Compulsory exercise 2: Group 27

TMA4268 Statistical Learning V2021

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

16 april, 2021

Problem 1

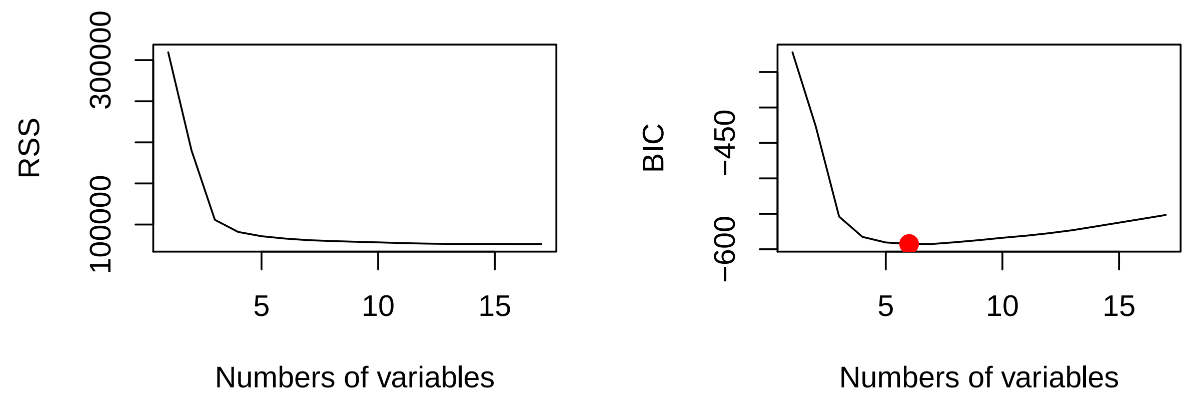
a) Multiple choice 1

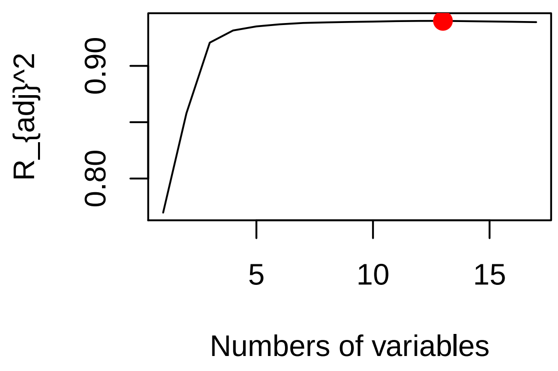
(i) TRUE
(ii) TRUE
(iii) TRUE
(iv) FALSE

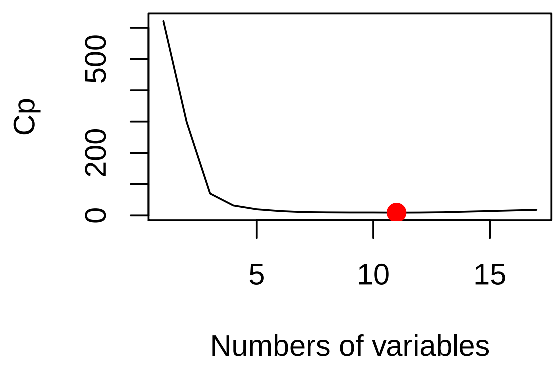
b) Best subset selection

For this task, we will perform best subset selection to obtain a reduced linear regression model based on the `catdat` dataset.

