EXAM V22 BAN404

Candidate 69

2023-05-09

```
# Loading libraries relevant for the exam
library(ISLR)
library(stargazer)
```

Task 1

First of all, we load in the required data set which will be used for this task. In this case, the data set is called `OJ` and is gathered from the `ISLR` package.

```
data(OJ, package = "ISLR")
```

a)

First of all, we will determine which variables are not fit for the analysis. I will use some inference gathered from the `?OJ` function in R, which describes the data set. This will form the basis of my reasoning.

I will assume that the week a purchase has been made does not form a causal relationship with which juice is bought. It is hard to determine whether or not there are any structural diffrences between the stores, but in order to remove any possibilities of multicolinearity, only the STORE variable will be kept from the variables showing stores.

```
var_remove <- c("WeekofPurchase", "Store7", "StoreID")    # Defining which columns should be removed
OJ <- OJ[, !names(OJ) %in% var_remove]
rm(var_remove)                                            # Deleting the redundant list from memory
```

Next, I will reprogram some of the classes of the variables.

```
data.frame(sapply(OJ,class))
```

```
##               sapply.OJ..class.
## Purchase                 factor
## PriceCH                 numeric
## PriceMM                 numeric
## DiscCH                  numeric
## DiscMM                  numeric
## SpecialCH               numeric
## SpecialMM               numeric
## LoyalCH                 numeric
## SalePriceMM             numeric
## SalePriceCH             numeric
## PriceDiff               numeric
## PctDiscMM               numeric
## PctDiscCH               numeric
## ListPriceDiff           numeric
```

```
## STORE                     numeric
```

From what I can gather, the `StoreID`, `SpecialCH` and `SpecialMM` should be reprogrammed as factors. The rest of the variables look right.

```
col_factor <- c("STORE", "SpecialCH", "SpecialMM")

OJ[, col_factor] <- lapply(OJ[, col_factor], factor)
rm(col_factor)
```

For all the tasks expect **f)**, the variable `LoyalCH` will be assumed to be unknown. I will therefore create two distinct data sets, one with the variable known and the other it will be unknown. X will mark the data which is unknown, and Z will mark the data where the Loyalty is known.

```
# Unknown Data set
xdata <- OJ[, !names(OJ) %in% c("LoyalCH")]

# The data set which "LoyalCH" is known is stored in zdata
zdata <- OJ
```

Now, we will use cross validation in order to perform the rest of the tasks.

```
{
# Setting the seed
set.seed(8655)

# Apply a 50/50 cross validation technique
n <- nrow(xdata)
ntrain <- floor(n/2)
ind <- sample(1:n, ntrain)

xtrain <- xdata[ind,]
xtest <- xdata[-ind,]
}
```

b)

In order to get a grasp of the data, I will present some measures of the central tendencies of the various variables. I will use the xdata data frame in order to do this initial analysis.

```
stargazer(xdata[xdata$Purchase == "CH", ], type = "text", align=T, digits=4, no.space=T)
```

```
##
## ================================================
## Statistic       N    Mean   St. Dev.    Min     Max
## ------------------------------------------------
## PriceCH        653 1.8693   0.0967   1.6900  2.0900
## PriceMM        653 2.1014   0.1152   1.6900  2.2900
## DiscCH         653 0.0689   0.1345   0.0000  0.5000
## DiscMM         653 0.0949   0.1821   0.0000  0.8000
## SalePriceMM    653 2.0066   0.2259   1.1900  2.2900
## SalePriceCH    653 1.8004   0.1539   1.3900  2.0900
## PriceDiff      653 0.2061   0.2455   -0.6700 0.6400
## PctDiscMM      653 0.0458   0.0871   0.0000  0.4020
## PctDiscCH      653 0.0364   0.0714   0.0000  0.2527
## ListPriceDiff  653 0.2321   0.1005   0.0000  0.4400
## ------------------------------------------------
```

The variables that I found promising: - PriceDiff - SpecialCH - SpecialMM - STORE

c)

Using the variables which were found interesting in the last task, I will create a logistic regression in order to try correctly classify which juice was bought at each purchase. As this is a classification problem, the logistic regression method is well suited, as it assign's a value between 0 and 1 for each observation based on the explanatory variables.

```
# The Regression Model
logfit1 <- glm(Purchase~PriceDiff+SpecialCH+SpecialMM+STORE, data = xtrain, family = "binomial")
summary(logfit1)
```

```
##
## Call:
## glm(formula = Purchase ~ PriceDiff + SpecialCH + SpecialMM +
##     STORE, family = "binomial", data = xtrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9969  -0.8883  -0.5910   1.0075   2.1749
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.81809    0.22455  -3.643 0.000269 ***
## PriceDiff   -2.38657    0.43749  -5.455 4.89e-08 ***
## SpecialCH1   0.07906    0.31467   0.251 0.801614
## SpecialMM1   0.12195    0.31250   0.390 0.696365
## STORE1       0.70404    0.32049   2.197 0.028037 *
## STORE2       1.15935    0.28318   4.094 4.24e-05 ***
## STORE3       1.90405    0.31126   6.117 9.52e-10 ***
## STORE4      -0.36091    0.36771  -0.982 0.326331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 714.98  on 534  degrees of freedom
## Residual deviance: 605.89  on 527  degrees of freedom
## AIC: 621.89
##
## Number of Fisher Scoring iterations: 4
```

The Special-variables seems to be not significant, which leads me to create a new regression with those removed:

```
# The Regression Model
logfit2 <- glm(Purchase~PriceDiff+STORE, data = xtrain, family = "binomial")
```

With the model fitted, it is good measure to test the results. I will make use of test accuracy and a confusion matrix.

```
# The Prediction on the test set
logpred <- predict(logfit2, newdata = xtest, type = "response")

# The Specified Threshold
th <- 0.5
```

```
predth <- logpred>th

# Confusion Matrix and accuracy
confMatrix <- table(predth, xtest$Purchase)
accuracy <- sum(diag(confMatrix))/sum(confMatrix)

# Displaying the Matrix
confMatrix
```

```
##
## predth   CH  MM
##   FALSE 261  86
##   TRUE   65 123
```

```
# Displaying the Accuracy
paste0("The accuracy of the model: ", round(accuracy * 100, 3), "%")
```

```
## [1] "The accuracy of the model: 71.776%"
```

Comparing this to the "baseline" prediction, which is the accuracy of the model if it predicts the most common answer to all new observations:

```
# Total number of observations for each occurence
obs <- table(xtest$Purchase)

# Measuring the baseline accuracy
base_acc <- obs[[1]]/(obs[[1]] + obs[[2]])
paste0("The baseline accuracy of the model: ", round(base_acc * 100, 3), "%")
```

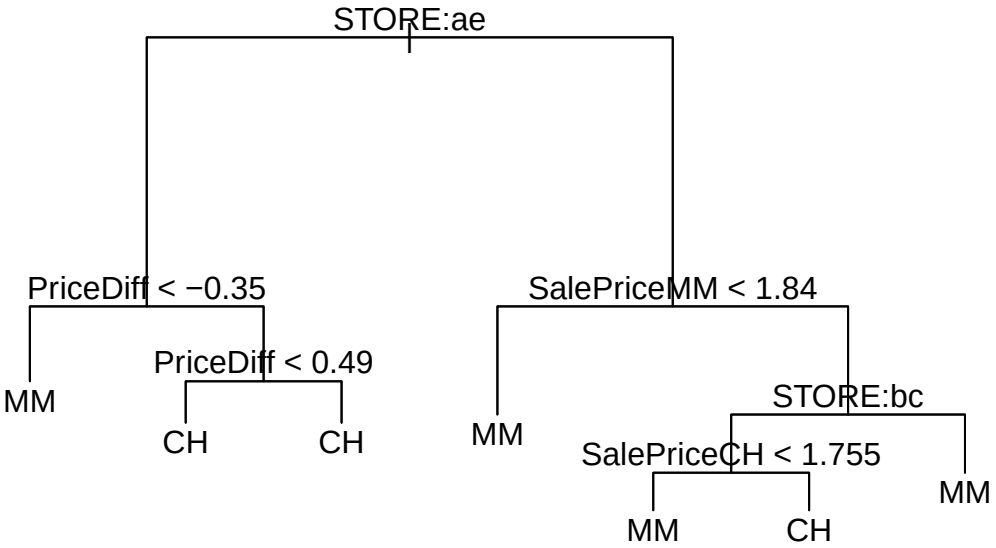
```
## [1] "The baseline accuracy of the model: 60.935%"
```

Comparing the model accuracy with the baseline accuracy, it shows some improvement, although it is not perfect in predicting which juice a given purchase included, although it is highly probable that is does explain some of the tendencies.

d)

In order to fit a classification tree to the data, I will make use of the same training and testing set established in the previous task. In order to extract inference from the decision tree, I will make a plot of it, explain how it can be used to make informed decisions, and make use of the accuracy and confusion matrix measures used in the previous task.

```
library(tree)

purtree <- tree(Purchase~., data=xtrain)
plot(purtree)
text(purtree, digits=3)
```



How to interpret a branch of the tree: A decision tree is a type of supervised learning algorithm that is mostly used for classification problems. It is made up of branch and leaf nodes. Branch nodes perform a logical check on the new data, which based on the output of TRUE/FALSE is sent further down the tree. If the logical check returns TRUE, the observation goes down the left side of the branch. When the observation reaches a leaf-node, the value of the leaf assigns a value to the new observation.

