Tentative solutions to the TMA4268 Statistical learning, May 2019 exam

Mette Langaas

June 11, 2019

Warming up

Problem 1: Warming up with an overview

[Maximal score: 3 points]

i) In this course we have considered two types of learning, A:*supervised* and B:*unsupervised*. In A both covariates and a known response is present, but in B only *covariates* are present. In A we have spent time studying two problem types: C *regression* and *classification*. Linear methods have been an important starting point for both problem types.

ii) In C we started with studying the relationship between one univariate response and one covariate in *simple linear regression*, and then we added more covariates, and got *multiple linear regression*. The least squares estimator was $\boldsymbol{\beta}$, which was *unbiased* and covariance matrix that was *dependent on the covariates* in the multiple linear regression model. We looked at including non-linear effects in the covariates by adding *polynomials*. Non-linear effects were also the topic of *K-nearest neighbour regression*. Interactions (the fact that the effect of one covariate on the response is dependent on the value of another covariate) were easily included using *regression trees*. Model selection was performed using *the AIC penalty* and *lasso regression*. More about regression in the last part of this exam.

iii) The starting point for classification was the sampling and the *diagnostic* paradigm. From the sampling paradigm we studied the *LDA and QDA*, and from the other paradigm many more methods, which we will work with soon.

iv) In *unsupervised learning* the goal is to discover interesting aspects about the measurements $x_1, x_2, \ldots, x_p$ when the y is not present. We have considered two types of methods: *principal component analysis* and *clustering*. Both methods have strong focus on finding *groups* in the data, and on *visualization*. The top figure to the right is an example of *hierarchical clustering*. The bottom figure to the right gives a biplot from *principal component analysis*.

Problem 2: Matching optimization criterion and method

[Maximal score: 2 points]

For each optimization criterion choose the correct method from the drop down menu:

a) Maximize M by choosing $\beta_0, \beta_1, \ldots, \beta_p$ subject to $\sum_{j=1}^{p} \beta_j^2 = 1$ and $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M$ for $i = 1, \ldots, n$: *maximal margin classifier*

b) $\text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2$: *least squares regression*

c) $\text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$: *ridge regression*

d) $\text{argmin}_{R_1(j,s), R_2(j,s)} \left[\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_1})^2 \right]$: *regression trees*

e) $\text{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$: *lasso regression*

f) Maximize M by choosing $\beta_0, \beta_1, \ldots, \beta_p, \varepsilon_1, \ldots, \varepsilon_n$ subject to $\sum_{j=1}^{p} \beta_j^2 = 1$ and $y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i)$ when $\varepsilon_i \geq 0$ for all $i = 1, \ldots, n$ and $\sum_{i=1}^{n} \varepsilon_i \leq C$: *support vector classifier.*

Basics of neural networks

Problem 3: Neural network — mathematical formula

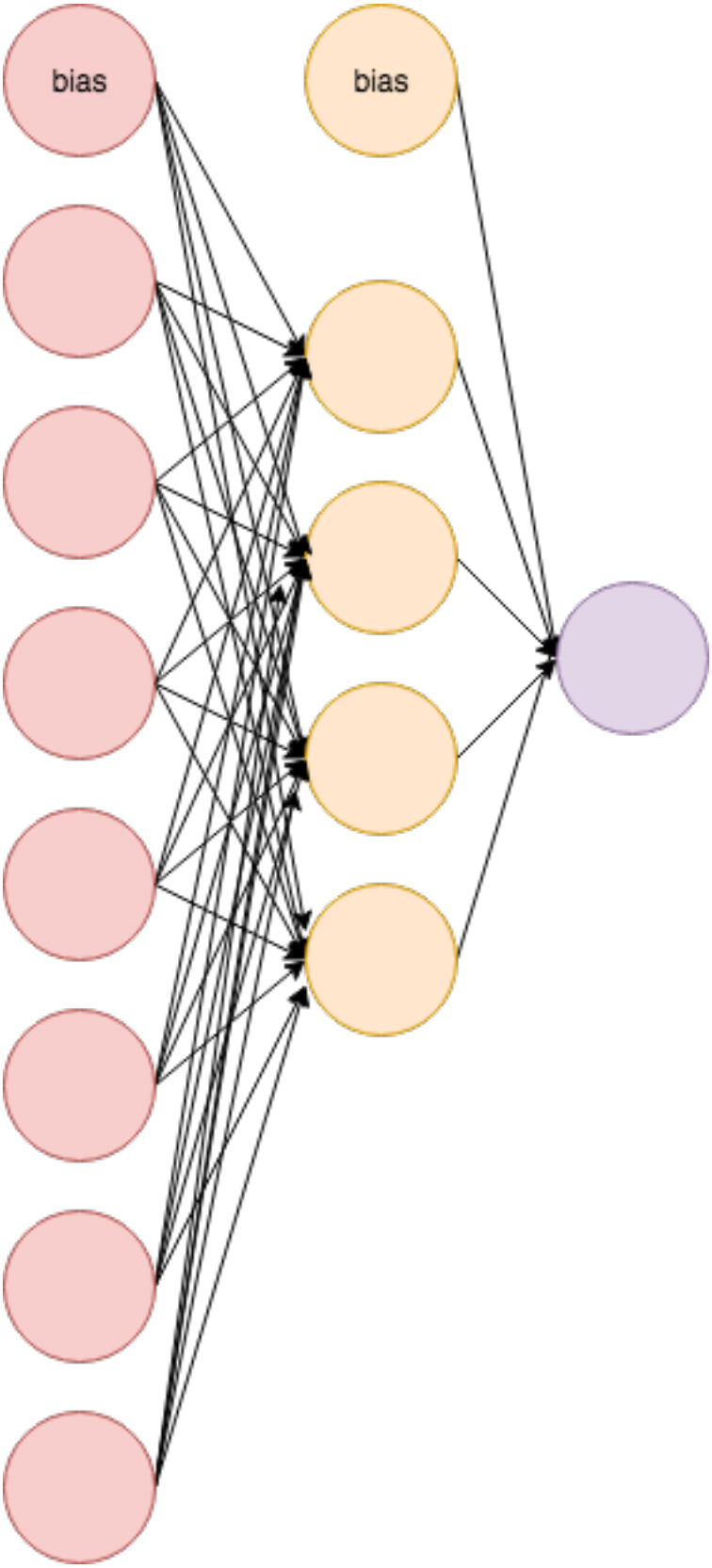
[Maximal score: 4 points]

a) Write down the mathematical formula for a feedforward network with one input layer, one hidden layer and one output layer, with the following architecture

- input layer: seven input nodes and one bias node for the hidden layer,
- hidden layer: four nodes, and a bias node for the output layer,
- output layer: one node.

Let the nodes in the hidden layer have sigmoid activiation function and the node in the output layer have linear activiation function.

A drawing of the network is provided.



b) Which type of problem can this network solve?

c) What would be an appropriate loss function (give mathematical formula)?

A:

$$\hat{y}_1(\mathbf{x}) = \beta_{01} + \sum_{m=1}^{4} \beta_{m1}(1 + \exp(-\alpha_{0m} - \sum_{j=1}^{7} \alpha_{jm} x_j))^{-1}$$

This network is appropriate for a regression problem with 7 covariates and one continuous response. There would be a need for a non-linear relationship between the covariates and the response due to the use of the non-linear sigmoid activiation function in the hidden layer. The universal approximation theorem says that - given enough nodes in the hidden layer and a squashing activation function- this network can approximate any (Borel measurable) function from one finite-dimensional space (our input layer) to another (our output layer) with any desired non-zero amount of error.

The use of a linear activation function to the output node suggests regression (and not classification), and for that a quadratic (mean squared error) loss function would be appropriate.

$$J(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_1(\mathbf{x}_i))^2$$

$\hat{y}_1(\mathbf{x}_i)$ is the output from the linear output node, and y_i is the response, and n is the number of training data.

Comments and common mistakes: Surprisingly many thought that having a sigmoid activation in the hidden layer and a linear activation in the output layer could be a suitable network for classification. Therefore, make sure you understand that all layers may have different activation functions (not just the last layer).

Problem 4: Neural network — understanding and estimation

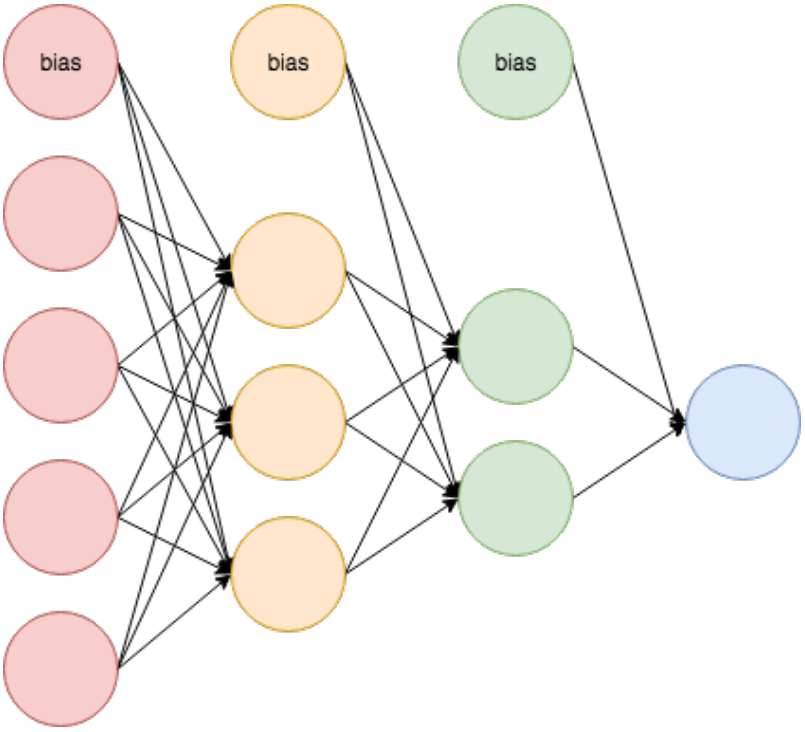
[Maximal score: 6 points]

a) Which network architecture and activation functions does this formula give? Include a drawing of the network architecture.

$$\hat{y}_1(\mathbf{x}) = [1 + \exp(-\beta_{01} - \sum_{m=1}^{2} \beta_{m1} \max(\gamma_{0m} + \sum_{l=1}^{3} \gamma_{lm} \max(\alpha_{0l} + \sum_{j=1}^{4} \alpha_{jl} x_j, 0), 0))]^{-1}$$

A:

- four input nodes (red), a bias term for the first hidden layer
- first hidden layer with 3 nodes (orange), ReLU activiation function, a bias term for the second hidden layer
- second hidden layer with 2 nodes (green) and ReLU activiation function, and bias node for the output node,
- one output node (blue) with sigmoid activiation function.



b) How many parameters are estimated in this network?

A: $(4+1)*3+(3+1)*2+(2+1)*1=26$

c) Which type of problem can this network be used to solve, and what would you use as loss function?

A: Since we have one node in the output layer we can't do classification with more than two classes. With sigmoid activation function in the output layer we would produce a number between 0 and 1, that we could think of this as modelling the probability of one of two classes and that classification (into two classes) is the task. If this is the case, then binary cross-entropy loss function is suitable.

However, we might also use this network for regression, if our response is restricted to the 0-1 interval (e.g. some kind of score), and then a quadratic loss function (mean squared error) is a natural choice.

