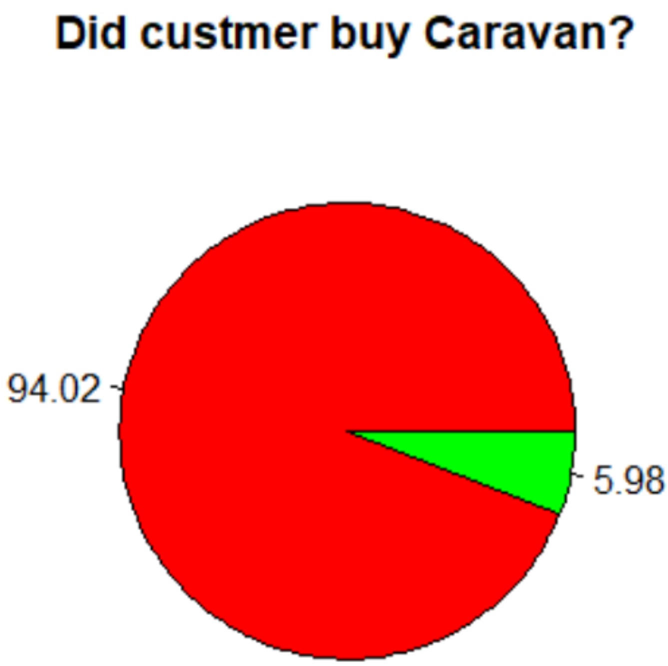
Rustam Fadeev - Homework 2

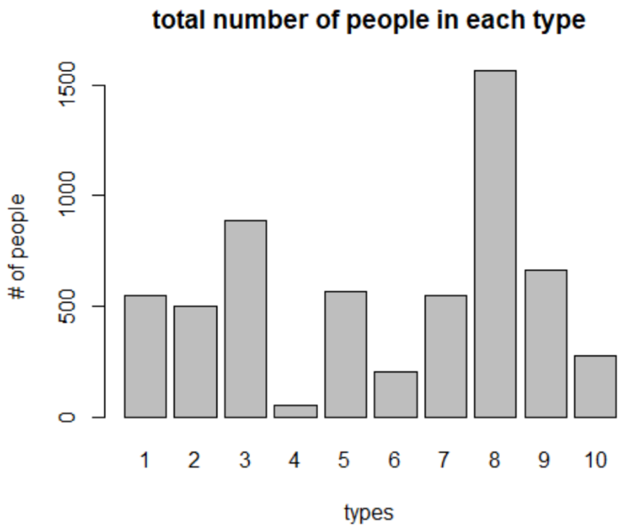
Preparation

```
library(ISLR)
 str(Caravan)
library(ROCR)
### Prepare ###
## 1000 for test data, rest for train
set.seed(321)
test_index <- sample(seq_len(nrow(Caravan)), size = 1000)

test <- Caravan[test_index, ]
train <- Caravan[-train_index, ]

### Check distribution of target atribute
target_attribute<-table(Caravan$Purchase) # Number of people who purchased and didn't
percent <- round(100*target_attribute/sum(target_attribute),2) # calculate %
colors=c("red","green")
pie(target_attribute,main = "Did custmer buy Caravan?",col=colors, labels = percent)

#
# We can see that only 6% of customers bought Caravan. Distribution is uneven
# and we should consider this
#
```

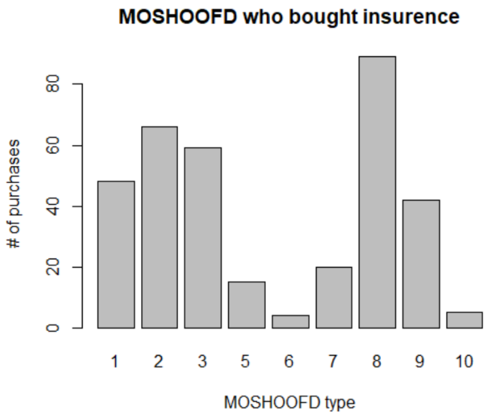


1a)

```
### 1A customer type MOSHOOFD

#customer_main_type<-table(Caravan$MOSHOOFD, Caravan$Purchase)
customer_main_type<-table(Caravan$MOSHOOFD) # total number of people in each type
barplot(customer_main_type,
        main = "total number of people in each type",
        xlab="types",
        ylab="# of people")
## We can see that most common type is 3,8,9
```