Confusion Matrix for the decision tree:

```
# The Prediction on the test set
treepred <- predict(purtree, newdata = xtest)

# The Specified Threshold
th <- 0.5

treepredth <- data.frame(treepred>th)
treepredth <- ifelse(treepredth$CH == TRUE, "CD", "MM")

# Confusion Matrix and accuracy
confMatrix <- table(treepredth, xtest$Purchase)
accuracy <- sum(diag(confMatrix))/sum(confMatrix)

# Displaying the Matrix
confMatrix
```

```
##
## treepredth  CH  MM
##         CD 238  69
##         MM  88 140
```

```
# Displaying the Accuracy
paste0("The accuracy of the model: ", round(accuracy * 100, 3), "%")
```

```
## [1] "The accuracy of the model: 70.654%"
```

e)

In this task we will try to maximize the threshold of predictions, in order to get the highest accuracy possible. I will use the decision tree as the basis for this operation.

```r
{
# Setting the seed
set.seed(4598)

# Apply a 50/50 cross validation technique
n <- nrow(xdata)
ntrain <- floor(n/2)
ind <- sample(1:n, ntrain)

xtrain <- xdata[ind,]
xtest <- xdata[-ind,]
}

purtree <- tree(Purchase~., data=xtrain)
treepred <- predict(purtree, newdata = xtest)


# Create a vector of threshold values to test
thresholds <- seq(0.1, 0.9, by = 0.1)

# Create an empty vector to store the accuracy scores
accuracy_scores <- numeric(length(thresholds))

# Loop through each threshold value and calculate the accuracy score
for (i in seq_along(thresholds)) {
  # Calculate the predicted values using the current threshold
  treepredth <- data.frame(treepred>thresholds[i])
  treepredth <- ifelse(treepredth$CH == TRUE, "CD", "MM")

  # Calculate the confusion matrix and accuracy score
  confMatrix <- table(treepredth, xtest$Purchase)
  accuracy_scores[i] <- sum(diag(confMatrix))/sum(confMatrix)
}

# Find the threshold value that gives the highest accuracy score
best_threshold <- thresholds[which.max(accuracy_scores)]

# Displaying the best threshold
paste0("The best threshold for this decision tree: ", best_threshold)
```

```
## [1] "The best threshold for this decision tree: 0.4"
```

f)

In this part of the task, I will use the data set denoted as `zdata` which contains the `LoyalCH` variable. This variable will be added to my analysis from the previous tasks in order to determine if this is a good predictor of `Purchase`.

In order to get a grasp of the values, We must first understand what it means. `LoyalCH` is a numeric variable which on a scale from 0-1 measures how loyal a customer is to the *CH* brand. if the value is 1, they are perfectly loyal, and if the value if 0, they are perfectly disloyal. This is important to remember when interpreting the output of the analysis.

Augmenting the Logistic Regression

```
# Setting the seed
set.seed(8655)

# Apply a 50/50 cross validation technique
n <- nrow(zdata)
ntrain <- floor(n/2)
ind <- sample(1:n, ntrain)

ztrain <- zdata[ind,]
ztest <- zdata[-ind,]

logfit2 <- glm(Purchase~PriceDiff+STORE+LoyalCH, data = ztrain, family = "binomial")
summary(logfit2)
```

```
##
## Call:
## glm(formula = Purchase ~ PriceDiff + STORE + LoyalCH, family = "binomial",
##     data = ztrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7072  -0.5733  -0.2495   0.5941   2.6592
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.6428     0.3715   7.114 1.13e-12 ***
## PriceDiff    -2.8546     0.4890  -5.837 5.30e-09 ***
## STORE1        0.2928     0.3683   0.795   0.4267
## STORE2        0.6149     0.3230   1.904   0.0569 .
## STORE3        0.5240     0.3872   1.353   0.1760
## STORE4        0.2908     0.4283   0.679   0.4971
## LoyalCH      -5.7318     0.5382 -10.651  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 714.98  on 534  degrees of freedom
## Residual deviance: 430.06  on 528  degrees of freedom
## AIC: 444.06
##
## Number of Fisher Scoring iterations: 5
```

According to this summary, the `LoyalCH` variable is a good predictor of the `Purchase` variable. In addition to this, the `STORE` variable is not now deemed significant in this context.

Next, I will make a confusion matrix based on these results, whilst also removing the `STORE` variable from the regression.

```
logfit3 <- glm(Purchase~PriceDiff+LoyalCH, data = ztrain, family = "binomial")

# The Prediction on the test set
logpred3 <- predict(logfit3, newdata = ztest, type = "response")
```

```r
# The Specified Threshold
th <- 0.5

predth <- logpred3>th

# Confusion Matrix and accuracy
confMatrix <- table(predth, ztest$Purchase)
accuracy <- sum(diag(confMatrix))/sum(confMatrix)

# Displaying the Matrix
confMatrix
```

```
##
## predth   CH  MM
##   FALSE 288  47
##   TRUE   38 162
```

```r
# Displaying the Accuracy
paste0("The accuracy of the model: ", round(accuracy * 100, 3), "%")
```

```
## [1] "The accuracy of the model: 84.112%"
```

This new model is much more accurate than the previous models!

Augmenting the decision tree classifying model

```r
# Setting the seed
set.seed(4598)

# Apply a 50/50 cross validation technique
n <- nrow(zdata)
ntrain <- floor(n/2)
ind <- sample(1:n, ntrain)

ztrain <- zdata[ind,]
ztest <- zdata[-ind,]

# Fitting the tree with the complete data set
purtree <- tree(Purchase~., data=ztrain)

# The Prediction on the test set
treepred <- predict(purtree, newdata = ztest)

# The Specified Threshold
th <- 0.4

treepredth <- data.frame(treepred>th)
treepredth <- ifelse(treepredth$CH == TRUE, "CD", "MM")

# Confusion Matrix and accuracy
confMatrix <- table(treepredth, ztest$Purchase)
accuracy <- sum(diag(confMatrix))/sum(confMatrix)

# Displaying the Matrix
confMatrix
```

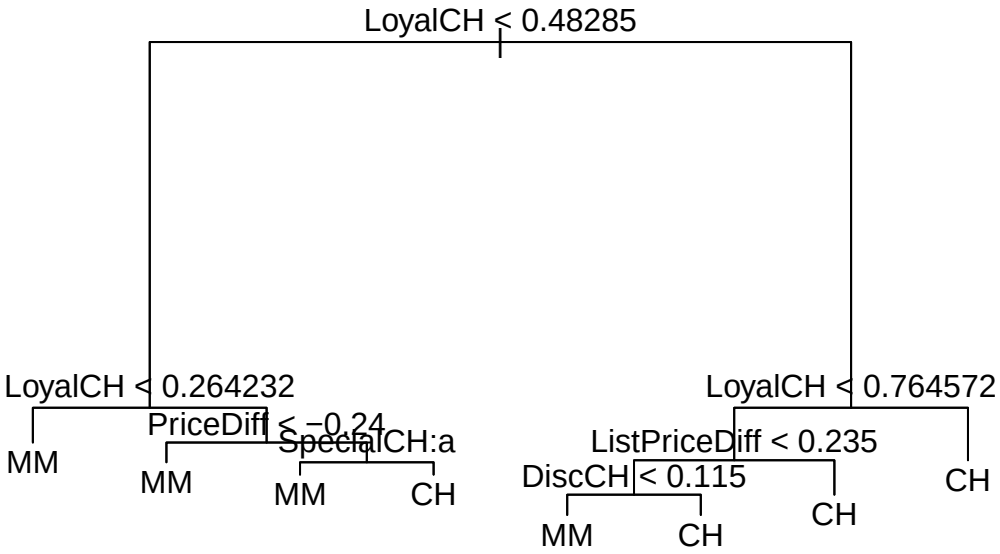
```
##
## treepredth  CH  MM
##         CD 288  66
##         MM  39 142
```

```
# Displaying the Accuracy
paste0("The accuracy of the model: ", round(accuracy * 100, 3), "%")
```

```
## [1] "The accuracy of the model: 80.374%"
```

The model is now more accurate than before as well! I can plot the new decision tree in order to see how it might have changed.

```
plot(purtree)
text(purtree, digits=3)
```



As is evident from the plotted tree, the `LoyalCH` variable is very important and significant when predicting using the decision tree classifier.

