
0.17
0.18
−7
−6
−5
−4
−3
−2
Log(λ)





This visualization helps in choosing an optimal lambda value , typically via the 1-standard error rule or by selecting the lambda that minimizes the cross-validation error.

Next, to select the best value for λ, that balances model simplicity and accuracy, for each fixed value of α we look at either the value that minimizes the cross-validation error (lambda.min) or the 1-SE rule (lambda.1se),

and then compare the selected models. We extract these λ values from cv.glmnet and then fit the final models on the full data set using these selected λ value. To compare the selected models based on their complexity (number of non-zero coefficients), predicted values, and MSE, we can use the coef function to inspect the coefficients, make predictions with the predict function, and then calculate MSE for each model. Table 2 summarizes all that. The choice between lambda.min and lambda.1se involves a trade-off between model simplicity and predictive accuracy. lambda.1se typically leads to simpler models (with potentially slightly higher MSE). We observe that as α increases from 0.0 to 1.0, the number of non-zero coefficients varies. The model with $\alpha = 0.0$ (ridge regression) maintains 10 non-zero coefficients for both lambda.min and lambda.1se. For other values of α (moving towards lasso regression), the number of non-zero coefficients tends to decrease, indicating sparser models. This is especially visible for $\alpha = 1.0$ with only 5 or 6 non-zero coefficients, suggesting that lasso regularization enforces the most sparsity. The least complex model is the one with $\alpha = 0.2$ using the lambda.1se criterion, having only 5 non-zero coefficients, whereas the most complex models with respect to the number of coefficients are all those with $\alpha = 0.0$. Regarding predicited values, the cv_mse does not vary significantly across different α values, indicating that the change in α is not drastically affecting the model's predictive ability in this case. The lowest cv_mse for lambda.min appears at $\alpha = 1.0$, suggesting that the lasso model at its optimal lambda achieves a marginally better fit in terms of MSE. For the lambda.1se criterion, the cv_mse is slightly higher compared to the lambda.min, which is expected as the 1-SE rule tends to select a simpler and more generalizable model at the expense of a slight increase in error. In the end, the choice between lambda.min and lambda.1se would depend on whether the priority is on the lowest possible MSE or on model simplicity.

alpha	lambda	criterion	non_zero_coefficients	cv_mse
0.0	0.0505227	min	10	0.1416001
0.0	0.7502457	1se	10	0.1468714
0.2	0.0241132	min	7	0.1414149
0.2	0.2049025	1se	5	0.1467689
0.4	0.0132321	min	6	0.1414132
0.4	0.1124401	1se	5	0.1461814
0.6	0.0088214	min	6	0.1414171
0.6	0.0822686	1se	5	0.1466378
0.8	0.0066160	min	6	0.1414202
0.8	0.0617014	1se	5	0.1464145
1.0	0.0052928	min	6	0.1414225
1.0	0.0493612	1se	5	0.1462448

Last, we inspect the best solution for $\alpha = 1$ (lasso regression) using the 1-SE rule. This approach tends to prefer simpler models with fewer non-zero coefficients, which can be advantageous for interpretability and generalization. To see which variables were selected (i.e., have non-zero coefficients) and their estimated coefficients, we use the coef function from the glmnet package. We observe here only 4 variables are selected, and all have rather small coefficients ans thus little influence on the response variable 'lwage76'. To assess the goodness-of-fit, we then calculate the correlation between the predicted values from the model and the observed values. Values closer to 1 or -1 indicating a stronger linear relationship between predictions and actual outcomes. In this case the correlation lies at 0.4293. This indicates a moderate positive linear relationship between the predicted and observed values.The model has some predictive power, but it is not capturing all of the variability in the dependent variable.

```
## [1] "Non-zero coefficients (including intercept):"
```

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 5.542382e+00
## smsa66      .
## smsa76      .
## nearc2      .
```

