Compulsory exercise 2: Group 24

TMA4268 Statistical Learning V2019

Silje Anfindsen and Clara Panchaud

20 March, 2020

Problem 1

a)

Let's find the ridge regression estimator. Remember that $\hat{\beta}_{Ridge}$ minimizes $RSS + \lambda \sum_{j=1}^{p} \beta_j^2$. Let's rewrite this in matrix notation.

$$\begin{aligned}
min_\beta\{(y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta\} = &\quad \text{develop the expression} \\
min_\beta\{y^T y - 2\beta^T X^T y + \beta^T X^T X\beta + \lambda\beta^T\beta\} &\quad \text{take the derivative with respect to beta and set equal to 0} \\
-2X^T y + 2X^T X\beta + 2\lambda\beta = 0 & \\
(X^T X + 2\lambda I)\beta = X^T y & \\
\beta = (X^T X + \lambda I)^{-1} X^T y &
\end{aligned}$$

Therefore the estimator is $\hat{\beta}_{Ridge} = (X^T X + \lambda I)^{-1} X^T y$.

b)

To find the expected value and the variance-covariance matrix of $\hat{\beta}_{Ridge}$ we need to remember the distribution of y, $y \sim N(X\beta, \sigma^2 I)$. Therefore we get the expected value:

$$E(\hat{\beta}_{Ridge}) = E((X^T X + \lambda I)^{-1} X^T y) = (X^T X + \lambda I)^{-1} X^T E(y) = (X^T X + \lambda I)^{-1} X^T X\beta$$

and the variance-covariance matrix:

$$\begin{aligned}
Var(\hat{\beta}_{Ridge}) = Var((X^T X + \lambda I)^{-1} X^T y) = &\quad \text{by property of the variance} \\
(X^T X + \lambda I)^{-1} X^T Var(y)((X^T X + \lambda I)^{-1} X^T)^T = &\quad \text{develop the expression} \\
\sigma^2 (X^T X + \lambda I)^{-1} X^T X (X^T X + \lambda I)^{-1} &
\end{aligned}$$

c)

TRUE, FALSE, FALSE, TRUE

d)

```
library(ISLR)
library(leaps)
library(glmnet)
```

We want to work with the *College* data. First we split it into a training and a testing set.

```
set.seed(1)

#make training and testing set
train.ind = sample(1:nrow(College), 0.5 * nrow(College))
college.train = College[train.ind, ]
college.test = College[-train.ind, ]

#the structure of the data
str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

Now we will apply forward selection, using *Outstate* as a response. We have 18 variables including the response so we will obtain a model including up to 17 variables.

```
nb_predictors<-17
forward<-regsubsets(Outstate~.,college.train,nvmax=17,method="forward")
sum<-summary(forward)
```

In Figure 1 we can look at the RSS and the adjusted R^2 in order to pick the number of variables that gives the optimal result. Remember that if the difference is not very significant we would rather pick the simplest model. It seems like 5 variables would be good here.

```
par(mfrow=c(1,2))
plot(sum$rss,xlab="Number of Variables",ylab="RSS",type="l")
plot(sum$adjr2,xlab="Number of Variables",ylab="Adjusted RSq",type="l")
```

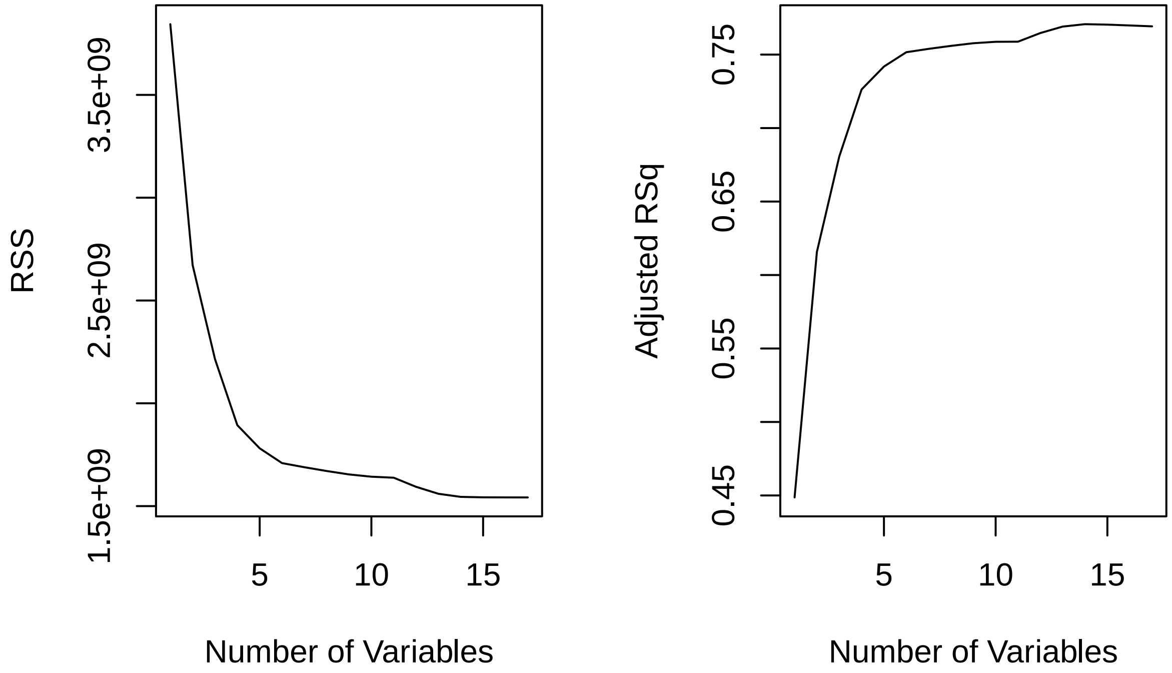


Figure 1: Comparison of models with different number of variables.

Below are the chosen variables when we decide to include 5 variables in the reduced model.