Task 2

In this part of the exam I will make use of the `Computers` data set from the `Ecdat` package. I will predict the variable `price` using the data contained in the data set.

First of all, I will load the data and clean the working memory which contains the data from the previous task.

```
# Removing leftover data
rm(list = ls())

# Loading in the data
data(Computers, package = "Ecdat")
xdata <- Computers
```

a)

First of all, we will determine if the variables has to be re-coded as factors.

	price	speed	hd	ram	screen	cd	multi	premium	ads	trend
	Min. : 999	25 : 289	Min. : 80	2 : 192	14:1806	no :1720	no :2694	no : 313	Min. : 39.0	Min.
	1st Qu.:1779	33 :1029	1st Qu.: 214	4 :1160	15:1030	yes:1409	yes: 435	yes:2816	1st Qu.:162.0	1st Q
	Median :2138	50 : 502	Median : 340	8 :1129	17: 293	NA	NA	NA	Median :246.0	Med
	Mean :2216	66 :1008	Mean : 413	16: 494	NA	NA	NA	NA	Mean :220.7	Mean
	3rd Qu.:2595	75 : 61	3rd Qu.: 528	24: 145	NA	NA	NA	NA	3rd Qu.:275.0	3rd Q
	Max. :5399	100: 240	Max. :2100	32: 9	NA	NA	NA	NA	Max. :339.0	Max

```
unique_count <- function(df){apply(df, 2, function(x) length(unique(x)))}
len <- unique_count(xdata)
len
```

```
##   price   speed      hd     ram  screen      cd   multi premium     ads   trend
##     808       6      59       6       3       2       2       2      34      35
```

The variables which has less than 5 unique observations could be recoded as factors. This is done like this:

```
for(i in 1:length(len)) if(len[i]<=6) xdata[,i]=as.factor(xdata[,i])
```

This re-coded the `screen` variable as a factor variable, as it only had 3 unique observations.

Then, we are able to remove variables which cannot be used in the analysis. I will make use of the `?Computers` function in order to get a better understanding of variables in the data set. From this we can assume that the number of listings is not relevant for the analysis. The other variables might prove useful later.

```
var_remove <- c("ads")                              # Defining which columns should be removed
#xdata <- xdata[, !names(xdata) %in% var_remove]
rm(var_remove)                                       # Deleting the redundant list from memory
```

With the variables that are going to be used established, we can split the data into a training and test set based on a 50/50 cross validation technique.

```
# Set the seed
set.seed(2652)

# Splitting the data 50/50 into a training and test set
n <- nrow(xdata)
ntrain <- floor(n/2)
ind <- sample(1:n, ntrain)

train <- xdata[ind,]
test <- xdata[-ind,]
```

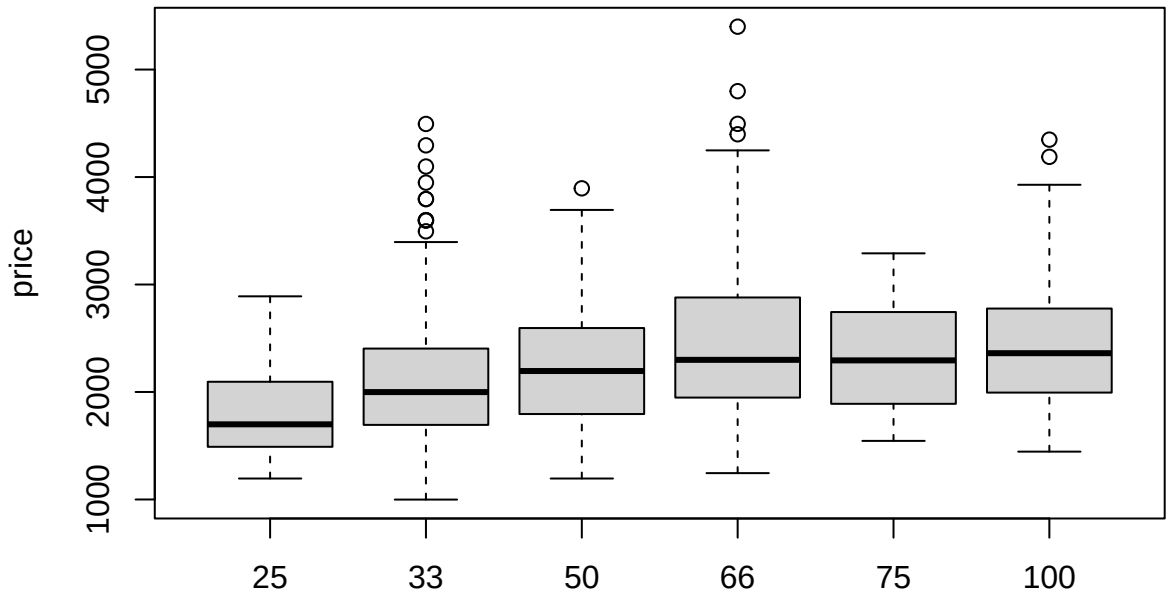
b)

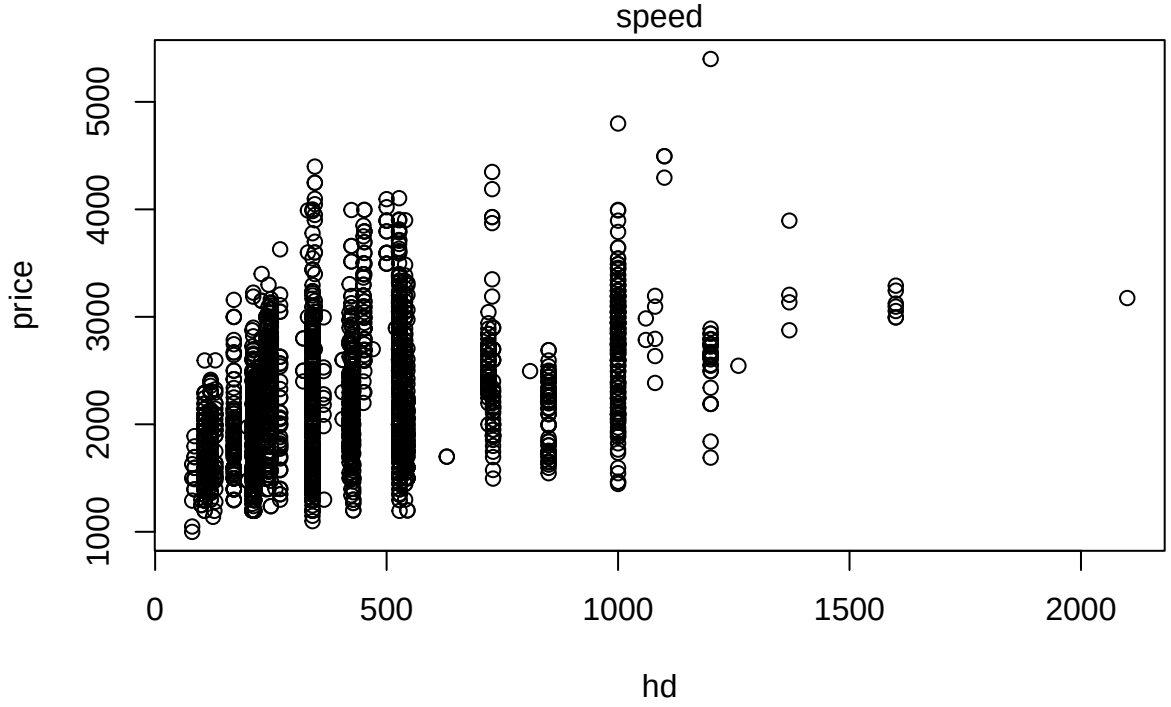
In this task I will use descriptive statistics in order to find some promising predictors of the price.

