Compulsory Exercise 2 – Solutions

TMA4268 Statistical Learning V2020

Martina Hall, Michail Spitieris, Stefanie Muff, Department of Mathematical Sciences, NTNU

Hand out date: March 16, 2020

Last changes: 25.03.2020

The submission deadline is Thursday, 2. April 2020, 23:59h using Blackboard

Problem 1 (10p)

a) (2p)

Solution

We want to minize the function $\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2$, with respect to β. Using matrix notation

$$\begin{aligned}\sum_{i=1}^{n}(y_i - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 &= (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta \\ &= y^Ty - \beta^TX^Ty - y^TX\beta + \beta^Tx^TX\beta + \lambda\beta^T\beta \\ &= y^Ty - \beta^TX^Ty - \beta^TX^Ty + \beta^TX^TX\beta + \beta^T\lambda I\beta \\ &= y^Ty - 2\beta^TX^Ty + \beta^T(X^TX + \lambda I)\beta\end{aligned}$$

We differentiate with repect to the vector β and we get

$$-2X^Ty + 2(X^TX + \lambda I)\beta = 0,$$

and finally

$$\hat{\beta}_{Ridge} = (X^TX + \lambda I)^{-1}X^Ty$$

b) (2p)

Find the expected value and the variance-covariance matrix of $\hat{\beta}_{Ridge}$ (1P each).

Solution

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

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

c) (2P) - Multiple choice

Which of the following statements are true, which false?

(i) For a problem that $p > n$ we can use forward selection but not backward.

(ii) Ridge Regression exploits the trade-off between variance and bias in the MSE, but Lasso doesn't.

(iii) Ridge and Lasso perform variable selection.

(iv) As the tuning parameter λ in ridge regression increases, the variance decreases and the bias increases.

Solution TRUE - FALSE - FALSE - TRUE

(i) We cannot include more than n variables, thus backward selection does not work. (ii) Both ridge and Lasso exploit the trade-off (iii) No, but with Lasso we could (iv) The larger λ, the more we punish large β. This reduces variance but increases bias.

d) (2p)

For the remaining regression tasks we will use the `College` data from the `ISLR` package, which consists of 18 variables and 777 observations (for more details see `?College`). First we split into training and testing sets (50% of the data in each) using the following code:

```
library(ISLR)
set.seed(1)
train.ind = sample(1:nrow(College), 0.5 * nrow(College))
college.train = College[train.ind, ]
college.test = College[-train.ind, ]
```

```
str(College)
```

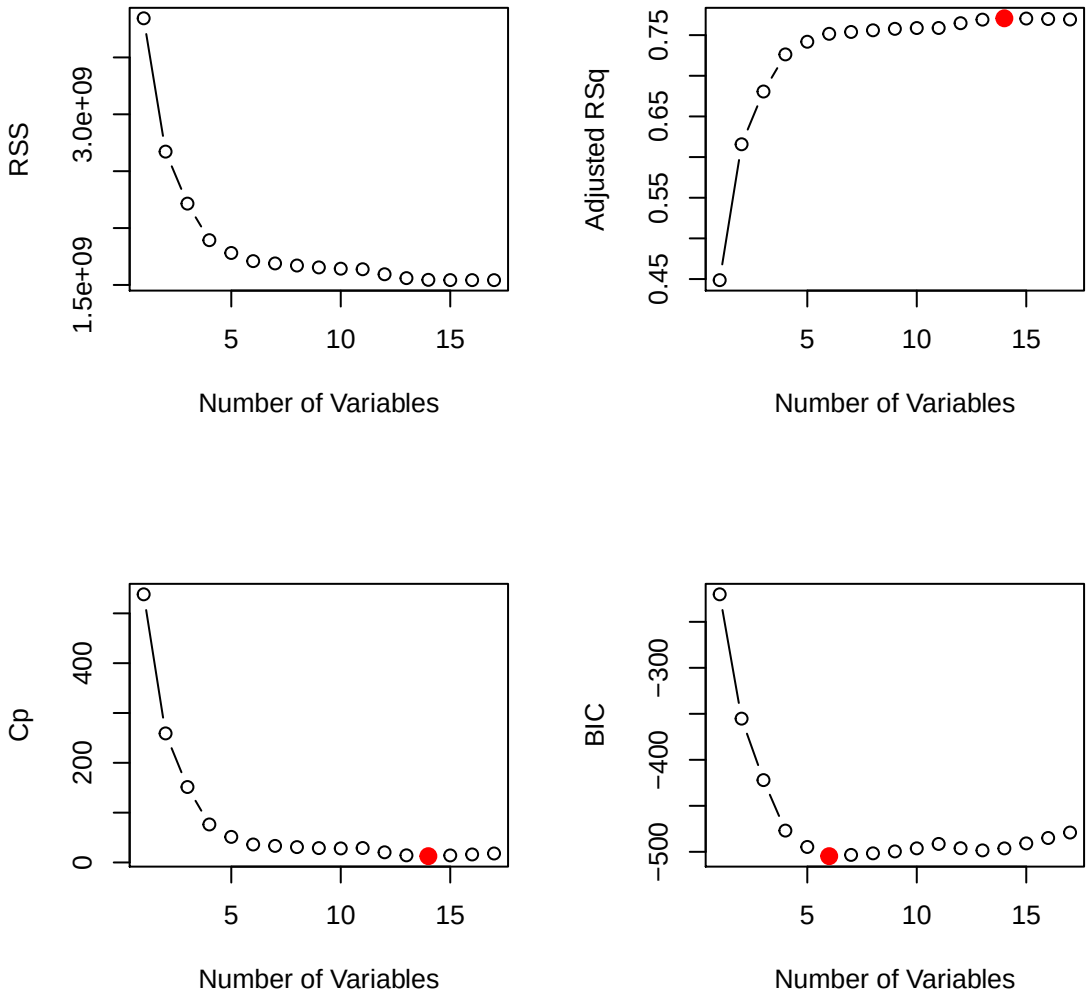
Using the out-of-state tuition (variable `Outstate`) as response, apply forward selection in order to identify a satisfactory model that uses only a subset of all the variables in the dataset in order to predict the response. Choose a model according to one of the criteria that you know and briefly say why. Write down the model and report the MSE on the test set.

Solution

```
library(leaps)
for.reg = regsubsets(Outstate ~ ., data = college.train, nvmax = 17, method = "forward")
for.reg.summary = summary(for.reg)
```

```
# Plot RSS, Adjusted R^2, C_p and BIC
par(mfrow = c(2, 2))
plot(for.reg.summary$rss, xlab = "Number of Variables", ylab = "RSS", type = "b")
plot(for.reg.summary$adjr2, xlab = "Number of Variables", ylab = "Adjusted RSq",
    type = "b")
bsm_best_adjr2 = which.max(for.reg.summary$adjr2)
points(bsm_best_adjr2, for.reg.summary$adjr2[bsm_best_adjr2], col = "red", cex = 2,
    pch = 20)
plot(for.reg.summary$cp, xlab = "Number of Variables", ylab = "Cp", type = "b")
bsm_best_cp = which.min(for.reg.summary$cp)
points(bsm_best_cp, for.reg.summary$cp[bsm_best_cp], col = "red", cex = 2, pch = 20)
```

```
bsm_best_bic = which.min(for.reg.summary$bic)
plot(for.reg.summary$bic, xlab = "Number of Variables", ylab = "BIC", type = "b")
points(bsm_best_bic, for.reg.summary$bic[bsm_best_bic], col = "red", cex = 2,
    pch = 20)
```




```
par(mfrow = c(1, 1))
```

According to BIC forward selection suggests a model with 6 predictors, although other criteria tend to favour larger models. In fact, you could see that for 7 or more predictors the test error is slightly smaller than for the six-variable model that we choose here (and if you justify why you go on with a larger model, that is ok). The model with 6 predictors contains the following variables

```
coef(for.reg, 6)
```

```
##   (Intercept)    PrivateYes    Room.Board      Terminal   perc.alumni
## -4726.8810613  2717.7019276     1.1032433    36.9990286    59.0863753
##        Expend     Grad.Rate
##     0.1930814    33.8303314
```

and is thus given as follows:

$$Y = \beta_0 + \beta_1 \cdot \text{PrivateYes} + \beta_2 \cdot \text{Room.Board} + \beta_3 \cdot \text{Terminal} + \beta_4 \cdot \text{perc.alumni} + \beta_5 \cdot \text{Expend} + \beta_6 \cdot \text{Grad.Rate} + \epsilon \ ,$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

The MSE is obtained as follows:

```
predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
    xvars = names(coefi)
    mat[, xvars] %*% coefi
}

mse.for.reg = mean((college.test$Outstate - predict.regsubsets(for.reg, college.test,
    id = 6))^2)
mse.for.reg
```

```
## [1] 3844857
```

Note: Different solutions will be graded as correct (as long as the justification is ok).