```
nb_selected_pred<-5
variables<-names( coef( forward,id=nb_selected_pred ) )
variables
```

```
## [1] "(Intercept)" "PrivateYes"  "Room.Board"  "perc.alumni" "Expend"
## [6] "Grad.Rate"
```

We will now find the reduced model as well as the MSE (mean squared error) on the test set.

```
#fit the reduced model
reduced.model<-lm(Outstate~Private+Room.Board+Grad.Rate+perc.alumni+Expend, data =college.train)
summary(reduced.model)
```

```
##
## Call:
```

```
## lm(formula = Outstate ~ Private + Room.Board + Grad.Rate + perc.alumni +
##     Expend, data = college.train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7293.1 -1537.5  -159.9  1286.7  9254.8
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.711e+03  5.155e+02  -5.259 2.41e-07 ***
## PrivateYes   2.250e+03  2.787e+02   8.075 8.92e-15 ***
## Room.Board   1.241e+00  1.205e-01  10.296  < 2e-16 ***
## Grad.Rate    3.855e+01  7.850e+00   4.910 1.35e-06 ***
## perc.alumni  6.446e+01  1.113e+01   5.792 1.45e-08 ***
## Expend       2.182e-01  2.317e-02   9.417  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2159 on 382 degrees of freedom
## Multiple R-squared:  0.7452, Adjusted R-squared:  0.7419
## F-statistic: 223.4 on 5 and 382 DF,  p-value: < 2.2e-16
```

The reduced model is

$$\begin{aligned}\textbf{Outstate} = &-2711.4329907 + 2250.1100562\textbf{Private} + 1.2410466\textbf{Room.Board}\\ &+ 38.5491289\textbf{Grad.Rate} + 64.4580901\textbf{perc.alumni} + 0.218216\textbf{Expend},\end{aligned}$$

```
#find test MSE
p<-predict(reduced.model,newdata=college.test)
mse_fwd <- mean(((college.test$Outstate)-p)^2)
mse_fwd
```

```
## [1] 4112680
```

The test MSE is

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(\hat{x_i}))^2 = 4.1126804 \times 10^6$$

e)

We will now select a model for the same dataset as in (d) but this time with the Lasso method. Again, we use both a training and testing set for the data.

```
#Make a x matrix and y vector for both the training and testing set
x_train<-model.matrix(Outstate~.,college.train)[,-1]
y_train<-college.train$Outstate
x_test<-model.matrix(Outstate~.,college.test)[,-1]
y_test<-college.test$Outstate
```

In order to select the best value for the tuning parameter λ we will use cross validation.

```r
set.seed(5555)

#perform the Lasso method and choose the best model using CV
lasso.mod = glmnet(x_train,y_train,alpha=1) #lasso method on train set
cv.lasso = cv.glmnet(x_train,y_train,alpha=1) #CV on train set
lambda.best = cv.lasso$lambda.min #select best lambda
lambda.best
```

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

```r
#find the test MSE
predictions<-predict(lasso.mod,s=lambda.best,newx=x_test)
mse_lasso <- mean((predictions-y_test)^2) #test MSE
mse_lasso
```

```
## [1] 3717020
```

%%Check if log scale!! From cross validation we can observe that the optimal tuning parameter is $\lambda = 5.0932009$ as this is the parameter that minimizes the MSE for the training set.

The test MSE is now 3.7170197×10^6, which is lower than what we found for the reduced model using forward selection in d).

The lasso yields sparse models which involves only a subset of variables. Lasso performs variable selection by forcing some of the coefficient estimates to be exactly zero. The selected variables that was not put to zero are displayed below.

```r
c<-coef(lasso.mod,s=lambda.best,exact=TRUE)
inds<-which(c!=0)
variables<-row.names(c)[inds]
variables
```