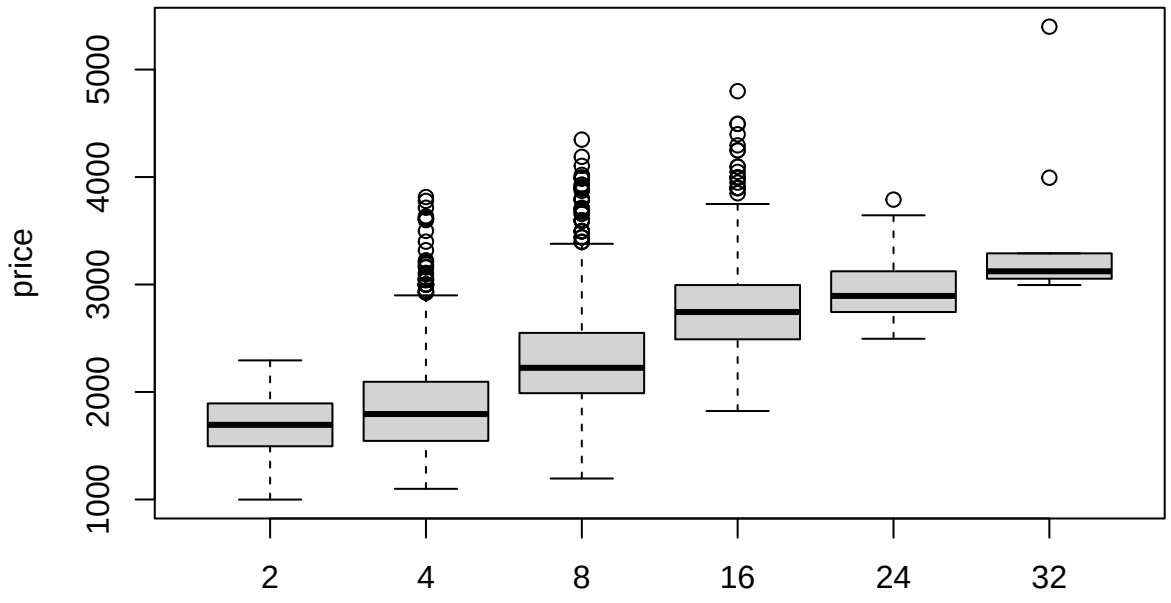
```
library(knitr)
library(kableExtra)

kable(summary(train)) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

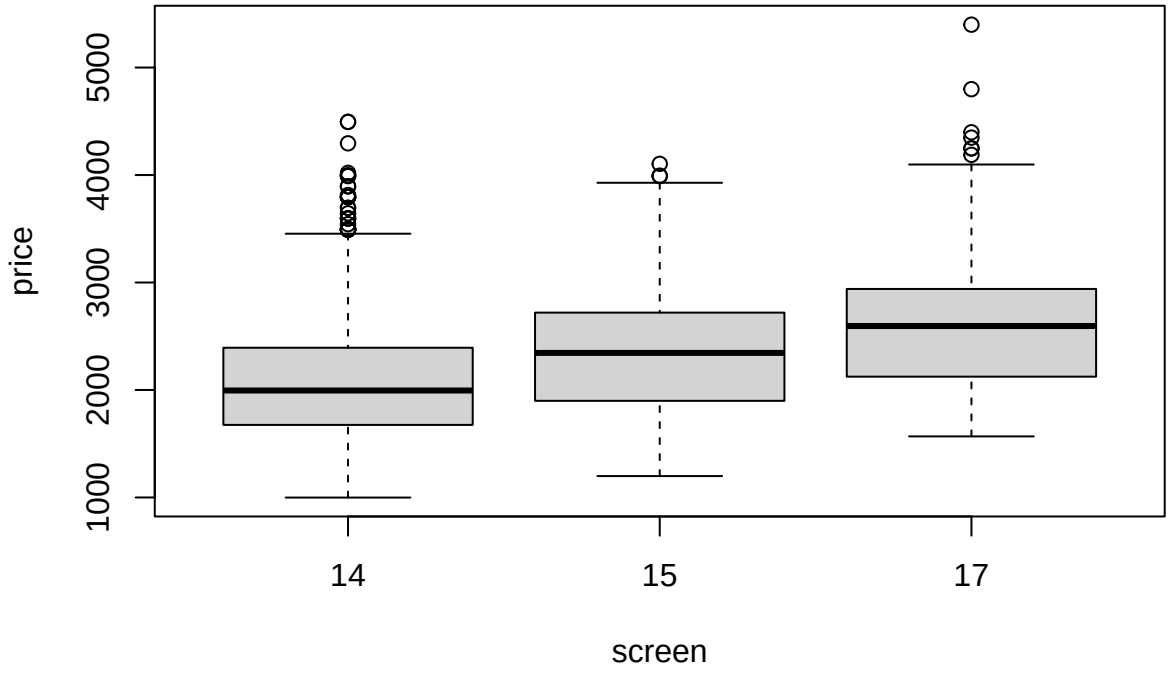
It is also useful to show the price plotted against each variable, in order to look for signs of correlation.

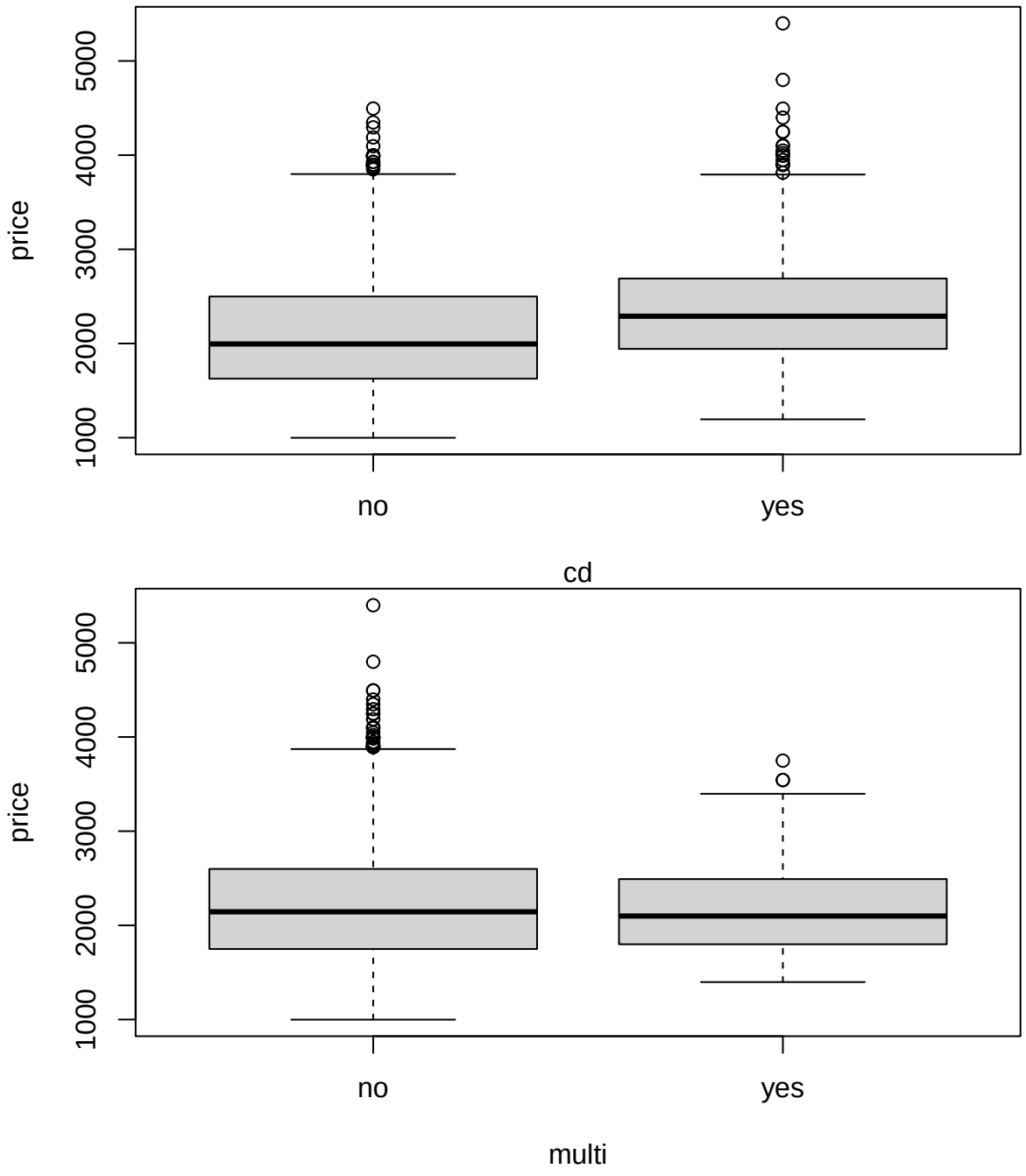
```
plot(price~., data = train)
```

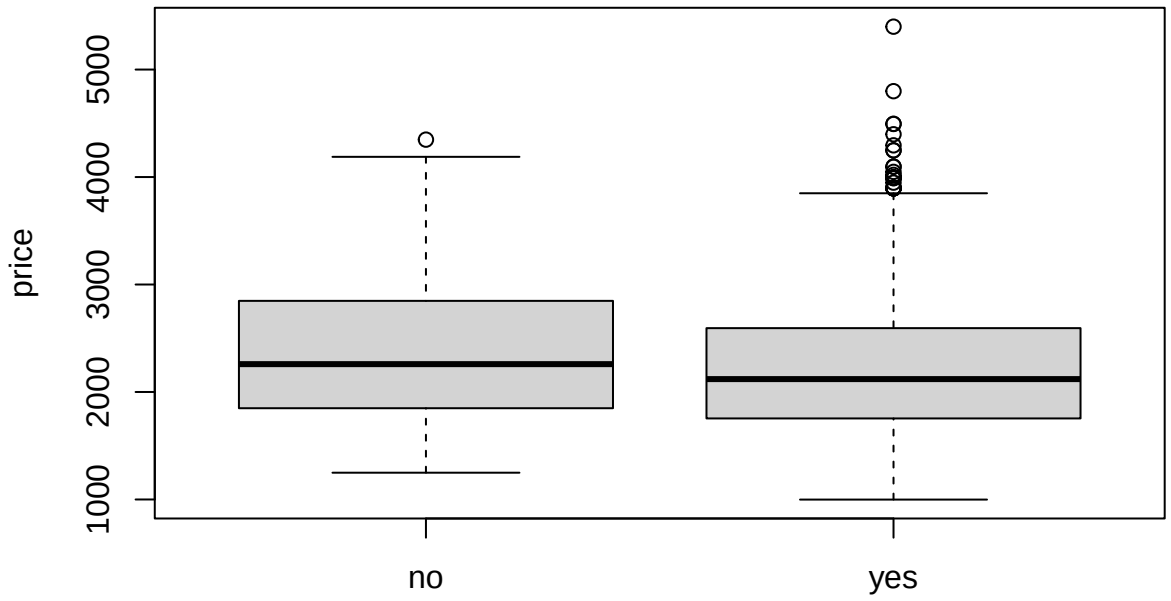

price
1000
2000
3000
4000
5000
25
33
50
66
75
100
speed

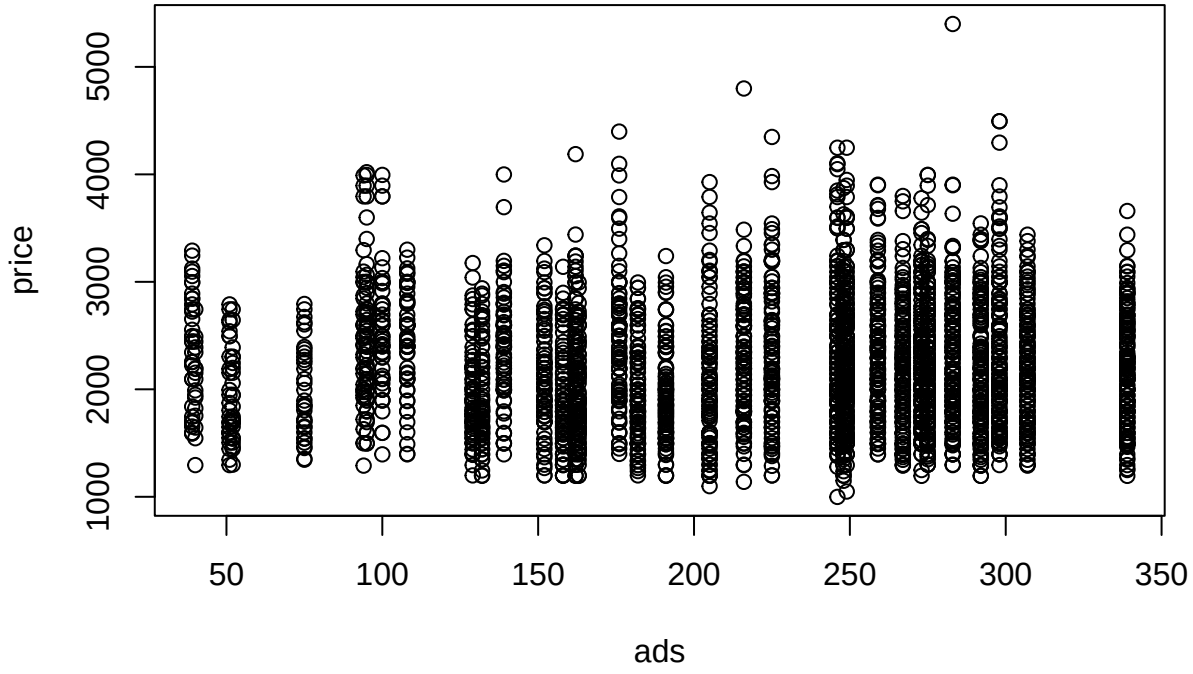

price
1000
2000
3000
4000
5000
0
500
1000
1500
2000
hd

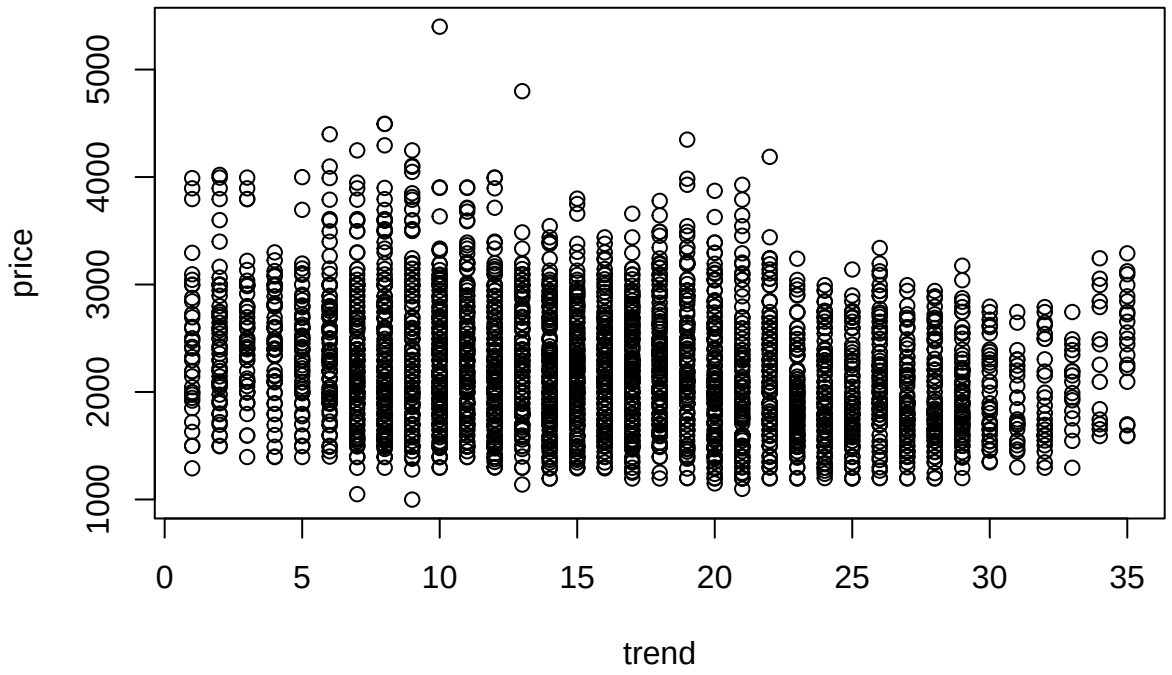

price
1000
2000
3000
4000
5000
2
4
8
16
24
32
ram


price
1000
2000
3000
4000
5000
14
15
17
screen


price
1000
2000
3000
4000
5000
no
yes
cd
price
1000
2000
3000
4000
5000
no
yes
multi


price
1000
2000
3000
4000
5000
no
yes
premium


price
1000
2000
3000
4000
5000
50
100
150
200
250
300
350
ads



From the outputted plots, the relationship between `price` and the variables `cd`, `hd`, `ram` and `screen` looks like the ones which could be promising predictors, as there is a change in the central tendancy of the price based on some of the factors described.

c)

Now I can run a linear regression on all explanatory variables. After this I will make use of appropriate performance measures of the prediction.