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

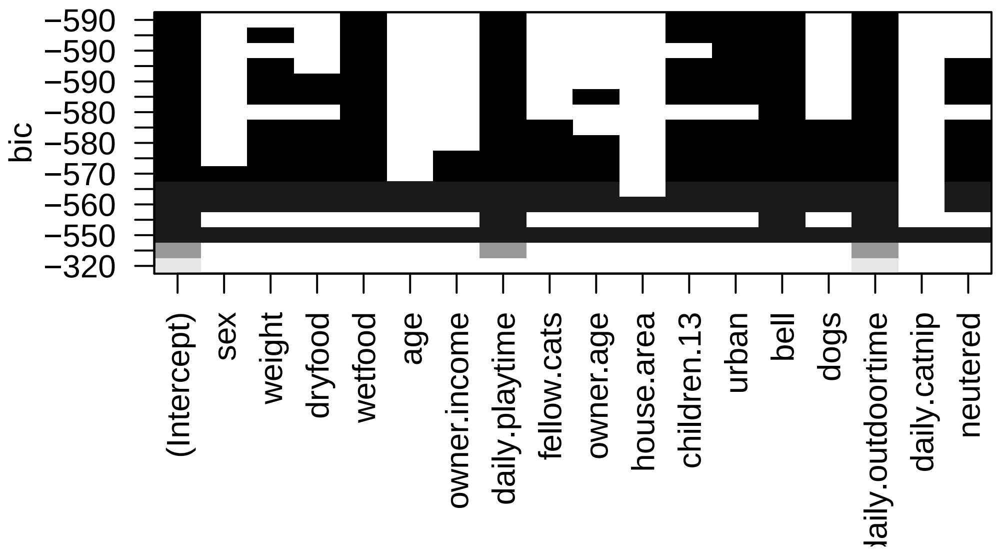






The first plot shows that the RSS decreases for increasing number of predictors, which is expected since RSS always will increase when adding more predictors to the model. When wanting to minimize the Bayesian Information Criterion (BIC), one sees that the subset containing six predictors turns out as 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.

The below 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.

```r
#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`.

```r
#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)
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)

#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 shrunken 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 `weight, dryfood, wetfood, owner.income, daily.playtime, owner.age, house.area children.13, urban, bell, dogs, daily.outdoortime, neutered` and the omitted once are `sex, age, fellow.cats, daily.catnip`.

The test mean square error for the lasso regression model was found to be `298.42`.

d) The shrinking parameter λ

For increased values of λ the slopes gets smaller and smaller until they reach zero as $\lambda \to \infty$. 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) Intercept and full model

