

Compulsory exercise 2: Group 27

TMA4268 Statistical Learning V2021

Maren Bråthen Kristoffersen, Vilde Marie Skårdal Jensen and Viveka Priya Simhan

13 april, 2021

Problem 1

a) Multiple choice 1

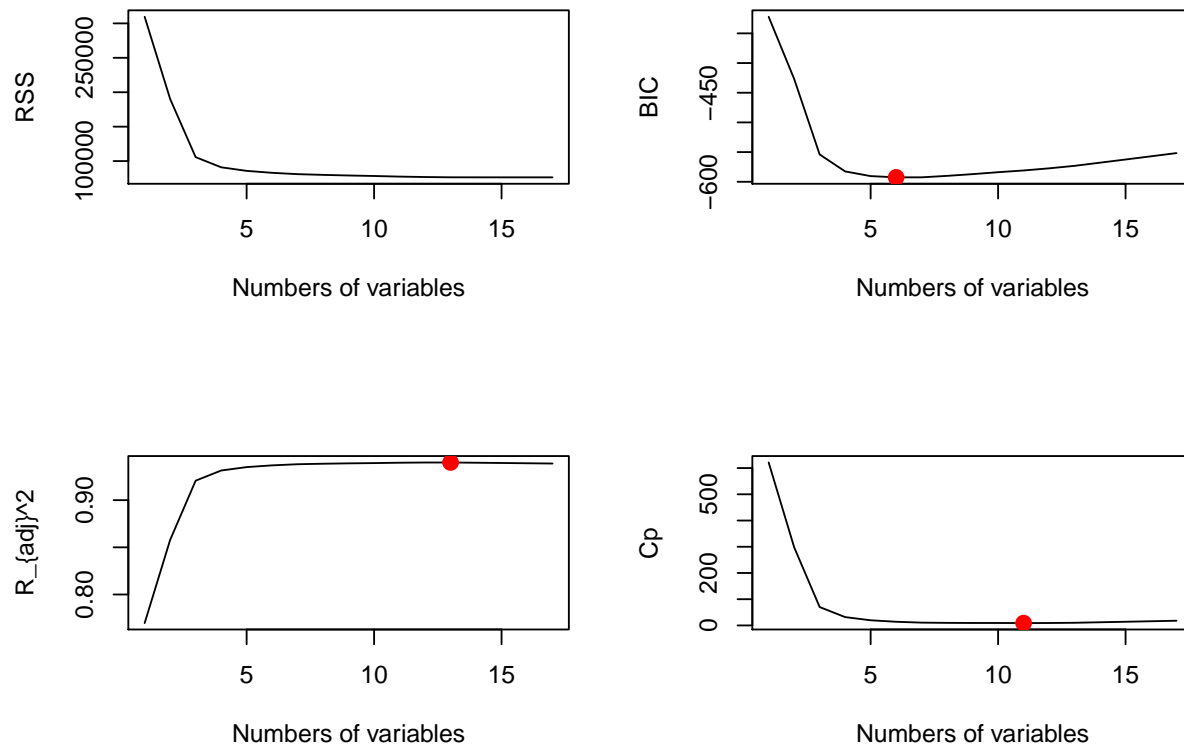
- (i) TRUE
- (ii) TRUE
- (iii) TRUE
- (iv) FALSE

b) Best subset selection

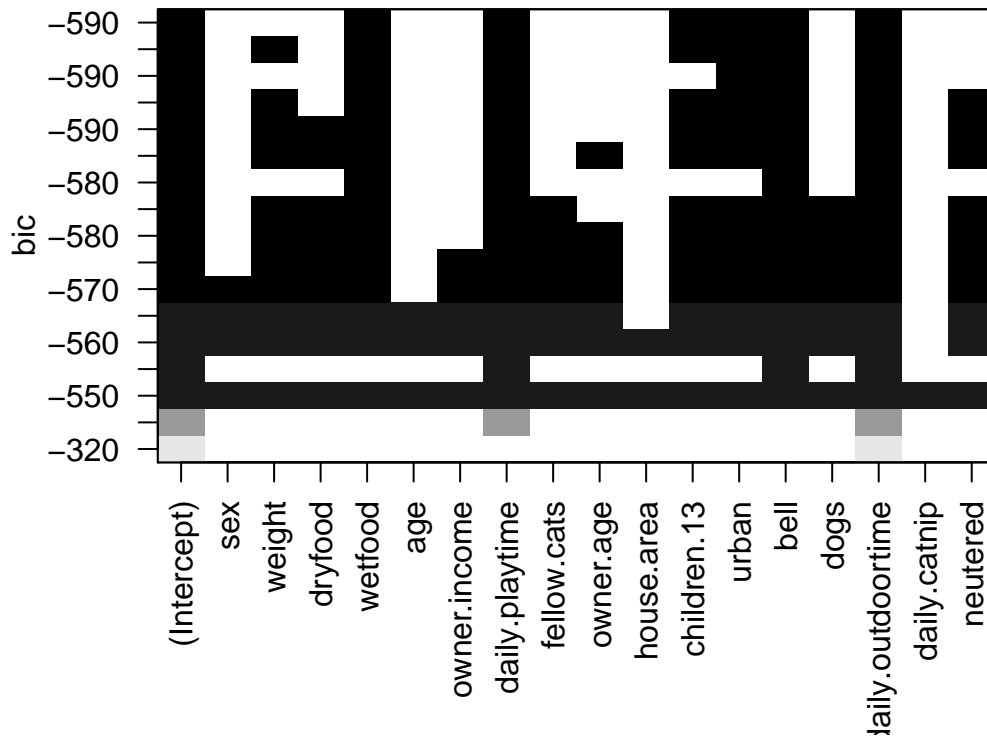
For this task, we will perform best subset selection to obtain a reduced linear regression model based on the `catdat` dataset. Below, we have used the function `regsubsets()` to implement the best subset selection method.

```
numPred = 17
BSS = regsubsets(birds~., data = catdat.train, nvmax = numPred)
BSS_summary = summary(BSS)
```

By observing the graphs below, that are illustrating the optimal numbers of predictors for different criteria, we want to determine which model choice criterion to use.