```
fit1 <- lm(price~., data = train)
summary(fit1)
```

```
##
## Call:
## lm(formula = price ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1021.57  -169.67   -12.89   138.07  1217.82
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2229.90935   35.64852  62.553  < 2e-16 ***
## speed33      291.87376   17.30884  16.863  < 2e-16 ***
## speed50      459.17175   19.44809  23.610  < 2e-16 ***
## speed66      588.47432   18.27995  32.192  < 2e-16 ***
## speed75      579.60641   37.75926  15.350  < 2e-16 ***
## speed100     784.14314   24.47870  32.034  < 2e-16 ***
## hd             0.78803    0.03813  20.665  < 2e-16 ***
## ram4          73.74723   20.83271   3.540 0.000406 ***
## ram8         352.25242   23.25438  15.148  < 2e-16 ***
## ram16        768.45720   27.74069  27.701  < 2e-16 ***
## ram24        966.28352   41.28142  23.407  < 2e-16 ***
```

```
## ram32       1330.59024   98.02365  13.574  < 2e-16 ***
## screen15      32.16443   10.77289   2.986 0.002852 **
## screen17     394.76119   16.53332  23.877  < 2e-16 ***
## cdyes         49.34966   12.70561   3.884 0.000105 ***
## multiyes     101.16274   15.14113   6.681 2.79e-11 ***
## premiumyes  -560.24360   16.41405 -34.132  < 2e-16 ***
## ads            0.59842    0.06861   8.722  < 2e-16 ***
## trend        -50.88145    0.84221 -60.414  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 255.2 on 3110 degrees of freedom
## Multiple R-squared:  0.8116, Adjusted R-squared:  0.8106
## F-statistic: 744.5 on 18 and 3110 DF,  p-value: < 2.2e-16
```

First of all, the summary of the regression shows that all explanatory variables are significant in the analysis. This is not an outragious statement as the computer market is very decentralized in the sense that many different companies make up a small part of a given computer, where the products they are selling are priced consistently. There might be just a few companies which control the production of RAM and CPUs (speed) which might explain why there is a consitant change in price when such components are "upgraded" in the spec sheet.

In order to get a better understanding of the regression, I will first measure the testMSE of this models predictions.

```
pred1 <- predict(fit1, newdata = test, type = "response")

testMSE <- mean((test$price - pred1)^2)
```

When there are many explanatory variables, models are prone to overfitting. I will therefore use a sub-setting method which might make it clear as to how many variables are needed in order to accurately predict the price of a computer.

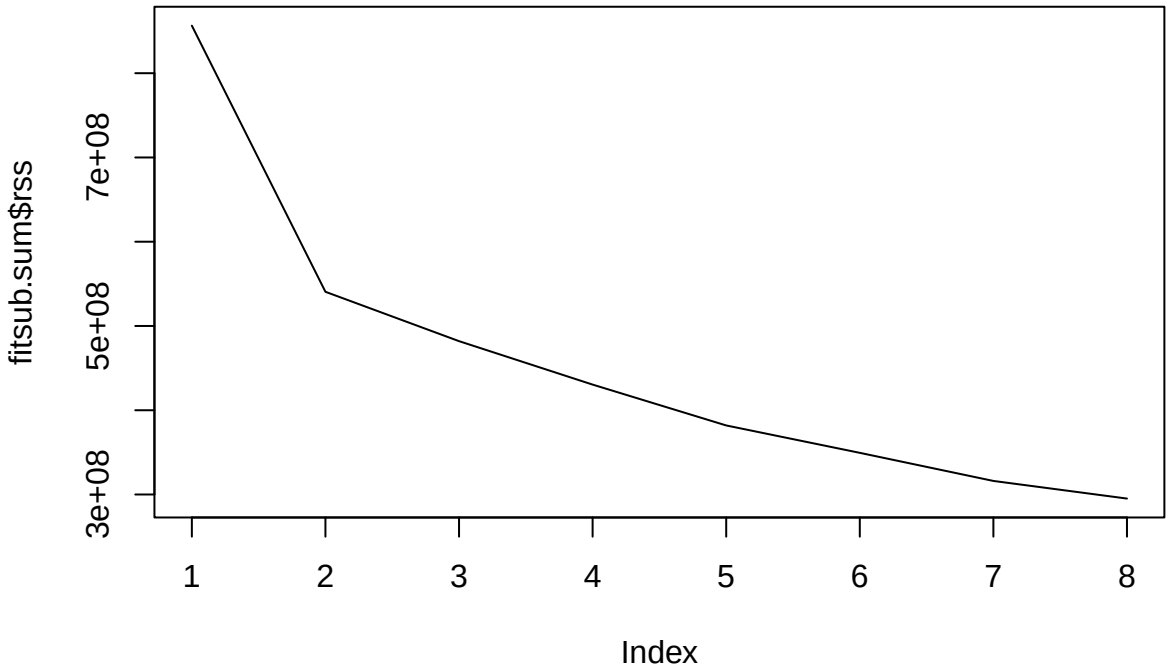
```
library(leaps)

fitsub <- regsubsets(price~., data = train)
fitsub.sum <- summary(fitsub)
fitsub.sum$outmat
```

```
##          speed33 speed50 speed66 speed75 speed100 hd  ram4 ram8 ram16 ram24
## 1  ( 1 ) " "     " "     " "     " "     " "      " " "*"  " "  " "   " "
## 2  ( 1 ) " "     " "     " "     " "     " "      "*" " "  " "  " "   " "
## 3  ( 1 ) " "     " "     " "     " "     " "      "*" " "  " "  " "   " "
## 4  ( 1 ) " "     " "     " "     " "     " "      "*" " "  " "  " "   " "
## 5  ( 1 ) " "     " "     " "     " "     " "      "*" " "  " "  "*"   " "
## 6  ( 1 ) " "     " "     "*"     " "     " "      "*" " "  " "  "*"   " "
## 7  ( 1 ) " "     " "     "*"     " "     "*"      "*" " "  " "  "*"   " "
## 8  ( 1 ) " "     " "     "*"     " "     "*"      "*" "*"  " "  "*"   " "
##          ram32 screen15 screen17 cdyes multiyes premiumyes ads trend
## 1  ( 1 ) " "   " "      " "      " "   " "      " "        " " " "
## 2  ( 1 ) " "   " "      " "      " "   " "      " "        " " "*"
## 3  ( 1 ) " "   " "      " "      " "   " "      "*"        " " "*"
## 4  ( 1 ) " "   " "      "*"      " "   " "      "*"        " " "*"
## 5  ( 1 ) " "   " "      "*"      " "   " "      "*"        " " "*"
## 6  ( 1 ) " "   " "      "*"      " "   " "      "*"        " " "*"
## 7  ( 1 ) " "   " "      "*"      " "   " "      "*"        " " "*"
```

```
## 8  ( 1 ) " "   " "      "*"      " "   " "      "*"        " " "*"
```

These stars show which variable is included in each model, where the algorythm chooses the variables which produces the lowest RSS first. RAM, trend and speed looks like the three variables which the model has chosen first. We can also create a graph to look at the change in RSS for each iteration of the subsetting.