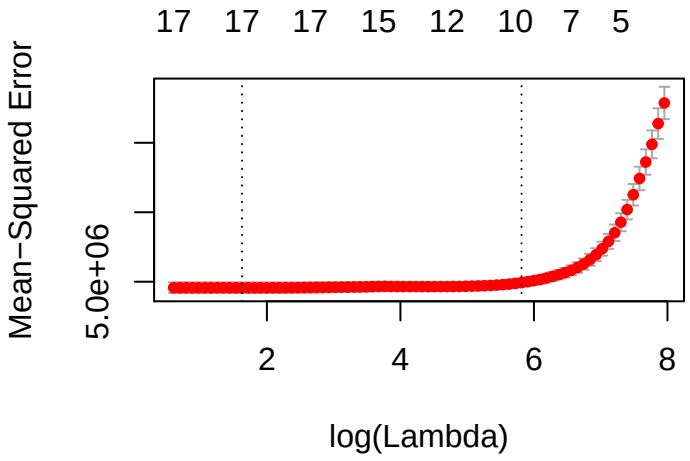
e) (2p)

Now do model selection using the same dataset as in (d) using the Lasso method. How did you select the tuning parameter λ? Report the set of variables that was selected and the MSE on the test set.

Solution

Lasso Regression using cross-validation to find λ:

```
library(glmnet)
x.train = model.matrix(Outstate ~ ., data = college.train)
y.train = college.train$Outstate
set.seed(4268)
cv.lasso = cv.glmnet(x.train, y.train, alpha = 1)
plot(cv.lasso)
```




```
(one.se.lamdda.lasso = cv.lasso$lambda.1se)
```

```
## [1] 335.0986
```

```
(one.se.lamdda.lasso = cv.lasso$lambda.min)
```

```
## [1] 5.093201
```

MSE

```r
x.test = model.matrix(Outstate ~ ., data = college.test)
y.test = college.test$Outstate
# In the following, we could also use cv.lasso$lambda.1se
lasso = glmnet(x.train, y.train, alpha = 1, lambda = cv.lasso$lambda.min)
mse.lasso = mean((y.test - predict(lasso, newx = x.test))^2)
mse.lasso
```

```
## [1] 3717510
```

All the variables that are in the final model have a non-zero coefficient:

```r
predict(lasso, type = "coefficients")[1:19, ]
```

```
##   (Intercept)   (Intercept)    PrivateYes          Apps        Accept
## -1.131804e+03  0.000000e+00  2.202298e+03 -3.325007e-01  7.781907e-01
##        Enroll     Top10perc     Top25perc   F.Undergrad   P.Undergrad
## -5.755156e-01  5.272733e+01 -1.919842e+01 -4.345397e-02 -5.937967e-02
##    Room.Board         Books      Personal           PhD      Terminal
##  1.087612e+00 -9.331645e-01 -3.003483e-01  3.242927e+00  3.107177e+01
##     S.F.Ratio   perc.alumni        Expend     Grad.Rate
## -6.857566e+01  4.689425e+01  1.476347e-01  2.463614e+01
```

Problem 2 (9p)

a) (2p) - Multiple choice

Which of the following statements are true, which false?

(i) A regression spline of order 3 with 4 knots has 8 basis functions.
(ii) A regression spline with polynomials of degree $M-1$ has continous derivatives up to order $M-2$, but not at the knots.
(iii) A natural cubic spline is linear beyond the boundary knots.
(iv) A smoothing spline is (a shrunken version of) a natural cubic spline with knots at the values of all data points x_i for $i = 1, \ldots, n$.

Solution

FALSE - FALSE - TRUE - TRUE

(i) We need 7 basis functions, but estimate 8 parameters (including the intercept, which is not a basis function). (ii) The derivative at the knots is also continuous. (iii) and (iv) are true.

b) (2p)

Write down the basis functions for a cubic spline with knots at the quartiles q_1, q_2, q_3 of variable X.

Solution

The quartiles are the 3 cut points that divide the data into 4 intervals of same length. We have a spline with $M = 4$ and $k = 3$, which results the following 6 basis functions

$$X, X^2, X^3, (X - q_1)^3_+, (X - q_2)^3_+, (X - q_3)^3_+,$$

where q_1, q_2 and q_3 are the quantile cut points.

c) (2p)

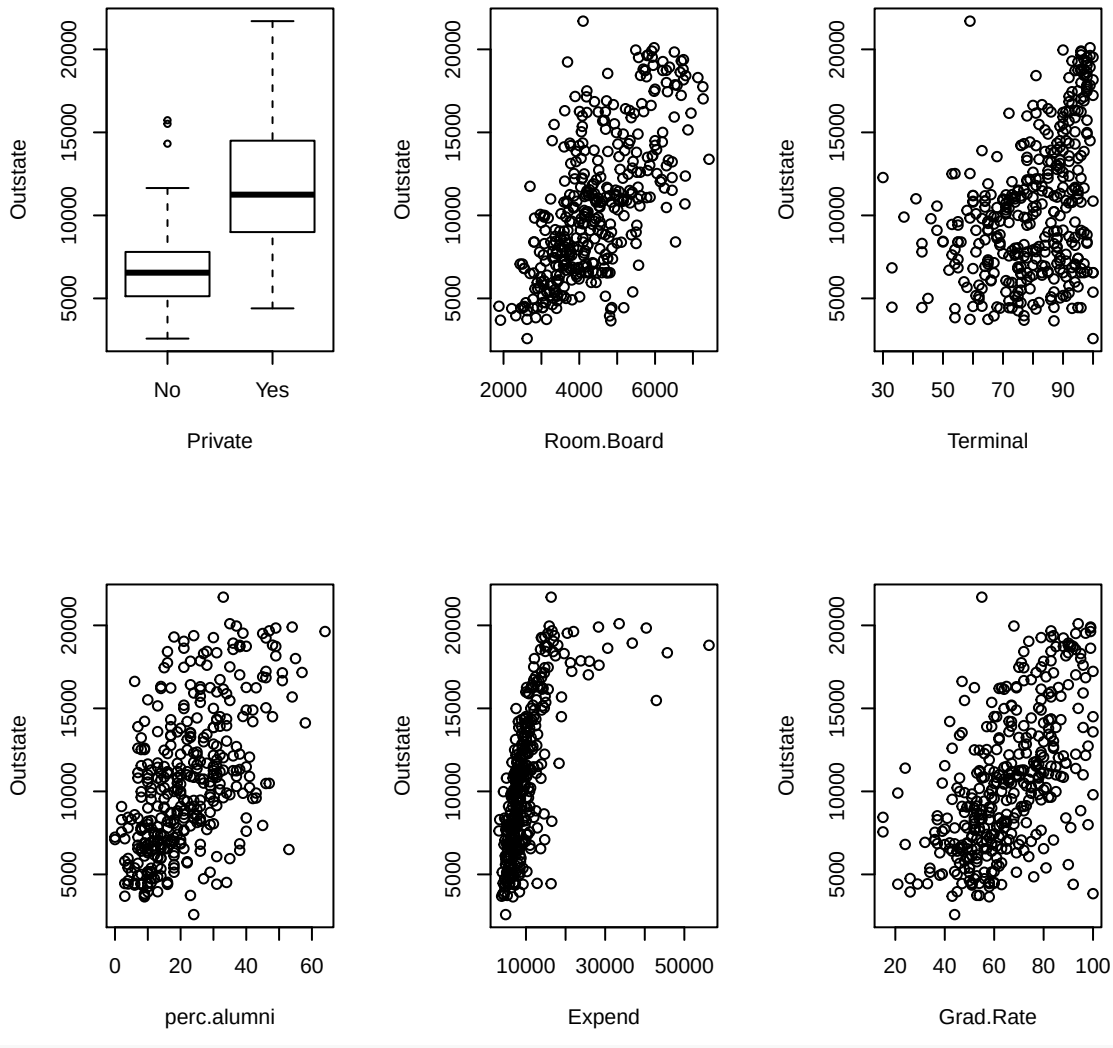
We continue with using the `College` dataset that we used in problem 1. Investigate the relationships between `Outstate` and the following 6 predictors (using the training dataset `college.train`):

```
## [1] "Private"     "Room.Board"  "Terminal"    "perc.alumni" "Expend"
## [6] "Grad.Rate"
```