The first plot shows that the RSS decreases for increasing number of predictors, which is expected since RSS always will increase when adding more predictors to the model. When wanting to minimize the Bayesian Information Criterion (BIC), one sees that the subset containing six predictors turns out as the best subset. For the adjusted R-squared and C_p , one can see that there is no specific subset that stands out to be the best, and it looks like all subsets containing between 5 and 17 predictors results in approximately the same model performance. However, all the plots indicates that the reduced model should include at least the five predictors `wetfood`, `daily.playtime`, `urban`, `bell` and `daily.outdoortime`. Kept in mind that we want our model to be as simple as possible, we have chosen BIC as model choice criterion.



```
##      (Intercept)      wetfood      daily.playtime      children.13
##      106.097330      -10.209847      -10.311803      4.224025
##      urban      bell      daily.outdoortime
##      -10.573511      -49.364863      1.142209
```

The above plot illustrates the best subsets ranked based on BIC, where the subset that minimize the BIC is at the top. Thus, the selected variables for the satisfactory model are wetfood, daily.playtime, children.13, urban, bell and daily.outdoortime.

Furthermore, we want to determine the mean square error of the reduced linear regression model based on the selected variables above.

```
#linear regression using model criterion BIC
kept_predictors <- names(coef(BSS, id = BSS_best_bic))
kept_predictors <- kept_predictors[!kept_predictors %in% "(Intercept)"]
BSS_design_matrix <- model.matrix(birds~., catdat.train)[, kept_predictors]
BSS_data_train <- data.frame(birds = catdat.train$birds, BSS_design_matrix)

LinReg_BIC <- lm(birds~., BSS_data_train)

#make predictions on the test set
BSS_design_matrix_test <- model.matrix(birds~., catdat.test)[, kept_predictors]
BSS_predictions <- predict(LinReg_BIC, newdata = as.data.frame(BSS_design_matrix_test))
```

```

#compute test squared errors
BSS_squared_errors <- (catdat.test$birds - BSS_predictions)^2
squared_errors <- data.frame(BSS_squared_errors = BSS_squared_errors)

#the test MSE
MSE_BSS <- mean(BSS_squared_errors)

```

The test mean square error for the reduced model using best subset selection with 6 predictors is 299.88.

c) Lasso Regression

Now, we are going to obtain a lasso regression model of the dataset catdat.

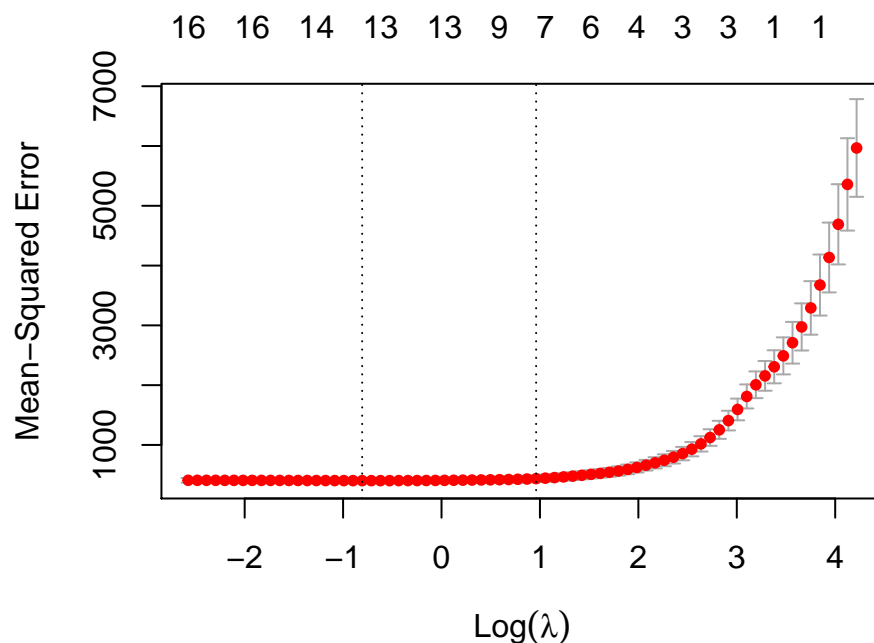
```

#make matrices of the training and testing data
x.train <- model.matrix(birds~., data = catdat.train)[, -1]
y.train <- catdat.train$birds
x.test <- model.matrix(birds~., data = catdat.test)[, -1]
y.test <- catdat.test$birds

#fits a generalized linear model with lasso penalty.
lasso_mod <- glmnet(x.train, y.train, alpha = 1)

#k-fold cross validation to choose lambda
set.seed(1)
cv.out = cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.out)

```



```

bestlam.Lasso<- cv.out$lambda.min

lasso_predictions = predict(lasso_mod, s = bestlam.Lasso, newx= x.test)
lasso_square_errors <- as.numeric((lasso_predictions - y.test)^2)
squared_errors_lasso<- data.frame(lasso_square_errors = lasso_square_errors, squared_errors)

#determine the coefficients used in the lasso regression
lasso.coef = predict(lasso_mod, type = "coefficients", s = bestlam.Lasso)[1:18,]
all_coefs = lasso.coef
nonzero_coefs = all_coefs[all_coefs!=0]

#the test MSE
MSE_lasso <- mean(lasso_square_errors)

```

For this method, λ is chosen using cross-validation, where the λ that minimizes the mean squared error is selected and was found to be $\lambda = 0.4465155$. A beneficial property of the lasso regression is that predictors can be shrunk all the way to zero, meaning that parameters that is not significant for estimating the response can be completely erased from the model. In our case, the non-zero coefficients using lasso regression are (Intercept), weight, dryfood, wetfood, owner.income, daily.playtime, owner.age, house.area children.13, urban, bell, dogs, daily.outdoortime, neuteredand the omitted once are sex, age, fellow.cats, daily.catnip.