```
plot(fitsub.sum$rss, type = "l")
```



By looking at these results, one can see that including just the first four variables which has been determined by the the subsetting method.

The testMSE of this model:

```
fit2 <- lm(price~ram+trend+speed+premium, data = train)
pred2 <- predict(fit2, newdata = test, type = "response")

testMSE2 <- mean((test$price - pred2)^2)

# Comparing the testMSE
testMSE
```

```
## [1] 68552.09
```

```
testMSE2
```

```
## [1] 91303.99
```

d)

In this task, I will do a log-transformation to the price variable in order to see if this would produce better results. I will make use of the testMSE measurements in order to interpret wheter or not this augmentation creates a better fit.

```
fit3 <- lm(log(price)~., data = train)
summary(fit3)
```

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

```
## lm(formula = log(price) ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.52687 -0.06917 -0.00127  0.06542  0.40280
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.661e+00  1.533e-02 499.654  < 2e-16 ***
## speed33      1.405e-01  7.445e-03  18.877  < 2e-16 ***
## speed50      2.200e-01  8.365e-03  26.298  < 2e-16 ***
## speed66      2.663e-01  7.862e-03  33.872  < 2e-16 ***
## speed75      2.835e-01  1.624e-02  17.454  < 2e-16 ***
## speed100     3.636e-01  1.053e-02  34.535  < 2e-16 ***
## hd           3.343e-04  1.640e-05  20.383  < 2e-16 ***
## ram4         4.000e-02  8.960e-03   4.464 8.33e-06 ***
## ram8         1.794e-01  1.000e-02  17.935  < 2e-16 ***
## ram16        3.494e-01  1.193e-02  29.283  < 2e-16 ***
## ram24        4.296e-01  1.776e-02  24.195  < 2e-16 ***
## ram32        5.124e-01  4.216e-02  12.154  < 2e-16 ***
## screen15     1.989e-02  4.633e-03   4.293 1.81e-05 ***
## screen17     1.691e-01  7.111e-03  23.774  < 2e-16 ***
## cdyes        4.113e-02  5.465e-03   7.527 6.78e-14 ***
## multiyes     4.555e-02  6.512e-03   6.995 3.24e-12 ***
## premiumyes  -2.496e-01  7.060e-03 -35.350  < 2e-16 ***
## ads          2.528e-04  2.951e-05   8.566  < 2e-16 ***
## trend       -2.303e-02  3.622e-04 -63.579  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1098 on 3110 degrees of freedom
## Multiple R-squared:  0.8238, Adjusted R-squared:  0.8228
## F-statistic:   808 on 18 and 3110 DF,  p-value: < 2.2e-16
```

From the summary, we can see that the R^2 is around 1 % higher than in the base regression.

In the case of log(price), the interpretation of the coefficients is slightly different. For example, if you have a coefficient of 0.5 for a predictor variable, then you can say that a one-unit increase in that predictor variable is associated with a **50% increase** in the expected value of price.

We can predict the price from here, and compare the predictions using testMSE.

```
logpred3 <- predict(fit3, newdata = test, type = "response")
pred3 <- exp(logpred3)

testMSE3 <- mean((test$price - pred3)^2)

paste0("The testMSE of the base regression:            ", round(testMSE, 2))
```

```
## [1] "The testMSE of the base regression:            68552.09"
```

```
paste0("The testMSE of the log transformed regression: ", round(testMSE3, 3))
```

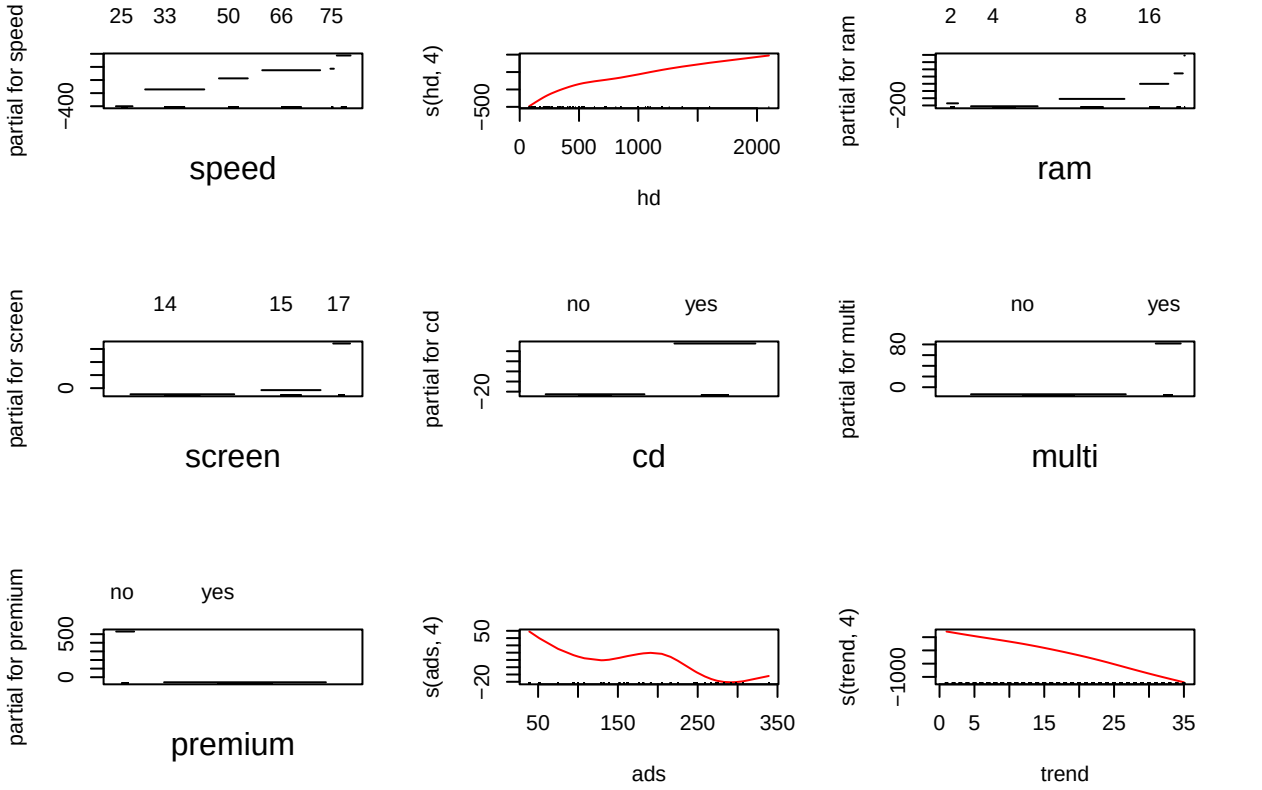
```
## [1] "The testMSE of the log transformed regression: 63695.091"
```

There is no significant change in the prediction power of log-transforming the price variable.

e)

This task will be broken up into two parts. First of all, a GAM model will be fitted to the data, and the results will be plotted and evaluated. The second part of the task will use this information, combined with general inference to lay out reasoning behind now allowing all variables to take on a non-linear relationship with price.

```
library(gam)

gamfit <- gam(price~speed+s(hd, 4)+ram+screen+cd+multi+premium+s(ads, 4)+s(trend, 4),
              data = train)
{
par(mfrow = c(3,3))
plot.Gam(gamfit, col="red")
}
```



One reason to not allow for some variables to have a non linear relationship with price is if the variable is a factor variable, as this represents a state variable, in which the observation can be in a state or not, which is not a suited for non-linear relationships.

The testMSE of this model can be calculated like this

```
predgam <- predict.Gam(gamfit, newdata = test)

testMSEgam <- mean((test$price - predgam)^2)
testMSEgam
```

```
## [1] 63255.03
```

Next, I will analyze whether there is evidence of a non-linear relationship with the response. In the figures above, we find some evidence of ads having a non-linear relationship with price. We test this by creating three GAM models: one without trend, one with trend as linear, and one with trend as non-linear. Then we

compare them using an ANOVA test.