Create some informative plots and say which of the variables seem to have a linear relationship and which not and might thus benefit from a non-linear transformation (like e.g. a spline).

Solution

```
v = c("Outstate", "Private", "Room.Board", "Terminal", "perc.alumni", "Expend",
    "Grad.Rate")
pl.var = subset(college.train, select = v)
par(mfrow = c(2, 3))
for (i in 2:ncol(pl.var)) plot(pl.var[, i], pl.var[, 1], type = "p", pch = 1,
    xlab = co.names[i - 1], ylab = "Outstate")
```




```
par(mfrow = c(1, 1))
```

The relationship between `Outstate` and each of the predictors `Room.Board`, `perc.alumni` and `Grad.Rate` seems linear, while the relationship between `Outstate` and `Terminal`, `Expend` is non-linear.

d) (3P)

(i) Fit polynomial regression model for `Outstate` with `Terminal` as the only covariate for a range of polynomial degrees ($d = 1, \ldots, 10$) and plot the results. Use the training data (`college.train`) for this task.

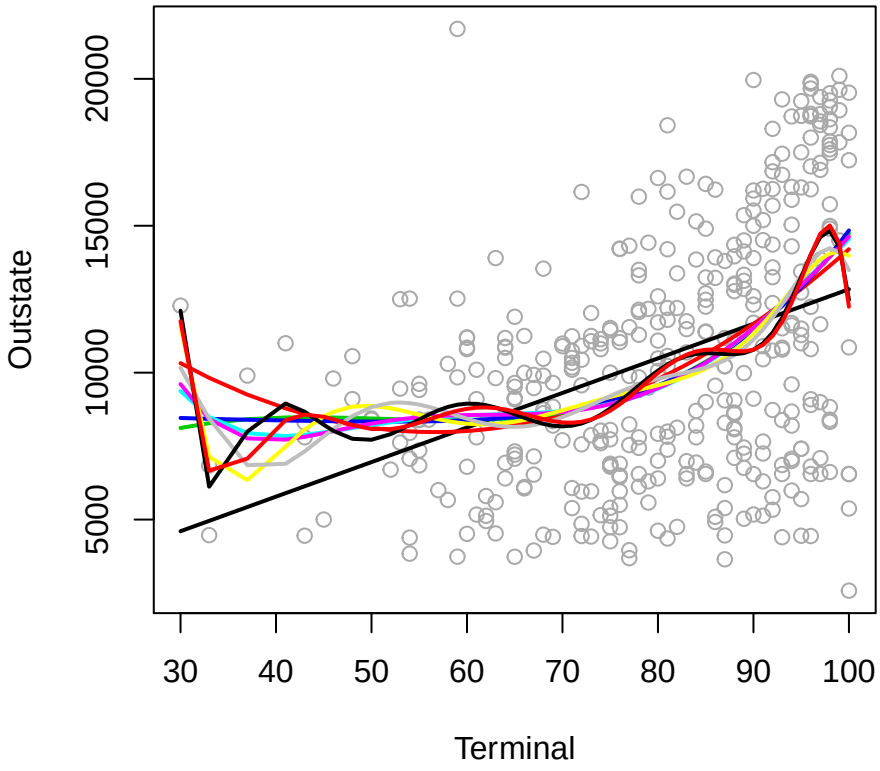
(ii) Still for the training data, choose a suitable smoothing spline model to predict `Outstate` as a function of `Expend` (again as the only covariate) and plot the fitted function into the scatterplot of `Outstate` against `Expend`. How did you choose the degrees of freedom?

(iii) Report the corresponding training MSE for (i) and (ii). Did you expect that?

Solution

(i) Polynomial Regression

```
attach(college.train)
mse_train = c()
plot(Terminal, Outstate, type = "p", col = "darkgrey")
for (i in 1:10) {
    poly.mod = lm(Outstate ~ poly(Terminal, degree = i), data = college.train)
    lines(Terminal[order(Terminal)], poly.mod$fitted.values[order(Terminal)],
        col = i, lwd = 2)
    mse_train[i] = mean(poly.mod$residuals^2)
}
```



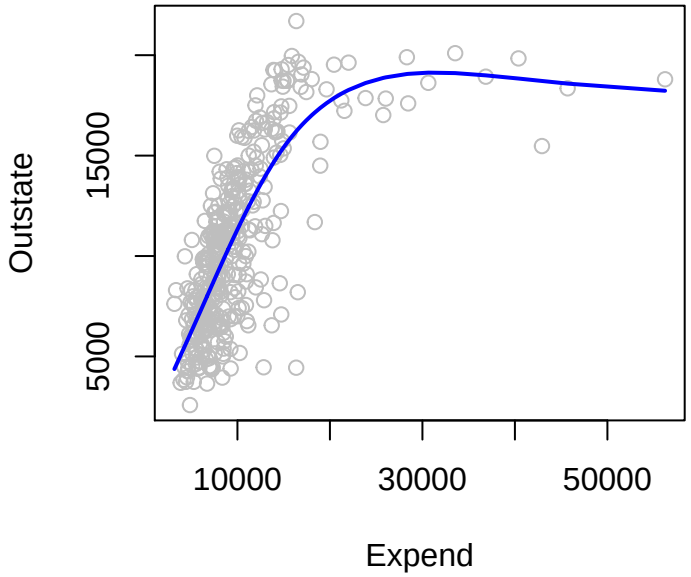

```
mse_train
```

```
##  [1] 15075161 14330586 14249448 14247330 14231485 14230392 14153207
##  [8] 14097911 13841526 13822205
```

As expected we see that the the MSE decreases monotonically for the training data as the df increases.

(ii) Smoothing Spline

```
attach(college.train)
df = range()
sm.sp.model = smooth.spline(Expend, Outstate, cv = TRUE)
pred.sm = predict(sm.sp.model, Expend)
plot(Expend, Outstate, col = "grey", type = "p", ylab = "Outstate", xlab = "Expend")
lines(sm.sp.model, col = "blue", lwd = 2)
```



(iii) The MSE values for `Terminal` as covariates decrease for higher-order polynomials, as to be expected (however, of course we know that test MSE would probably go up, but that's not the question here).

```
mse_train
```

```
##  [1] 15075161 14330586 14249448 14247330 14231485 14230392 14153207
##  [8] 14097911 13841526 13822205
```

On the other hand, the MSE when using `Expend` as a predictor in a smoothing spline model is much smaller, both for the training and test sets:

```
(MSE = mean((college.train$Outstate - pred.sm$y)^2))
```

```
## [1] 6871281
```

This is not surprising, at least when we look at the plots (i.e., the graph reveals that `Expend` is quite a good predictor).

Problem 3 (9p)

a) (2P) - Multiple choice

Which of the following statements are true, which false?