The test mean square error for the lasso regression model was found to be 298.4227495.

d)

For increased values of λ the slopes gets smaller and smaller until they reach zero as $\lambda \rightarrow \infty$. This means that one is left with a model containing only the intercept. On the other hand, when $\lambda = 0$ the lasso regression line will be the same as for the least squared regression line, i.e. the full multiple linear regression model, since the term that provides for penalization vanishes.

e)

We will now show that the test MSE of the best subset selection model and the lasso regression model indeed are better than the test MSE for the model with only intercept and the full model. We find these test MSE's the following way

```

#determine MSE for a model containing only the intercept
ILR <- lm(birds~1, data = catdat.train)
ILR_squared_errors <- (catdat.test$birds - summary(ILR)$coefficients[1])^2
MSE_ILR <- mean(ILR_squared_errors)

#determine MSE for a multiple linear regression
MLR <- lm(birds~., data = catdat.train)
MLR_predictions <- predict(MLR, newdata = catdat.test)
MLR_squared_errors <- (catdat.test$birds - MLR_predictions)^2
MSE_MLR <- mean(MLR_squared_errors)

```

The test mean square errors for respectively the model containing only intercept and the multiple linear regression model were found to be 4914.0693576 and 301.9185792. Both of these values are greater than the once obtained for best subset selection with BIC as model criteria and lasso regression. Hence, the models from b) and c) results in better predictions than both the full model and the model with no predictors.

f)

From the previous tasks we found the MSE for the four cases to be

##	the test MSE
## Best Subset Selection	299.8835
## Lasso Regression	298.4227
## Multiple Linear Regression	301.9186
## Intercept Linear Regression	4914.0694

From this tabulate, one sees that mean square errors the intercept linear regression stands out to be the absolute worst model to predict how many birds a cat will kill during a year. This is due to... - the model containing only intercept uses the mean of the 452 observations of numbers of birds kill by cats during a year as the predicted value. - since the MSE is $\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \dots$

Further, all the other MSE's results in a mean square error on about 300.

DOES IT FIT WITH WHAT YOU HAVE EXPECTED: This indeed fits with what was expected (??) From d), we have that lasso regression with $\lambda = 0$ equals the full model and with $\lambda \rightarrow \infty$ equals the model containing only intercept. Based on the MSE values, it is appropriate to use lasso regression on this dataset to use to get rid of coefficients that may be considered as noise when estimating the response.

Recall that lasso regression is lurt å bruke når the ikke er for mange prediktorer (i forhold til ridge som funker best med når det er mange...)

Best subset selection vs lasso regression then...

Forslag: From this tabulate, one sees that mean square errors for the intercept linear regression stands out to be the absolute worst model to predict how many birds a cat will kill during a year. This is due to the fact that this model only contains the intercept, which is determined using the mean of the 452 observations of numbers of birds kill by cats during a year as the predicted value. We expect this model to perform poorly, since the MSE for the model only containing the intercept is $\frac{1}{N} \sum_{i=1}^N (y_i - \bar{y})^2$, where \bar{y} is the mean number of birds killed by cats and the variance in the `catdat` data is... Further, all the other models have mean square errors around 300, where the full model has the highest MSE of those three. This is also as expected, since we would think that the reduced model would perform better than the full model, but not necessary significantly better.

Problem 2

a)

- (i) TRUE
- (ii) TRUE
- (iii) FALSE
- (iv) FALSE

b)

The basis functions for a cubic spline with knots at the quantiles q_1, q_2 of a variable X are given as

$$b_1(X) = X \qquad b_4(X) = (X - q_1)_+^3 \qquad (1)$$

$$b_2(X) = X^2 \qquad b_5(X) = (X - q_2)_+^3 \qquad (2)$$

$$b_3(X) = X^3 \qquad (3)$$

where b_4, b_5 are truncated power basis functions. These are defined

$$(X - q_i) = \begin{cases} (X - q_i) & \text{if } X > q_i \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