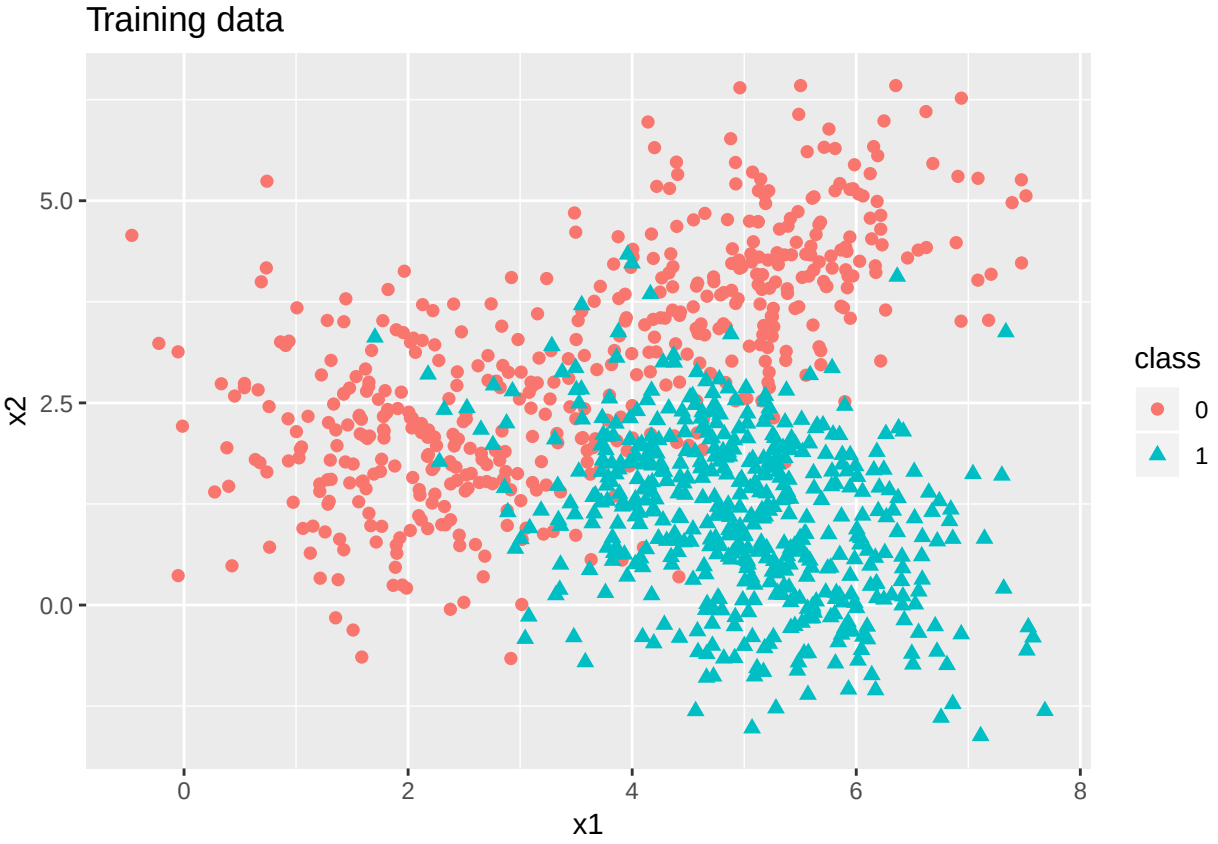
d) Discuss briefly how parameters are estimated in a feedforward neural network.

A:

- Minimize loss function based on gradient descent.
- Non-convex problem (when activation functions non-linear)
- Learning rate set by user (but modification below)
- Chain rule performed using backpropagation.
- Minibatch stochastic gradient descent most popular, where the training set is split randomly into batches of a given size, and fed to the optimzer. Gradient step performed after each minibatch.
- Learning rate modification: RMSprop is one modification where individually adapt the learning rates of all model parameters by scaling them with exponentially weighted moving average of historical values.
- Regularization can be added of L2 or L1 type, where L2 is called weight decay.
- If network is chosen too big, early stopping can be seen as a type of regularization.
- Drop-out layer also possible, supposed to act as a regularization.

Classification

We will study a simulated data set with two classes (labelled 0 and 1) and two numerical covariates x_1 and x_2. Let $\mathbf{x} = (x_1, x_2)$ be a column vector with the two covariates. A training set with 500 observations of each class is available, and a scatter plot is given below.



Problem 5: Classification with K-nearest neighbour

[Maximal score: 3 points]

a) Write down the mathematical formula for the probability of class 1 for the K-nearest neighbour classifier at a covariate value $\mathbf{x}_0$, and explain your notation.

A Given an integer K and a test observation (covariate value) $\mathbf{x}_0$, the KNN classification first identifies the K points in the training data that are closest (Euclidean distance) to $\mathbf{x}_0$, represented by $\mathcal{N}_0$. It then estimates probability of class 1 at $\mathbf{x}_0$ as the ratio of the number of class 1 observations in the training observations in $\mathcal{N}_0$ divided by K.

$$\hat{P}(Y = 1|\mathbf{X} = \mathbf{x}_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = 1)$$

Comment: we have not been very strict to say if this is P or $\hat{P}$, both is done on the module pages and in the textbook the P is used.

b) What does the rule of *majority vote* mean when applied to K-neareste neighbour classification? In our situation with two classes, what is the cut-off on the probability of class 1 equivalent of a majority vote?

A Majority vote means classifying a new observation with to the most occuring class among the K neighbours in the training data. With two classes this correponds to a cut-off on probability of class 1 at 0.5. We often use an odd number of neighbours so that we avoid having exacly 0.5 as probability.

c) List List one positive and one negative aspect of using K-nearest neighbour classification.

A

Positive: The possibility of non-linear class boundaries gives flexibility. Easy method to understand. Not training required, but the training set needs to be kept.

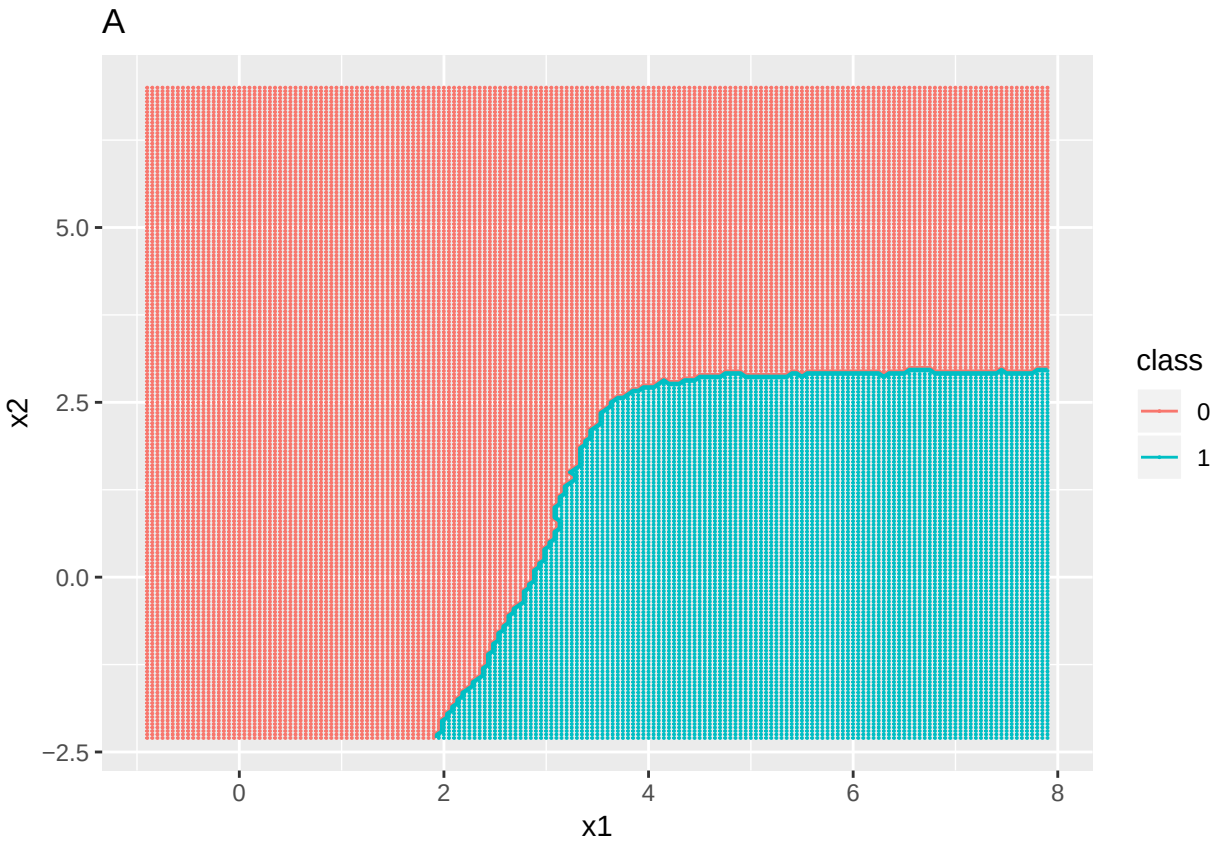
Negative: the number of neighbours is a hyperparameter and must be decided on. Curse of dimensionality: in high-dimensional space is any observation an outlier and the K nearest neighbours might be far away. Not so easy to interpret which features of the data the model has found since it is a local model.