```
## nearc4      .
## nearc4a     .
## nearc4b     .
## ed76        .
## ed66        4.554928e-02
## age76       1.443958e-03
## daded       .
## nodaded     .
## momed       .
## nomomed     .
## momdad14    .
## sinmom14    .
## step14      .
## south66     .
## south76     .
## famed       .
## black       .
## enroll76    .
## kww         6.283247e-03
## iqscore     3.616473e-05
## mar76       .
## libcrd14    .
## exp76       .
```

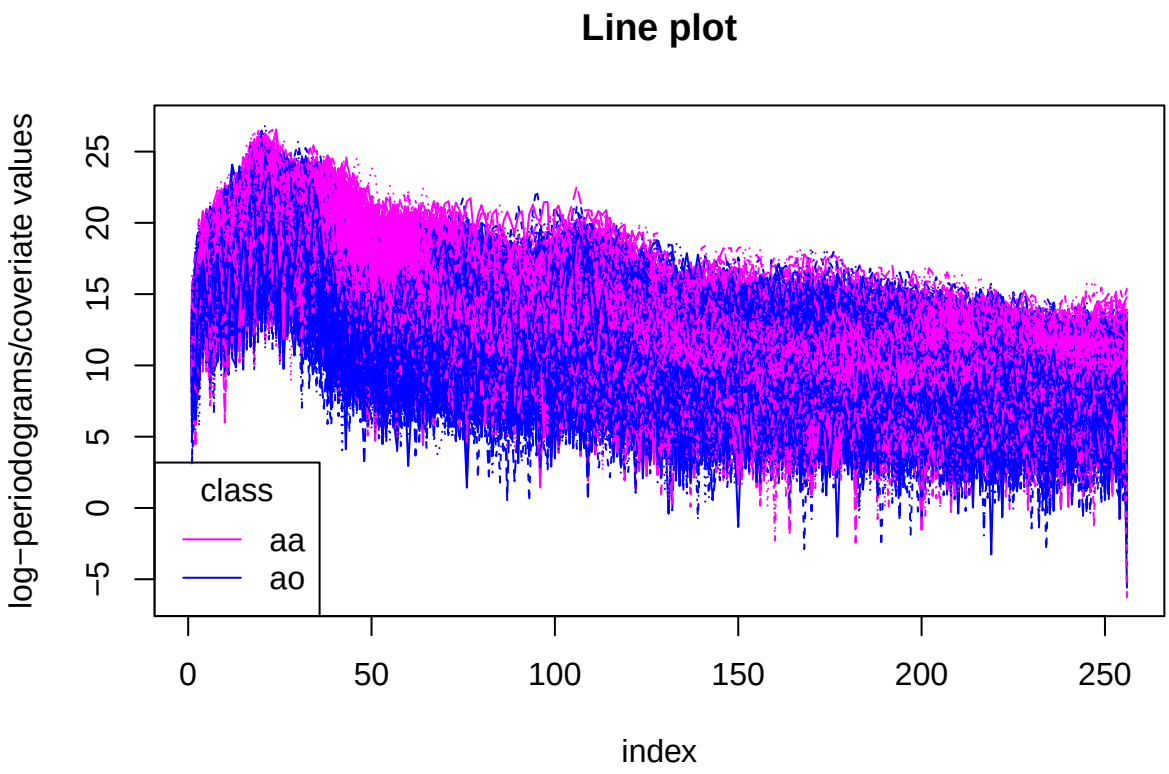
```
## [1] "Correlation between predicted and observed values for alpha = 1 (1-SE rule): 0.4293"
```

Exercise 4

Exercise 5

We load the acoustic-phonetic continuous speech corpus dataset *phoneme* from the package *ElemStatLearn*. There are five classes contained in the dataset. The covariates are log-periodograms of length 256. In the following two-group classification is performed using only the classes "aa" and "ao". Therefore, we subset the data accordingly.

Afterwards, we visualize the data by plotting the covariate values on the y-axis and the index on the x-axis using line plots. "aa" is plotted in *magenta* and "ao" in *blue*. "aa" seems to have overall slightly higher values than "ao".



We select 1000 samples as training data and use the remaining ones as test data.

We then fit a logistic regression model to the training data using all covariates and determine the misclassification rate and the average log-likelihood value on the training and test data.