c)

```
pr.train <- catdat.train[c("daily.outdoortime", "birds")]
deg = 1:10

plot(pr.train, col = 'darkgrey', main='Polynomial regression', xlab = 'Daily outdoortime', ylab='# of birds',
co = rainbow(length(deg))

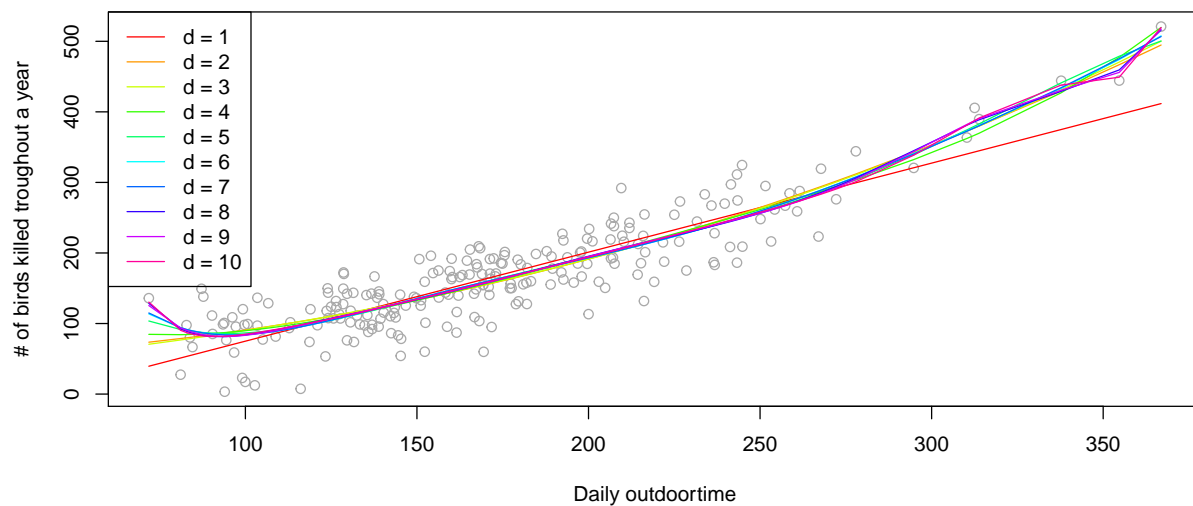
MSE.pr <- rep('numeric', length=length(deg))

for (d in deg) {
  mod <- lm(birds ~ poly(daily.outdoortime, d), pr.train)

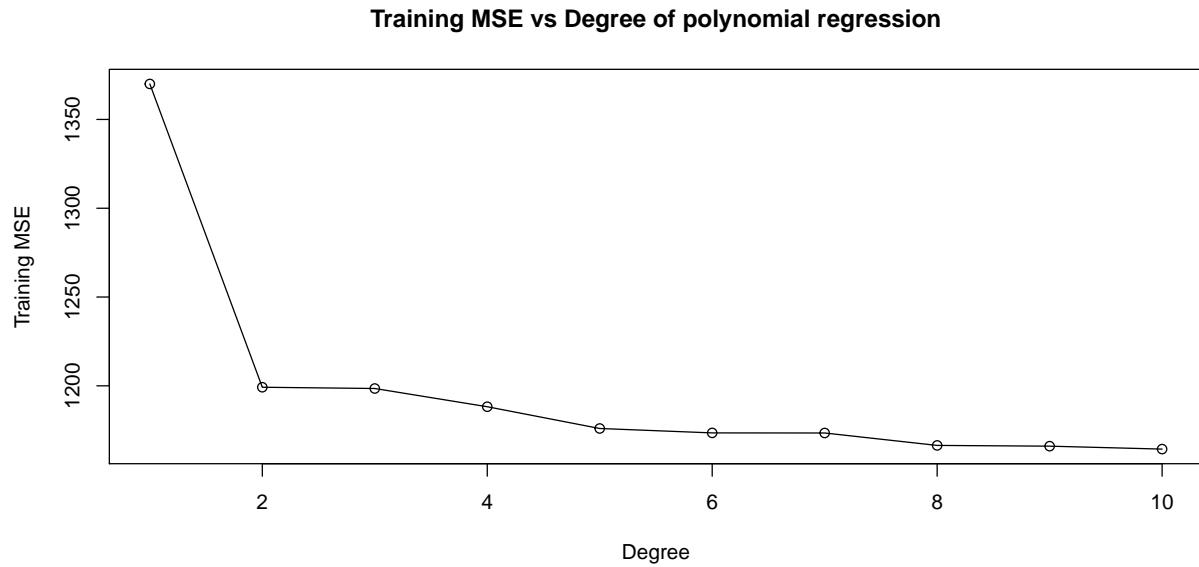
  lines(cbind(pr.train$daily.outdoortime, mod$fit)[order(pr.train$daily.outdoortime),], col=co[d])

  pred <- predict(mod, pr.train)
  MSE.pr[d] <- mean((pred-pr.train$birds)^2)
}
legend("topleft", legend = paste("d =", deg), lty=1, col=co)
```

Polynomial regression



```
plot(deg, MSE.pr, type="o", xlab="Degree", ylab="Training MSE", main="Training MSE vs Degree of polynomial r
```



Degree	Training MSE
1	1369.99652111442
2	1199.20159566985
3	1198.49876971155
4	1188.22703706041
5	1175.98070849339
6	1173.48472419999
7	1173.43712184647
8	1166.43680120411
9	1166.01882439429
10	1164.36762557602

Both the plot and the table of the training error show a decrease for an increasing number of degrees. There is an especially sharp decrease in the training MSE from $d = 1$ to $d = 2$. From the figure showing the polynomial regression for different degrees $d = 1, 2, \dots, 10$, we see that the models with $d > 1$ prove a markedly better fit for the training data points in the region `Daily_outdoortime` > 300 . This fits well with the training MSE being decidedly lower for those polynomial fits. From the figure we also can see that the polynomial fits of higher degrees seem to fit the individual data points very well, which might indicate some overfitting for those models. This is especially obvious for $d = 10$. However, since we are considering the training MSE, overfitting will lead to a decrease, so these models do have a lower training MSE than those with a lower degree.

Problem 3

a)

- (i) TRUE
- (ii) TRUE
- (iii) FALSE
- (iv) FALSE

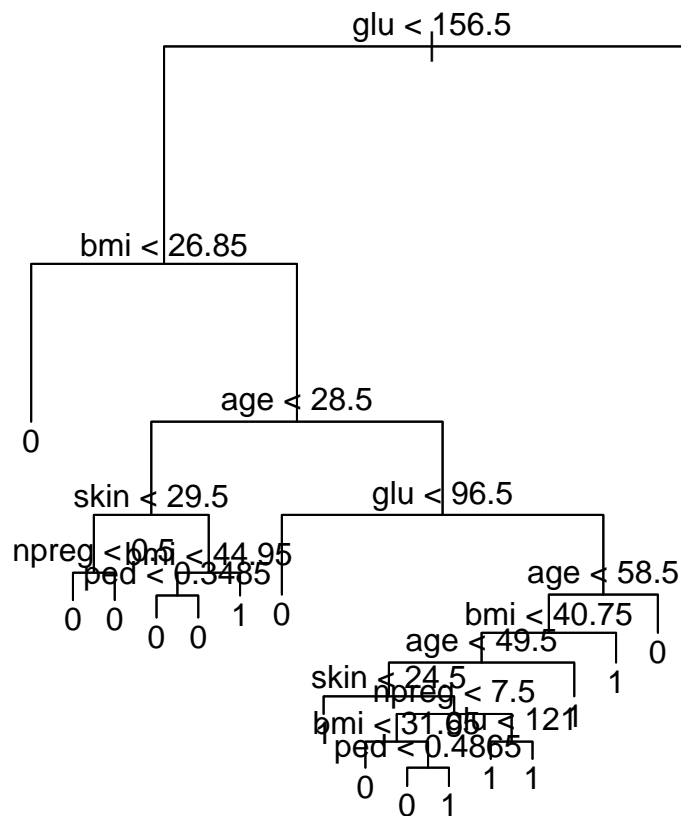
b)