```
##  [1] "(Intercept)" "PrivateYes"  "Apps"        "Accept"      "Enroll"
##  [6] "Top10perc"   "Top25perc"   "F.Undergrad" "P.Undergrad" "Room.Board"
## [11] "Books"       "Personal"    "PhD"         "Terminal"    "S.F.Ratio"
## [16] "perc.alumni" "Expend"      "Grad.Rate"
```

Problem 2

a)

FALSE, FALSE, TRUE, FALSE

b)

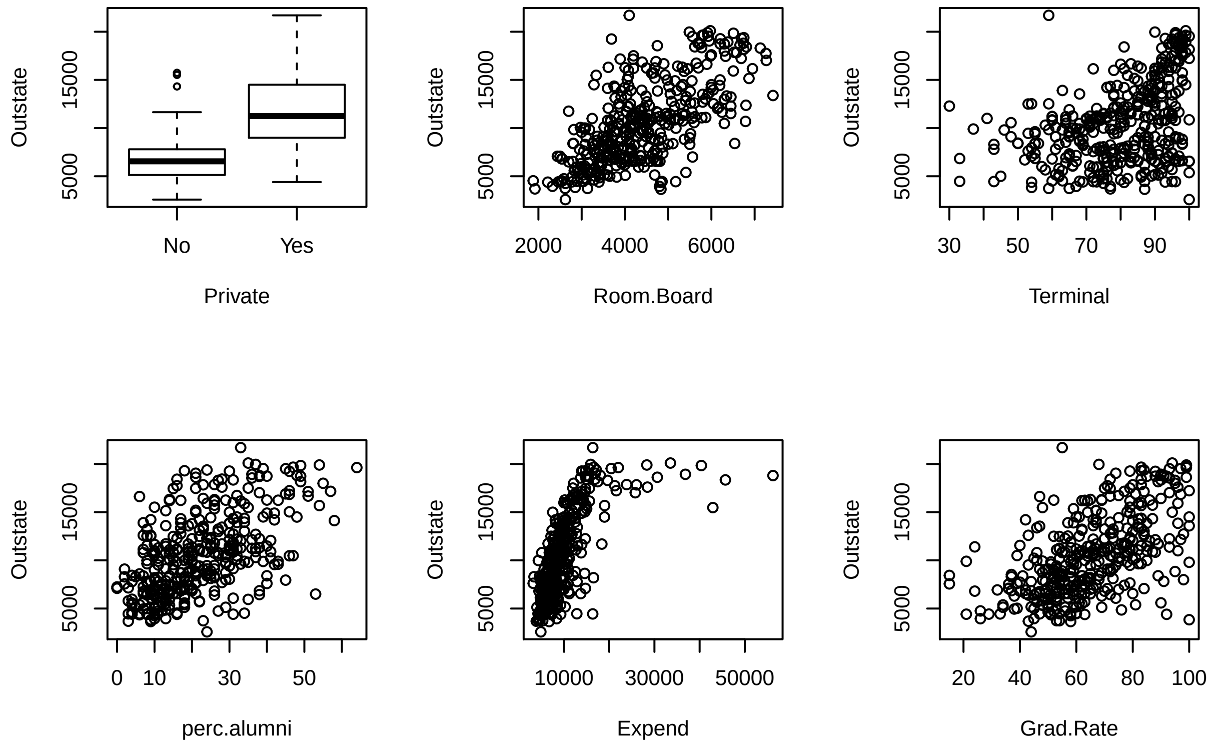
The basis functions for a cubic spline with knots at each quartile, of variable X are,

$$\begin{aligned} b_0(X) &= 1 & b_4(X) &= (X - q_1)_+^3 \\ b_1(X) &= x & b_5(x) &= (X - q_2)_+^3 \\ b_2(X) &= x^2 & b_6(X) &= (X - q_3)_+^3 \\ b_3(X) &= x^3 & & \end{aligned}$$

c)

We will now investigate the realtionship between *Outstate* and the 6 of the predictors, *Private*, *Room.Board*, *Terminal*, *perc.alumni*, *Expend*, and *Grad.Rate*.

```r
ds1 = college.train[c("Private", "Outstate")] #binary variable
ds2 = college.train[c("Room.Board", "Outstate")]
ds3 = college.train[c("Terminal", "Outstate")]
ds4 = college.train[c("perc.alumni", "Outstate")]
ds5 = college.train[c("Expend", "Outstate")]
ds6 = college.train[c("Grad.Rate", "Outstate")]