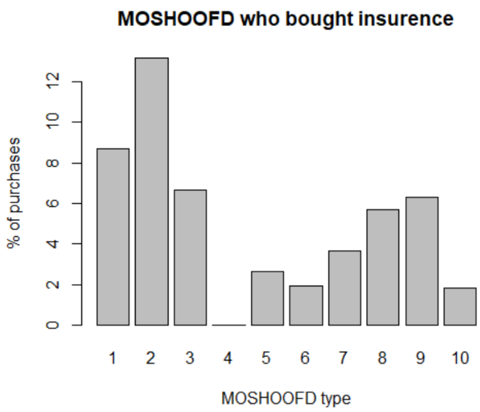



```
# number of people per type who bought insurence
customer_main_type_purchased<-table(Caravan$MOSHOOFD[Caravan$Purchase=="Yes"])

## We can see that type 4 didn't buy Caravan at all
## types 2,3 and 8 mostly were buying the insurence
barplot(customer_main_type_purchased,
        main = "MOSHOOFD who bought insurence",
        xlab="MOSHOOFD type",
        ylab="# of purchases")
```

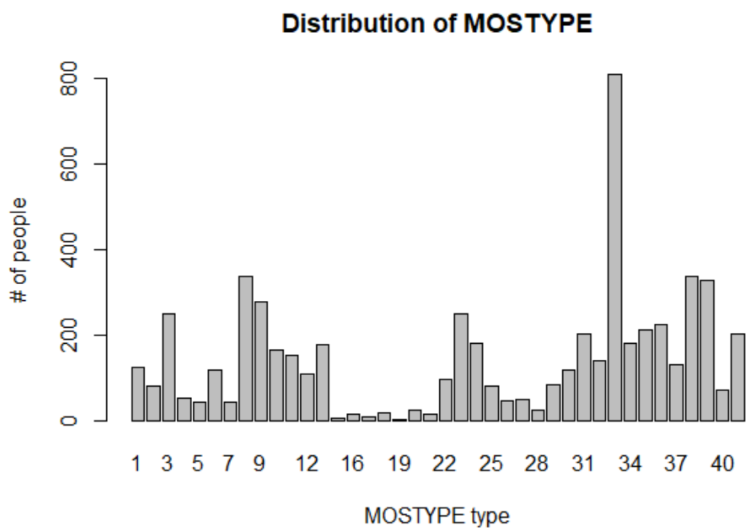



```
# Compute percentage. Note that 4th row is empty, so we have to combine several computations together
percent_first_half <- round(100*customer_main_type_purchased[1:3]/customer_main_type[1:3],2)
percent_forth <- 0
names(percent_forth)<-data.frame(4)
percent_second_half <- round(100*customer_main_type_purchased[4:9]/customer_main_type[5:10],2)
percent_cmt <- c(percent_first_half, percent_forth,percent_second_half)

#    1     2     3     4     5     6     7     8     9    10
#  8.70 13.15  6.66  0.00  2.64  1.95  3.64  5.69  6.30  1.81

# build plot of %
barplot(percent_cmt,
        main = "MOSHOOFD who bought insurence",
        xlab="MOSHOOFD type",
        ylab="% of purchases")
## We can see that by looking at %, most of type 2 people have insurence
```

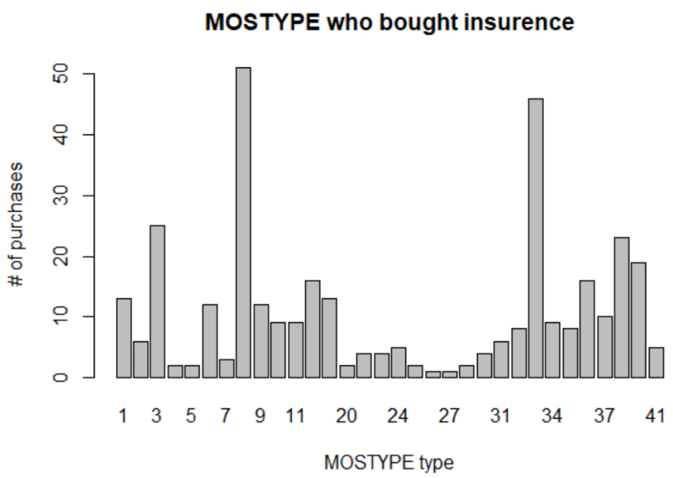
```
> percent_cmt
    1     2     3     4     5     6     7     8     9    10
 8.70 13.15  6.66  0.00  2.64  1.95  3.64  5.69  6.30  1.81
```




```
## Look at MOSTYPE
customer_sub_type<-table(Caravan$MOSTYPE) # total number of people in each type
barplot(customer_sub_type,
        main = "Distribution of MOSTYPE",
        xlab="MOSTYPE type",
        ylab="# of people")
## We can see that top three customer subtypes are 33, then 8, then 38
```