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

```r
#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 `4914.07` and `301.92`. Both of these values are greater than the ones 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) Comparing MSE for the models

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 the mean square error 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 fits what we have seen in earlier tasks, for example the plots in task b. This model only contains the intercept which is determined using the mean of the 452 observations of numbers of birds killed by cats during a year as the predicted value. Therefore we would expect this model to preform poorly with a dataset with high variance and we would expect better performance from models that include some or all of the covariates. As expected the mean square errors for these other models are significantly smaller at around 300, where the full model has the highest MSE of those three. This is also as expected, we would think that both best subset selection and lasso regression would perform somewhat better than the full model, since both methods resulted in reduces models. However, we would not necessarily expect significantly better performance.

Problem 2

a) Multiple choice 2

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

b) Basis functions for cubic splines

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

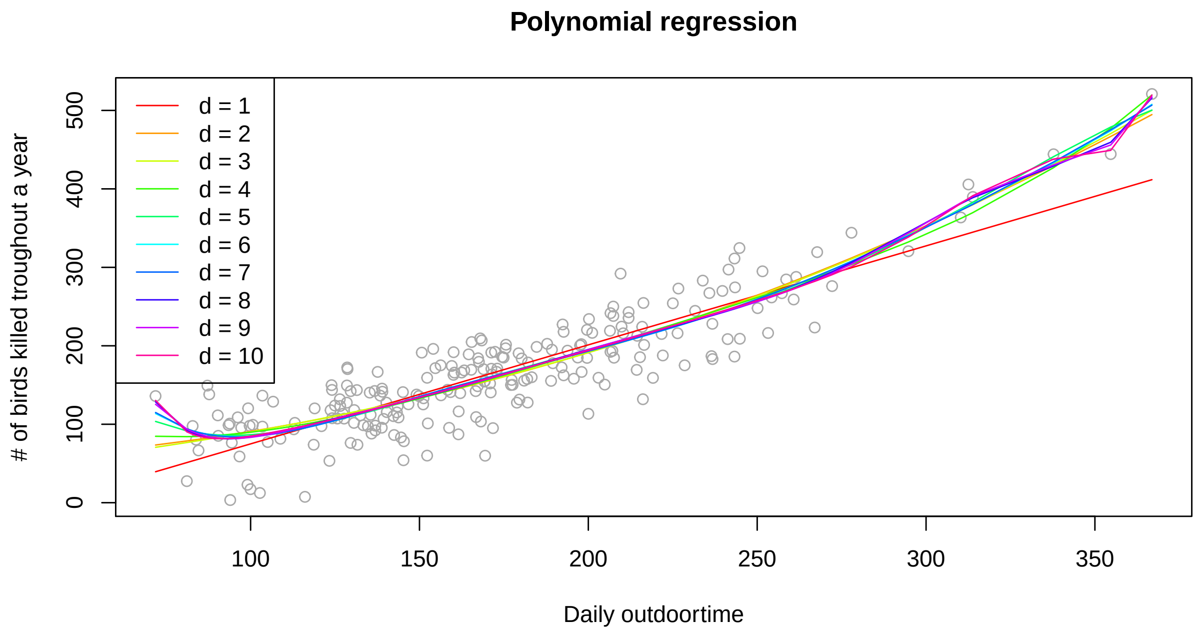
$$\begin{aligned} b_1(X) &= X & \quad b_4(X) &= (X - q_1)^3_+ \\ b_2(X) &= X^2 & \quad b_5(X) &= (X - q_2)^3_+ \\ b_3(X) &= X^3 \end{aligned}$$

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

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

c) Polynomial regression model

(i) The plot below shows polynomial regression models for different degrees d fitted for the `catdat` data using only `daily.outdoortime` as a covariate.



(ii) The training MSE for the models in (i) are found and shown in the table below.

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

The table of the training MSE 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 $d = 1, 2, ..., 10$, 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. Therefore, these models do have a lower training MSE than those with a lower degree, even though they might have higher test MSE.

Problem 3

a) Multiple choice 3

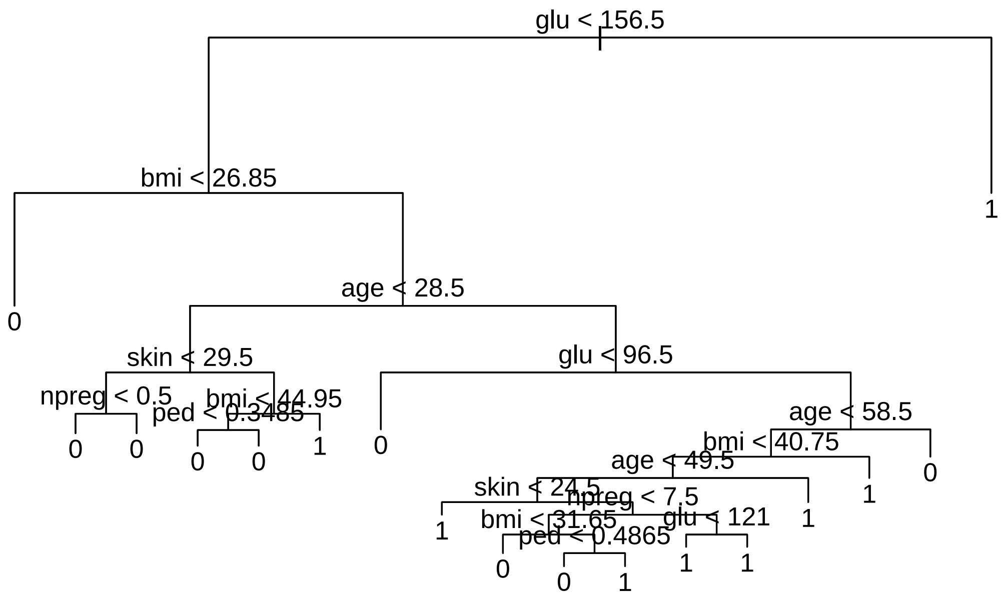
(i) TRUE
(ii) TRUE
(iii) FALSE
(iv) FALSE

b) Pruning the tree down to three leaves

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. For the tree in a, this results in that the pruned tree with three leaves will have splits at `age < 81.5` and `country: indonesia, japan, Korea`.

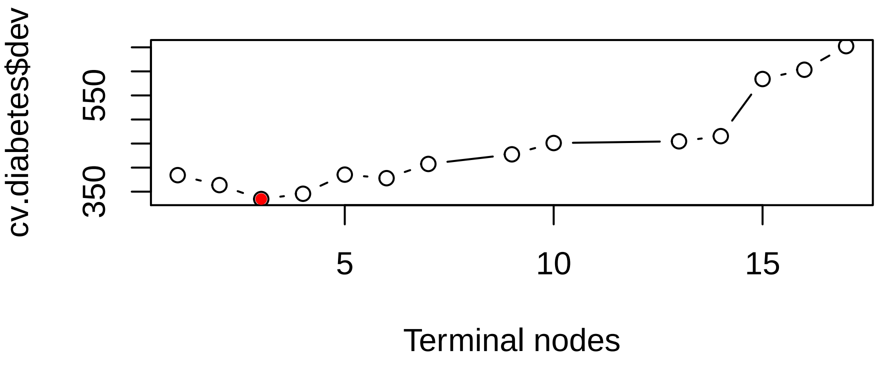
c) Tree-based methods to predict presence of diabetes

(i) The simple classification tree yields

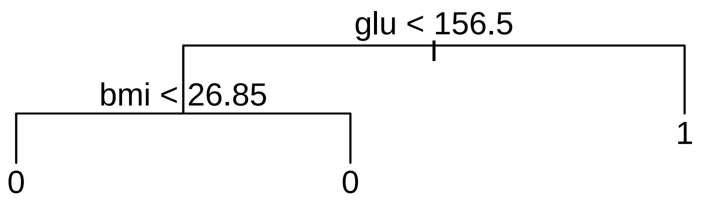