par(mfrow=c(2,3))
plot(ds1)
plot(ds2)
plot(ds3)
plot(ds4)
plot(ds5)
plot(ds6)
```

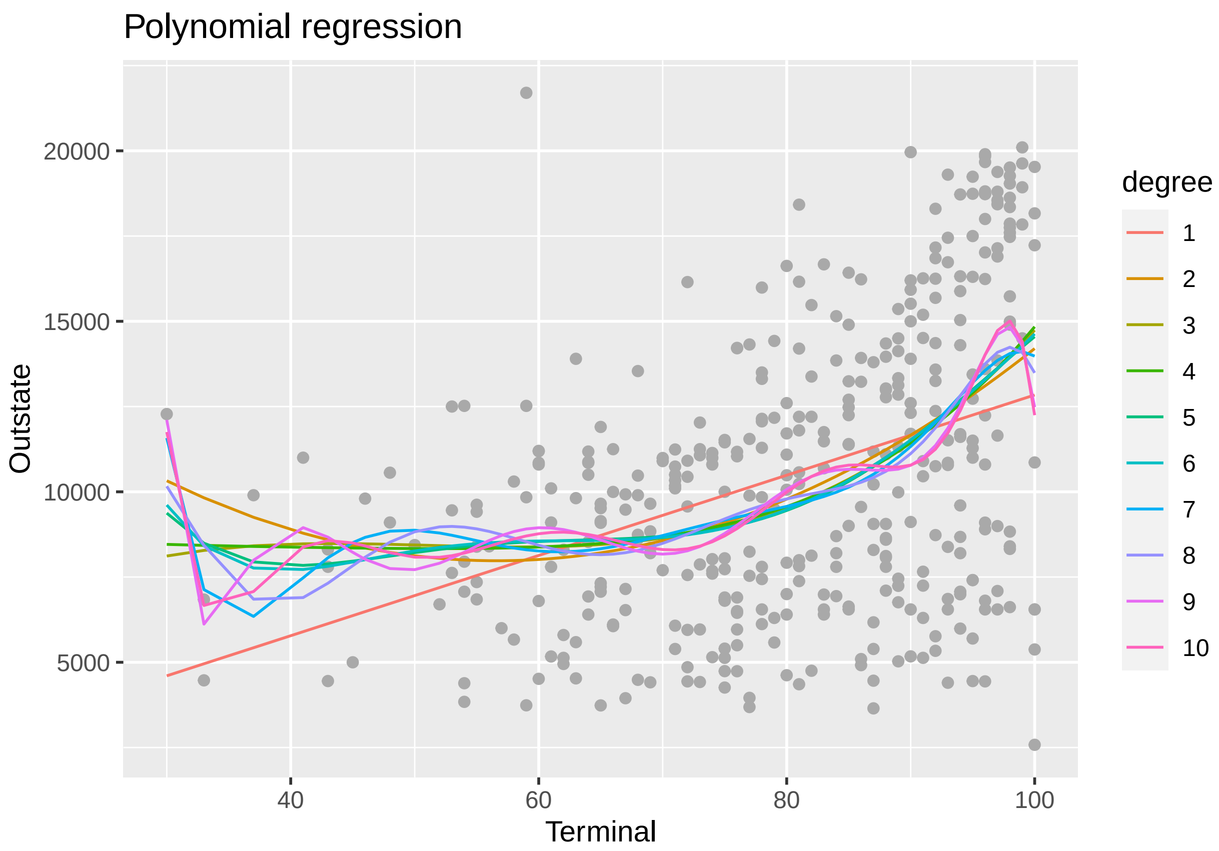


From each of the plots above we can conclude that at least *Terminal* and *Expend* seems to have a non-linear relationship with *Outstate*. These two variables therefore might benefit from a non-linear transformation. The others variables, *Room.Board*, *perc.alumni* and *Grad.Rate* seem to have a linear realtionship with the response variable. The binary variable *Private* is presented through a boxplot. Generally the data seem to cathegorize quite well into these two classes of private and public universites where the trend is a higher out-of-state tuition for private universites, except for some outliers for public universities where the outcome is very high and therefore can seem to belong to a private universities. Anyway, we cannot transform a binary variable.

d)

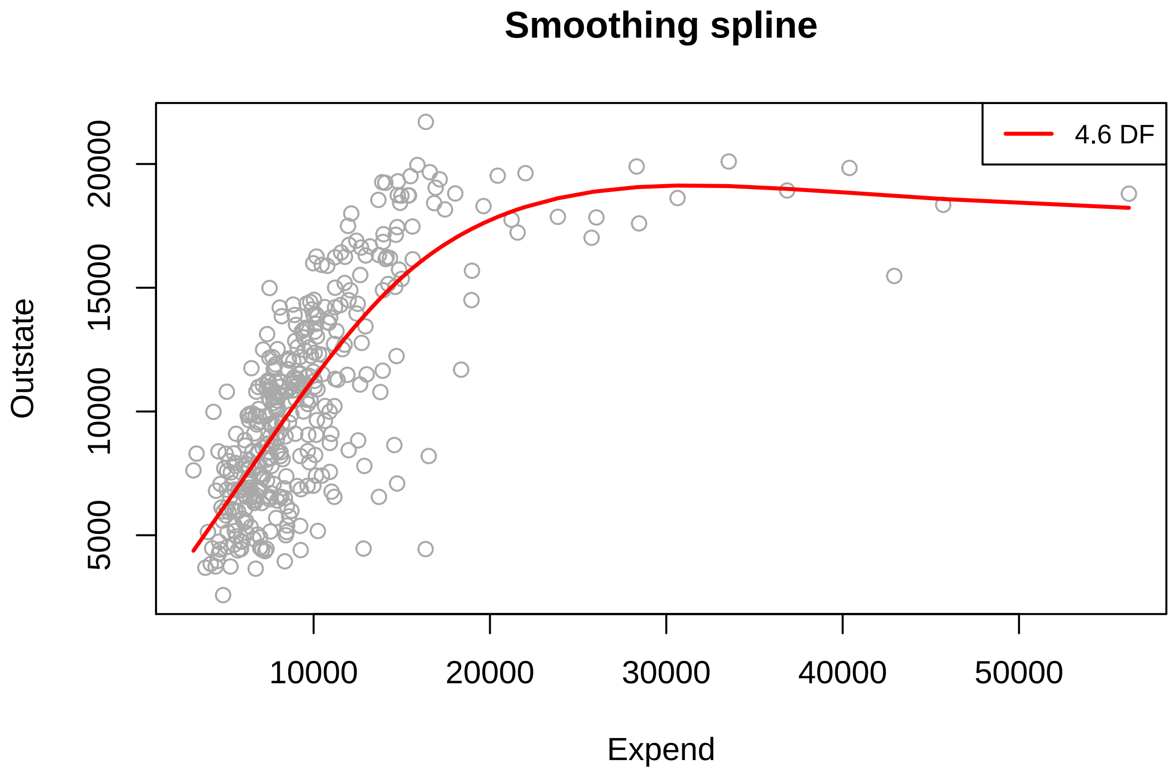
We will now fit several polynomial regression models for *Outstate* with *Terminal* as the only covariate. Each polynomial will have a degree from $d = 1, ..10$.