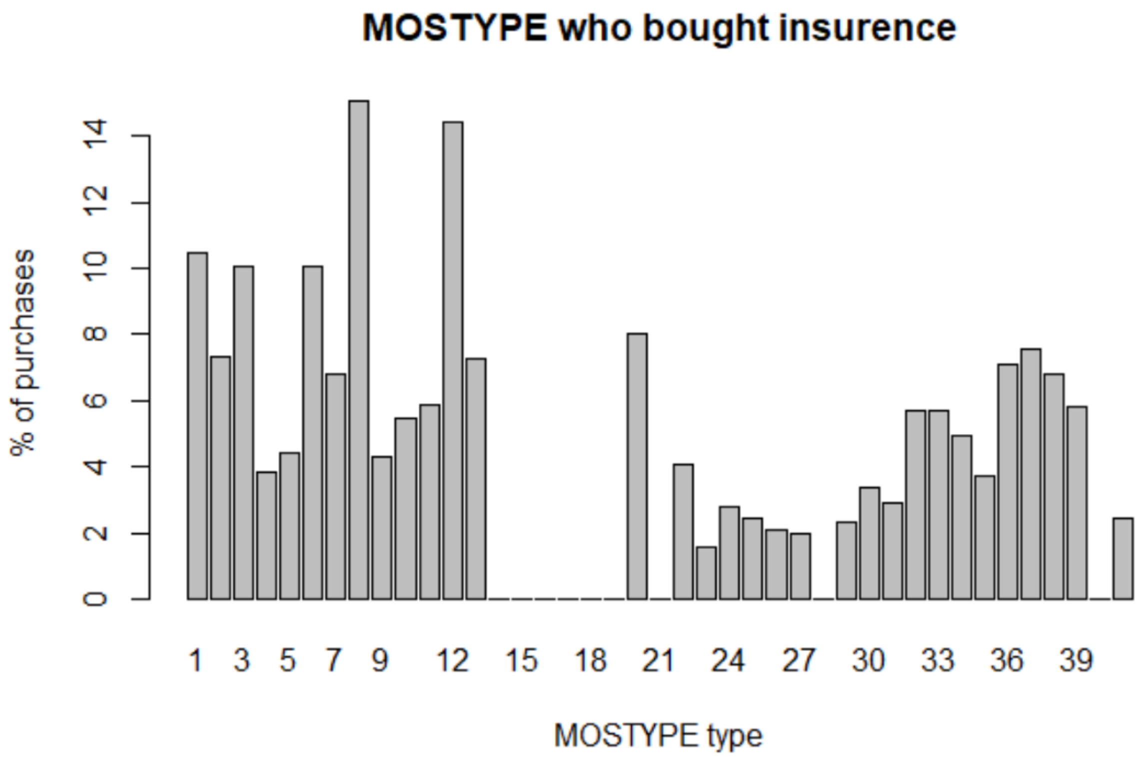

```
# number of people per type who bought insurence
customer_sub_type_purchased<-table(Caravan$MOSTYPE[Caravan$Purchase=="Yes"])

## We can see that type 4 didn't buy Caravan at all
## types 2,3 and 8 mostly were buying the insurence
barplot(customer_sub_type_purchased,
        main = "MOSTYPE who bought insurence",
        xlab="MOSTYPE type",
        ylab="# of purchases")

# We see that mostly people of subtype 8  (Middle class families) and 33 (Lower class large families) purchased caravan policy
```




```
# Compute percentage. Note many rows are empty, so we have to combine several computations together
# Row 14 in original set is also empty
percent_first <- round(100*customer_sub_type_purchased[1:13]/customer_sub_type[1:13],2)
percent_14_19 <- replicate(6, 0)
names(percent_14_19)<-data.frame(14,15,16,17,18,19)
percent_14_19 <- replicate(6, 0)
names(percent_14_19)<-data.frame(14,15,16,17,18,19)
percent_21 <- 0
names(percent_21)<-data.frame(21)
percent_28 <- 0
names(percent_28)<-data.frame(28)
percent_40 <- 0
names(percent_40)<-data.frame(40)
percent_20 <-round(100*customer_sub_type_purchased[14]/customer_sub_type[19],2)
percent_second <- round(100*customer_sub_type_purchased[15:20]/customer_sub_type[21:26],2)
percent_third <- round(100*customer_sub_type_purchased[21:31]/customer_sub_type[28:38],2)
percent_41 <-round(100*customer_sub_type_purchased[32]/customer_sub_type[40],2)
percent_cst <- c(percent_first, percent_14_19,percent_20, percent_21,percent_second, percent_28,percent_third, percent_40,percent_41)

# See top %
percent_cst_top <- percent_cst[order(percent_cst,decreasing = TRUE)]

# build plot of %
barplot(percent_cst,
        main = "MOSTYPE who bought insurence",
        xlab="MOSTYPE type",
        ylab="% of purchases")
## We can see that by looking at %,
## our top is type 8 (15.04%), type 12 (14.41%), type 1 (10.48%)/
```