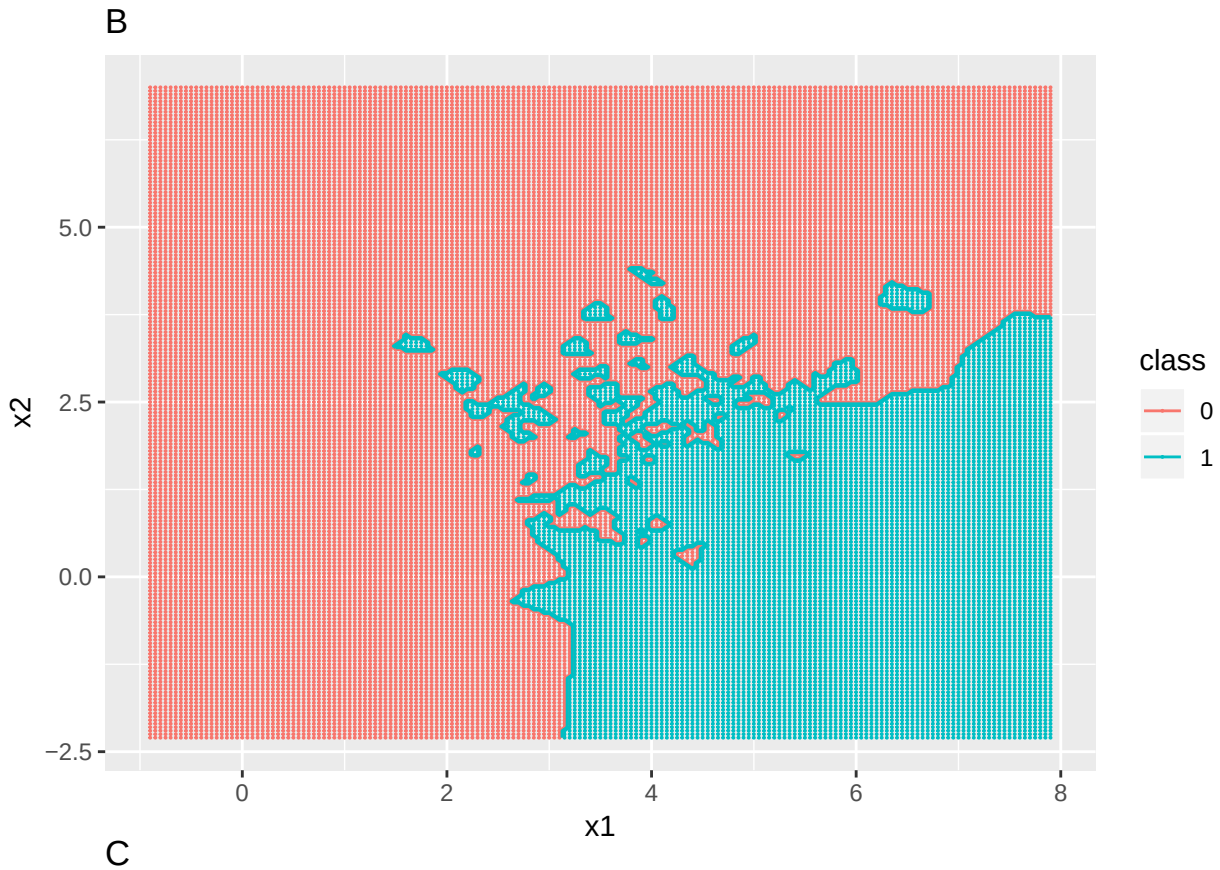
Problem 6: Class boundaries for KNN

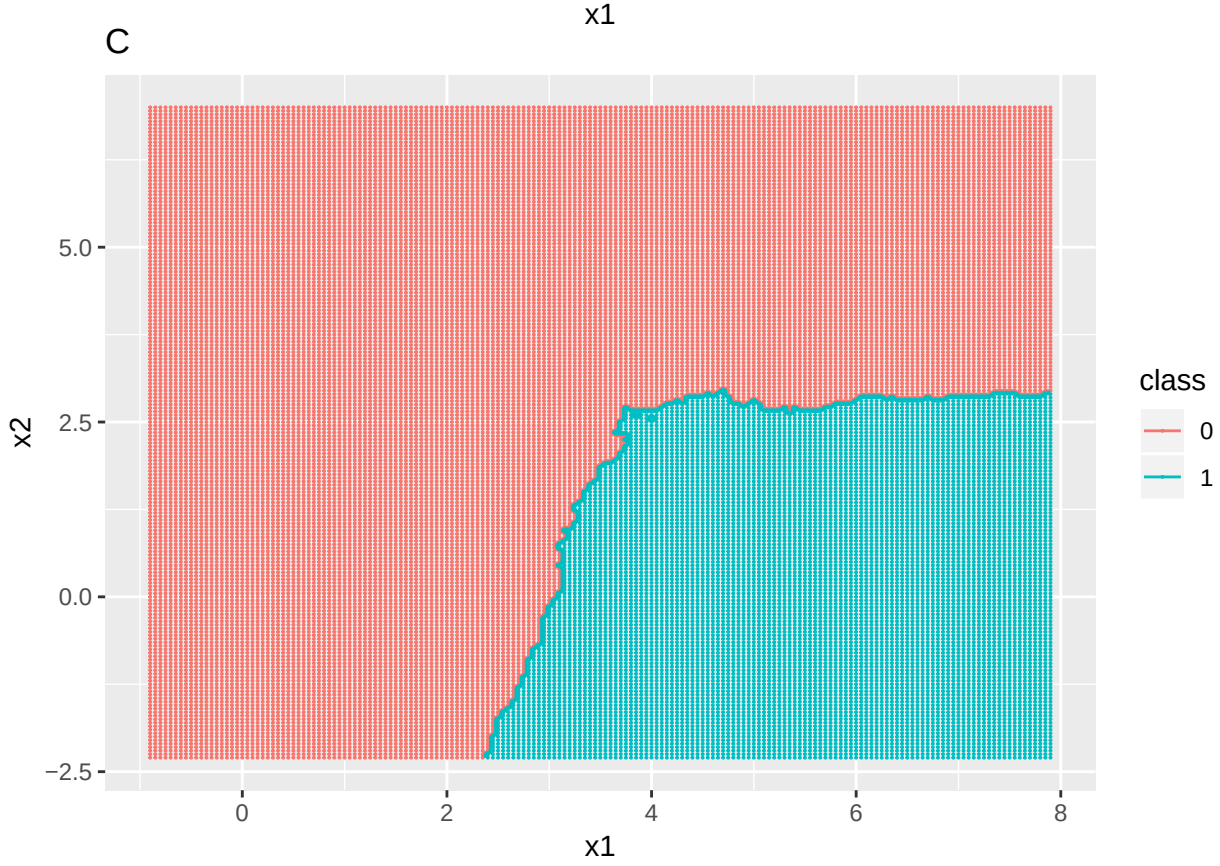
[Maximal score: 2 points]

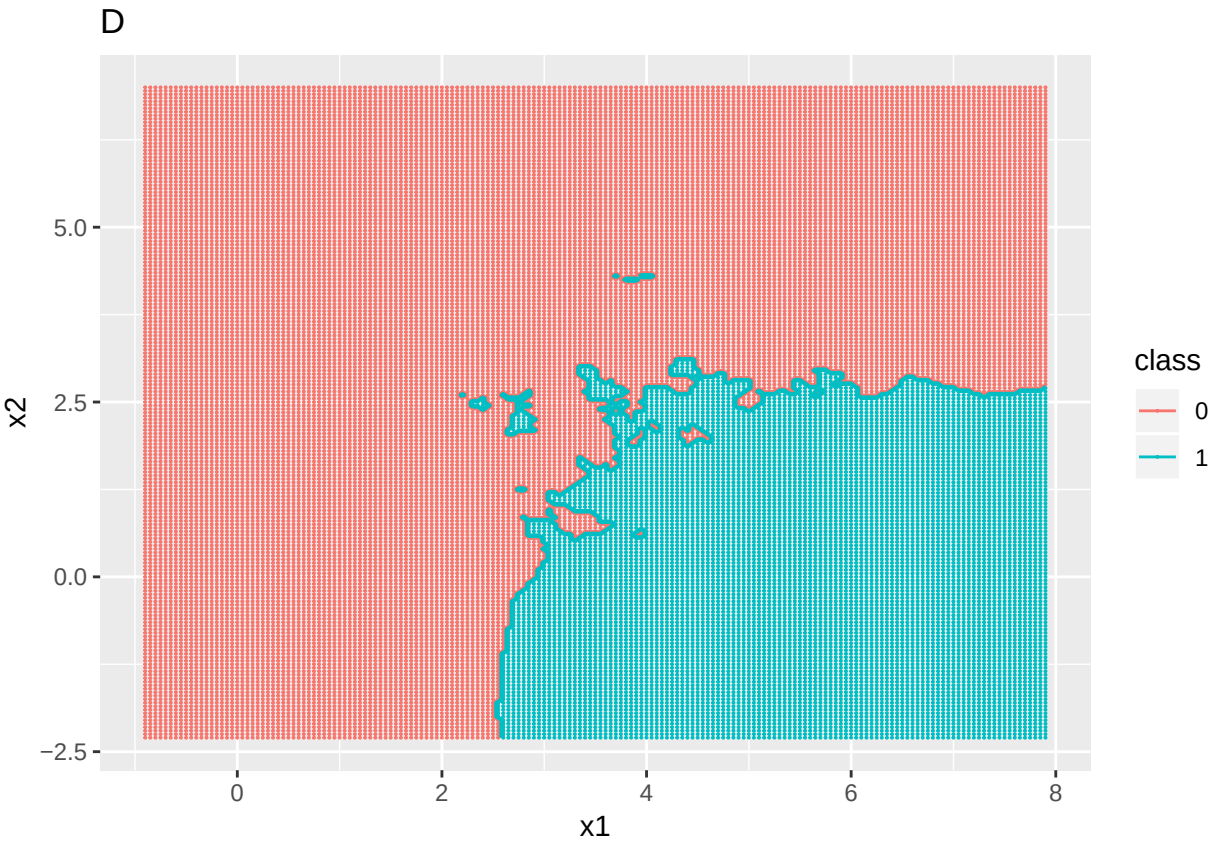
In the figure below we have used the K-nearest neighbour classification method, with K equal to 1, 3, 25 and 199, to get a classification boundary between class 0 and 1 based the training data.

Q Match panels A–D to values of K (1,3,25,199).




B
5.0
2.5
0.0
−2.5
x2
0
2
4
6
8
x1
class
0
1


C
5.0
2.5
0.0
−2.5
x2
0
2
4
6
8
x1
class
0
1



A:

- A: $K = 199$ since this is the smoothest boundary
- B: $K = 1$ since this is the most variable boundary
- C: $K = 25$ this is the second smoothest boundary
- D: $K = 3$ this is the second most variable boundary

Problem 7: Choosing K in K-nearest neighbour classification

[Maximal score: 5 points]

Crossvalidation can be used to choose a good value for K, and is performed the R-code and print-out.

a) Explain shortly what is done, and what the result is. Your explanation should include a drawing and the words: fold, error measure.

b) Which value of K (number of neighbours) would you choose? Elaborate

c) What is the one-standard-error-rule? Can you from the print-out use this rule to choose K? If not, what can be added or changed in the R-code to be able to use this rule?

```
# the training data contains 500 observations from each class
# train.x: a 1000 times 2 matrix with training covariates x1 and x2
# train.y: a 1000 vector with the class (0 or 1) as a factor for the
# training data

Kgrid = seq(1, 199, by = 2)
Kgrid
```

```
##   [1]   1   3   5   7   9  11  13  15  17  19  21  23  25  27  29  31  33
```

```
##  [18]  35  37  39  41  43  45  47  49  51  53  55  57  59  61  63  65  67
##  [35]  69  71  73  75  77  79  81  83  85  87  89  91  93  95  97  99 101
##  [52] 103 105 107 109 111 113 115 117 119 121 123 125 127 129 131 133 135
##  [69] 137 139 141 143 145 147 149 151 153 155 157 159 161 163 165 167 169
##  [86] 171 173 175 177 179 181 183 185 187 189 191 193 195 197 199
```

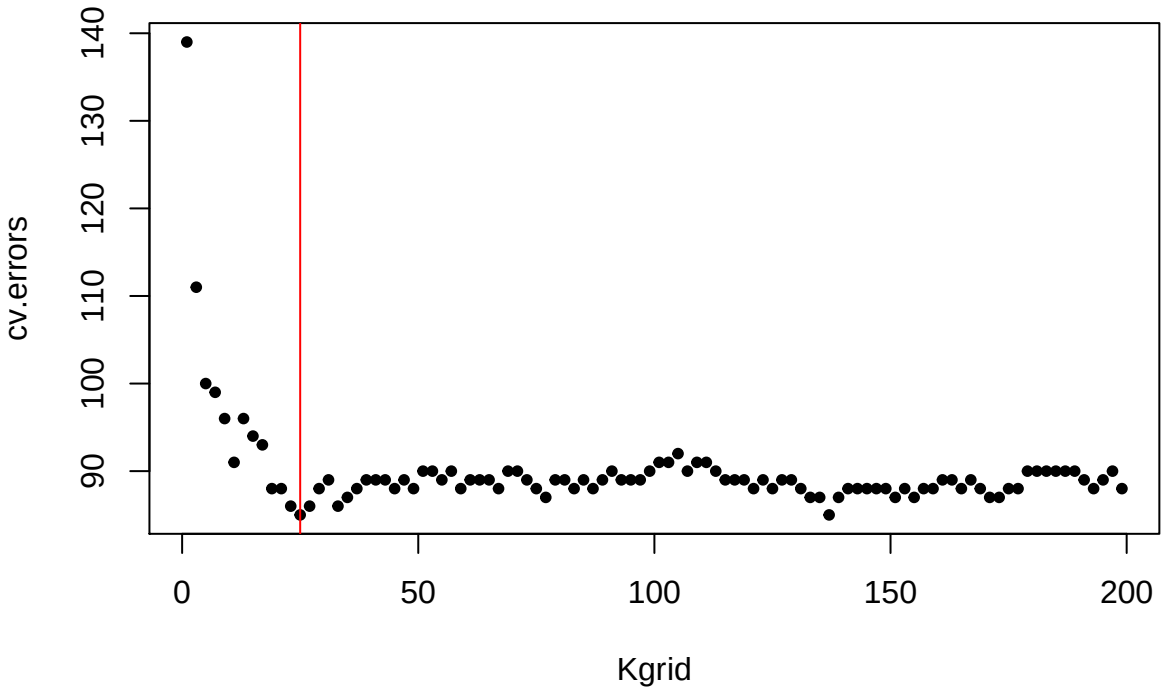
```
nmodels = length(Kgrid)
nfolds = 5

set.seed(4268)
folds <- createFolds(train.y, k = nfolds)
# the random sampling is done within the levels of y when y is a
# factor in an attempt to balance the class distributions within the
# splits.  folds[[1]] gives indecies for training observations in
# fold 1, folds[[2]] the same for fold 2 etc.

cv.errors = rep(0, nmodels)

for (i in 1:nfolds) {
    for (j in 1:nmodels) {
        pred = class::knn(train = train.x[-folds[[i]], ], test = train.x[folds[[i]],
            ], cl = train.y[-folds[[i]]], k = Kgrid[j])
        # gives the predicted class 0/1 for the validation fold
        cv.errors[j] = cv.errors[j] + sum(pred != train.y[folds[[i]]])
    }
}
cv.errors
```

```
##   [1] 139 111 100  99  96  91  96  94  93  88  88  86  85  86  88  89  86
##  [18]  87  88  89  89  89  88  89  88  90  90  89  90  88  89  89  89  88
##  [35]  90  90  89  88  87  89  89  88  89  88  89  90  89  89  89  90  91
##  [52]  91  92  90  91  91  90  89  89  89  88  89  88  89  89  88  87  87
##  [69]  85  87  88  88  88  88  88  87  88  87  88  88  89  89  88  89  88
##  [86]  87  87  88  88  90  90  90  90  90  90  89  88  89  90  88
```

```
Kgrid[which.min(cv.errors)]
```

```
## [1] 25
```

```
plot(x = Kgrid, y = cv.errors, pch = 20)
abline(v = 25, col = 2)
```