```r
library(ggplot2)

#make a dataframe
ds = College[c("Terminal", "Outstate")]
n = nrow(ds)

# chosen degrees
deg = 1:10

#now iterate over each degree d
dat = c()  #make a empty variable to store predicted values for each degree
MSE_poly = c(rep(0,10))  #make a empty variable to store MSE for each degree
for (d in deg) {
    # fit model with this degree
    mod = lm(Outstate ~ poly(Terminal, d), ds[train.ind, ])

    #dataframe for Terminal and Outstate showing result for each degree over all samples
    dat = rbind(dat, data.frame(Terminal = ds[train.ind, 1], Outstate = mod$fit,
                                degree = as.factor(rep(d,length(mod$fit)))))
    # training MSE
    MSE_poly[d] = mean((predict(mod, ds[-train.ind, ]) - ds[-train.ind, 2])^2)
}

# plot fitted values for different degrees
ggplot(data = ds[train.ind, ], aes(x = Terminal, y = Outstate)) +
    geom_point(color = "darkgrey") + labs(title = "Polynomial regression")+
    geom_line(data = dat, aes(x = Terminal, y = Outstate, color = degree))
```



We will now choice a suitable smoothing spline model to predict *Outstate* as a function of *Expend* and plot the fitted function.

```
library(splines)

#plot training set for Expend as only covariate
plot(college.train$Expend, college.train$Outstate, col = "darkgrey", main="Smoothing spline", xlab="Expe
#perform CV in order to find optimal number of df
fit = smooth.spline(college.train$Expend, college.train$Outstate, cv=TRUE)
df <- fit$df #choose df from CV
l <- fit$lambda

#add fitted function from smoothing spline
lines(fit, col="red", lwd=2)
legend("topright", legend=c("4.6 DF"), col="red", lty=1, lwd=2, cex=.8)
```

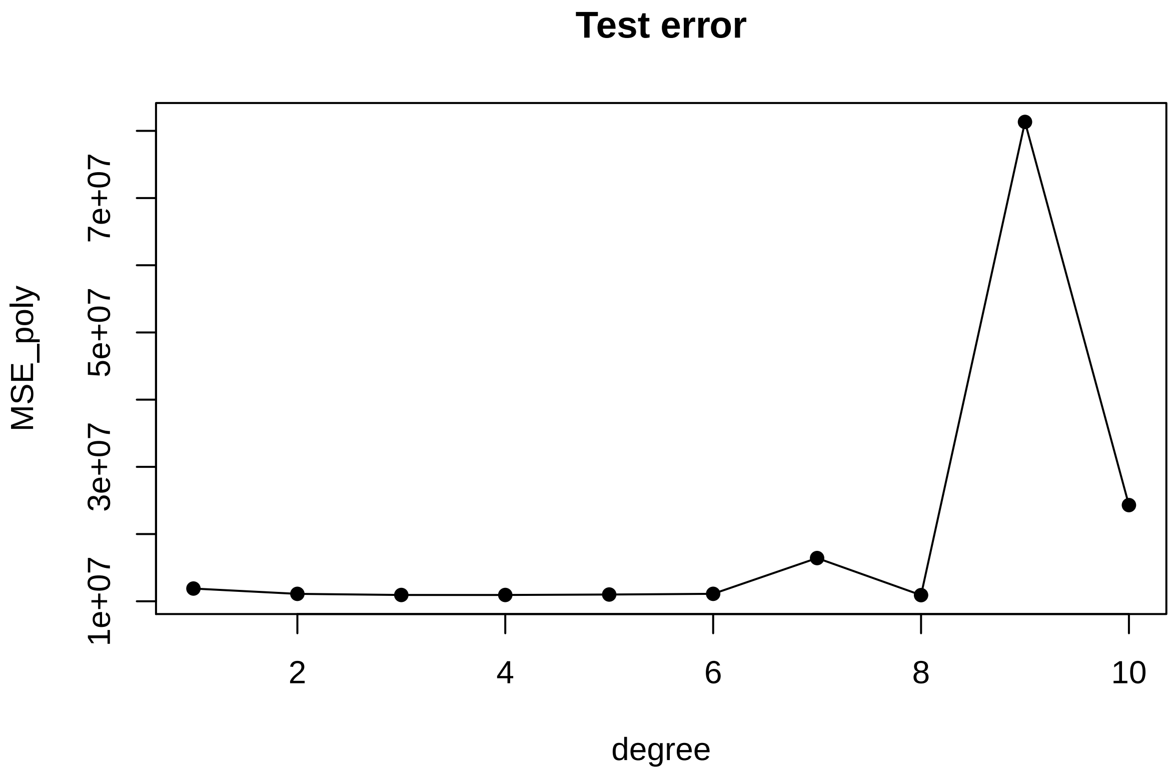


```
#training MSE
pred = predict(fit, newdata=college.train)
MSE_spline = mean( (college.train$Outstate - pred$y)^2 )
```

In order to choose a suitable model we did cross validation. The optimal number of degrees of freedom is $df = 4.660711$, which gives the smoothing parameter $\lambda = 0.0075385$

We will now calculate the training MSE for the polynomial regresson models and the smoothing spline model. For the polynomial regression models we find it easiest to look at the MSE by presenting a plot with MSE for each degree.