```
##             Truth
## Prediction    0   1
##            0 126  28
##            1  29  49
```

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

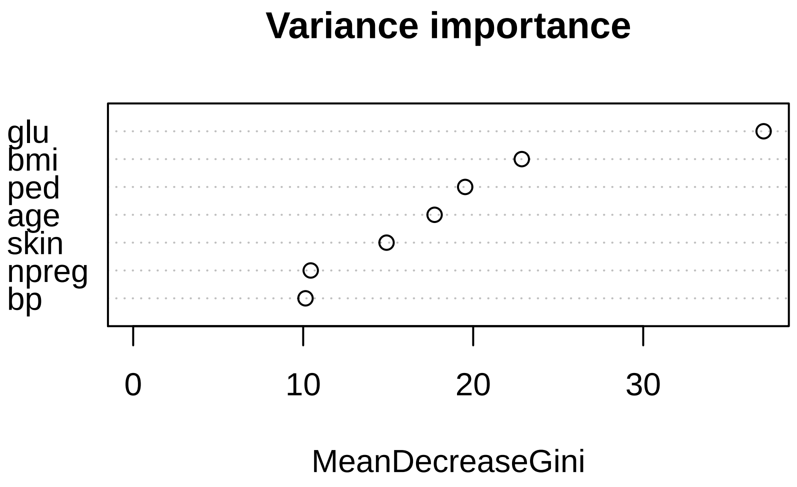


```
##           Truth
## Prediction   0   1
##          0 148  49
##          1   7  28
```

The misclassification error on the test set for this pruned tree was found to be `0.241`. Compared to the simple classification tree we see that the pruned tree have slightly lower misclassification error and classify more of the women without diabetes correctly.

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

```
##           Truth
## Prediction   0   1
##          0 133  32
##          1  22  45
```



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 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`.

Problem 4

a) Multiple choice 4

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

b) Support vector machine

(i) A SVM is a more suitable approach than logistic regression for this dataset since we have $p \gg n$ 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 ensembel 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 elimation. 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.

(iii) Support vector machine with a linear kernel will now be used to predict whether a child that has been successfully treated against leukemia later relapsed or not.

```
svm.leukemia.linear <- svm(Category~., data = d.leukemia.train, kernel = "linear",
                          cost = 1, scale = TRUE)
```

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

##                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 then the number of observations. Meaning that we are guaranteed to find a separating hyperplane.

From the confusion table 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 false negative. 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.

(iv) Now, the analysis will be repeated twice using the radial kernel

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

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

```
##                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`.