A:

(First we divide the data into a training set and a test set - and lock away the test set for model evaluation.)

We work now with the training set.

5-fold CV: we divide the training data randomly into 5 folds of size $n/5$ each and call the folds $i = 1$, to $i = 5$.

For each value of K we do the following.

For $i = 1, \ldots, 5$:

- use the $4n/5$ observations from the folds except fold i to define the K-neighbourhood $\mathcal{N}_0$ for each of the observations in the ith fold
- the observations in the ith fold is left out and is the validation set, there are $n/5$ observations - and we denote them (x_{0l}, y_{0l}),
- we then estimate $\hat{P}(Y = 1|\mathbf{X} = \mathbf{x}_{0l}) = \frac{1}{K} \sum_{j \in \mathcal{N}_0} I(y_j = 1)$ and classify to class 1 if the probability is above 0.5.
- We calculate the misclassification error in the ith fold of the validation set as of the number of misclassified observations is for the validation fold

The total error on the validation set is thus the sum of the number of misclassified observations in the validation set.

So, for each value of K we get an estimate of the validation misclassification. Finally, we choose the value of K that gives the lowest validation misclassification.

Choosing the K that gives the lowest misclassification error on the validation set we end up with $K = 25$ or also $K = 137$, both has a 85 (out of 1000) misclassified observations. The R-code outputs only $K = 25$ since the default is that if many values are equally small, the first is given.

The one-standard-error-rule is use to get a parsimonious model. Instead of choosing K that minimizes some error measure, we use the model that is least complex where the crossvalidation error of the model with K is less than or equal to mean(error)+sem(error) of the model with K* giving the minimum mean error. We can not find this from the print-out, because only the sum of misclassification errors over all folds is given and not the mean and standard error of the mean over the fold.

Comments: Many failed to say that the division into 5 folds were done randomly. For the 1se rule, many did not specify what we wanted the standard deviation of: the standard deviation of the crossvalidation error of the model with the smallest crossvalidation error.

Problem 8: Model evaluation on test set with 3-nearest neighbour classification

[Maximal score: 10 points]

A test set is available, with 500 observations from each of the two classes. We study classifiation at a value $\mathbf{x}_0$ in the test set.

a) For 3-nearest neighbour classification (that is, K-nearest neighbour classification with $K = 3$) what are the possible values for $P(Y = 1 \mid \mathbf{x}_0)$?

A $(0, 1/3, 2/3, 1)$

b) A classification rule may be to classify to class 1 if $P(Y = 1 \mid \mathbf{x}_0) \geq \frac{1}{2}$. For our test set this classification rule gave the following confusion matrix.

```
##       classto1
## test.y   0   1
##      0 464  36
##      1  50 450
```

Calculate the sensitivity and the specificity and explain what the two numbers mean.

A:

	Predicted -	Predicted +	Total
True -	True Negative TN	False Positive FP	N
True +	False Negative FN	True Positive TP	P
Total	N*	P*	

Sensitivity is the proportion of correctly classified positive observations: $\frac{\#\text{True Positive}}{\#\text{Condition Positive}} = \frac{\text{TP}}{\text{P}}$.

Specificity is the proportion of correctly classified negative observations: $\frac{\#\text{True Negative}}{\#\text{Condition Negative}} = \frac{\text{TN}}{\text{N}}$.

We would like that a classification rule (or a diagnostic test) have both a high sensitivity and a high specificity.

We choose to let class 1 denote the positive class and class 0 the negative class. This could as well be swapped.

Sensitivity: $TP/P = 450/500 = 0.9$, TP:classified as 1 and the truth is 1, P: truly 1. True positive rate.
Specificity: $TN/N = 464/500 = 0.928$, TN: classified as 0 and the truth is 0, N: truly 0. True negative rate.

The underlying statistical quantity is $P(\hat{Y} = 1 \mid Y = 1)$ for sensitivity and $P(\hat{Y} = 0 \mid Y = 0)$ for specificity.

Comment: Many students did not connect the "positive" with class 0 or 1.

c) What is a reciever-operator curve (ROC), and what can such a curve tell us?

A

- For all possible cut-offs on probability of class 1 calculate the sensitivity and specificity.
- Plot a graph with sensitivity on the vertical axis and 1-specificity on the horizontal axis. Add the points (0,0) (for specificity 1 and sensitivity 0) and (1,1) (for specificity 0 and sensitivity 1) and then draw lines between all points.
- It is not always clear which cut-off to choose for a specific problem, because maybe different losses are connected to the two decisions. Making an ROC-curve will give you the performance for many different cut-offs, adding flexibility to your evaluation.
- An ROC curve of a perfect classifier will hug the upper left corner of the graph, while random guessing will give an ROC with slope 1.

d) Draw a ROC curve based on the information given in the table above. Include the calculations you have made in your answer. In your drawing of the ROC curve the numerical values you calculated must be clearly visible.

```
table(test.y, prob = KNNprob)
```

```
##       prob
## test.y   0 0.333333333333333 0.666666666666667   1
##      0 391                73                22  14
##      1  14                36                54 396
```

A: From the table we get three possible confusion matrices.

The first is for a cut-off on probability of $\geq 1/3$, and we see that we get that by comparing column 1 with the sum of columns 2, 3 and 4.

```
tab = table(test.y, prob = KNNprob)
cbind(tab[, 1], tab[, 2] + tab[, 3] + tab[, 4])
```

```
##   [,1] [,2]
## 0  391  109
## 1   14  486
```

Sensitivity: $TP/P = 486/500 = 0.972$, Specificity: $TN/N = 391/500 = 0.782$,

This gives the sensitivity, specificity pair $(0.972, 0.782)$.

The second is a cut-off on probability of $\geq 2/3$ and then the confusion matrix is

```
tab = table(test.y, prob = KNNprob)
cbind(tab[, 1] + tab[, 2], tab[, 3] + tab[, 4])
```

```
##   [,1] [,2]
## 0  464   36
## 1   50  450
```

This is the same matrix that we got with cut-off 0.5 above, so that we have already calculated in b). And the we got

Sensitivity: $TP/P = 450/500 = 0.9$ Specificity: $TN/N = 464/500 = 0.928$

The third is a cut-off on probability of ≥ 1 and then the confusion matrix is

```
tab = table(test.y, prob = KNNprob)
cbind(tab[, 1] + tab[, 2] + tab[, 3], tab[, 4])
```

```
##   [,1] [,2]
## 0  486   14
## 1  104  396
```

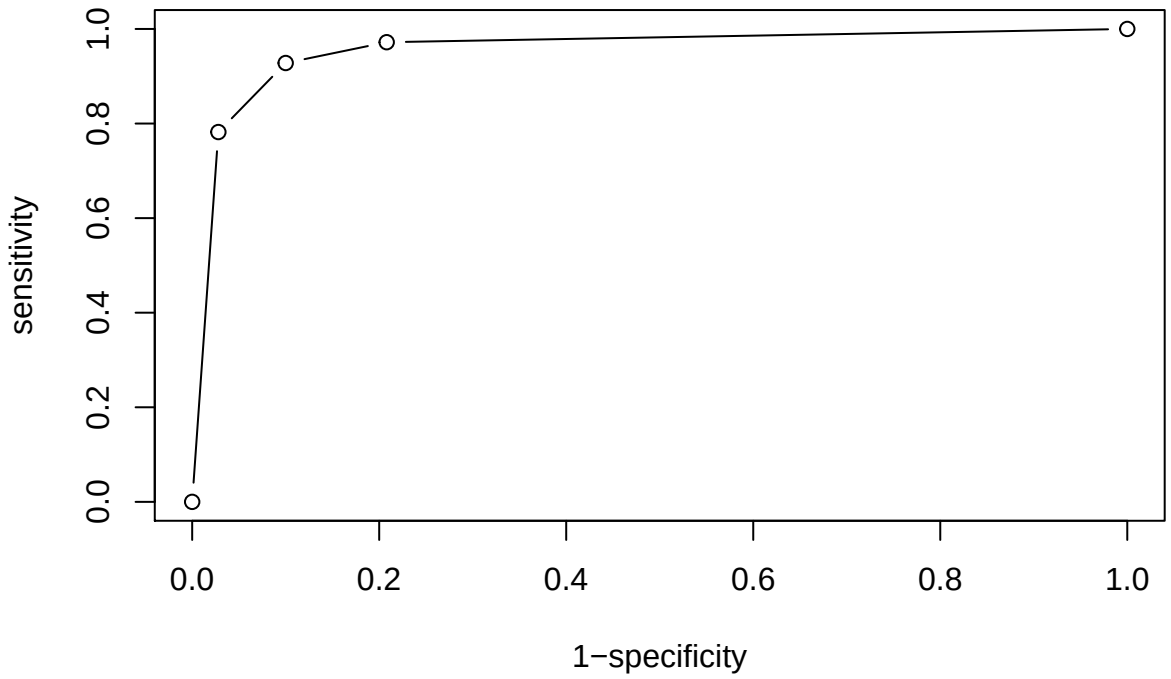
This gives

Sensitivity: $TP/P = 396/500 = 0.792$ Specificity: $TN/N = 486/500 = 0.972$

This gives us the following three points on the ROC:

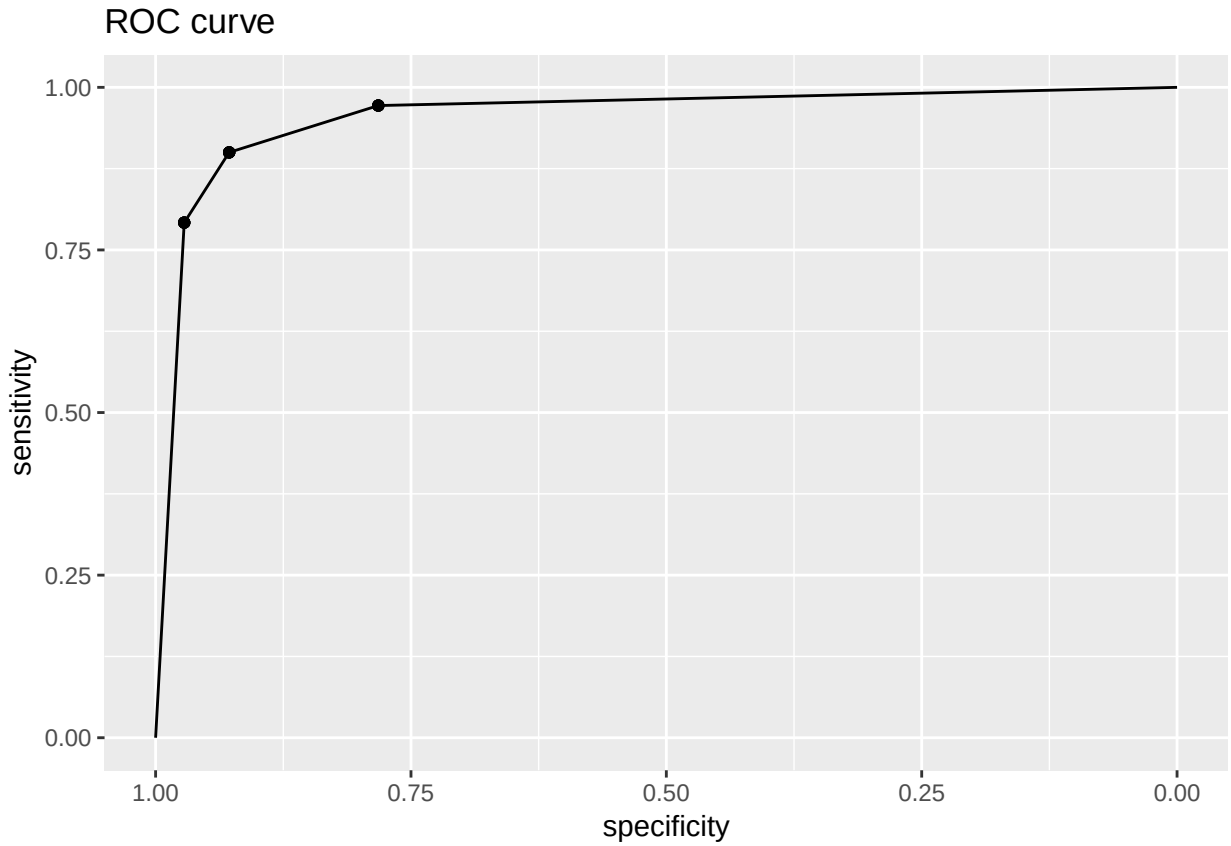
$(0.972, 0.782)$, $(0.9, 0.928)$, $(0.792, 0.972)$, in addition to the boundary points of $(0, 1)$ (classifying all observations as 0 gives 0 sensitivity and 1 specificity) and $(1, 0)$ (classifying all observations as 1 gives 1 sensitivity and 0 specificity). We then plot these 5 points on a curve with sensitivity on the vertical axis and (1-specificity) on the horisonal axis.

```
plot(c(0, 1 - 0.972, 1 - 0.9, 1 - 0.792, 1), c(0, 0.782, 0.928, 0.972,
    1), xlim = c(0, 1), ylim = c(0, 1), type = "b", xlab = "1-specificity",
    ylab = "sensitivity")
```



Checking the results with the pROC library built-in functions.

```
library(pROC)
library(ggplot2)
KNN3roc = roc(response = test.y, predictor = KNNprob, grid = FALSE)
sens = c(0.972, 0.9, 0.792)
spec = c(0.782, 0.928, 0.972)
ggroc(KNN3roc) + ggtitle("ROC curve") + geom_point(x = -spec[1], y = sens[1]) +
    geom_point(x = -spec[2], y = sens[2]) + geom_point(x = -spec[3],
    y = sens[3])
```




```
pROC::auc(KNN3roc)
```

```
## Area under the curve: 0.956
```

```
KNN3roc$sensitivities
```

```
## [1] 1.000 0.972 0.900 0.792 0.000
```

```
KNN3roc$specificities
```

```
## [1] 0.000 0.782 0.928 0.972 1.000
```

Comments and common mistakes:

- There are no other cut-offs that would give other sensitivity and specificity pairs.

Problem 9: Bayes decision rule

[Maximal score: 10 points]

a) What is a Bayes classifier, Bayes decision boundary and Bayes error rate?

A: Bayes classifier: classify to the most probable class (the class with the highest posterior probability) gives the minimum expected 0/1 loss. This requires that we know the posterior probability, which we usually don't know - except for situations when we have simulated data ourselves. The Bayes optimal boundary is the boundary for the Bayes classifier and the error rate (on a future test set drawn from the distribution of the classes) for the Bayes classifier is the Bayes error rate. The Bayes error rate is related to the irreducible error (but bias-variance decomposition is for quadratic loss).

Comment: The Bayes classifier use the posterior probability, not the class conditional probability (formuals below) to classify.

b)

The data set was simulated as follows:

- Prior class probabilities: $\pi_0 = P(Y = 0) = 0.5$ and $\pi_1 = P(Y = 1) = 0.5$
- Class conditional probabilites:
  - $f_0(\mathbf{x}) = 0.5 \cdot \frac{1}{2\pi|\Sigma_{01}|} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{01})^T\Sigma_{01}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{01})] + 0.5 \cdot \frac{1}{2\pi|\Sigma_{02}|} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{02})^T\Sigma_{02}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{02})]$
  - $f_1\mathbf{x}) = \frac{1}{2\pi|\Sigma_1|} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)]$
- with mean vectors $\mu_{01} = \begin{bmatrix}2\\2\end{bmatrix}$, $\mu_{02} = \begin{bmatrix}5\\4\end{bmatrix}$ and $\mu_1 = \begin{bmatrix}5\\1\end{bmatrix}$, and
- covariance matrices $\Sigma_{01} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$,$\Sigma_{02} = \begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}$, and $\Sigma_1 = \begin{bmatrix}1 & -0.3\\-0.3 & 1\end{bmatrix}$

How would you proceed to find the Bayes decision boundary? Write down the equation to be solved, explain what are the unknowns. You are not asked to solve the equation.

A

The class boundary should be at:

$$P(Y = 1 \mid \mathbf{x}) = P(Y = 0 \mid \mathbf{x})$$

We then use the Bayes rule to write out these two probabilities:

$$P(Y = 1 \mid \mathbf{x}) = \frac{f_1(\mathbf{x})\pi_1}{f_0(\mathbf{x})\pi_0 + f_1(\mathbf{x})\pi_1}$$

$$P(Y = 0 \mid \mathbf{x}) = \frac{f_0(\mathbf{x})\pi_1}{f_0(\mathbf{x})\pi_0 + f_1(\mathbf{x})\pi_1}$$

$$\frac{f_0(\mathbf{x})\pi_0}{f_0(\mathbf{x})\pi_0 + f_1(\mathbf{x})\pi_1} = \frac{f_1(\mathbf{x})\pi_1}{f_0(\mathbf{x})\pi_0 + f_1(\mathbf{x})\pi_1}$$

$$f_0(\mathbf{x})\pi_0 = f_1(\mathbf{x})\pi_1$$

When $\pi_0 = \pi_1$ this further gives

$$f_0(\mathbf{x}) = f_1(\mathbf{x})$$

inserting the true formula for each of these:

$$0.5 \cdot \frac{1}{2\pi|\Sigma_{01}|} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{01})^T\Sigma_{01}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{01})] + 0.5 \cdot \frac{1}{2\pi|\Sigma_{02}|} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_{02})^T\Sigma_{02}^{-1}(\mathbf{x}-\boldsymbol{\mu}_{02})]$$
$$= \frac{1}{2\pi|\Sigma_2|} \exp[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)]$$

where all mean vectors and covariance matrices are known. Then, we find pairs $\mathbf{x} = (x_1, x_2)$ where this equation is satisfied. This was done on a grid- calculating $f_0(\mathbf{x})$ and $f_1(\mathbf{x})$ in each point in the grid and reporting a boundary point if $f_0(\mathbf{x}) = f_1(\mathbf{x})$.

Comment: We here assume that we know the means and covariance vectors, and the the **x**s are the only unknown in deciding on the boundary.

c) The Bayes decision boundary is shown in the plot below. The Bayes error rate was found to be 8%. Explain what this means.

A: When the true data generating mechanism is known and this is used to produce the optimal Bayes decision boundary then the misclassification for new data simulated is 8%.

Comment: Just saying that this is the misclassification rate and not relating this to the optimal rate did not give any points for this question.

```
##
##     1
## 33099
```

d) What would happen to the Bayes decision boundary and Bayes error rate if we change the prior probabilities of the two classes to $\pi_0 = 0.9$ and $\pi_1 = 0.1$? Elaborate.

The Bayes decision boundary is at

$$\frac{f_0(\mathbf{x})\pi_0}{f_0(\mathbf{x})\pi_0 + f_1(\mathbf{x})\pi_1} = \frac{f_1(\mathbf{x})\pi_1}{f_0(\mathbf{x})\pi_0 + f_1(\mathbf{x})\pi_1}$$

$$f_0(\mathbf{x})\pi_0 = f_1(\mathbf{x})\pi_1$$

and now (unlike above) we can not cancel out the prior probabilities, and means that our Bayes decision boundary will move.

Since the prior for class 0 is increasing this means that the Bayes decision boundary will move to make the area for classifying to class 0 larger. This means that the boundary will move to the right and down.

This will again make the Bayes error rate change, but since the priors are changing (from which we draw new data) is it not clear if the error rate will increase or descrease. Simulations, below, show that the error rate will decrease - here to approx 3%.

```
##
##     1
## 33099
```

e) The method we have considered in this course are K-nearest neighbour classification, linear discriminant analysis, quadratic discriminant analysis, logistic regression, classification tree, bagged trees, random forest, maximal margin classifier, support vector classifier, support vector machine, feedforward neural network. Now that you know what the Bayes decision boundary looks like, which of the classification methods that we have studied in this course do you think would provide the best solution to our classification problem? Justify your answer.

A The KNN-classification boundary for K=199 is not far from the Bayes boundary. The test error for K=199 was not given, so we can not compare to 8%. But below you see that the error rate is close to 8.

Another method that might be ok is the regression tree - since the tree easily gives rectangular decision boundaries. In general methods that give non-linear boundaries which are smooth would be good.

Comment: The mixture of two multivariate normal distributions is not a multivariate normal distribution, and QDA assumes both classes are multivariate normal. This means that QDA is not an appropriate method.

```
classif = class::knn(train.x, test = test.x, cl = train.y, k = 199, prob = TRUE)
prob = attr(classif, "prob")
KNNprob = ifelse(classif == "0", 1 - prob, prob)
classto1 = rep(0, 1000)
classto1[KNNprob >= 0.5] = 1
table(test.y, classto1)
```

```
##       classto1
## test.y   0   1
##      0 458  42
##      1  34 466
```

```
confusionMatrix(data = factor(classto1), reference = test.y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 458  34
##          1  42 466
##
##                Accuracy : 0.924
##                  95% CI : (0.9058, 0.9397)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.848
##
##  Mcnemar's Test P-Value : 0.422
##
##             Sensitivity : 0.9160
##             Specificity : 0.9320
##          Pos Pred Value : 0.9309
##          Neg Pred Value : 0.9173
##              Prevalence : 0.5000
##          Detection Rate : 0.4580
##    Detection Prevalence : 0.4920
##       Balanced Accuracy : 0.9240
##
##        'Positive' Class : 0
##
```

Regression with bike rentals

We will look at a data set of daily counts of bike rentals in a bike sharing system in Washington DC in the first two years of operation, matched with data on climate, season and type of day.