The length of the branches in the tree corresponds to how much the RSS decreases along that branch, i.e. the longer the branch the more reduced RSS get. When we make a regression tree we want the RSS to be as small as possible. Hence, we want to keep the branches which are the longest. This results in that the pruned tree with three leaves will have splits at `age < 81.5` and `country: indonesia, japan, Korea`.

c)

(i)

The simple classification tree yields

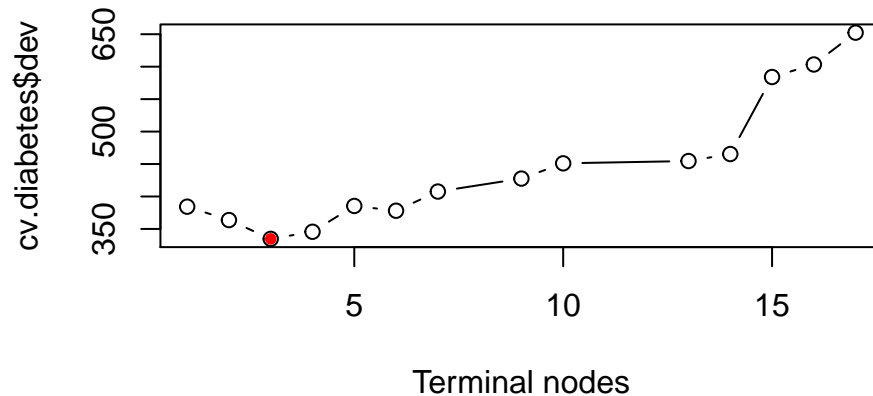


```
##          response.test
## yhat.tree    0    1
##           0 126  28
##           1   29  49
```

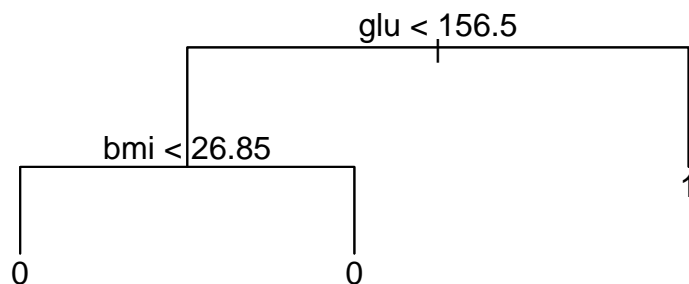
The misclassification error on the test set for this simple classification tree is 0.246.

(Vet ikke om vi vil ha med dette, da jeg spurte om det sa hun at det var den andre feilen som ble spurt om i oppgaven, men at man sikkert kunne ha med begge...)

Furthermore, cost-complexity pruning using 10-fold cross validation was applied on the training dataset resulting in that three leaves would be optimal numbers of leaves for the pruned tree, as shown below.



This resulted in the following pruned tree.



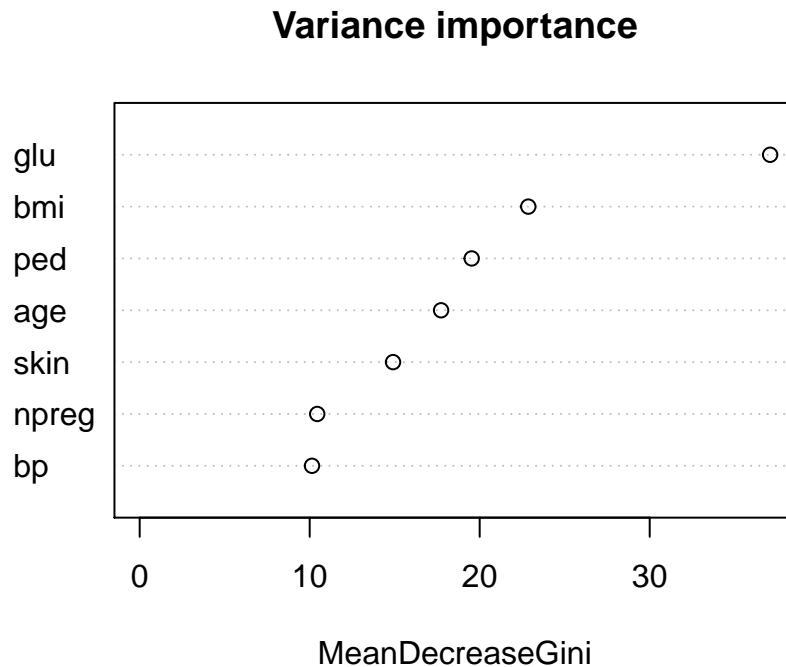
```
##           Truth
## Prediction  0   1
##           0 148 49
##           1   7 28
```

The misclassification error on the test set for this pruned tree was found to be 0.241. (usikker på om det er noe mer som burde skrives her...)

(ii) Now, the classification tree will be constructed using the random forest approach.

```
##           response.test
## yhat.rf    0    1
##           0 133  32
##           1  22  45

## [1] 0.2327586
```



When making this classification tree, the number of variables randomly sampled as candidates at each split `mtry` was chosen to be equal to \sqrt{p} , which in our case is $\sqrt{7} \approx 2.646 \approx 3$. Hence, we choose `mtry` = 3. The number of trees `ntree` = 500 is not a tuning parameter but is chosen large enough. The misclassification error on the test set when using random forest to train the model (??) was found to be 0.233.