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

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

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

For the support vector classifier with radial kernal 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) Inner product representation of the polynomial kernel

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

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

This can be written in terms of an inner product as

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

where

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

So, $h(X) = (1, \sqrt{2}X_1, \sqrt{2}X_2, X_1^2, \sqrt{2}X_1X_2, X_2^2)$.

Problem 5

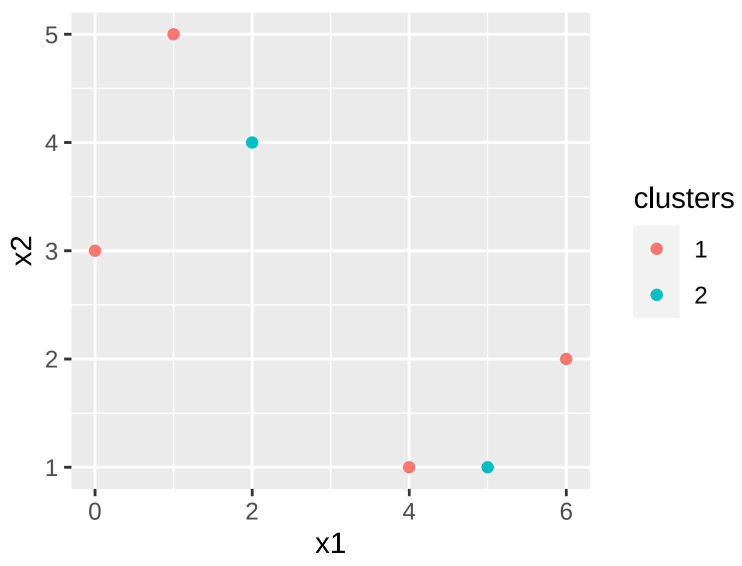
a) Multiple choice 5

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

b) K-means clustering

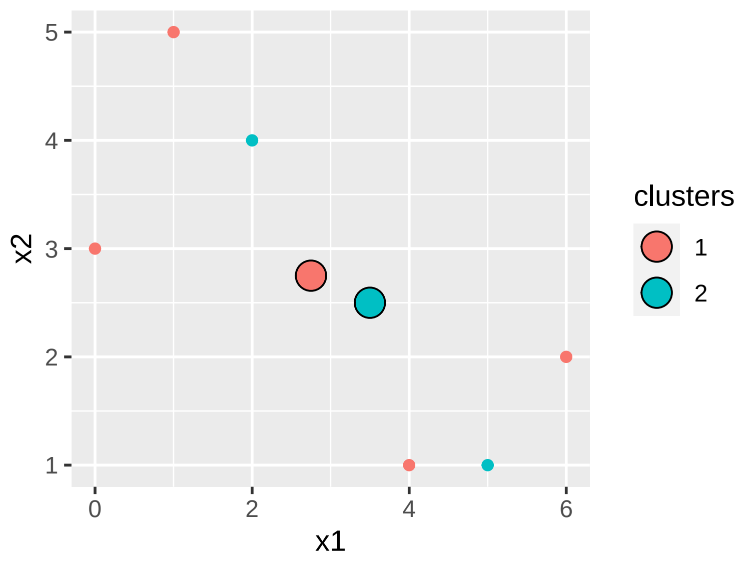
(i) First, we randomly assign a cluster to each observation and obtain the following plot.

```r
set.seed(1)
x1 = c(1, 2, 0, 4, 5, 6)
x2 = c(5, 4, 3, 1, 1, 2)
n = 6
K = 2

clusters = sample(1:K, n, replace = TRUE)
df = data.frame(x1,x2,clusters = as.factor(clusters))
ggplot(df, aes(x1,x2, color=clusters)) +geom_point()
```