- `y`: daily count of number of bike rentals (our response)
- `year`: (0: 2011, 1: 2012)
- `season`: factor with four levels (1: spring, 2: summer, 3: autumn, 4: winter)
- `holiday`: (0: not holiday, 1: holiday)
- `notworkday`: (0: neighter weekend or holiday, 1: weekend or holiday)
- `weather`: factor with three levels (1: clear to partly cloudy, 2: mist and/or clouds, 3: snow, rain, thunderstorm)
- `temp`: normalized temperature in Celsius
- `wind`: normalized wind speed

The data was divided randomly into a training set of size 500 and a test set of size 231. The training data is presented in the pairs-plot below.

More information at http://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset and also look at the article linking and preprocessing the data: Fanaee, Hadi and Gama (2013), Event labeling combining ensemble detectors and background knowledge, in *Progress in Artificial Intelligence*, http://dx.doi.org/10.1007/s13748-013-0040-3.

Problem 10: Smoothing spline and polynomials

[Maximal score: 6 points]

We first use the daily count of number of bike rentals as response and temperature as the only covariate.

a) What is the idea behind a smoothing spline, and which criterion (loss and penalty) is the smoothing spline minimizing? What is the connection between a smoothing spline and a cubic spline?

A: The idea of smoothing splines is that we find a function $f(x)$ that fits the observations well resulting in a small RSS $= \sum_{i=1}^{n}(y_i - f(x_i))^2$. However, we need another requirement in addition to the RSS. Otherwise, our prediction will be a function that interpolates all the observations, with RSS $= 0$. Smoothing splines minimize the RSS, but also reduce the variance in the prediction. A smoothing spline is the function f that minimizes

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int f''(t)^2 \mathrm{d}t,$$

where λ is a non-negative tuning parameter. The first term is loss and puts f close to the observations, while the second term penalizes variability.

Note that $\int f''(t)^2 \mathrm{d}t$ is the total squared change of $f'(t)$, that is how much f turns. The higher λ is, the smoother f will be. In the limit when $\lambda \to \infty$, f is the straight line we would get from linear least squares regression. Conversely, when $\lambda = 0$, f interpolates all the observations. Thus, λ controls the bias-variance trade-off.

Splines have knots, so that is the case also for smoothing splines. A smoothing spline is a shrunken natural cubic spline with knots at the unique values of x.

A total of five curves were fitted:

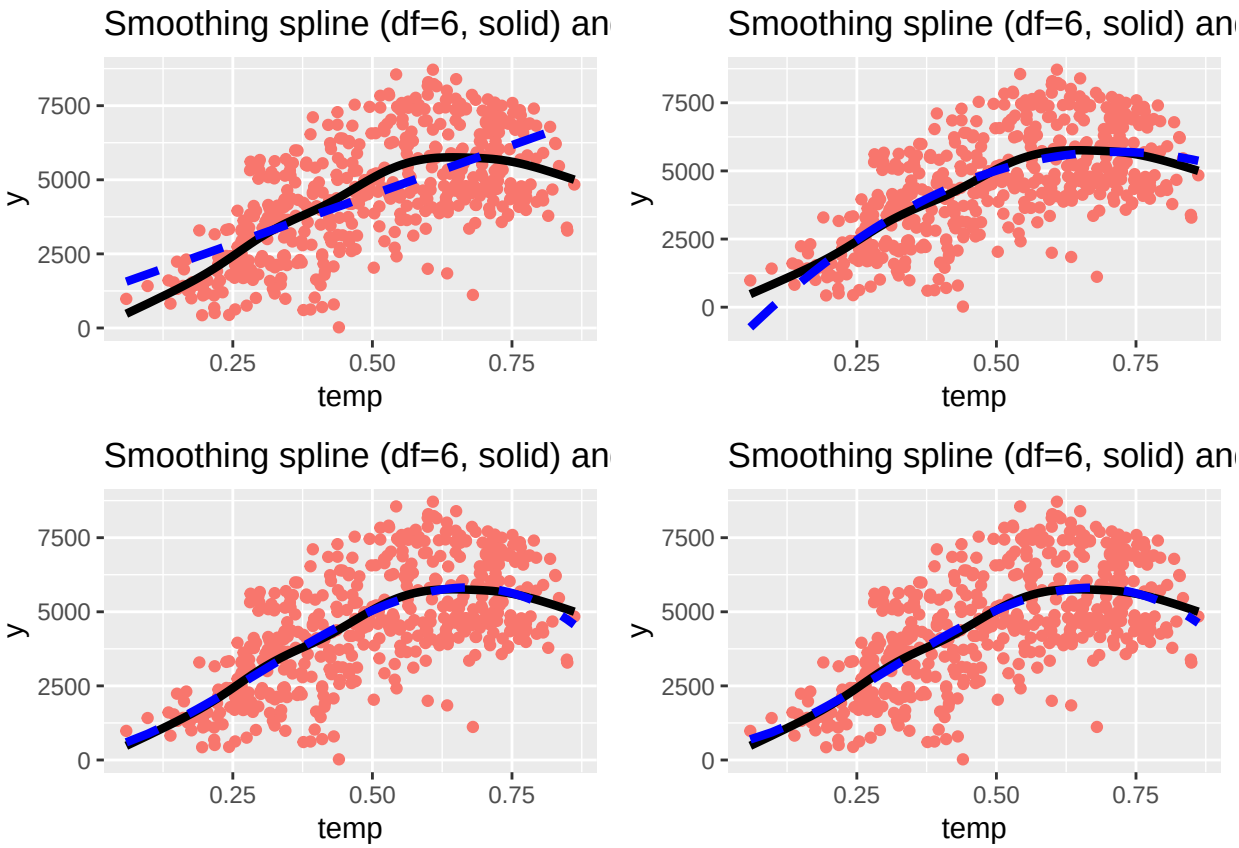
- a smoothing spline with 6 effective degrees of freedom was fit (solid, black)
- polynomial of degrees 1, 2, 3 and 4 (dashed, blue)

See the results in the four panels below.

b) What does it mean that the smoothing spline has 6 effective degrees of freedom? How would you evaluate the different model fits in the four panels? We will next fit a multiple linear regression with all available covariates. What is your suggestion for how to handle the `temp` covariate?

A: A smoothing spline is a linear smoother, and can be written $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$, where S is called a smoother matrix. The degrees of freedom of a smoothing spline is equal to the number of basis functions, while the effective degrees of freedom is equal to the sum of the diagonal elements of the smoother matrix. The effective degrees of freedom is smaller than the degrees of freedom due to the penalty on the smoothness of the curve.

The plot show that a quadratic or cubic polynomial can be a good fit for the regression. We now look at marginal fits, so this may change when we add more covariates. But as a starting point this should be good.



Problem 11: Multiple linear regression

[Maximal score: 4 points]

A multiple linear regression was fitted to the data, with linear effects of all covariates, and in addition also a quadratic and cubic effect of temperature. Dummy variable coding (treatment contrast) was used for the factors, with the lowest level (season=1 and weather=1) as the reference category.

A pdf with R-code and print-out is found to the left.

a) Write down the mathematical formula for the regression on matrix form. What is the dimension of the design matrix? Write out the row of the design matrix for an observation with

- `year`=1
- `season`=1
- `holiday`=0
- `notworkday`=0
- `temp`=0.3
- `weather`=3
- `wind`=0.3

A

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

The design matrix has dimension n times $p+1$ where $p+1 = 13$ The columns for the given observation is

- Intercept:1

- Year: 1
- Season: three columns with 0
- Holiday: 0
- Notworkday: 0
- Temperature: three columns with 0.3, 0.3^2 and 0.3^3 (or othogonalized versions there-of if you instead would use `poly`)
- Weather: two columns, first with 0 and second with 1
- Wind: 0.2

This gives the row vector (given the ordering above)

$$(1, 1, 0, 0, 0, 0, 0, 0.3, 0.3^2, 0.3^3, 0, 1, 0.2)$$

In the R-code a different ordering was given: `season+year+holiday+notworkday+weather+temp+I(temp^2)+I(temp^3)+wind` which would give the row vector

$$(1, 0, 0, 0, 1, 0, 0, 0, 1, 0.3, 0.3^2, 0.3^3, 0.2)$$

b) What are the assumptions made for the error terms in a normal multiple linear regression model? Why might we think that the response, which is a count, can be modelled to be normally distributed? Comment briefly on the two residual plots.

A:

$$\boldsymbol{\varepsilon} \sim N_n(\mathbf{0}, \sigma^2\mathbf{I})$$

Error terms are independent between observations and with the same variance.

A count is often seen as the result of a Poisson process and for a high value of the intensity of the Poisson process the normal approximation might be used.

However, the residual plots are not looking that good. We also have negative fitted values (negative number of bicycles rented). There might be unexplained variablity for small fitted valuse and the residuals are far from normal.

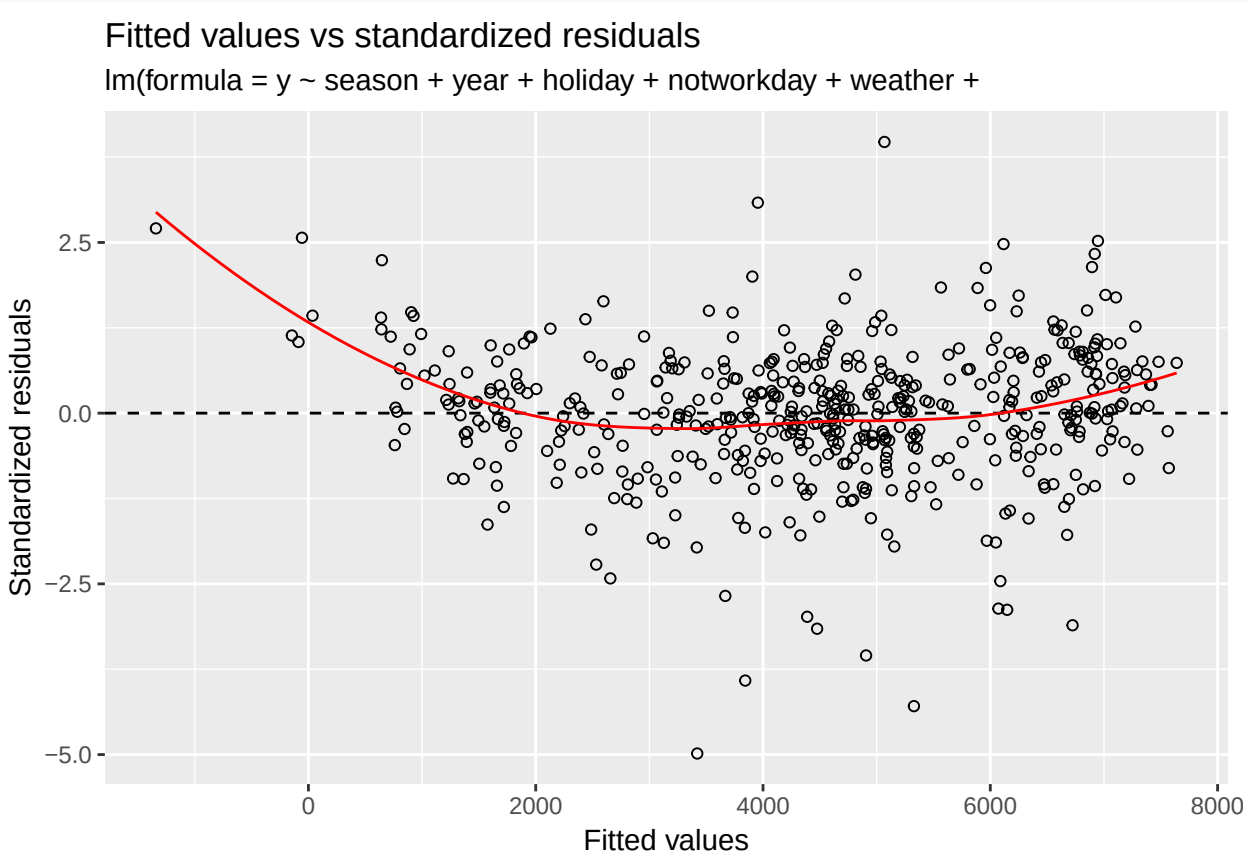
Comment: The data are taken over 2 years, but randomly divided into training and testing. There might be serial correlations in the counts (time series structure) that we have not handled with our regression model. But, since the date was not included and the data were in random order it was not possible for the student to assess serial correcation.

```
fitlm = lm(y ~ season + year + holiday + notworkday + weather + temp +
    I(temp^2) + I(temp^3) + wind, data = train)
# I(temp^2) just means temp^2, but in a model formula this extra I()
# is needed
summary(fitlm)
```