From the variance importance plot above, we can see that `glu` and `bmi` has the highest `MeanDecreaseGini`. Hence, they are the most influential predictors of diabetes. This corresponds well to the tree we made in 3c) (i) where the two first splits were based on `glu` and `bmi`.

(usikker på type = 2, MeanDecreaseAccuracy)

Problem 4

a)

- (i) TRUE
- (ii) TRUE
- (iii) FALSE
- (iv) TRUE

b)

(i) A SVM is a more suitable approach than logistic regression for this dataset since we have $p \gg 1$ and therefore always can find a separating hyperplane, excepting exact feature ties.

Another method that could be used instead of a SVM is tree-based classification, since it also allows for a higher number of dimensions than observations. It then would be beneficial to use an ensemble method like random forest to reduce the variance.

(ii) The paper suggests using ensemble learning with SVMs to do feature selection on genomic data through elimination. That is, using bootstrapping multiple SVM models are built and then aggregated to form one final model which is used to eliminate genomic features unimportant for patient classification.

```
svm.leukemia.linear <- svm(Category~., data = d.leukemia.train, kernel = "linear", cost = 1, scale = TR
#training error
y.leukemia.train <- predict(svm.leukemia.linear, newdata = d.leukemia.train)
table.leukemia.train <- table(Predicted_train = y.leukemia.train, Truth_train = d.leukemia.train$Category)
table.leukemia.train
```

(iii)

```
##              Truth_train
## Predicted_train Non-Relapse Relapse
##      Non-Relapse      30      0
##      Relapse        0      15
```

```
#test error
y.leukemia.test <- predict(svm.leukemia.linear, newdata = d.leukemia.test)
table.leukemia.test <- table(Predicted_test = y.leukemia.test, Truth_test = d.leukemia.test$Category)
table.leukemia.test
```

```
##              Truth_test
## Predicted_test Non-Relapse Relapse
##      Non-Relapse      8      4
##      Relapse        1      2
```

The training misclassification error rate for the support vector classifier with linear kernel using $C = 1$ is 0 and the test misclassification error is 0.333. It is not surprising that the training error rate is equal to zero, since the number of covariates are much bigger than the number of observations. Meaning that we are guaranteed to find a separating hyperplane. This can result in overfitting. (usikker på om vi skal ha med den siste setningen...)

From the confusion for the test set, we can see that the most errors are due to classifying children that have been successfully treated against leukemia and later relapse as non-relapse, i.e. the most common type of error is Type-?-error. It is more important to classify all of the children who are going to relapse correctly, than the children who are not going to relapse. Hence, we do not think this classification method is successful.

```
#C = 1, gamma = 10^-2
svm.leukemia.radial1 <- svm(Category~., data = d.leukemia.train, kernel = "radial", cost = 1, gamma = 10^-2)

#training error
y.leukemia.train1 <- predict(svm.leukemia.radial1, newdata = d.leukemia.train)
table.leukemia.train1 <- table(Predicted_train = y.leukemia.train1, Truth_train = d.leukemia.train$Category)
table.leukemia.train1
```

(iv)

```
##              Truth_train
## Predicted_train Non-Relapse Relapse
##      Non-Relapse      30      0
##      Relapse         0      15
```

```
#test error
y.leukemia.test1 <- predict(svm.leukemia.radial1, newdata = d.leukemia.test)
table.leukemia.test1 <- table(Predicted_test = y.leukemia.test1, Truth_test = d.leukemia.test$Category)
table.leukemia.test1
```

```
##              Truth_test
## Predicted_test Non-Relapse Relapse
##      Non-Relapse      9      6
##      Relapse         0      0
```

The training misclassification error rate for the support vector classifier with radial kernel using $C = 1$ and $\gamma = 10^{-2}$ is 0 and the test misclassification error rate is 0.4.

```
#C = 1, gamma = 10^-5
svm.leukemia.radial2 <- svm(Category~., data = d.leukemia.train, kernel = "radial", cost = 1, gamma = 10^-5)

#training error
y.leukemia.train2 <- predict(svm.leukemia.radial2, newdata = d.leukemia.train)
table.leukemia.train2 <- table(Predicted_train = y.leukemia.train2, Truth_train = d.leukemia.train$Category)
table.leukemia.train2
```

```
##              Truth_train
## Predicted_train Non-Relapse Relapse
##      Non-Relapse      30      15
##      Relapse         0      0
```

```
#test error
y.leukemia.test2 <- predict(svm.leukemia.radial2, newdata = d.leukemia.test)
table.leukemia.test2 <- table(Predicted_test = y.leukemia.test2, Truth_test = d.leukemia.test$Category)
table.leukemia.test2
```

```
##              Truth_test
## Predicted_test Non-Relapse Relapse
##      Non-Relapse      9      6
##      Relapse         0      0
```

For the support vector classifier with radial kernel using $C = 1$ and $\gamma = 10^{-5}$, the training misclassification error rate is 0.333 and the test misclassification error rate is 0.4.

Both of the support vector classifiers with radial kernel, classify all the children in the test set as non-relapse. Since training misclassification error rate for the support vector classifier using $\gamma = 10^{-2}$ is 0, is this classifier most likely overfitted. On the other hand, the support vector classifier using $\gamma = 10^{-5}$ classify all the children in the training set as non-relapse as well. This corresponds well with the fact that a smaller γ leads to the decision boundaries to be smoother, i.e. with a smaller γ overfitting will be less likely. In our case the support vector classifier with linear kernel will be the best classifier, since the training misclassification rate is smaller and it is able to classify some of the children who relapse correctly.

c)

The polynomial kernel for the feature space with inputs X_1 and X_2 and degree $d = 2$ is

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_{i'}) &= \left(1 + \sum_{j=1}^2 x_{ij}x_{i'j}\right)^2 = (1 + x_{i1}x_{i'1} + x_{i2}x_{i'2})^2 \\ &= 1 + 2x_{i1}x_{i'1} + 2x_{i2}x_{i'2} + (x_{i1}x_{i'1})^2 + 2x_{i1}x_{i'1}x_{i2}x_{i'2} + (x_{i2}x_{i'2})^2. \end{aligned}$$