```
##      4031       614       746      3934      1411       389
## 0.9991013 1.0000000 0.9999994 0.3806535 0.9996663 0.9999328
```

```
##            9           17           20           24           28           32
## 2.950911e-02 7.930664e-01 9.999960e-01 1.851190e-06 3.470267e-02 9.562946e-03
```

```
##
##   0   1
## 416 584
```

```
##
##   0   1
## 313 404
```

```
## 'log Lik.' -209.0755 (df=257)
```

```
## [1] -209.0755
```

```
## [1] -773.5824
```

```
## [1] 0.081
```

```
## [1] 0.2301255
```

The complexity of this model can be reduced by restricting the regression coefficients to vary only smoothly over the covariates, i.e., regression coefficients for close covariates are similar. To achieve this, we use splines. We create a 12-dimensional model matrix X^* based on natural cubic splines which we use to fit the logistic

regression model instead of the 256-dimensional X. Specifically, we fit a logistic regression model to the training data using X^* as model matrix and determine the misclassification rate and the average log-likelihood value on the training and test data. The likelihood function of the (multivariate) Bernoulli distribution is given by

$$\prod_{i=1}^{n} p^{x_i}(1-p)^{1-x_i}$$

Taking the logarithm gives us the log-lieklihood.

$$\sum_{i=1}^{n} x_i \log(p) + (1-x_i)\log(1-p)$$

```
##      4031       614       746      3934      1411       389
## 0.9768718 0.9712144 0.9760902 0.8522399 0.5827366 0.9200335
```

```
##           9          17          20          24          28          32
## 0.69368774 0.82430451 0.94831697 0.08391863 0.85637294 0.15925690
```