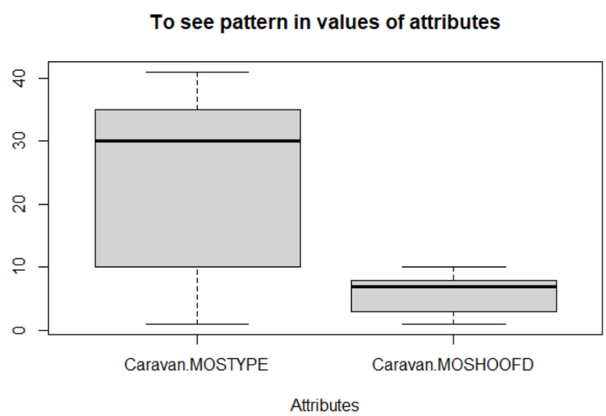
Some of percent_cst_top

```
> percent_cst_top
    8    12     1     6     3    20    37     2    13
15.04 14.41 10.48 10.08 10.04  8.00  7.58  7.32  7.26
```



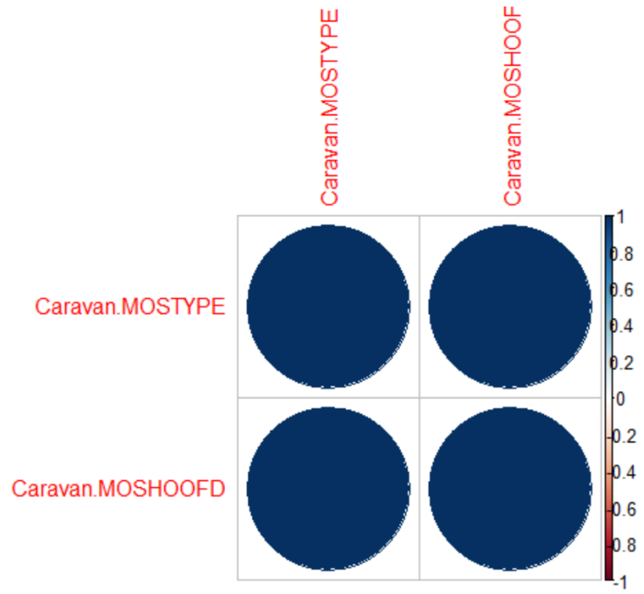
1b

```
### 1b Comparing MOSTYPE and MOSHOOFD
library(psych)
Comparing<-data.frame(Caravan$MOSTYPE,Caravan$MOSHOOFD)
round(describe(Comparing),3) #

boxplot(Comparing,main ="To see pattern in values of attributes",
        xlab = "Attributes")
```




```
library(corrplot)
cor(x=Comparing$Caravan.MOSTYPE,y=Comparing$Caravan.MOSHOOFD)
# [1] 0.9926719
corrplot(cor(Comparing))
# We clearly see positive correlation between MOSTYPE and MOSHOOFD
```

```
# We can use MOSTYPE to predict MOSHOOFD and vice versa if needed
## We can observe that people who purches insurnce is mostly Middle/Low class traditional families.
```