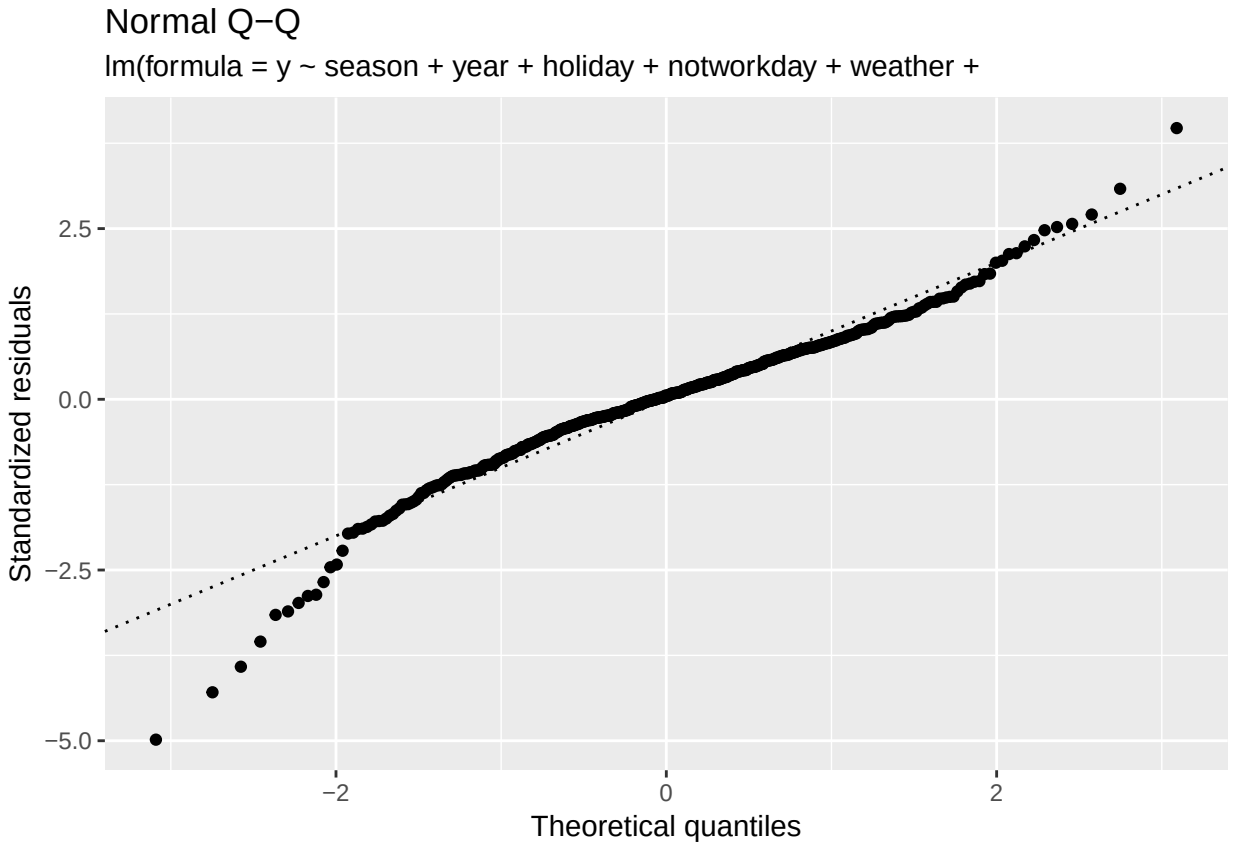
```
##
## Call:
## lm(formula = y ~ season + year + holiday + notworkday + weather +
##     temp + I(temp^2) + I(temp^3) + wind, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3399.3  -344.9    36.9   429.2  2767.6
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2902.20     481.36   6.029 3.26e-09 ***
```

```
## season2        828.76     121.20   6.838 2.41e-11 ***
## season3       1093.71     155.38   7.039 6.62e-12 ***
## season4       1286.18     104.99  12.251  < 2e-16 ***
## year          2052.39      64.82  31.663  < 2e-16 ***
## holiday       -479.99     199.13  -2.410   0.0163 *
## notworkday    -100.66      70.22  -1.434   0.1524
## weather2      -705.99      69.99 -10.087  < 2e-16 ***
## weather3     -2495.95     186.56 -13.379  < 2e-16 ***
## temp        -14643.01    3430.61  -4.268 2.37e-05 ***
## I(temp^2)    57717.61    7464.37   7.732 6.10e-14 ***
## I(temp^3)   -47873.21    5071.47  -9.440  < 2e-16 ***
## wind         -2726.69     425.89  -6.402 3.61e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 709.4 on 487 degrees of freedom
## Multiple R-squared:  0.8707, Adjusted R-squared:  0.8675
## F-statistic: 273.2 on 12 and 487 DF,  p-value: < 2.2e-16
```

```
ggplot(fitlm, aes(.fitted, .stdresid)) + geom_point(pch = 21) + geom_hline(yintercept = 0,
    linetype = "dashed") + geom_smooth(se = FALSE, col = "red", size = 0.5,
    method = "loess") + labs(x = "Fitted values", y = "Standardized residuals",
    title = "Fitted values vs standardized residuals", subtitle = deparse(fitlm$call))
```




```
ggplot(fitlm, aes(sample = .stdresid)) + stat_qq(pch = 19) + geom_abline(intercept = 0,
    slope = 1, linetype = "dotted") + labs(x = "Theoretical quantiles",
    y = "Standardized residuals", title = "Normal Q-Q", subtitle = deparse(fitlm$call))
```




```
library(nortest)
ad.test(rstudent(fitlm))
```

```
##
##  Anderson-Darling normality test
##
## data:  rstudent(fitlm)
## A = 3.9514, p-value = 7.525e-10
```

```
trainerrorlm = mean((predict(fitlm) - train$y)^2)
predlm = predict(fitlm, newdata = test)
testerrorlm = mean((predlm - test$y)^2)
c(trainerrorlm, testerrorlm)
```

```
## [1] 490198.7 568906.1
```

Problem 12: Ridge regression

[Maximal score: 10 points]

A ridge regression was fitted to the bike rental data, and R-code and print-out is found in the pdf-file in the left panel.

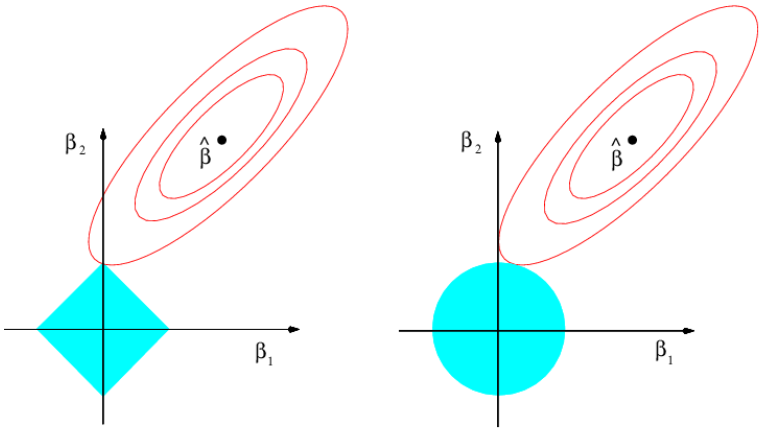
a) Write down the optimization problem for the ridge regression, and illustrate with a drawing. How was the penalty parameter chosen in the analysis in the pdf-file? What was the optimal value?

A:

Drawing: constrained optimization, minimize least squares error under the constraint that $\sum_{j=1}^{p} \beta_j^2 \leq C$ -

drawn as a circle. The penalty parameter is found by 10-fold cross-validation using the 1se rule, and the optimal value was 196.

Figure 6.7, right panel is the illustration asked for.



b) Write down the formula for the ridge regression estimator and derive the mean and covariance matrix for the estimator. Specify the assumptions you make and comment on your findings.

A: Estimator:

$$\tilde{\beta} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$$

We need to assume that we study a regression problem

$$Y = f(\mathbf{x}) + \varepsilon, \text{ where } \quad \mathrm{E}(\varepsilon) = 0 \text{ and } \quad \mathrm{Var}(\varepsilon) = \sigma^2.$$

and the error terms are independent for each observation pair $(\mathbf{x}_i, Y_i)$. Further, we know that the true function is a linear combination of observed covariates $f(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta}$.

$$\mathrm{E}(\widetilde{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathrm{E}(\mathbf{Y}) = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}\boldsymbol{\beta}$$

$$\mathrm{Cov}(\widetilde{\boldsymbol{\beta}}) = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathrm{Cov}(\mathbf{Y})((\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T)^T$$
$$= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\sigma^2\mathbf{I}\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1} = \sigma^2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}$$

c) In ridge regression regularization is performed by adding a penalty term to a loss function. Why would we want to add a penalty term to the loss function? What is the connection between regularization and the bias–variance trade–off?

A:

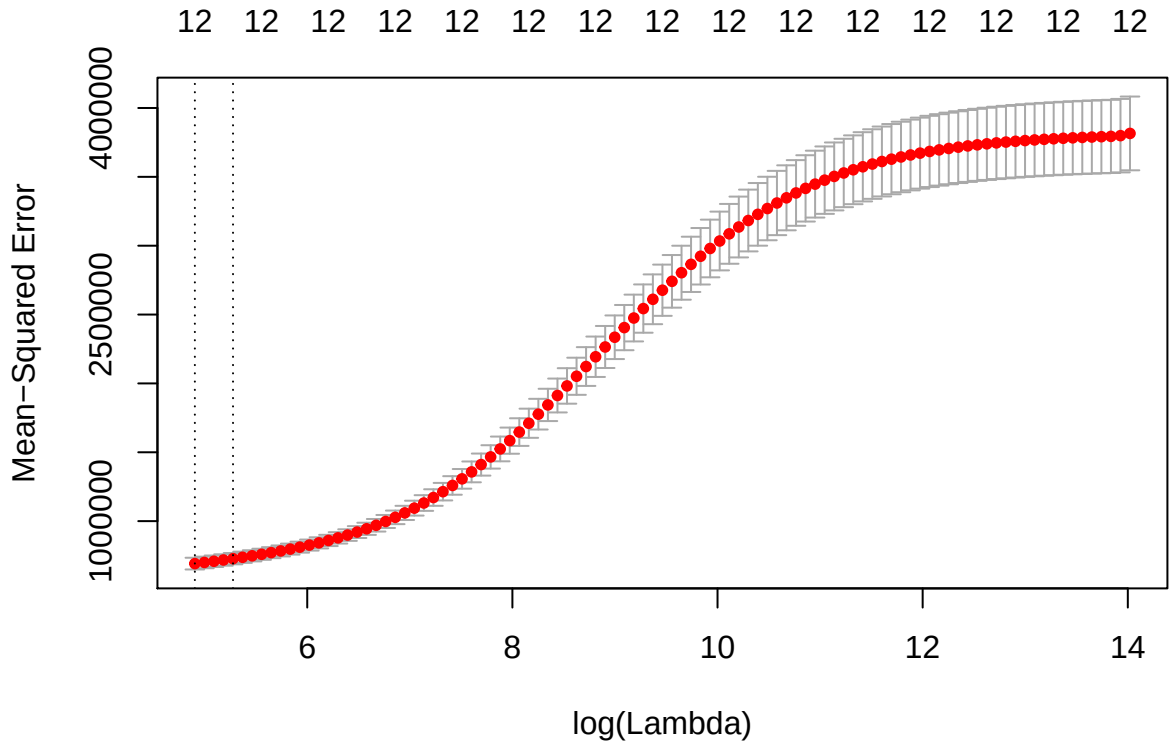
In some situations, for example with highly correlated data or many covariates, the variance of the regression coefficients are very high, and by shrinking the parameter estimates (to become biased) the sum of the squared bias and the variance will be smaller for the ridge than for the least squares. This means that we trade-off increase in squared bias towards decrease in variance.

d) In the bottom of the pdf-file there is a comparison between the estimated parameters of the multiple linear regression and the ridge regression. Comment on the comparison. Would you prefer to use multiple linear regression or ridge regression in the analysis of the bike rental data?