This can be written as

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_{i'}) &= h_1(x)h_1(x') + h_2(x)h_2(x') + h_3(x)h_3(x') + h_4(x)h_4(x') + h_5(x)h_5(x') + h_6(x)h_6(x') \\ &= \sum_{j=1}^6 h_j(x)h_j(x') = \langle h(x), h(x') \rangle, \end{aligned}$$

where

$$\begin{aligned} h_1(x)h_1(x') &= 1 \implies h_1(x) = 1 \\ h_2(x)h_2(x') &= 2x_{i1}x_{i'1} \implies h_2(x) = \sqrt{2}x_{i1} \\ h_3(x)h_3(x') &= 2x_{i2}x_{i'2} \implies h_3(x) = \sqrt{2}x_{i2} \\ h_4(x)h_4(x') &= x_{i1}^2x_{i'1}^2 \implies h_4(x) = x_{i1}^2 \\ h_5(x)h_5(x') &= 2x_{i1}x_{i2}x_{i'1}x_{i'2} \implies h_5(x) = \sqrt{2}x_{i1}x_{i2} \\ h_6(x)h_6(x') &= x_{i2}^2x_{i'2}^2 \implies h_6(x) = x_{i2}^2. \end{aligned}$$

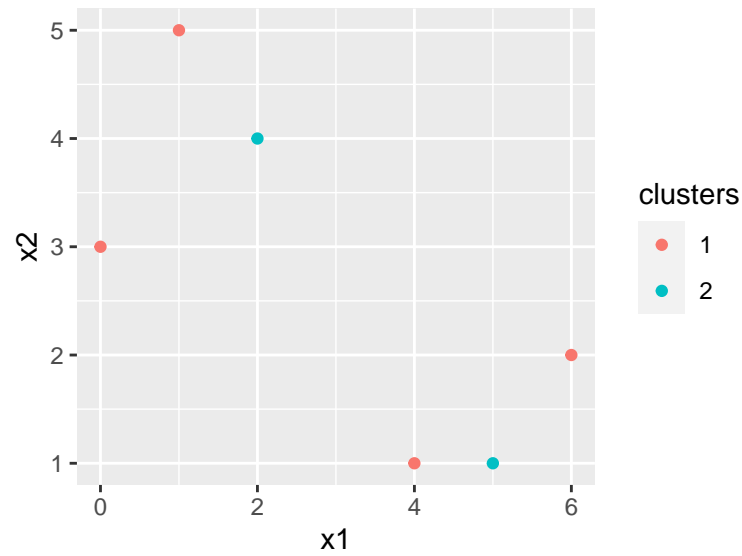
Problem 5

a)

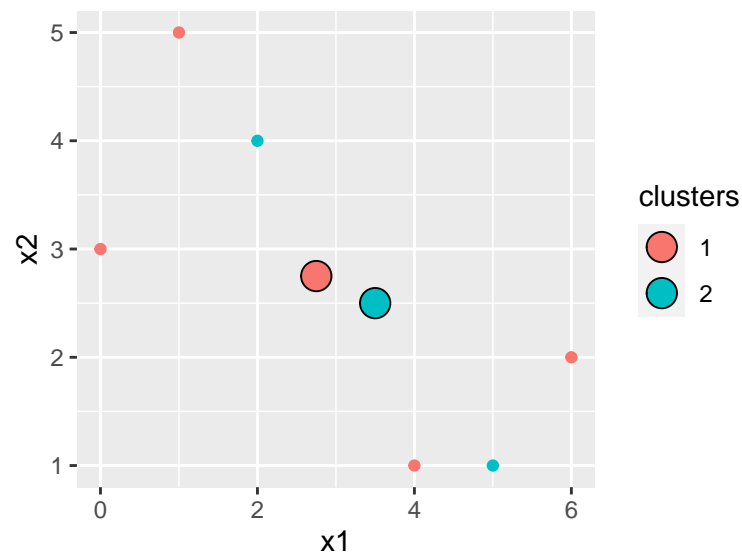
- (i) TRUE
- (ii) FALSE
- (iii) FALSE
- (iv) FALSE

b)

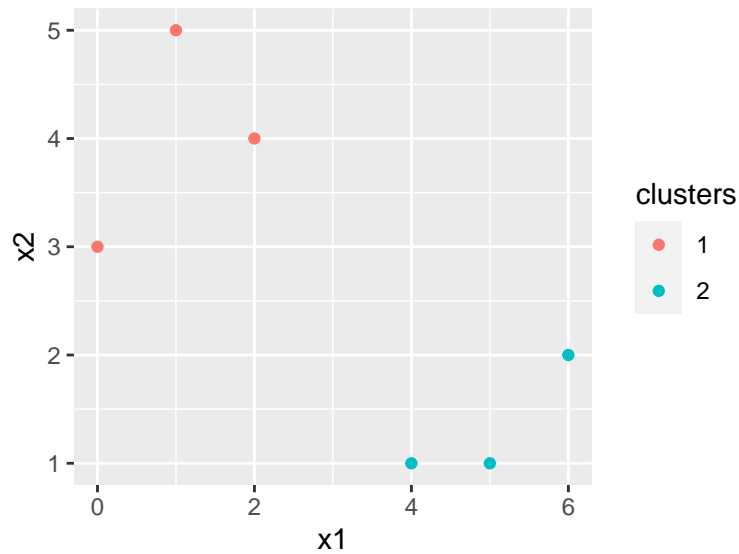
- (i) First, we randomly assign a cluster to each observation and obtain the following plot.



(ii) Secondly, we calculate the centroids for each cluster and add them to the plot.



(iii) Lastly, we assign the observations new clusters based on which centroid is closest in euclidain distance and obtain the following plot.