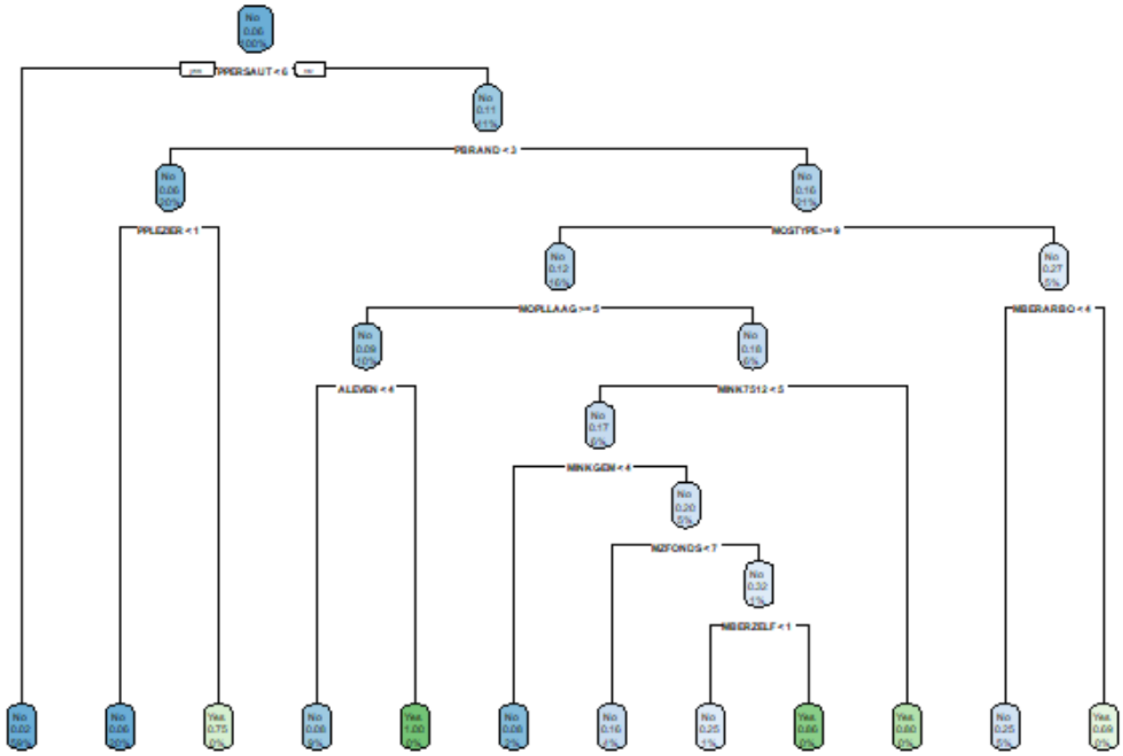
Task 2

2a

```
### TASK 2 ###

### Decision Tree ###
library(rpart)
library(rpart.plot)
#library(RColorBrewer)
library(ROCR)       # plotting ROC curve
library(crossval)   # evaluation
train_no_purchase <- train[,1:85]
train_dt<-train
train_dt$Purchase = as.factor(train_dt$Purchase)
tree.model <- rpart(Purchase ~., data = train_dt,control=rpart.control(minsplit=20, minbucket=1, cp=0.005))
rpart.plot(tree.model)
printcp(tree.model)
bestcp<-tree.model$cptable[which.min(tree.model$cptable[,"xerror"]),"CP"]
# [1] 0.005454545

test_dt <- test[,1:85]
prediction <- predict(tree.model, test_dt, type="class")
table(prediction)
confusionMatrix(test$Purchase, prediction)
# Accuracy : 0.948
# Balanced Accuracy : 0.64159
# Kappa : 0.0315
```