```
#MSE for polynomial regression models (1-10)
plot(1:10,MSE_poly, type = "o", pch = 16, xlab = "degree", main = "Test error")
```



```
min(MSE_poly)
```

```
## [1] 10914136
```

```
#MSE for smoothing spline
MSE_spline
```

```
## [1] 31072818
```

The training MSE for the two methods are (discuss is it is as expected!)

Problem 3

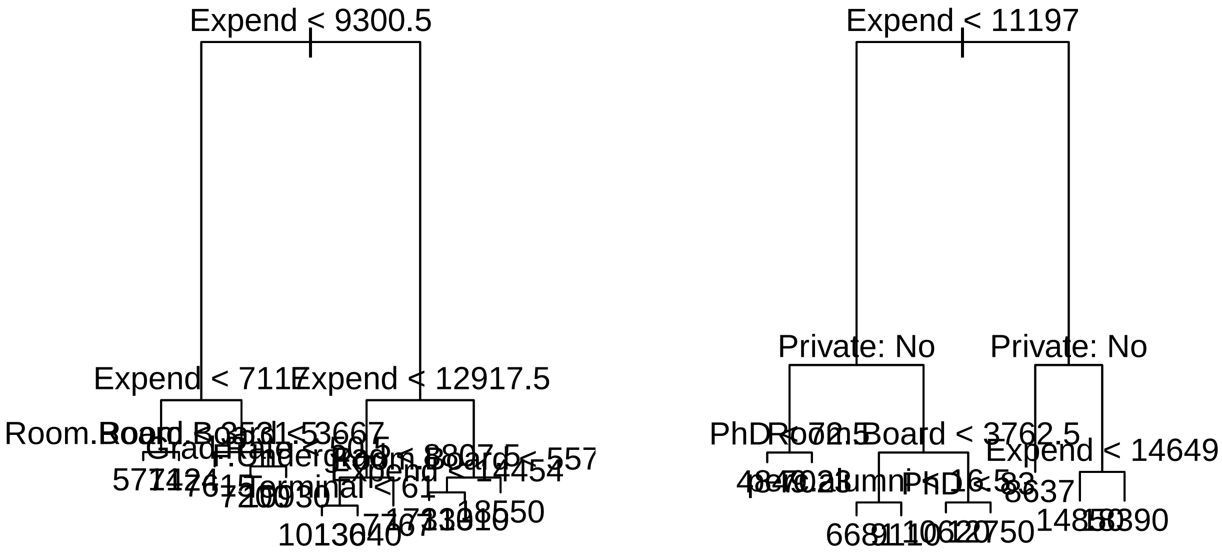
a)

FALSE, TRUE, TRUE, FALSE

b)

To predict *Outstate* we first try fitting the simple tree-based method, Regression Trees. Since we have learned that decision trees often suffer from high variance we will dividie our training dato into two new sets and fit a tree on each. If this is the case we will get two quite different trees.

```
library(tree)
set.seed(1)

#make new training and testing set - random split
new.train.ind = sample(1:nrow(college.train), 0.5 * nrow(college.train))
new.college.train1 = college.train[new.train.ind, ]
new.college.train2 = college.train[-new.train.ind, ]

#fit a tree on the full data set and two new trees with the two new traning sets
full.tree = tree(Outstate ~ .,college.train) #full tree
tree.mod1 = tree(Outstate ~ .,new.college.train1)
tree.mod2 = tree(Outstate ~ .,new.college.train2)

#plot the two tree based on a smaller part of the training set
par(mfrow=c(1,2))
plot(tree.mod1)
text(tree.mod1, pretty = 0)
plot(tree.mod2)
text(tree.mod2, pretty = 0)
```



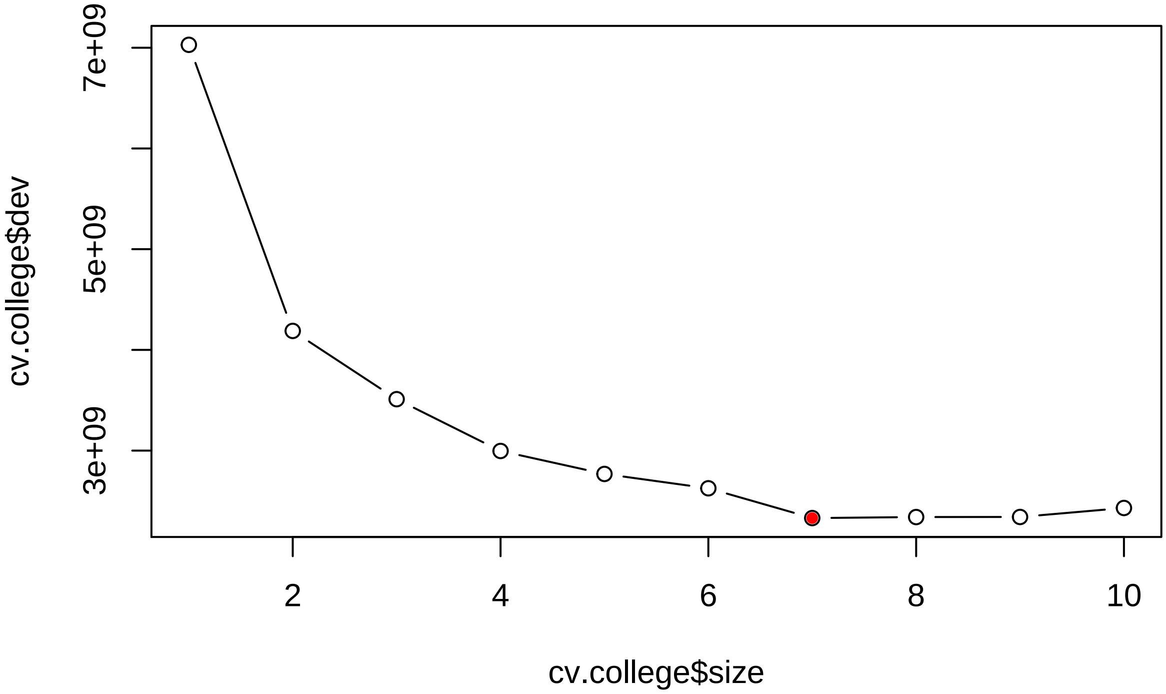
```
#find test MSE for full tree
yhat = predict(full.tree, newdata = college.test, n.trees = 500)
mse_pure <- mean((yhat- college.test$Outstate)^2)
mse_pure
```

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

maybe we dont need to actually do the pruning, and go directly boosting?

Now we try pruning and check whether this will improve performance.