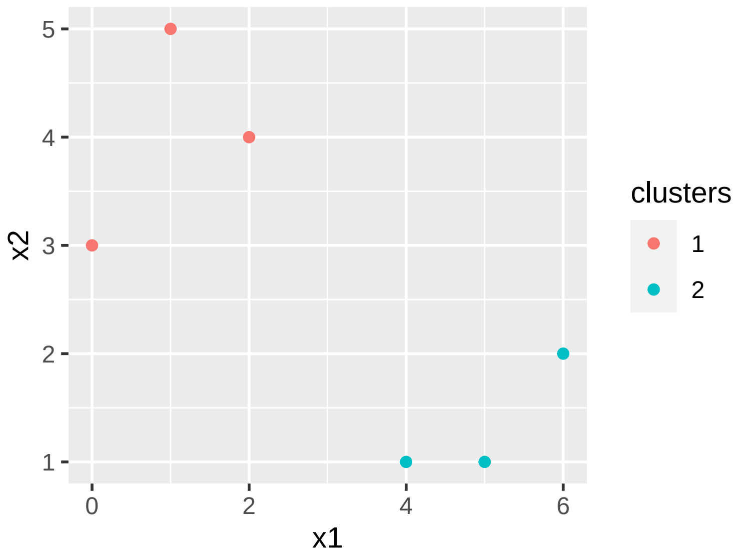
(ii) Secondly, we calculate the centroids for each cluster and add them to the plot.

```
centroid.cluster1_x1 = mean(df$x1[df$clusters == 1])
centroid.cluster1_x2 = mean(df$x2[df$clusters == 1])
centroid.cluster2_x1 = mean(df$x1[df$clusters == 2])
centroid.cluster2_x2 = mean(df$x2[df$clusters == 2])

df.centroid = data.frame(x1 = c(centroid.cluster1_x1, centroid.cluster2_x1),
                         x2 = c(centroid.cluster1_x2, centroid.cluster2_x2),
                         clusters = as.factor(c(1,2)))

ggplot() + geom_point(data = df, aes(x1,x2, color=clusters)) +
  geom_point(data = df.centroid, aes(x1,x2, fill = clusters),
             size = 5, shape = 21, color = "black")
```



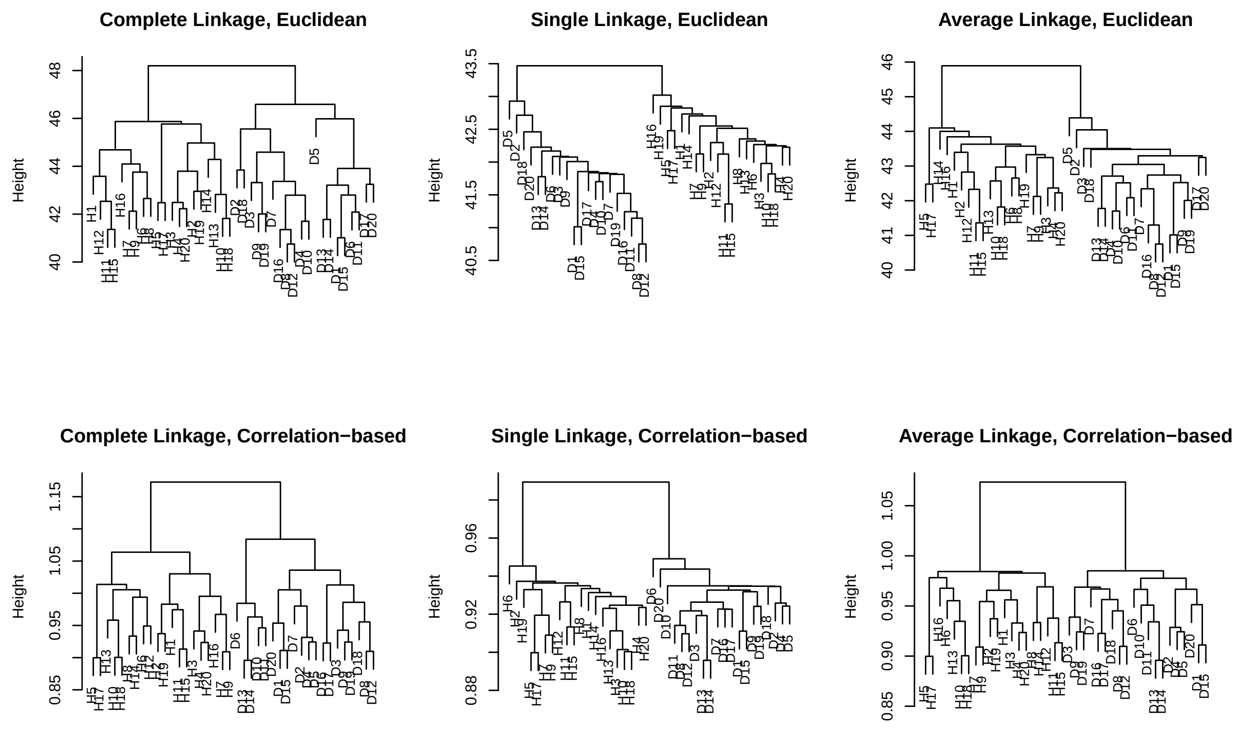
(iii) Lastly, we assign the observations new clusters based on which centroid is closest in euclidean distance and obtain the following plot.

```r
for (i in 1:n){
  euclid.cluster1 = (df$x1[i] - df.centroid$x1[1])^2 + (df$x2[i] - df.centroid$x2[1])^2
  euclid.cluster2 = (df$x1[i] - df.centroid$x1[2])^2 + (df$x2[i] - df.centroid$x2[2])^2
  if (euclid.cluster1 < euclid.cluster2){
    df$clusters[i] = 1
  }
  else{
    df$clusters[i] = 2
  }
}

ggplot() + geom_point(data = df, aes(x1,x2, color=clusters))
```