```
##
##   0   1
## 415 585
```

```
##
##   0   1
## 270 447
```

```
## 'log Lik.' -404.6102 (df=13)
```

```
## [1] -404.6102
```

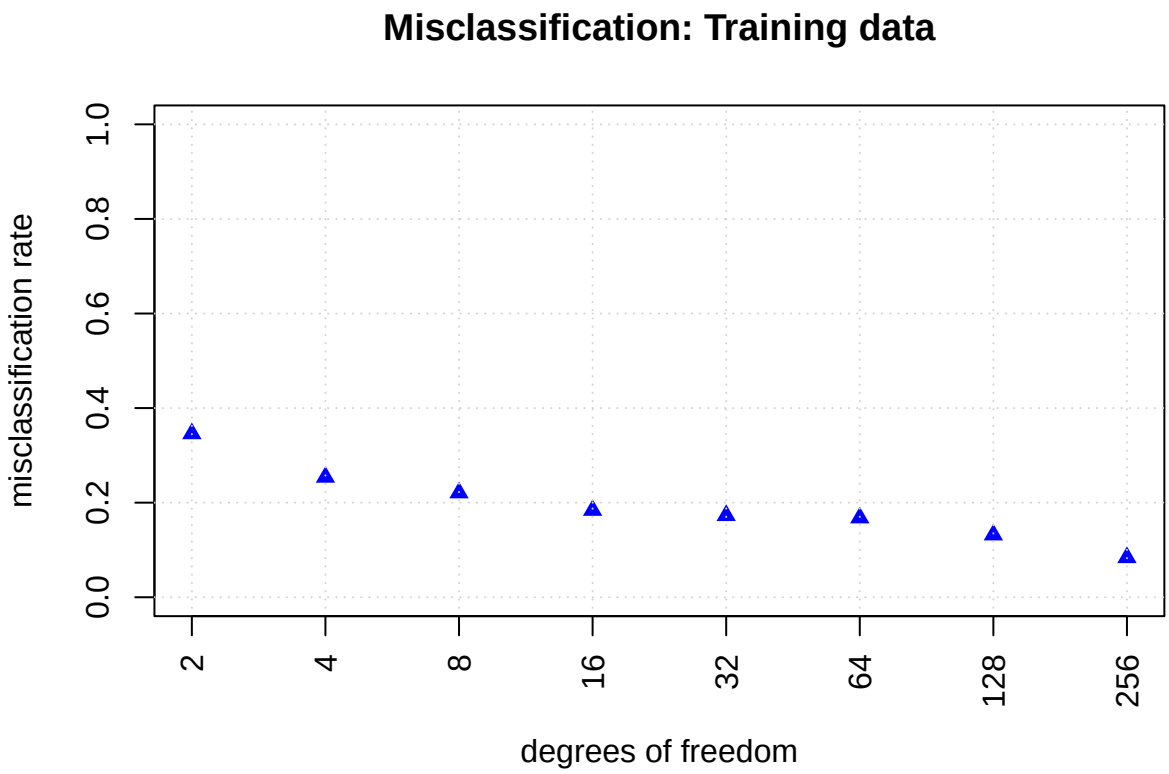
```
## [1] -272.9746
```

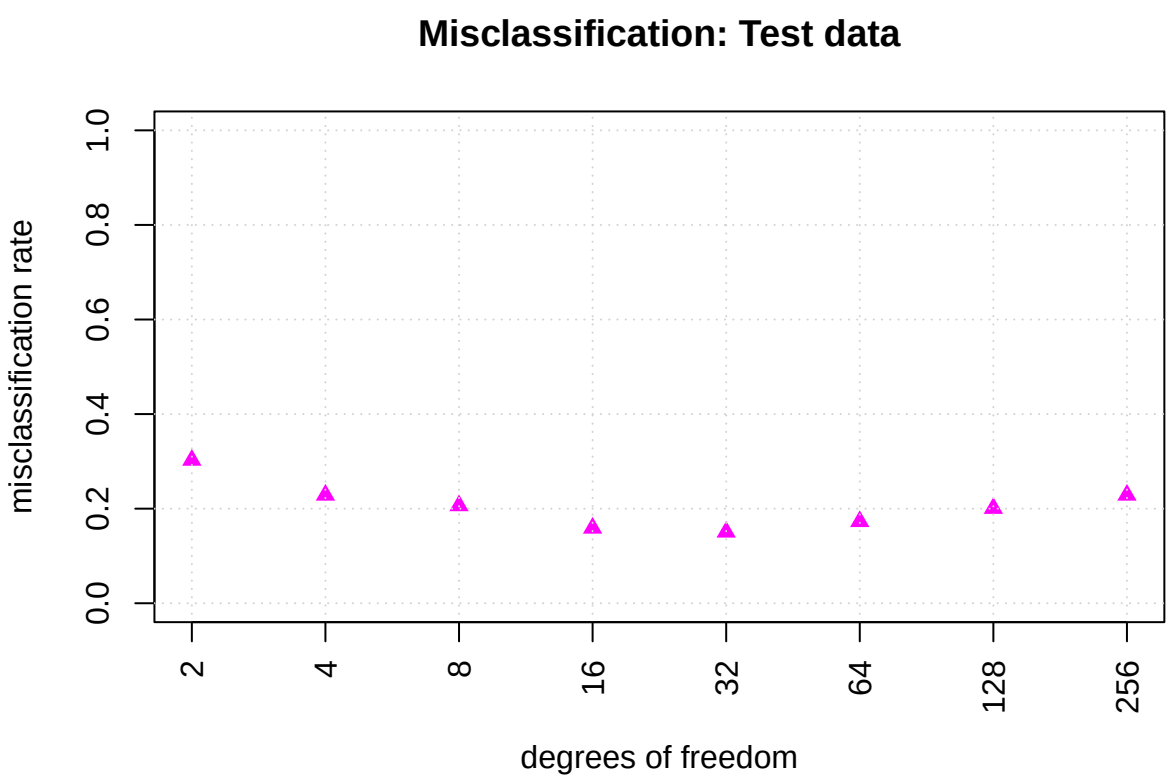
```
## [1] 0.186
```

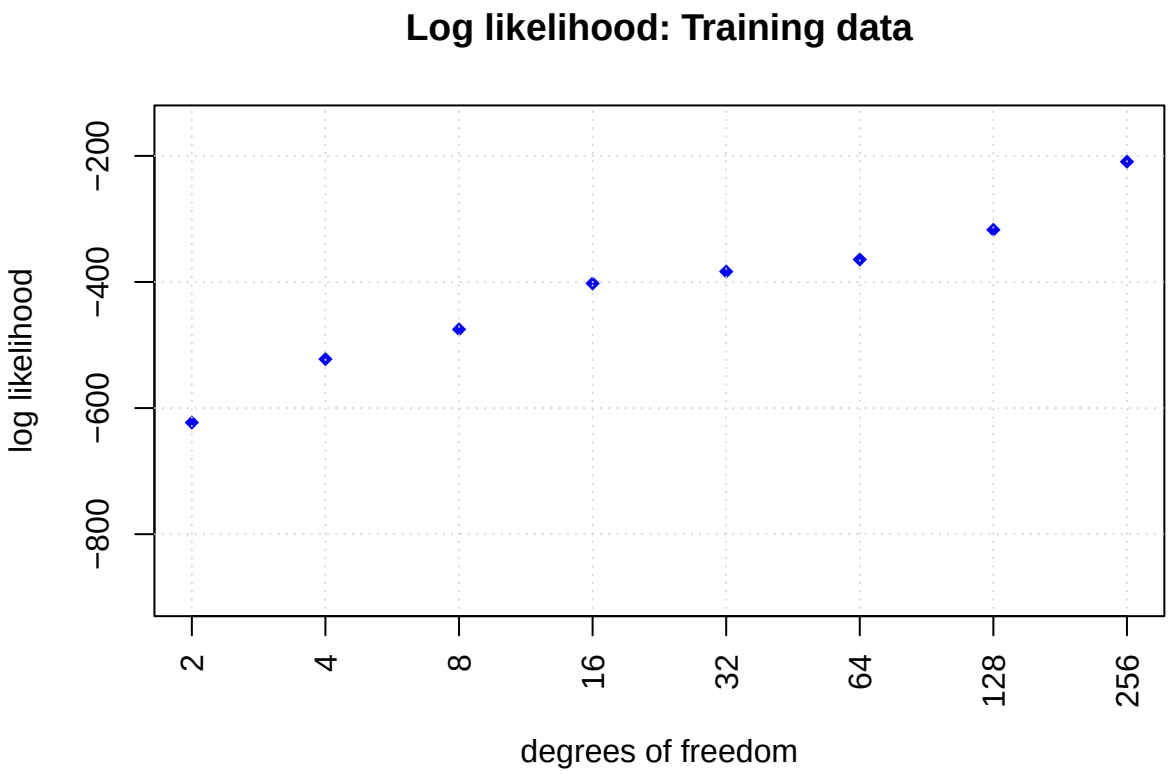
```
## [1] 0.1617852
```

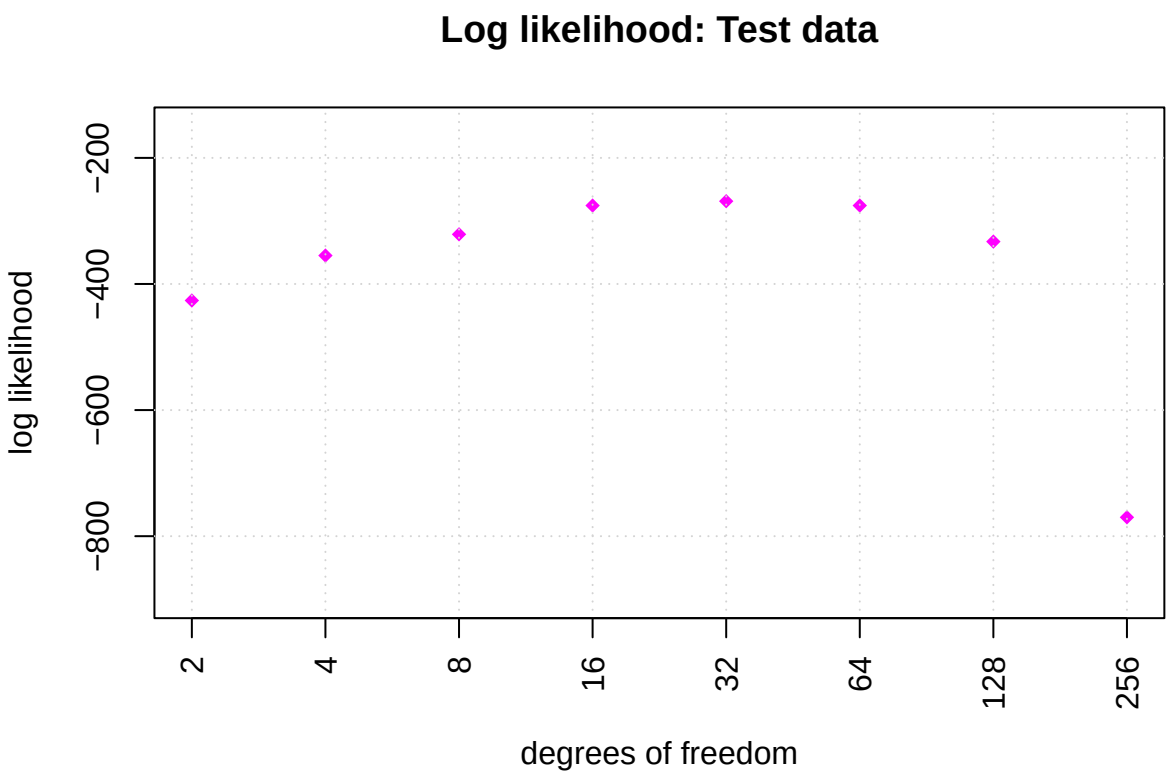
We vary the degrees of freedom in the spline basis expansion using 2 to the power of 1 to 8, i.e., 2,4,8,16,32,64,128 and 256. Again, we calculate the misclassifiation rate and the mean log-likelihood on the training and test data for each of the fitted models.

Now we are able to compare the misclassification rates and mean log-likelihoods based on training and test data sets visually for the different degrees of freedom.


