```
#### Exam 1 - Assignment 1
### Libraries
library(tree)
library(glmnet)

### Setup
set.seed(12345)
data.crx = read.csv("crx.csv")
data.crx$Class = as.factor(data.crx$Class)
n = dim(data.crx)[1]
ids = sample(1:n, n*0.8)
train = data.crx[ids,]
test = data.crx[-ids,]

### Function
plot_tree = function(tree){
  plot(tree)
  text(tree, pretty=0)
}
# Use the following error function to compute the test error for the
LASSO and tree models
# E =∑i Yi*logp̂i +(1-Yi)*log(1- p̂i)
E = function(p, y){
  return(sum(y*log(p)+(1-y)*log(1-p)))
}

### Implementation
## Task 1
# Fit decision tree
tree1 = tree(Class~., data=train)
# Plot resulting tree
plot_tree(tree1)
# Remove second observation
train2 = train[-2,]
# Repat fitting and plotting
tree2 = tree(Class~., data=train2)
plot_tree(tree2)

# Comment:
# This observation "ruins" the purity of all the observations with
A11>=0.5
# And this the decision tree must add more splits until it finds a
pure set

## Task 2
# Selected # of leaves by cross validation
set.seed(12345)
cv.tree = cv.tree(tree1)
plot(cv.tree$size, cv.tree$dev, type="b", col="blue",
     main="Cross Validation results, Deviance vs. # leaves")
min = cv.tree$size[which.min(cv.tree$dev)] # 10 leaves
# Comment:
# According to the plot 10 leaves provides the minimum deviance,
# However, the deviance does not reduce greatly from 3 to 10
# Finally, less leaves reduce the risk of overfitting and thus I will
select 3 leaves
# since the reduction in deviance is not that great whereas the risk
of overfitting is increasing

# Prune the tree
tree.pruned = prune.tree(tree1, best=min)
plot_tree(tree.pruned)

## Task 3
x.train = model.matrix(~ .-1, train[,-16])
y.train = train[,16]
# See the coefficients to ensure everything is correct
lasso = glmnet(x.train, y.train, family="binomial", alpha=1)
plot(lasso, xvar="lambda", TRUE, main="Lasso Regression")

# Select best number of variables through cross-validation
set.seed(12345)
lasso.cv = cv.glmnet(x.train, y.train, family="binomial", alpha=1)
plot(lasso.cv, xvar="lambda", label=TRUE, main="Cross-Validation
LASSO", xlab="LogLambda")
lambda.min = lasso.cv$lambda.min # "best" lambda = 0, no
penalty/regularization #0.00860862
coef(lasso.cv, s="lambda.min") # 23 variables selected
# Comment:
# Selected the binomial family since the output is a factor with two
levels with a binary output
# The best deviance was achieved with the penalty factor lambda =
0.008608627
# 23 variables are selected at the optima,
# By looking at the plot we can see that the optimal models does
indeed look statstically
# significantly better than the model with the smallest penalty factor
of lambda
# Looking at the confidence bands it seems highly unlikely that the
smallest penalty factor
# would have a superior model than the optimal model calculated here

## Task 4
# Make predictions
# Tree
p.tree = predict(tree1, test, type="vector")[,1]
p.pruned.tree = predict(tree.pruned, test, type="vector")[,1] # To get
the prediction probabilities
# Lasso
x.test = model.matrix(~ .-1, test[,-16])
p.lasso = predict(lasso.cv, x.test, s="lambda.min", type="response") #
To get predictions as probabilities

# Calculate E
y.test = as.numeric(test[,16])-1
E.tree = E(p.tree, y.test) # -Inf, since predicts 100% wrong at
occasions, overfitted
E.pruned.tree = E(p.pruned.tree, y.test) # -251.5017054
E.lasso = E(p.lasso, y.test) # -41.9737668
```

```
# Comment:
# Lasso outperforms the decision Trees
# This criterion is sometimes more reasonable to use than
missclassification rate since
# it includes information about how certain the predictions are, where
confident eg. p=0.99 -> R=-0.01005034
# correct predictions are punished less than "lucky" correct
predictions eg. p=0.51 -> R = -0.6733446
# Both make the correct prediction given that class is 1 but the more
confident prediction contributes
# less to the error.
# Suitible when you want to know which model is the most confident in
its decision making
```