A:

The ridge parameters are shrunken versions of the least squares parameters (except for the intercept which is not shrunken). The test set error is smaller for least squares that for ridge, and I would prefer the least squares estimators.

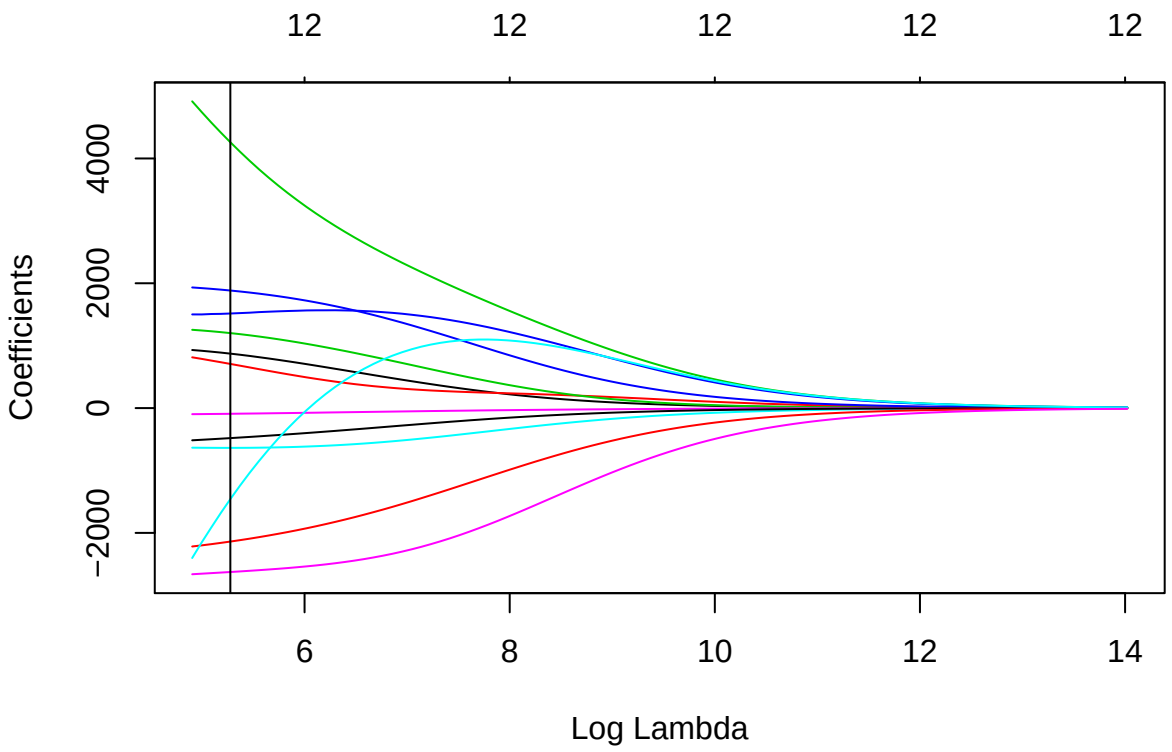
```
library(glmnet)
# set up model matrix, not include the intercept term because glmnet
# doesn't want that
model = formula(~season + year + holiday + notworkday + weather + temp +
    I(temp^2) + I(temp^3) + wind)
mm = model.matrix(model, data = train)[, -1]
set.seed(4268)  #for reproducibility
cvfitridge = cv.glmnet(mm, train$y, alpha = 0, standardize = TRUE, nfolds = 10)
plot(cvfitridge)
```




```
print(cvfitridge$lambda.1se)
```

```
## [1] 195.6469
```

```
# also plot how coeffients change with (log)lambda
fitridgegrid = glmnet(mm, train$y, lambda = cvfitridge$lambda, alpha = 0,
    standardize = TRUE)
plot(fitridgegrid, xvar = "lambda")
abline(v = log(cvfitridge$lambda.1se))
```




```
# look at final model
fitridge = glmnet(mm, train$y, lambda = cvfitridge$lambda.1se, alpha = 0,
    standardize = TRUE)
coef(fitridge)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  1365.14088
## season2       870.69235
## season3       704.21687
## season4      1201.04463
## year         1883.85953
## holiday      -637.79542
## notworkday    -91.24635
## weather2     -481.02879
## weather3    -2138.55327
## temp         4245.95260
## I(temp^2)    1545.05199
## I(temp^3)   -1469.95619
## wind        -2626.19893
```

```
# mse on training and test set
trainerrorridge = mean((predict(fitridge, newx = mm) - train$y)^2)
predridge = predict(fitridge, newx = model.matrix(model, data = test)[,
    -1])
testerrorridge = mean((predridge - test$y)^2)
c(trainerrorridge, testerrorridge)
```

```
## [1] 688158.6 731022.4
```

```
# estimated regression coefficients for least squares (lm) and ridge
# regression (glmnet)
```

```
res = cbind(fitlm$coefficients, coef(fitridge))
colnames(res) = c("lm", "ridge")
res
```

```
## 13 x 2 sparse Matrix of class "dgCMatrix"
##                       lm       ridge
## (Intercept)   2902.1995  1365.14088
## season2        828.7597   870.69235
## season3       1093.7146   704.21687
## season4       1286.1808  1201.04463
## year          2052.3907  1883.85953
## holiday       -479.9857  -637.79542
## notworkday    -100.6592   -91.24635
## weather2      -705.9897  -481.02879
## weather3     -2495.9517 -2138.55327
## temp        -14643.0085  4245.95260
## I(temp^2)    57717.6057  1545.05199
## I(temp^3)   -47873.2054 -1469.95619
## wind         -2726.6905 -2626.19893
```

```
# and remembering the errors for lm and ridge
c(trainerrorlm, testerrorlm)
```

```
## [1] 490198.7 568906.1
```

```
c(trainerrorridge, testerrorridge)
```

```
## [1] 688158.6 731022.4
```

Problem 13: Regression tree

[Maximal score: 5 points]

We have fitted a regression tree to the training data, and R-code and print-out is found in the pdf-file in the left panel.

```
library(tree)
fitT = tree(y ~ ., data = train)
summary(fitT)
```

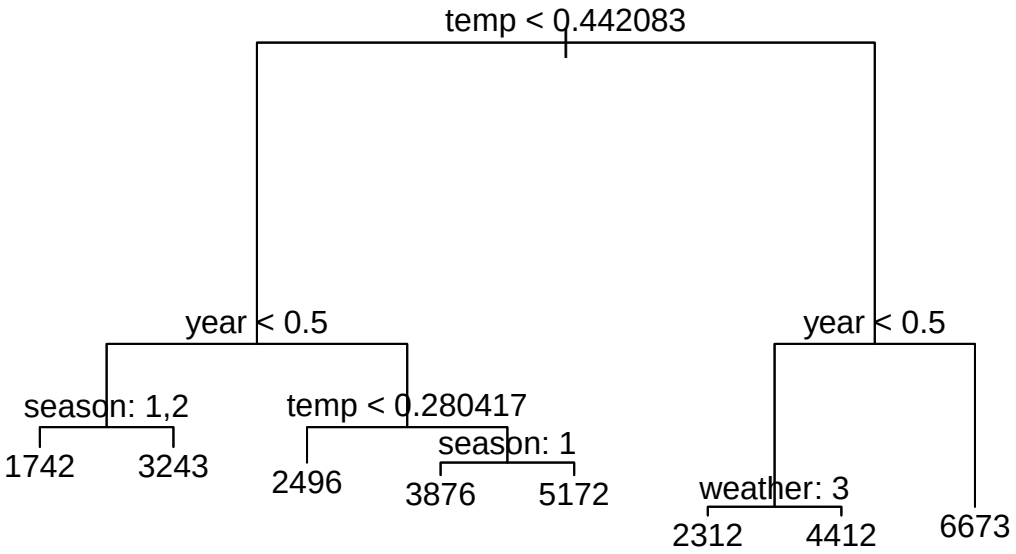
```
##
## Regression tree:
## tree(formula = y ~ ., data = train)
## Variables actually used in tree construction:
## [1] "temp"    "year"    "season"  "weather"
## Number of terminal nodes:  8
## Residual mean deviance:  774800 = 381200000 / 492
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -5150.0  -430.7   126.0     0.0   504.4  2110.0
```

```
fitT
```

```
## node), split, n, deviance, yval
##       * denotes terminal node
##
```

```
##  1) root 500 1.895e+09 4493
##    2) temp < 0.442083 210 5.322e+08 3073
##      4) year < 0.5 110 1.009e+08 2138
##        8) season: 1,2 81 3.150e+07 1742 *
##        9) season: 4 29 2.128e+07 3243 *
##      5) year > 0.5 100 2.293e+08 4101
##       10) temp < 0.280417 25 1.599e+07 2496 *
##       11) temp > 0.280417 75 1.274e+08 4636
##         22) season: 1 31 1.819e+07 3876 *
##         23) season: 2,4 44 7.866e+07 5172 *
##    3) temp > 0.442083 290 6.319e+08 5522
##      6) year < 0.5 143 9.863e+07 4339
##       12) weather: 3 5 5.695e+05 2312 *
##       13) weather: 1,2 138 7.677e+07 4412 *
##      7) year > 0.5 147 1.382e+08 6673 *
```

```
plot(fitT)
text(fitT, pretty = TRUE)
```



a) Using the fitted regression tree, what would you predict as the number of bikes rented on a day with the following covariates:

- `year`=1
- `season`=1
- `holiday`=0
- `notworkday`=0
- `temp`=0.3
- `weather`=3
- `wind`=0.3

Remember: at splits the criterion at the node is to the left in the tree.

A:

Top node go to left (temp < 0.44), then next go to right (year>0.5), then to right (temp>0.28), and finally season=1 so go left. Then we are in a node where the mean is 3876.

b) The top split in the regression tree is on temp <0.442083 (left) and temp 0.442084 (right). Why is this the top split? Include a mathematical formula in your argument.

A

This is the split that minimizes the following formula (over all covariates and splits)

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

The tree is fitted using a greedy algorithm.

c) What are similarities and differences between the greedy tree fitting algorithm and forward stepwise selection in regression?

A

Similarities:

- Both methods proceed in a forward greedy fashion. First only the mean of all observations is our prediction, then we choose to include either a split or a covariate, which makes the RSS go down the most.

Differences:

- The linear regression is linear in parameters and in covariates, while the tree builds uses step functions in the covariates (so not linear in covariates). Therefore the linear regression might be better at fitting relationships that are linear in the covariates, and the tree rectangular regions.
- The linear regression does not include the same covariate more than one time, while in the tree the covariate can be included several times, with different splits.