Misclassification: Training data
misclassification rate
0.0
0.2
0.4
0.6
0.8
1.0
2
4
8
16
32
64
128
256
degrees of freedom


Misclassification: Test data
misclassification rate
0.0
0.2
0.4
0.6
0.8
1.0
2
4
8
16
32
64
128
256
degrees of freedom


Log likelihood: Training data
log likelihood
−800
−600
−400
−200
2
4
8
16
32
64
128
256
degrees of freedom



The more we restrict the regression coefficients to vary only smoothly over the covariates (lower degrees of freedom), the higher is the misclassification rate for the training data. Theoretically, higher smooth avoids overfitting to the training data. But it can also loose important information. By looking at the misclassification rate of the test data, we can see that medium levels of degrees of freedom (16-64) result in the lowest misclassification rate. Low smoot, or in other words high degrees of freedom, show the expected overfitting. Too much sommot, low degrees of freedom, seems to loose important information and hence also increases the misclassification rate on the test data.

On the training data, the log-likelihood worsens with the smooth of the covariates. Looking at the log-likelihood of the test data we see that medium levels of degrees of freedom (16-64) are the best choice. They avoid overfitting and don't loose too much information. Note the very low log-likelihood due to overiftting for 256 degrees of freedom. On the other end, with high smooth the loss isn't as high.