(i) Regression trees cannot handle categorical predictors.
(ii) Regression and classification trees are easy to interpret.
(iii) The random forest approaches improves bagging, because it reduces the variance of the predictor function by decorrelating the trees.
(iv) The number of trees B in bagging and random forests is a tuning parameter.

Solution: FALSE - TRUE - TRUE - FALSE

b) (4P)

Select one method from Module 8 (tree-based methods) in order to build a good model to predict `Outstate` in the `College` dataset that we used in problems 1 and 2. Explain your choice (pros/cons?) and how you chose the tuning parameter(s). Train the model using the training data and report the MSE for the test data.

Solution

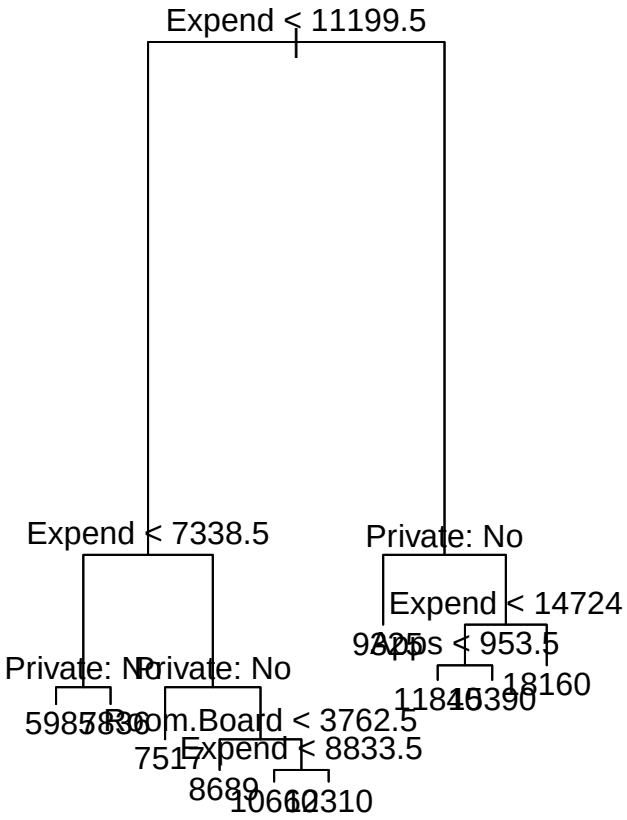
The students should explain the pros/cons of the method they chose and how they chose the tuning parameters.

Regression tree

```
library(tree)
tree.mod = tree(Outstate ~ ., college.train)
summary(tree.mod)
```

```
##
## Regression tree:
## tree(formula = Outstate ~ ., data = college.train)
## Variables actually used in tree construction:
## [1] "Expend"     "Private"    "Room.Board" "Apps"
## Number of terminal nodes:  10
## Residual mean deviance:  3971000 = 1.501e+09 / 378
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -7594.00 -1236.00    71.59     0.00  1263.00  6803.00
```

```
plot(tree.mod)
text(tree.mod, pretty = 0)
```



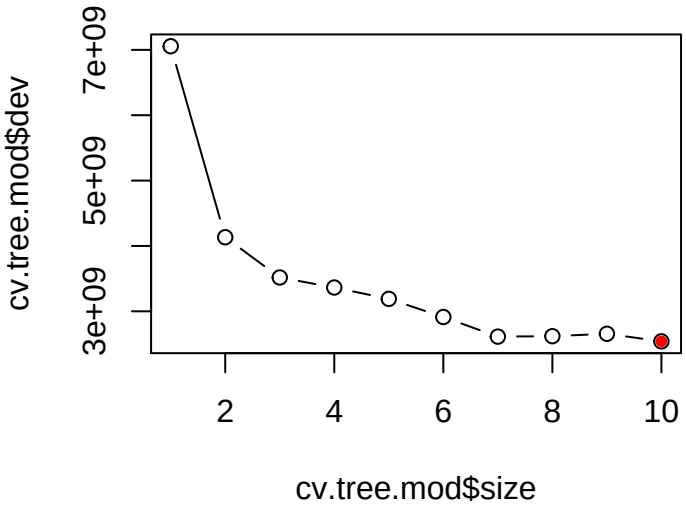

```
mse.full.tree = mean((college.test$Outstate - predict(tree.mod, college.test))^2)
mse.full.tree
```

```
## [1] 4185923
```

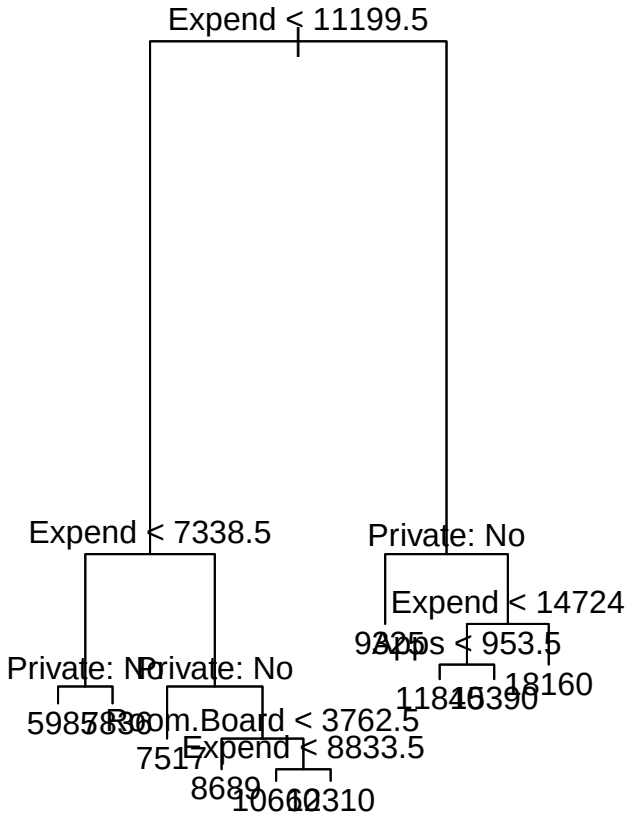
Pruning

First cross-validate and then prune the tree:

```
set.seed(4268)
cv.tree.mod = cv.tree(tree.mod)
tree.min = which.min(cv.tree.mod$dev)
best = cv.tree.mod$size[tree.min]
plot(cv.tree.mod$size, cv.tree.mod$dev, type = "b")
points(cv.tree.mod$size[tree.min], cv.tree.mod$dev[tree.min], col = "red", pch = 20)
```




```
pr.tree = prune.tree(tree.mod, best = 10)
plot(pr.tree)
text(pr.tree, pretty = 0)
```




```
mse.pr.tree = mean((college.test$Outstate - predict(pr.tree, college.test))^2)
mse.pr.tree
```

```
## [1] 4185923
```

Bagging and random forest

```
set.seed(4268)
library(randomForest)
bag.model = randomForest(Outstate ~ ., data = college.train, mtry = ncol(college.train) -
    1, importance = TRUE)
mse.bagging = mean((college.test$Outstate - predict(bag.model, college.test))^2)
rf.model = randomForest(Outstate ~ ., data = college.train, mtry = 6, importance = TRUE)
mse.rf = mean((college.test$Outstate - predict(rf.model, college.test))^2)
mse.bagging
```

```
## [1] 2770371
```

```
mse.rf
```

```
## [1] 2605711
```

```
# varImpPlot(bag.model) varImpPlot(rf.model)
```

Boosting

In boosting there are different choices for the number of trees, the interaction depth and λ. The students should at least mention that these are potential tuning parameters.

```
set.seed(4268)
library(gbm)
boost = gbm(Outstate ~ ., college.train, distribution = "gaussian", n.trees = 1000,
    interaction.depth = 4, shrinkage = 0.01)
yhat.boost = predict(boost, newdata = college.test, n.trees = 1000)
mse.boost = mean((yhat.boost - college.test$Outstate)^2)
mse.boost
```

```
## [1] 2576982
```

c) (2p)

Compare the results (tests MSEs) among all the methods you used in Problems 1-3. Which method perform best in terms of prediction error? Which method would you choose if the aim is to develop an interpretable model?