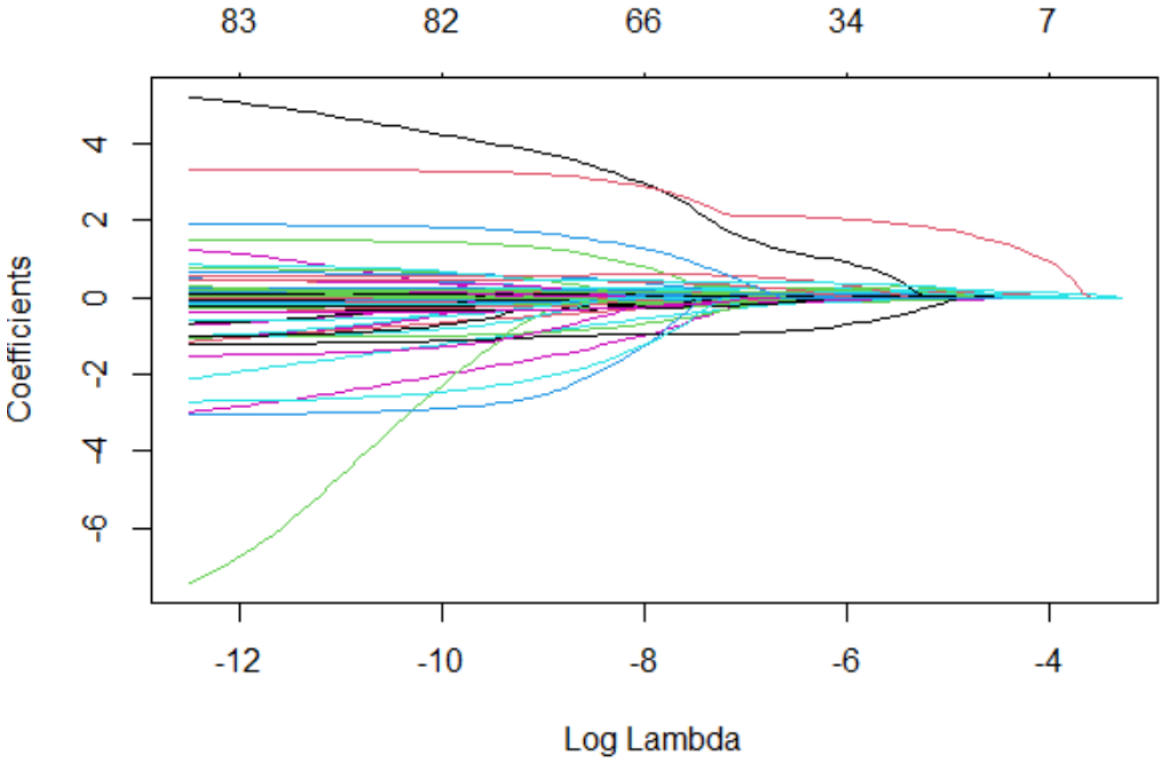
```
### OUTPUT OF printcp(tree.model)
Classification tree:
rpart(formula = Purchase ~ ., data = train_dt, control = rpart.control(minsplit = 20,
    minbucket = 1, cp = 0.005))

Variables actually used in tree construction:
 [1] ALEVEN   MBERARBO MBERZELF MINK7512 MINKGEM
 [6] MOPLLAAG MOSTYPE  MZFONDS  PBRAND   PPERSAUT
[11] PPLEZIER

Root node error: 330/5508 = 0.059913

n= 5508
```

```
         CP nsplit rel error xerror     xstd
1 0.0054545      0   1.00000 1.0000 0.053374
2 0.0050505      5   0.97273 1.0152 0.053751
3 0.0050000     11   0.94242 1.0242 0.053975
```