```
set.seed(1)
cv.college = cv.tree(full.tree)
tree.min = which.min(cv.college$dev)
best = cv.college$size[tree.min]
plot(cv.college$size, cv.college$dev, type = "b")
points(cv.college$size[tree.min], cv.college$dev[tree.min], col = "red", pch = 20)
```



We see that 7 is on the same level of deviance as for size 10, since this is the smallest number for the size we pick 7. Let us now prune the tree to make it size 7.

```
pr.tree = prune.tree(tree.mod1, best = 7)
#plot(pr.tree)
#text(pr.tree, pretty = 0)

#find test mse
yhat.prune = predict(pr.tree, newdata = college.test, n.trees = 500)
mse.prune <- mean((yhat.prune- college.test$Outstate)^2)
mse.prune
```

```
## [1] 6599665
```

done with pruning (maybe remove the pruning part ?)

We observe that the two decision trees above are quite different where different variables act as the most important factors for splitting. We conclude that the regression tree therefore suffer from high variance. We will try the boosting approach in order to train the tree and decrease the variance. The boosting approach grows several trees where each tree is grown using information from previously grown trees. The final predicitor is a weighted sum of the trees. We have three tuning parameters: the number of trees B, the shrinkage parameter λ and the number of splits in each tree (interaction depth) d. These parameters can be decided with cross validation.

```
library(gbm)
set.seed(1)

#fit a boosted tree to the training data - use CV in order to decide the tuning parameters
boost.cv = gbm(Outstate ~ .,data=college.train, distribution="gaussian", cv.folds = 10)
boost.cv$n.trees
```

```
## [1] 100
```

```
#check relative importance of variables
summary(boost.cv, plotit = FALSE)
```

```
##                     var    rel.inf
## Expend           Expend 48.7606794
## Room.Board   Room.Board 16.7531438
## Private         Private 11.5500639
## Grad.Rate     Grad.Rate  8.6394004
## perc.alumni perc.alumni  5.3690550
## Books             Books  1.4002392
## P.Undergrad P.Undergrad  1.2444213
## F.Undergrad F.Undergrad  1.1475416
## PhD                 PhD  0.8009306
## Top10perc     Top10perc  0.7754256
## Top25perc     Top25perc  0.6867535
## Personal       Personal  0.6569089
## Enroll           Enroll  0.6280889
## Terminal       Terminal  0.4908188
## Apps               Apps  0.4825551
## S.F.Ratio     S.F.Ratio  0.3442051
## Accept           Accept  0.2697689
```

We notice that *Expend* has a very high relative influence and therefore seems to be the most important variable in the data. We will now look at the test MSE for the method.

```
yhat.boost = predict(boost.cv, newdata = college.test)
```

```
## Using 91 trees...
```

```
mse_boost <- mean((yhat.boost - college.test$Outstate)^2)
mse_boost
```

```
## [1] 3114600
```

The test MSE for the boosted tree, $MSE_{boosted} = 3.1145996 \times 10^6$ is smaller than for the Regression tree approach, $MSE_{non-boosted} = 4.1859232 \times 10^6$. Therefore the boosted tree will probably do a better prediciton of the response. **Write something about pros/cons for the chosen method (boosting) maybe compared to pure regression trees.**

c)

We will now compare the test MSEs among the methods used on the data set *College* so far. That is: the two linear model selection methods, forward selection and Lasso method, non-linear methods, polynomial regression and smoothing splines and at last the tree-based method Boosted regression trees.

```
MSE <- c(mse_fwd,mse_lasso,min(MSE_poly),MSE_spline,mse_pure, mse_boost)
Method <- c("Forward selection", "Lasso", "Polynomial regression", "Smoothing spline", "Regression Tree
df <- data.frame(Method, MSE)
kable(df)
```

Method	MSE
Forward selection	4112680
Lasso	3717020
Polynomial regression	10914136
Smoothing spline	31072818
Regression Tree	4185923
Boosting	3114600

The method performing best in terms of prediction error is the boosted regression tree. But if the aim is to develop a interpretable model we would probably have chosen the regression tree.

Problem 4

Start by loading the data of *diabetes* from a population of women.

```
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)

In order to anseer on the Mulitple choice we have to present the training data.

```
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
```