Solution

```
cbind(mse.for.reg, mse.lasso, mse.full.tree, mse.bagging, mse.rf, mse.boost)
```

```
##      mse.for.reg mse.lasso mse.full.tree mse.bagging  mse.rf mse.boost
## [1,]     3844857   3717510       4185923     2770371 2605711   2576982
```

Random Forest and Boosting are the best performing methods in terms of prediction accuracy. However, if the aim is to develop an *interprentable model*, we should prefer forward regression, Lasso or a simple regression tree. Here, forward regression leads to a slightly lower test MSE than Lasso, and it is simpler as it inclues only 6 predictors.

Problem 4 (12P)

We will use the classical data set of *diabetes* from a population of women of Pima Indian heritage in the US, available in the R `MASS` package. The following information is available for each woman:

- diabetes: `0`= not present, `1`= present
- npreg: number of pregnancies
- glu: plasma glucose concentration in an oral glucose tolerance test
- bp: diastolic blood pressure (mmHg)
- skin: triceps skin fold thickness (mm)
- bmi: body mass index (weight in kg/(height in m)^2)
- ped: diabetes pedigree function.
- age: age in years

We will use a training set (called `d.train`) with 300 observations (200 non-diabetes and 100 diabetes cases) and a test set (called `d.test`) with 232 observations (155 non-diabetes and 77 diabetes cases). Our aim is to make a classification rule for the presence of diabetes (yes/no) based on the available data. You can load the data as follows:

```
id <- "1Fv6xwKLSZHldRAC1MrcK2mzdOYnbgv0E"  # google file ID
d.diabetes <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id))
d.train = d.diabetes$ctrain
d.test = d.diabetes$ctest
```

a) (2P) - Multiple choice

Start by getting to know the *training data*, by producing summaries and plots. Which of the following statements are true, which false?

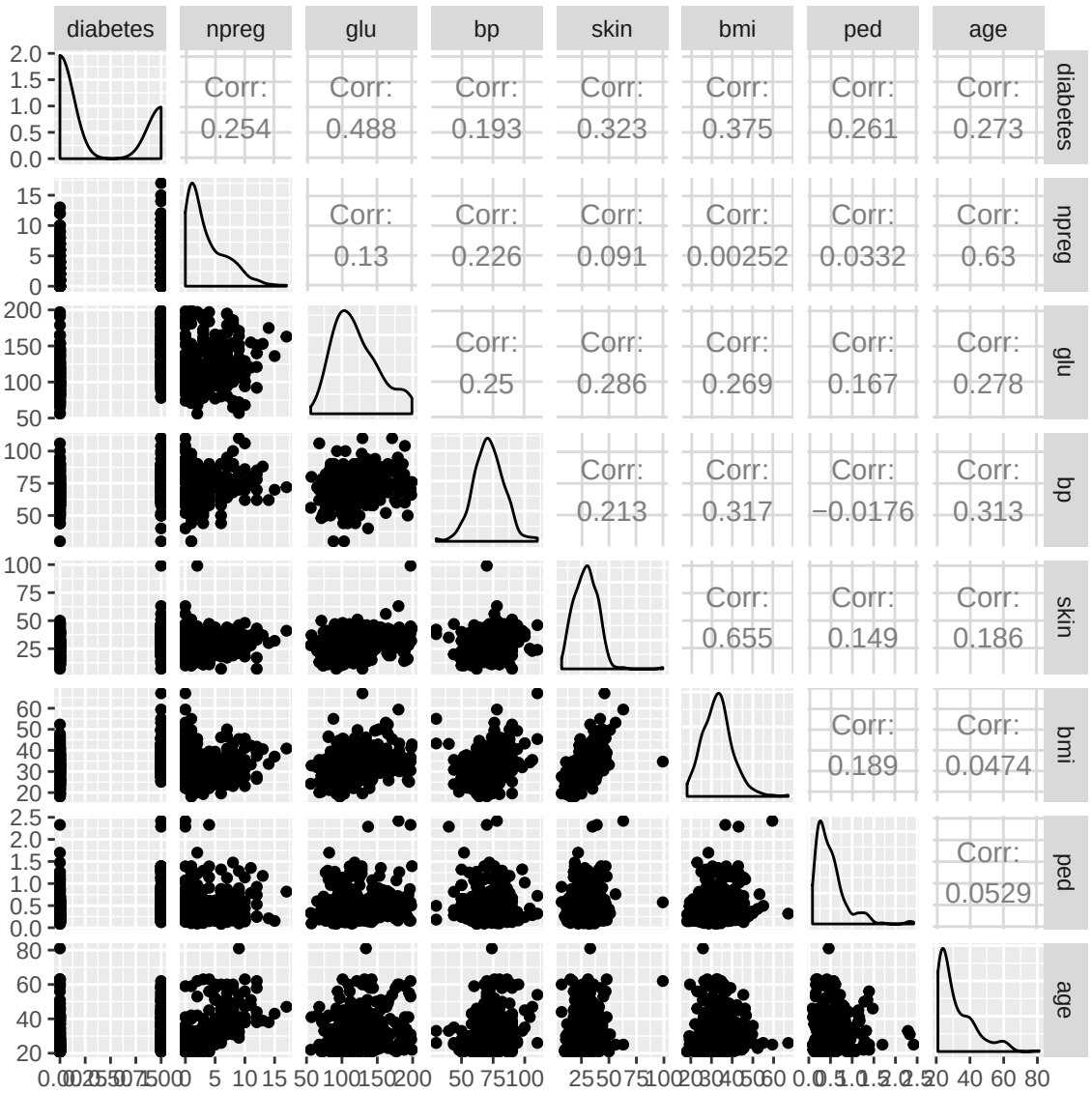
(i) Females with high glucose levels and higher bmi seem to have a higher risk for diabetes.
(ii) Some women had up to 17 pregnancies.
(iii) BMI and triceps skin fold thickness seem to be positively correlated.
(iv) The distribution of the number of pregnancies per woman seems to be a bit skewed and a transformation of this variable could therefore be appropriate.

Solution:

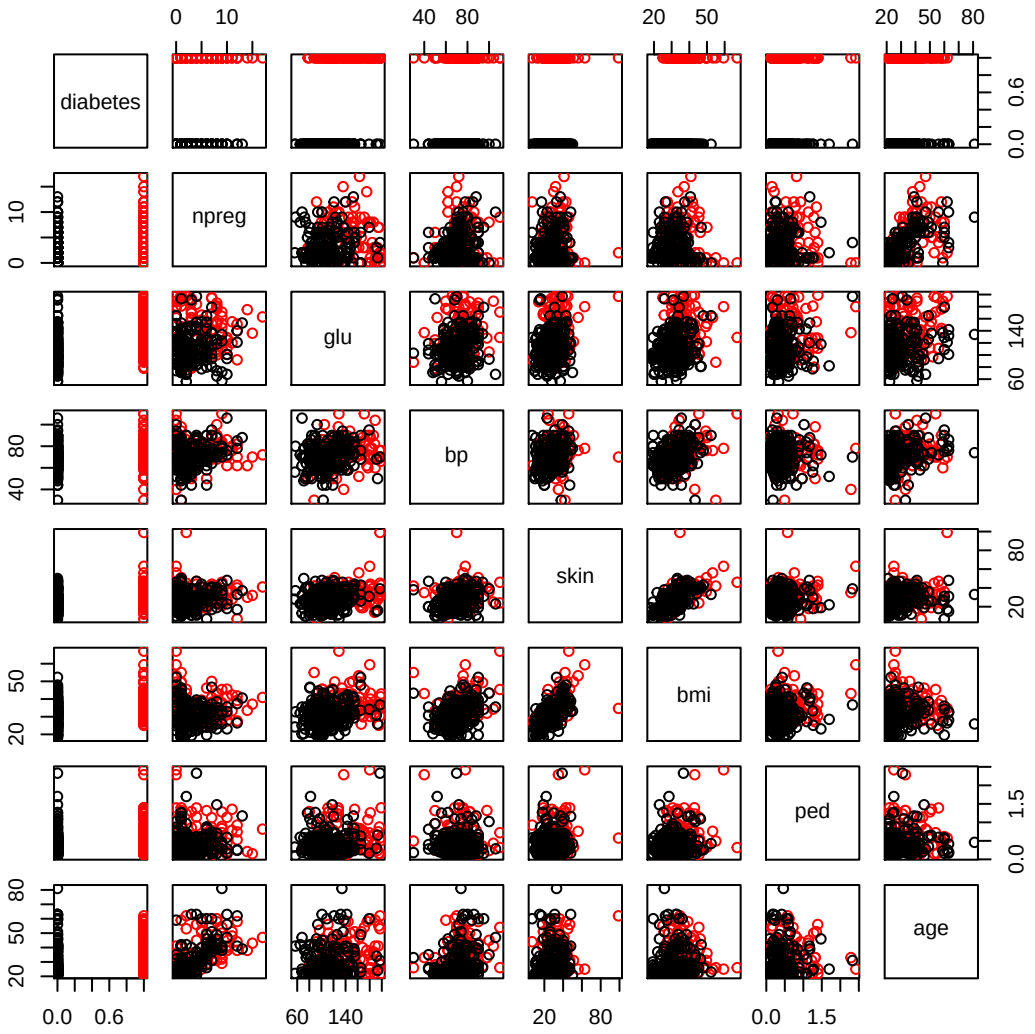
TRUE - TRUE - TRUE - TRUE

To see why all statements are correct, the following graphs and summaries might help:

```
# not to be included
library(GGally)
ggpairs(d.train)
```




```
pairs(d.train, col = d.train$diabetes + 1)
```




```
summary(d.train)
```

```
##     diabetes          npreg             glu              bp
##  Min.   :0.0000   Min.   : 0.000   Min.   : 56.00   Min.   : 30.00
##  1st Qu.:0.0000   1st Qu.: 1.000   1st Qu.: 96.75   1st Qu.: 64.00
##  Median :0.0000   Median : 2.000   Median :114.00   Median : 71.00
##  Mean   :0.3333   Mean   : 3.467   Mean   :120.13   Mean   : 71.56
##  3rd Qu.:1.0000   3rd Qu.: 5.250   3rd Qu.:140.25   3rd Qu.: 80.00
##  Max.   :1.0000   Max.   :17.000   Max.   :199.00   Max.   :110.00
##       skin             bmi              ped              age
##  Min.   : 7.00   Min.   :18.20   Min.   :0.0850   Min.   :21.00
##  1st Qu.:22.00   1st Qu.:27.98   1st Qu.:0.2567   1st Qu.:23.00
##  Median :29.00   Median :32.80   Median :0.4150   Median :27.00
##  Mean   :29.14   Mean   :33.03   Mean   :0.5004   Mean   :31.55
##  3rd Qu.:36.00   3rd Qu.:37.12   3rd Qu.:0.6210   3rd Qu.:37.25
##  Max.   :99.00   Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

b) (4P)

Fit a support vector classifier (linear boundary) and a support vector machine (radial boundary) to find good functions that predict the diabetes status of a patient. Use cross-validation to find a good cost parameter (for

the linear boundary) and a good combination of cost *and* γ parameters (for the radial boundary). Report the confusion tables and misclassification error rates for the test set in both cases. Which classifier do you prefer and why? (Do not use any variable transformations or standardizations to facilitate correction).

R-hints: The response variable must be converted into a factor variable before you continue.

```
d.train$diabetes <- as.factor(d.train$diabetes)
d.test$diabetes <- as.factor(d.test$diabetes)
```

To run cross-validation over a grid of two tuning parameters, you can use the `tune()` function where `ranges` defines the grid points as follows:

```
tune(..., formula, kernel = ..., ranges = list(cost = c(...), gamma = c(...)))
```

Solution: Linear boundary:

```
set.seed(4268)
library(e1071)

d.train$diabetes <- as.factor(d.train$diabetes)
d.test$diabetes <- as.factor(d.test$diabetes)

CV_svm_linear <- tune(svm, diabetes ~ npreg + glu + bp + skin + bmi + ped +
    age, data = d.train, kernel = "linear", ranges = list(cost = c(0.001, 0.01,
    0.05, 0.1, 0.5, 1, 5, 10, 50)))
```

The following command gives a long list with an overview over the performance for the different values of cost (output not printed).

```
summary(CV_svm_linear)
```

```
bestmod_linear <- CV_svm_linear$best.model
summary(bestmod_linear)
```

```
##
## Call:
## best.tune(method = svm, train.x = diabetes ~ npreg + glu + bp +
##     skin + bmi + ped + age, data = d.train, ranges = list(cost = c(0.001,
##     0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.05
##
## Number of Support Vectors:  159
##
##  ( 78 81 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

For the radial boundary we have to use `kernel="radial"`, and cross-validate over a grid of cost and γ parameters.

```
set.seed(123)
CV_svm_radial <- tune(svm, diabetes ~ npreg + glu + bp + skin + bmi + ped +
    age, data = d.train, kernel = "radial", ranges = list(cost = c(0.001, 0.01,
    0.05, 0.1, 0.5, 1, 5, 10, 50), gamma = c(0.001, 0.01, 0.1, 1, 10)))
```

The following command gives a long list with an overview over the performance for the different sets of parameters (output not printed).

```
summary(CV_svm_radial)
```

```
bestmod_radial <- CV_svm_radial$best.model
summary(bestmod_radial)
```

```
##
## Call:
## best.tune(method = svm, train.x = diabetes ~ npreg + glu + bp +
##     skin + bmi + ped + age, data = d.train, ranges = list(cost = c(0.001,
##     0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50), gamma = c(0.001, 0.01,
##     0.1, 1, 10)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##
## Number of Support Vectors:  153
##
##  ( 76 77 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

Prediction error for linear and radial kernels:

```
ypred_linear = predict(bestmod_linear, d.test)
(tt1 <- table(predict = ypred_linear, truth = d.test[, "diabetes"]))
```

```
##        truth
## predict   0   1
##       0 137  36
##       1  18  41
```

```
1 - sum(diag(tt1))/sum(tt1)
```

```
## [1] 0.2327586
```

```
ypred_radial = predict(bestmod_radial, d.test)
(tt2 <- table(predict = ypred_radial, truth = d.test[, "diabetes"]))
```