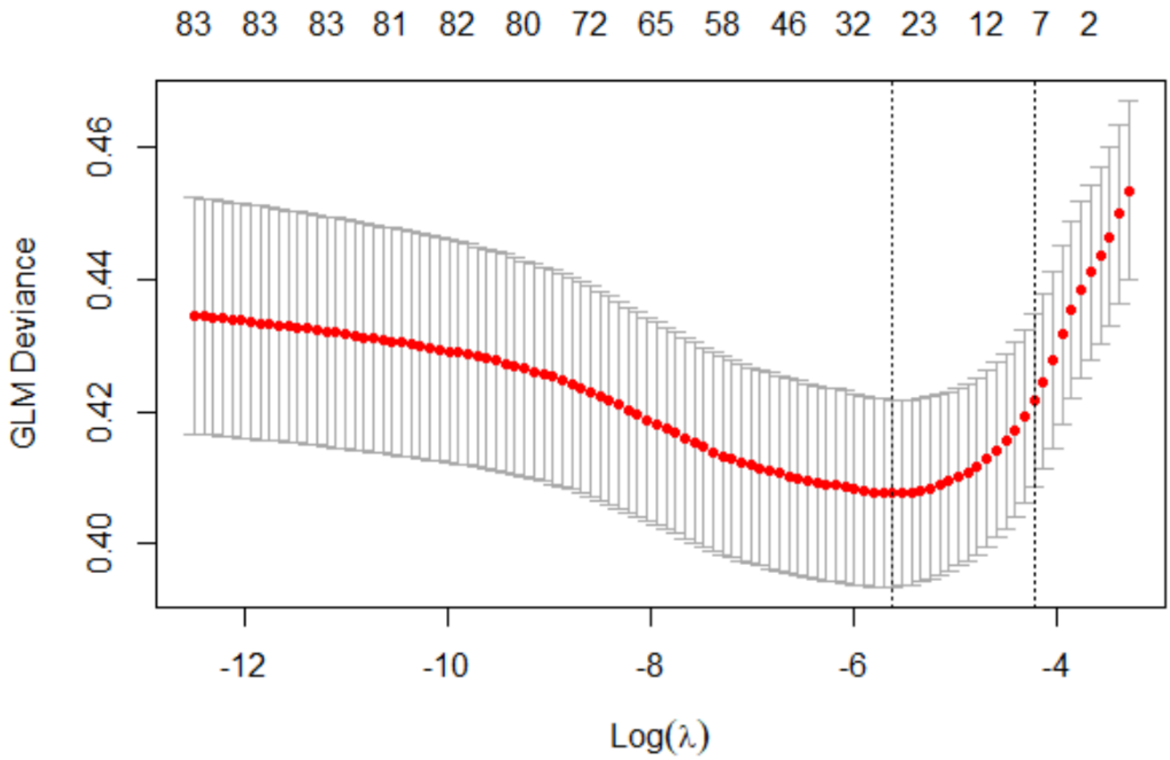


2c

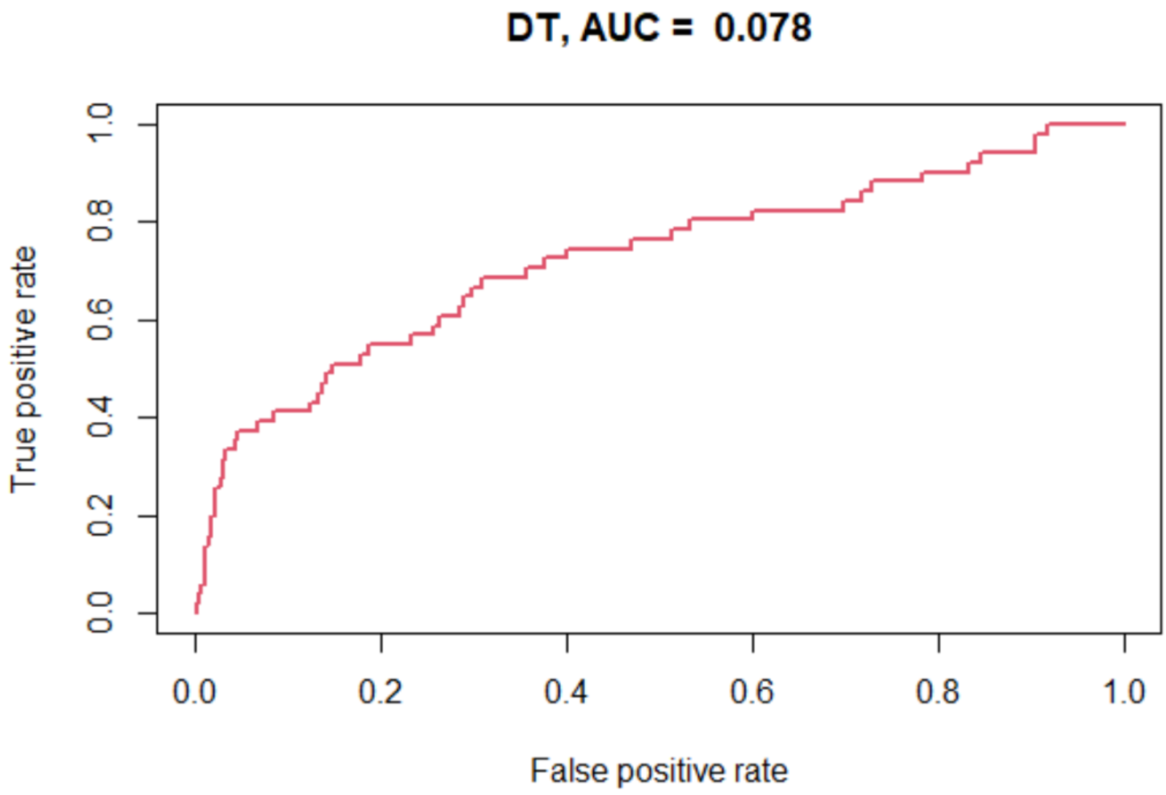
```
###  Regression

# Use glmnet to build Lasso model
# alpha=1 means to consider lasso, not its regression

# all attributes as matrix minus "intercept"
train_dt.matrix<-model.matrix(train$Purchase~., data=train)[,-1]
# Fit lasso model in training
L.model<-glmnet(train_dt.matrix, train$Purchase, alpha=1, family=binomial)
# coeff vs Log Lambda
plot(L.model, xvar="lambda")

## Looking at graph we can see that higher lambda leads to coefficients going to 0
## Now to choose the best lambda, we use cross-validation
```




```
set.seed(123)
# Fit lasso model in training with cross-validation
L.model.CV <- cv.glmnet(train_dt.matrix, train$Purchase, alpha=1,k=10, family=binomial("logit"))
# MSE vs log Lambda
plot(L.model.CV)
# find minimum lambda
lambda_min <- L.model.CV$lambda.min
# 0.003609857

## Now, using this minimum lambda we can train model again and exclude all 0-coeff attributes
## It will help building lighter and better model
```




```
# compute # of attributes with non-zero coeff
sum(predict(L.model, s=lambda_min, type="coefficients")!=0)-1
# 24

##
### Analyse Linear Regression Model
##
library(modelr)
test.matrix<-data.matrix(test[1:85])

LinReg.predict<-predict(L.model.CV, s=lambda_min,newx=test.matrix,type="response")
LinReg.pred <- prediction(LinReg.predict,test$Purchase)
performanceLR <- performance(LinReg.pred, fpr.stop=0.2, measure = "tpr",
                             x.measure = "fpr")
cutoffs <- data.frame(cut=performanceLR@alpha.values[[1]],
                      fpr=performanceLR@x.values[[1]],
                      tpr=performanceLR@y.values[[1]])

## Show AUC with fpr>0.2
pauc.dt <- round(performance(LinReg.pred, measure = "auc",fpr.stop=0.2)@y.values[[1]], 3)
plot(performance(LinReg.pred, measure='tpr', x.measure='fpr'),
     main=paste('DT, AUC = ', pauc.dt), col=2, lwd = 2)
## AUC = 0.078 with fpr.stop=0.2

## Log performance
perf.acc.log <- performance(LinReg.pred, measure = "acc", )
perf.tpr.log <- performance(LinReg.pred, measure = "tpr")
perf.fpr.log <- performance(LinReg.pred, measure = "fpr")
```