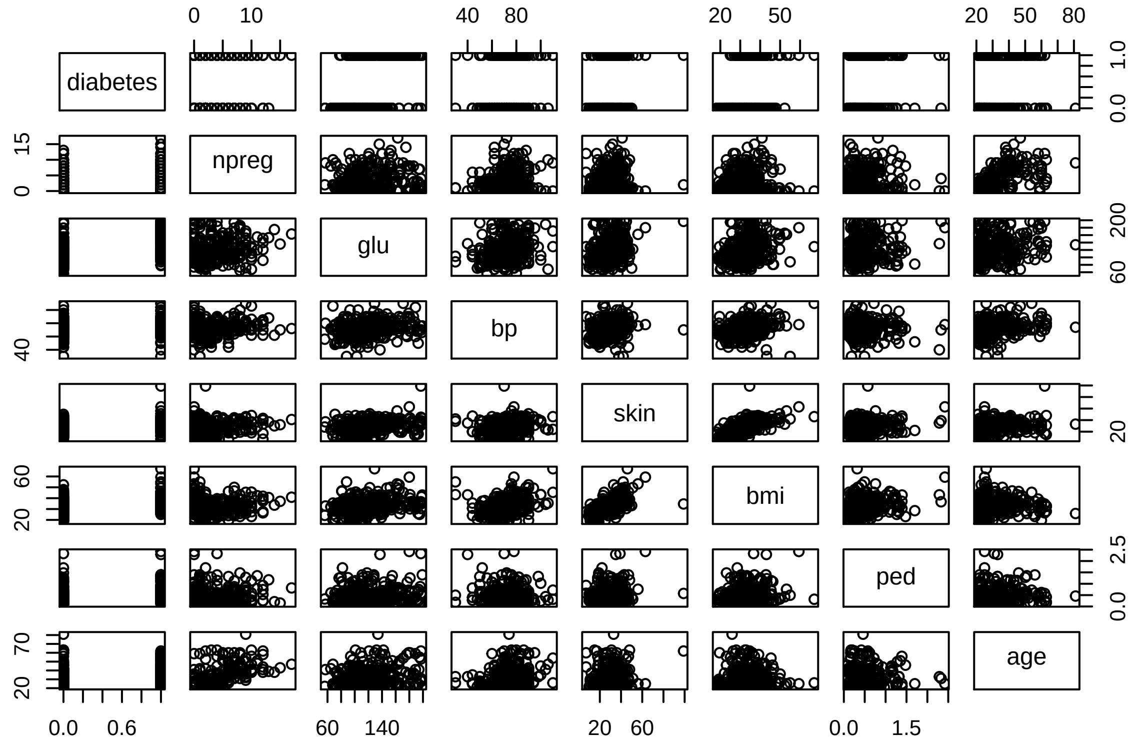
```
max(d.train$npreg)#max nr of pregnancies
```

```
## [1] 17
```

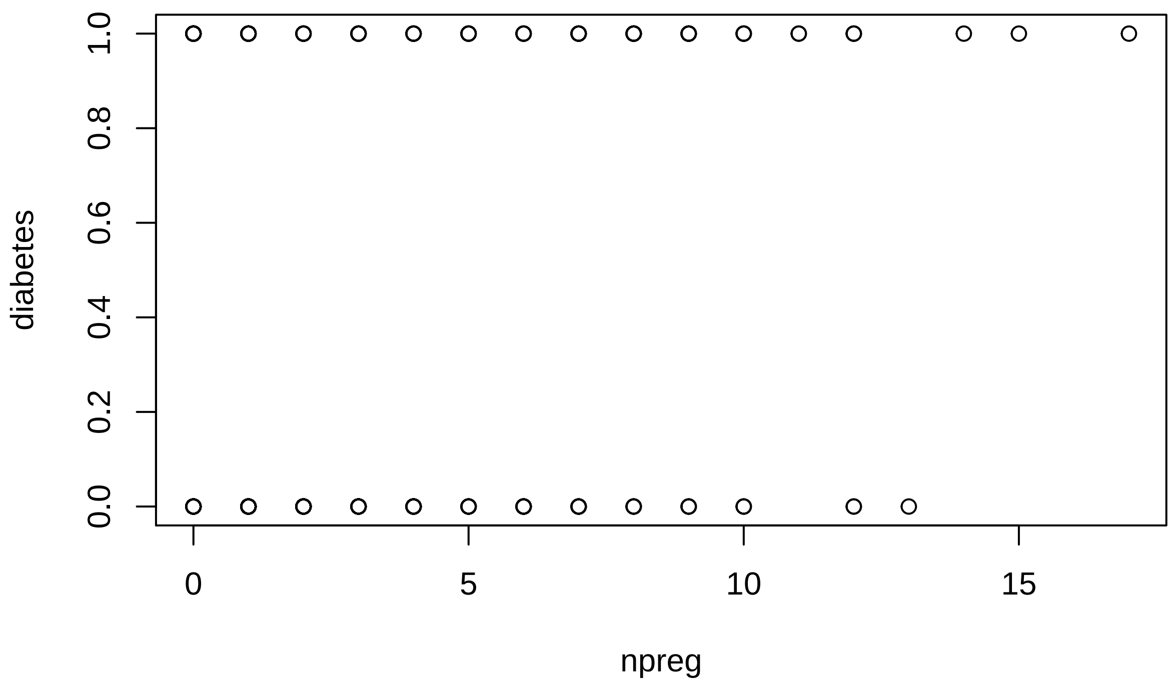
```
head(d.train) #overview of data
```

```
##     diabetes npreg glu bp skin  bmi   ped age
## 339        1     7 109 80   31 35.9 1.127  43
## 270        1     7 152 88   44 50.0 0.337  36
## 210        1     9 119 80   35 29.0 0.263  29
## 117        1     9 112 82   32 34.2 0.260  36
## 390        1     8 151 78   32 42.9 0.516  36
## 217        1     2  90 68   42 38.2 0.503  27
```

```
plot(d.train) #look at correlation between variables
```



```r
plot(diabetes~npreg, data=d.train)
```



TRUE, TRUE, TRUE, FALSE (not sure about the first and last)

b)

c)

d)

Problem 5

a)

b)

c)

d)

e)

f)

References

James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. An Introduction to Statistical Learning with Applications in R. New York: Springer.