c) Hierarchical clustering

The following plot shows dendrograms obtained from hierarchical clustering with complete, single and average linkage for both Euclidean distance and correlation-based distance.



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)
```

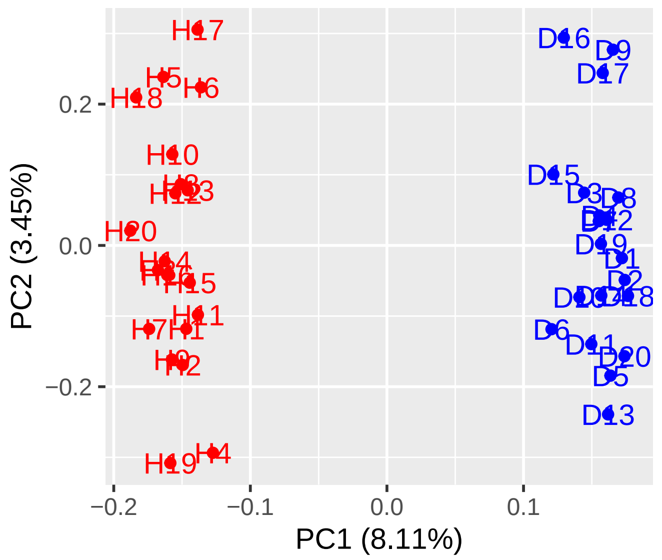
```
##            TrueTissue
## CE_2groups  D  H
##          D 20  0
##          H  0 20
```

For all the six combinations of linkages and distance measures, the tissues were clustered into two groups that coincide with whether the tissues came from a healthy patient or a diseased patient. All methods were able to correctly cluster the two groups of tissues from the patients. This is shown in the table for the hierarchical clustering using complete linkages and euclidean distance, and was also the case for all the other methods, which can be seen in the dendrograms above. Hence, all the hierarchical clustering methods performed equally good and there is no hierarchical clustering method that stands out as the best.

e) Principal components analysis

(i) The samples are plotted based on the first and second principal component.

```
pca.GeneData <- prcomp(GeneData,scale = TRUE)
HvD <- c(rep(1,20),rep(2,20))
autoplot(pca.GeneData, data = GeneData,col=c("red","blue")[HvD],label=TRUE)
```



(ii) Now, we want to determine how much of the variance in the dataset the first five principal components explain.

```
pca.var <- pca.GeneData$sdev^2
pve <- pca.var/sum(pca.var)
explained_var <- sum(pve[1:5])
```

The first 5 PCs explain 21.1% of the variance.

f) Principal component analysis

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.

```
PC1.loadings <- pca.GeneData$rotation[,1]
high_var <- sort(abs(PC1.loadings),decreasing = TRUE)
```

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
```