```
##        truth
## predict   0   1
##       0 136  35
##       1  19  42
```

```r
1 - sum(diag(tt2))/sum(tt2)
```

```
## [1] 0.2327586
```

It looks like both SVMs have a similar misclassification error rate. It is thus probably better to choos the simpler (i.e., the linear) model.

c) (2P)

Compare the performance of the two classifiers from b) to *one other classification method* that you have learned about in the course. Explain your choice and report the confusion table and misclassification error rate on the test set for your chosen method and interpret what you see. What are advantages/disadvantages of your chosen method with respect to SVMs?

Solution: We do not provide detailed solutions here. Possibilities involve

- The KNN classifier
- Logistic regression
- LDA or QDA
- Classification trees

Most other methods have the advantage that they provide actual class probabilities, and not just a simple category prediction like SVMs (this is something all replies should mention). On the other hand, SVMs are very powerful for nonlinear boundaries. However, in the present example it looks like a radial boundary does not lead to an improvement over a linear boundary (see b), so a simple logistic regression has the advantage that it allows for easy interpretation of how each variable impacts the class probabilities.

d) (2P) - Multiple choice

Which of the following statements are true, which false?

(i) Under standard conditions, the maximal margin hyperplane approach is equivalent to a linear discriminant analysis.
(ii) Under standard conditions, the support vector classifier is equivalent to quadratic discriminant analysis.
(iii) Logistic regression, LDA and support vector machines tend to perform similar when decision boundaries are linear, unless classes are linearly separable.
(iv) An advantage of logistic regression over SVMs is that it is easier to do feature selection and to interpret the results.

Solution:

FALSE - FALSE - TRUE - TRUE

Comment on (i) and (ii): The support vector classifer is not equivalent to a LDA or QDA. One reason to understand why is because only a limited number of data points (the support vectors) influences the estimation of the classifier function of a support vector classifier, while LDA and QDA take all data points into account by estimating the class mean and covariance matrices from the data.

e) (2P) Link to logistic regression and hinge loss.

Look at slides 71-73 of Module 9. Show that the loss function

$$\log(1 + \exp(-y_i f(\boldsymbol{x}_i)))$$

is the deviance for the $y = -1, 1$ encoding in a logistic regression model.
Hint: $f(\boldsymbol{x}_i)$ correponds to the linear predictor in the logistic regression approach.

Solution: Using that $\mathrm{P}(Y=1) = \frac{\exp(f)}{1+\exp(f)}$ and $\mathrm{P}(Y=-1) = \frac{1}{1+\exp(f)}$, the contribution of $(\boldsymbol{x}_i, y_i)$ to the likelihood is

$$\mathrm{L}(f; (\boldsymbol{x}_i, y_i)) = \left(\frac{\exp(f)}{1+\exp(f)}\right)^{\mathrm{I}(y=1)} \cdot \left(\frac{1}{1+\exp(f)}\right)^{\mathrm{I}(y=-1)} .$$

Therefore, the negative log-likelihood (which is proportional to the deviance) is given as

$$\begin{aligned} -\log \mathrm{L}(f; (\boldsymbol{x}_i, y_i)) =& -\mathrm{I}(y_i = 1) \log\left(\frac{\exp(f(x_i))}{1+\exp(f(x_i))}\right) \\ & -\mathrm{I}(y_i = -1) \cdot \log\left(\frac{1}{1+\exp(f(x_i))}\right) \\ =& \mathrm{I}(y_i = 1) \underbrace{[\log\left(1+\exp(f(x_i)) - f(x_i)\right]}_{(\star)} \\ & + \mathrm{I}(y_i = -1) \cdot \log\left(1+\exp(f(x_i))\right) , \end{aligned}$$

where the second line is an algebraic reformulation of the first one. Rewriting the $(\star)$ component as

$$(\star) = \log\left(1+\exp(f(x_i)\right) - \log(\exp(f(x_i))) = \log\left(\frac{1+\exp(f(x_i))}{\exp(f(x_i))}\right) = \log\left(\exp(-f(x_i)) + 1\right) ,$$

we get

$$\begin{aligned} -\log \mathrm{L}(f; (\boldsymbol{x}_i, y_i)) =& \mathrm{I}(y_i = 1) \log\left(1+\exp(-y_i f(x_i))\right) + \\ & \mathrm{I}(y_i = -1) \log\left(1+\exp(-y_i f(x_i))\right) . \end{aligned}$$

This expression can now be summarized to the result

$$\log\left(1+\exp(-y_i f(x_i))\right) .$$

Problem 5 (10P)

The following dataset consists of 40 tissue samples with measurements of 1,000 genes. The first 20 tissues come from healthy patients and the remaining 20 come from a diseased patient group. The following code loads the dataset into your session with colum names decribing if the tissue comes from a diseased or healthy person.

```
id <- "1VfVCQvWt121UN39NXZ4aR9Dmsbj-p9OU"  # google file ID
GeneData <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = F)
colnames(GeneData)[1:20] = paste(rep("H", 20), c(1:20), sep = "")
colnames(GeneData)[21:40] = paste(rep("D", 20), c(1:20), sep = "")
row.names(GeneData) = paste(rep("G", 1000), c(1:1000), sep = "")
```

a) (2P)

Perform hierarchical clustering with complete, single and average linkage using **both** Euclidean distance and correlation-based distance on the dataset. Plot the dendograms. Hint: You can use `par(mfrow=c(1,3))` to plot all three dendograms on one line or `par(mfrow=c(2,3))` to plot all six together.

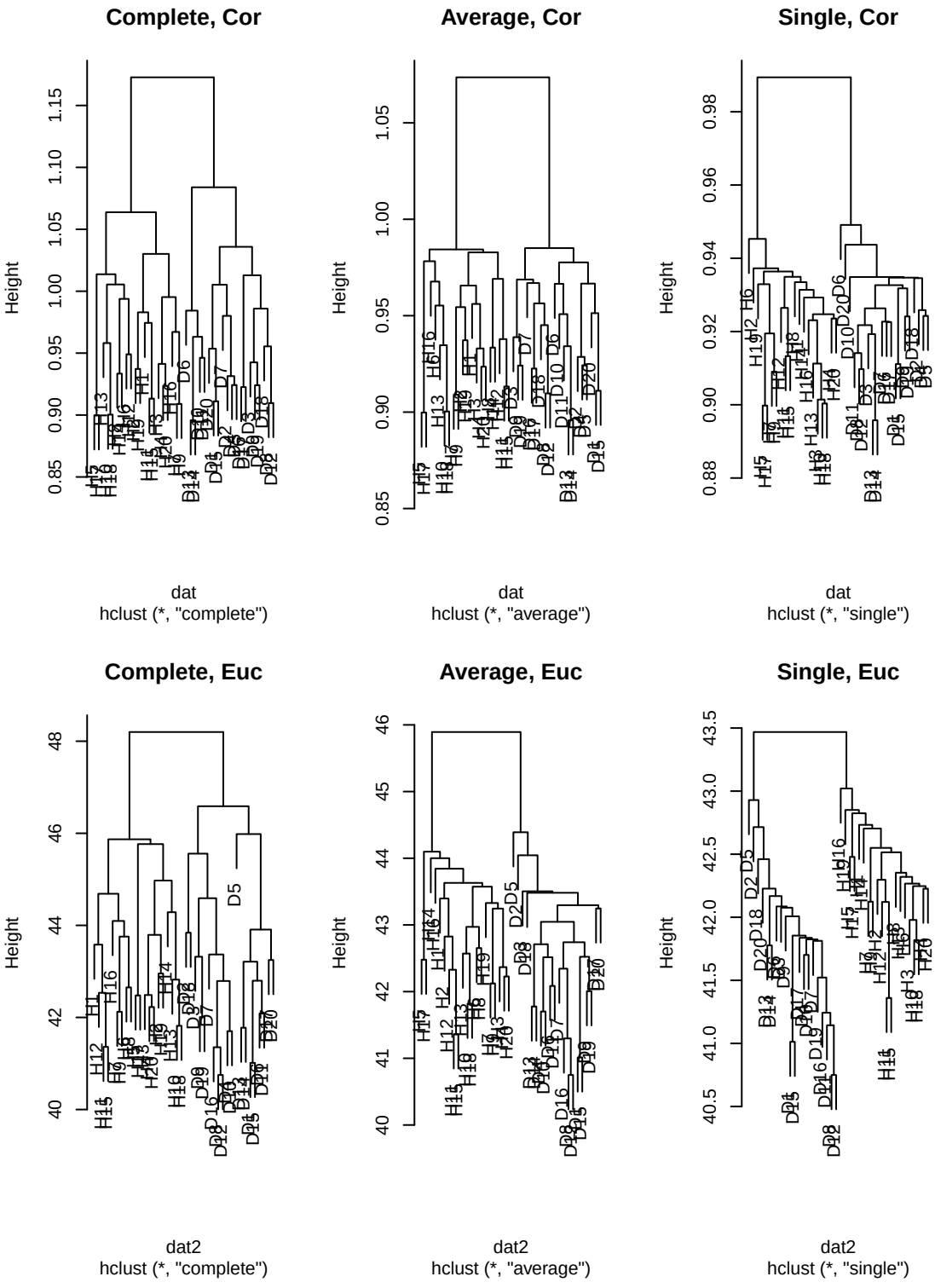
Solution:

```
GeneData = t(GeneData)
GeneData <- scale(GeneData)
# dat = as.dist(t(d)) dat = as.dist(1-cor(d)) correlation based
dat = as.dist(1 - cor(t(GeneData)))
# euclidean distance
dat2 = dist((GeneData), method = "euclidian")

# hier.clust
hc.complete = hclust(dat, method = "complete")
hc.average = hclust(dat, method = "average")
hc.single = hclust(dat, method = "single")

hc.complete2 = hclust(dat2, method = "complete")
hc.average2 = hclust(dat2, method = "average")
hc.single2 = hclust(dat2, method = "single")
```

```
par(mfrow = c(2, 3))
plot(hc.complete, main = "Complete, Cor")
plot(hc.average, main = "Average, Cor")
plot(hc.single, main = "Single, Cor")
plot(hc.complete2, main = "Complete, Euc")
plot(hc.average2, main = "Average, Euc")
plot(hc.single2, main = "Single, Euc")
```



b) (2P)

Use these dendograms to cluster the tissues into two groups. Compare the groups with respect to the patient group the tissue comes from. Which linkage and distance measure performs best when we know the true state of the tissue?

Solution:

```
# cut tree into two braches
clustComp = cutree(hc.complete, k = 2)
clustSing = cutree(hc.single, k = 2)
clustAv = cutree(hc.average, k = 2)

clustComp2 = cutree(hc.complete2, k = 2)
clustSing2 = cutree(hc.single2, k = 2)
clustAv2 = cutree(hc.average2, k = 2)

# actual group:
trueGroup = c(rep(1, 20), rep(2, 20))
Comp = table(trueGroup, clustComp)
Sing = table(trueGroup, clustSing)
Av = table(trueGroup, clustAv)
Comp2 = table(trueGroup, clustComp2)
Sing2 = table(trueGroup, clustSing2)
Av2 = table(trueGroup, clustAv2)

errorRate = function(data) {
    return((sum(data) - sum(diag(data)))/(sum(data)))
}

error = c(errorRate(Comp), errorRate(Sing), errorRate(Av), errorRate(Comp2),
    errorRate(Sing2), errorRate(Av2))
error
```

```
## [1] 0 0 0 0 0 0
```

All three linkages with Euclidean distance have zero errors - so it doesn't matter which one we choose next.

c) (1P)

With Principal Component Analysis, the first principal component loading vector solves the following optimization problem,

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j1} x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

Explain what ϕ, p, n and x are in this optimization probelm and write down the formula for the first principal component scores.

Solution:

The first principal component scores ($z_{i,1}$) takes the form,

$$z_{i1} = \phi_{11} x_{i1} + \phi_{2,1} x_{i2} + \cdots + \phi_{p,1} x_{ip}$$

where

- ϕ_{j1} is the loading vector for covariate j
- x_{ij} is the covariate j for sample i
- p is the number of covariates
- n is the sample size

d) (2P)

(i) (1P) Use PCA to plot the samples in two dimensions. Color the samples based on the tissues group of patients.

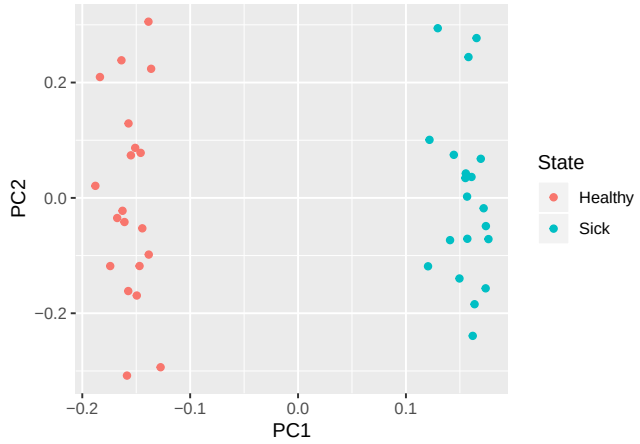
(ii) (1P) How much variance is explained by the first 5 PCs?

Solution:

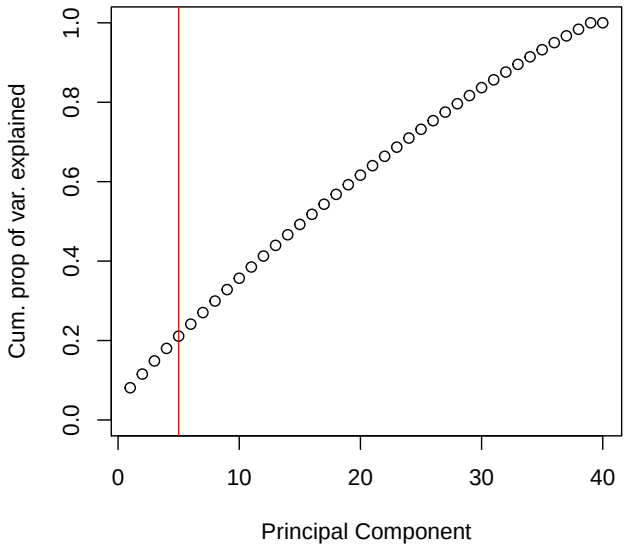
```
# transform and add column for state (diseased/healthy)
PCdata = as.data.frame((GeneData))

PCdata$State = c(rep("Healthy", 20), rep("Sick", 20))

library(ggfortify)
pr.out = prcomp(PCdata[, 1:1000], scale = T)
autoplot(pr.out, data = PCdata, colour = "State")
```




```
# Proportion of variance explained
pr.var = pr.out$sdev^2
pve = pr.var/(sum(pr.var))
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cum. prop of var. explained",
    ylim = c(0, 1), type = "b")
abline(v = 5, col = "red")
```



```r
cumsum(pve)[5]
```

```
## [1] 0.2109659
```

The first five prinicipal component explain 21.1% of the variance in the data. However, we see from the plot that the first PC clearly seperates the two groups.

e) (1P)

Use your results from PCA to find which genes that vary the most accross the two groups.

Solution:

Look at loadings/rotation vector of PCA and find which genes that contribute the most.

```r
# which genes contribute the most to PC1?
ImpGenes = names(sort(abs(pr.out$rotation[, 1]), decreasing = T)[1:10])
pr.out$rotation[match(ImpGenes, row.names(pr.out$rotation)), 1]
```

```
##       G502       G589       G565       G590       G600       G551
## 0.09485044 0.09449766 0.09183823 0.09173169 0.09167322 0.08768360
##       G593       G538       G584       G509
## 0.08758616 0.08745400 0.08690858 0.08661015
```

All these genes contribute with positive loadings for PCA, meaning that high values of these genes contribute to the disease group and low values point to the healthy group.

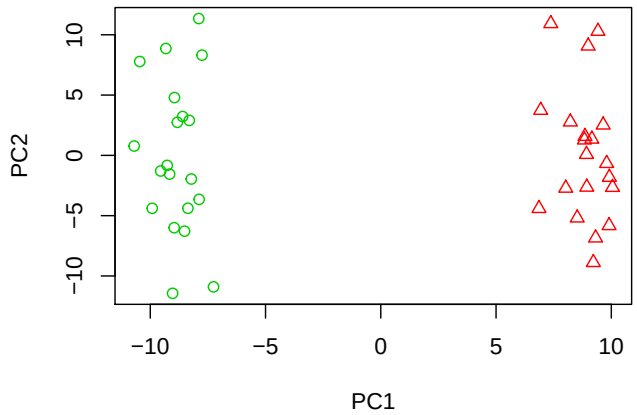
f) (2P)

Use K-means to seperate the tissue samples into two groups. Plot the values in a two-dimensional space with PCA. What is the error rate of K-means?

Solution:

```r
km.out = kmeans((GeneData), 2, nstart = 20)

plot(pr.out$x[, c(1:2)], col = (km.out$cluster + 1), pch = trueGroup)
```




```r
error = sum(km.out$cluster != trueGroup)/(length(km.out$cluster))
error
```

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

Gives 0 missclassifications.