Task 3

As we have seen from the tree

```
Variables actually used in tree construction:
 [1] ALEVEN   MBERARBO MBERZELF MINK7512 MINKGEM
 [6] MOPLLAAG MOSTYPE  MZFONDS  PBRAND   PPERSAUT
[11] PPLEZIER
```

At the same time lasso chose these attributes

```
predict(L.model, s=lambda_min, type="coefficients")!=0

(Intercept) |
MGEMLEEF    |
MGODGE      |
MRELGE      |
MOPLHOOG    |
MOPLLAAG    |
MBERBOER    |
MBERMIDD    |
MHHUUR      |
MAUT1       |
MINK7512    |
MINK123M    |
MINKGEM     |
MKOOPKLA    |
PWAPART     |
PPERSAUT    |
PGEZONG     |
PBRAND      |
PFIETS      |
AWALAND     |
ATRACTOR    |
AZEILPL     |
APLEZIER    |
```

```
AFIETS      |
ABYSTAND    |
```

We can see that only these attributes appear in both models:

MINK7512, MINKGEM, MOPLLAAG, PBRAND,PPERSAUT

MINK7512 represents medium-high income(75→122.000)

MINKGEM also shows us information about the income, but average

MOPLLAAG is lower-level education

PBRAND represents contribution fire policies

Finally, PPERSAUT represents contribution car policies

Lasso is using much more attributes for the predictions.

Task 4

```
### Predict given test vector
Dtest <- read.csv("https://ufal.mff.cuni.cz/~holub/2021/docs/caravan.test.1000.csv", sep="\t", header=FALSE)
dim(Dtest)
str(Dtest)

FinalTest.matrix<-data.matrix(Dtest)
RegressionTest<-predict(L.model.CV, s=lambda_min,newx=FinalTest.matrix,type="response")
RT <- RegressionTest
oneHP<-RegressionTest[order(RegressionTest,decreasing = TRUE)][100]
## This way we can see sorted %s. We see that 100th % is 0.1296366
## Everything lower than this will be 0, other 100 values will be 1
RT[RT>=oneHP]<-1
RT[RT<oneHP]<-0

## Create file
write.table(RT,file="T.prediction.txt",row.names = FALSE)
```