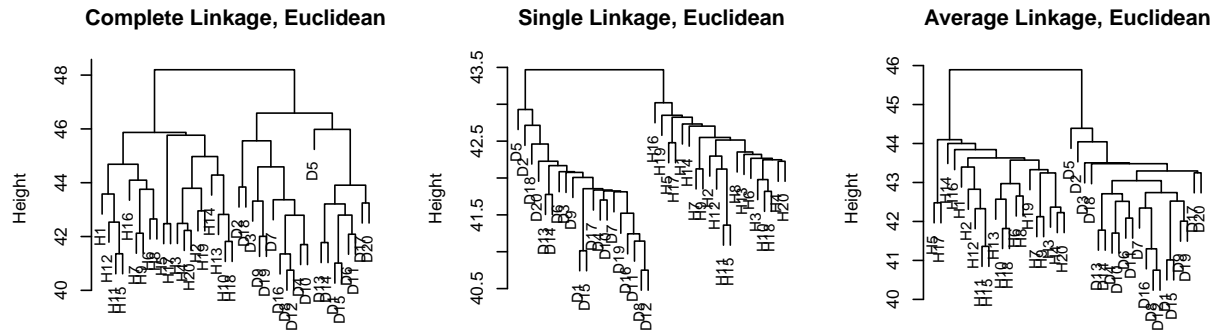
c)

The following code shows how hierarchical clustering with complete, single and average linkage was implemented both using Euclidean distance and correlation-based distance.

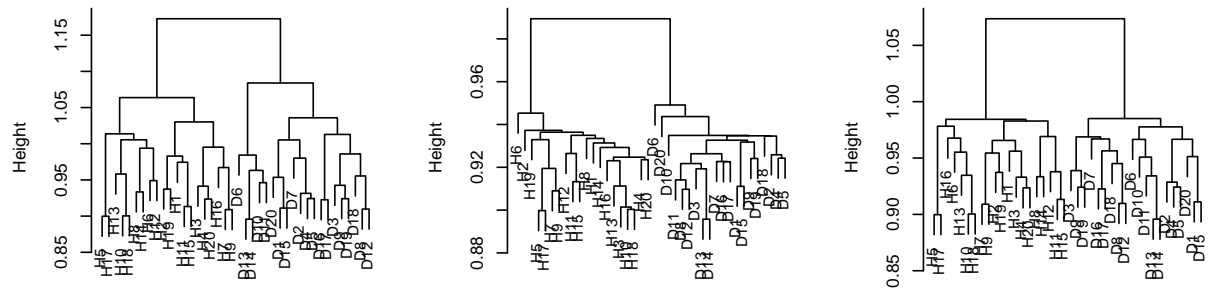
```
#hierarchical clustering Euclidean distance
CE_hclust = hclust(dist(GeneData), method = "complete")
SE_hclust = hclust(dist(GeneData), method = "single")
AE_hclust = hclust(dist(GeneData), method = "average")

#hierarchical clustering using correlation-based distance
correlation_dist = as.dist(1-cor(t(GeneData)))
CC_hclust = hclust(correlation_dist, method = "complete")
SC_hclust = hclust(correlation_dist, method = "single")
AC_hclust = hclust(correlation_dist, method = "average")
```

The following plot shows dendrograms for the investigated choices for linkage and distance.



complete Linkage, Correlation-based distSingle Linkage, Correlation-based distAverage Linkage, Correlation-based dist



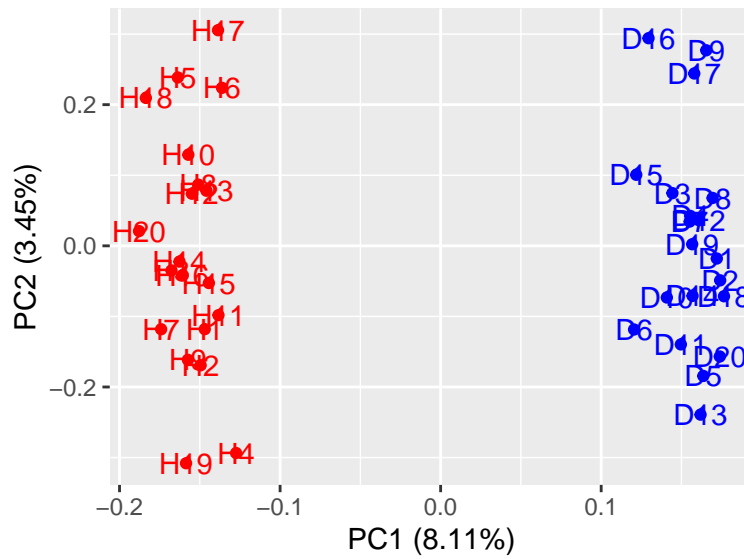
d) Based on the dendrograms from c), the tissues were clustered into two groups the following way.

```
#cut the clusters in two groups
CE_2groups = cutree(CE_hclust, 2)
SE_2groups = cutree(SE_hclust, 2)
AE_2groups = cutree(AE_hclust, 2)
CC_2groups = cutree(CC_hclust, 2)
SC_2groups = cutree(SC_hclust, 2)
AC_2groups = cutree(AC_hclust, 2)
```

For all the six combinations of linkages and distance measures, the tissues were clustered into groups that coincide with whether the tissues came from a healthy patient group or the tissues came from a diseased patient group (mulgens litt rar setning). Hence, all the hierarchical clustering methods performed equally good and there is no hierarchical clustering method that stands out as the best.

%legge inn kode som presenterer the two clusters in each case in a fifi way.

e)



(i)

(ii) The first 5 PCs explain 21.1% of the variance.

f)

From the plot of the two first principal components, we see that the groups differ a lot along the first principal component. To find the genes that vary the most across the two groups we can therefore look at the genes that contribute the most to the first principal component.

The five genes that have the highest variance across the two groups are thus

##	G502	G589	G565	G590	G600
##	0.09485044	0.09449766	0.09183823	0.09173169	0.09167322