```
gamfit1 <- gam(price~speed+s(hd, 4)+ram+screen+cd+multi+premium+s(ads, 4),
               data = train)
gamfit2 <- gam(price~speed+s(hd, 4)+ram+screen+cd+multi+premium+s(ads, 4)+trend,
               data = train)
gamfit3 <- gam(price~speed+s(hd, 4)+ram+screen+cd+multi+premium+s(ads, 4)+s(trend, 4),
               data = train)


anova(gamfit1, gamfit2, gamfit3, test = "F")
```

```
## Analysis of Deviance Table
##
## Model 1: price ~ speed + s(hd, 4) + ram + screen + cd + multi + premium +
##     s(ads, 4)
## Model 2: price ~ speed + s(hd, 4) + ram + screen + cd + multi + premium +
##     s(ads, 4) + trend
## Model 3: price ~ speed + s(hd, 4) + ram + screen + cd + multi + premium +
##     s(ads, 4) + s(trend, 4)
##   Resid. Df Resid. Dev     Df  Deviance        F    Pr(>F)
## 1      3105  399810416
## 2      3104  187079668 1.0000 212730747 3565.955 < 2.2e-16 ***
## 3      3101  184993358 3.0002   2086311   11.657 1.355e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From this output, all models are significant. Although model 2 has the lowest p-value and the highest F-statistic. From this we can assume that the gamfit2 model is the best, with trend as linear.

f)

Backfitting is a simple iterative procedure used to fit a generalized additive model (GAM), which is a type of non-parametric regression model that can capture nonlinear relationships between predictors and outcome variables1. A GAM has the form:

$$y_i = f_1(x_{i1}) + f_2(x_{i2}) + ... + f_p(x_{ip}) + \epsilon_i$$

where y_i is the outcome variable, x_{ij} are the predictor variables, f_j are smooth functions of a single predictor, and ϵ_i is the error term.

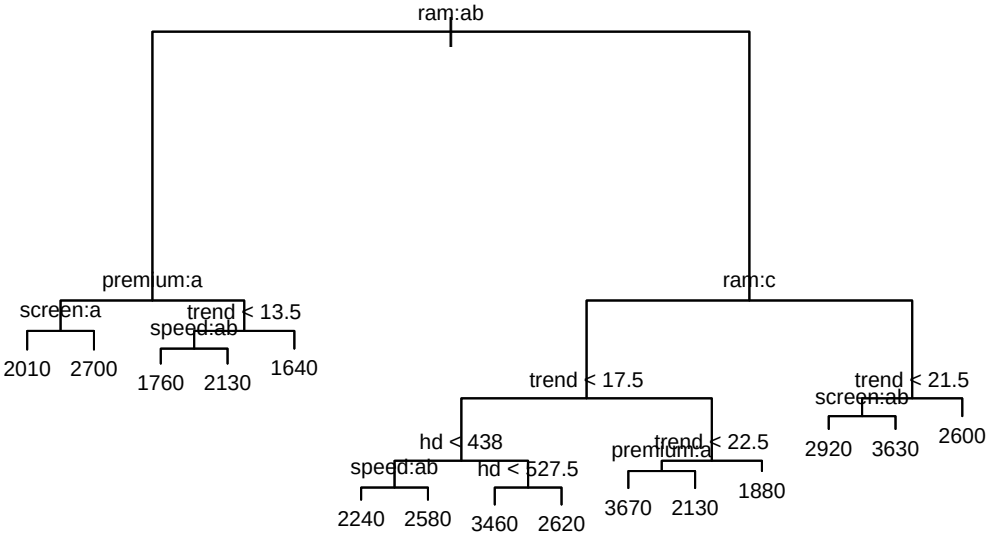
The backfitting algorithm works by initializing the function estimates and then updating each one in turn by fitting the residuals of all the others using a smoothing operator, such as a cubic spline or a local polynomial regression1. The algorithm stops when the function estimates converge to a solution that minimizes the expected squared error.

Backfitting can be used to fit a GAM with ordinary least square regression when the outcome variable is continuous and normally distributed. In this case, the smoothing operator can be chosen to be a linear smoother that minimizes the sum of squared errors1. However, backfitting can also be extended to fit a GAM with other types of regression models, such as logistic or Poisson regression, by using a local quasilikelihood approach.

g)

Bagging is short for bootstrap aggregation. In this task we will make use of this method to predict price in the Computers data set. We start by making a simple decision tree.

```
library(tree)

simple.tree <- tree(price~., data=train)

{
plot(simple.tree)
text(simple.tree, digits=3, cex = 0.7)
}
```



The testMSE for this simple tree is:

```
simple.pred <- predict(simple.tree, newdata = test)
msesimp <- mean((test$price - simple.pred)^2)
msesimp
```

```
## [1] 102882.8
```

Bagged tree with paralellization

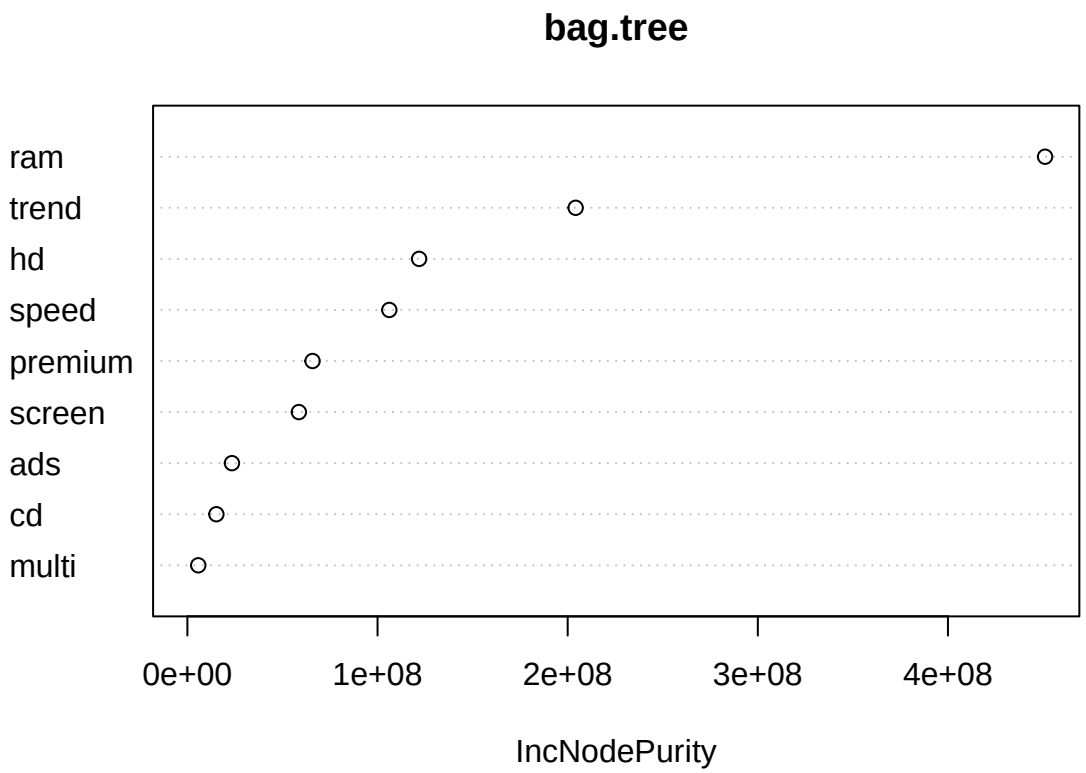
```
library(randomForest)
library(doParallel)

# Establishing parallel computation
cores <- parallel::detectCores()
cluster <- makeCluster(min(cores, 8))
registerDoParallel(cluster)

# The bagged decision tree model
bag.tree <- randomForest(price~., data=train, mtry=10)
bag.pred <- predict(bag.tree, newdata=test)
mse.bag=mean((test$price-bag.pred)^2)

# Stopping the paralellization
stopCluster(cluster)

# Displaying the testMSE
mse.bag
```

```
## [1] 29290.47
```

Using the bagged tress, the testMSE went down to 1/3 of what it was with a simple decision tree.

Variable importance

```
varImpPlot(bag.tree)
```



The varImpPlot function in R shows the variable importance of each predictor in a random forest model. The importance can be measured by two criteria: Mean Decrease Accuracy and Mean Decrease Gini. The former is the average decrease in prediction accuracy when a variable is permuted in the out-of-bag samples, while the latter is the average decrease in node impurity when a variable is used for splitting. A higher value of either criterion means that the variable is more important for the model performance. In this case, "ram", "hd", "trend" and "screen" have the highest values of both criteria, which means that they are the most influential predictors in the model

h)

Bagging is an ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. The idea behind bagging is that combining the predictions of many models trained on different subsets of the data will lead to better overall performance than using any single model alone. Bagging can be used with any type of model, but it is particularly effective with decision trees, which tend to have high variance. By training many decision trees on different subsets of the data and averaging their predictions, bagging can help to reduce overfitting and improve the accuracy of predictions