Lab 8 - Decission Trees

An Introduction to Statistical Learning

```
pacman::p_load(tree, ISLR)
```

1. Fitting Classification Trees

The process has the following steps:

1. Fit the tree to the training data.
2. Find the optimal tree complexity using cross-validation.
3. Make predictions on the testing subset.

The `Carseats` dataset will be used.

Classification and Regression Trees are fitted using the `tree` library.

```
summary(Carseats)
```

```
##      Sales          CompPrice       Income        Advertising
##  Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000
##  1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000
##  Median : 7.490   Median :125   Median : 69.00   Median : 5.000
##  Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635
##  3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000
##  Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000
##    Population        Price        ShelveLoc       Age          Education
##  Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00   Min.   :10.0
##  1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75   1st Qu.:12.0
##  Median :272.0   Median :117.0   Medium:219   Median :54.50   Median :14.0
##  Mean   :264.8   Mean   :115.8                Mean   :53.32   Mean   :13.9
##  3rd Qu.:398.5   3rd Qu.:131.0                3rd Qu.:66.00   3rd Qu.:16.0
##  Max.   :509.0   Max.   :191.0                Max.   :80.00   Max.   :18.0
##  Urban        US
##  No :118   No :142
##  Yes:282   Yes:258
##
##
##
##
```

```
attach(Carseats)
```

First we split the data in train and testing subsets:

```
set.seed(1)
train = sample(1:nrow(Carseats), dim(Carseats)[1]/2)
```

We will classify the data according to `Sales`; as it's a continuous variable, we'll encode it as a binary variable, with `ifelse()`, using the mean as an approximate threshold. The resulting array must be converted to a categorical variable using `factor()`:

```r
High = ifelse(Sales > 8, 'Yes', 'No')
High = factor(High)
```

We merge the new variable `High` in the dataset:

```r
Carseats$High = High
```

Fitting the model

We fit the decission tree using all the variables but `Sales`:

```r
tree.carseats = tree(High~.-Sales, data = Carseats, subset = train)
summary(tree.carseats)
```

```
##
## Classification tree:
## tree(formula = High ~ . - Sales, data = Carseats, subset = train)
## Variables actually used in tree construction:
## [1] "Price"      "Population"  "US"           "CompPrice"    "Advertising"
## [6] "Income"     "ShelveLoc"   "Age"
## Number of terminal nodes:  20
## Residual mean deviance:  0.4549 = 81.89 / 180
## Misclassification error rate: 0.105 = 21 / 200
```

The *residual mean deviance* is the deviance divided by $n - |T_0|$, being n the number of observations and T_0 the number of terminal nodes (reported by `summary()`). A small deviance indicates that the tree provides a god fit to the training data.

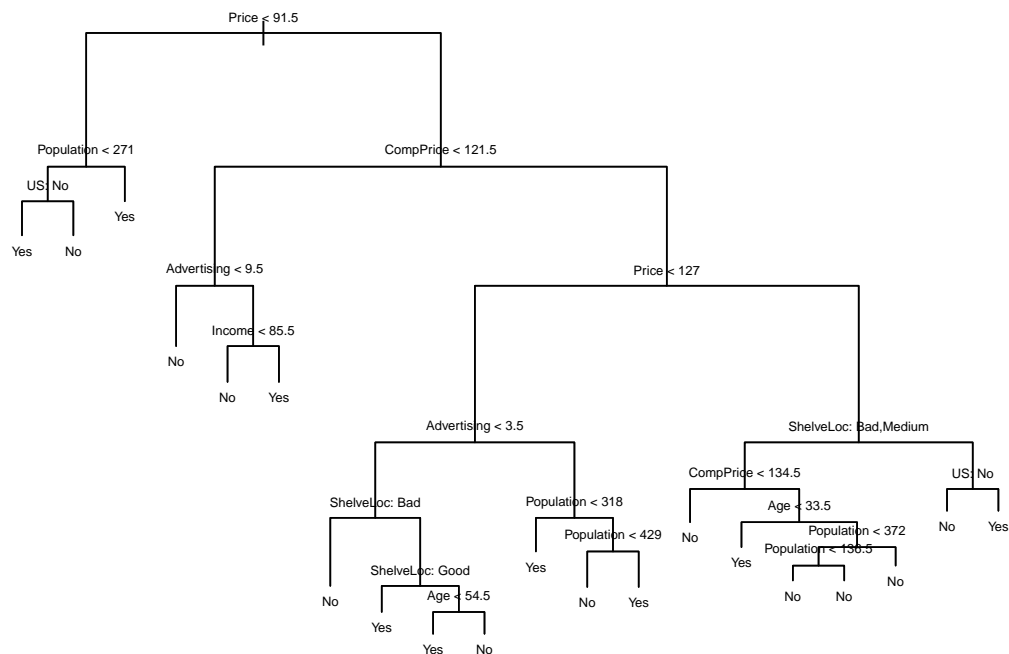`summary()` also includes the *misclassification error rate*, which for *classification trees* is given by:

$$-2 \sum_m \sum_k n_{mk} \log \hat{p}_{mk}$$

where n_{mk} is the number of observations in the mth terminal node that belong to the kth class.

`tree()` selects the relevant variables and exclude the rest from the model (in this case, `Population`, `Education` and `Urban` have been excluded)

Plotting the tree

```r
plot(tree.carseats)
text(tree.carseats, pretty = 0, cex = .5)
```

Price < 91.5

Population < 271   CompPrice < 121.5

US: No   Yes

Yes   No

Advertising < 9.5   Price < 127

Income < 85.5

No

No   Yes

Advertising < 3.5   ShelveLoc: Bad,Medium

ShelveLoc: Bad   Population < 318   CompPrice < 134.5   US: No

Population < 429   No   Age < 33.5   No   Yes

ShelveLoc: Good   Age < 54.5   Yes   Population < 372

No   Yes   No   Yes   Yes   Population < 136.5   No

Yes   Yes   No   No   No

ShelveLoc appears to be an important indicator for Sales, because the first branch differenciates Good locations from Medium and Bad locations.

Making predictions

For classification trees, type="class" tells predict() to return the *actual class prediction*:

```
tree.pred = predict(tree.carseats, newdata = Carseats[-train,], type = 'class')
table(tree.pred, High[-train])
```

```
##
## tree.pred No Yes
##       No  84  37
##       Yes 35  44
```

The tree makes correct classifications for the 64% of the testing data:

```
mean(tree.pred == High[-train])
```

```
## [1] 0.64
```

Pruning the tree with Cross-Validation

Finding the optimal level of complexity

The optimal level of complexity for a fitted tree can be determined using cross-validation, with the cv.tree() function. The default metric used to guide cross-validation is the deviance, but we can specify that we want

the missclassification error rate with `FUN=prune.misclass`:

```
set.seed(42)
cv.carseats = cv.tree(tree.carseats, FUN = prune.misclass)
```

The result includes the following fields:

```
cv.carseats
```

```
## $size
## [1] 20 18 10  8  6  4  2  1
##
## $dev
## [1] 52 52 51 50 52 53 73 84
##
## $k
## [1] -Inf  0.0  0.5  1.5  2.0  4.0 12.0 19.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

`size` is the number of terminal nodes of each tree considered. `dev` is actually the cross-validation error rate, as indicated with `FUN`. `k` is the cost-complexity parameter used, which corresponds to α in equation 8.4.

The 8-node tree has the lowest error rate, so we will use it as the final tree.

We can plot the error rate as a function of both `k` and `size`:

```
par(mfrow=c(1, 2))
plot(cv.carseats$size, cv.carseats$dev, type='b'); grid()
plot(cv.carseats$k, cv.carseats$dev, type='b'); grid()
```
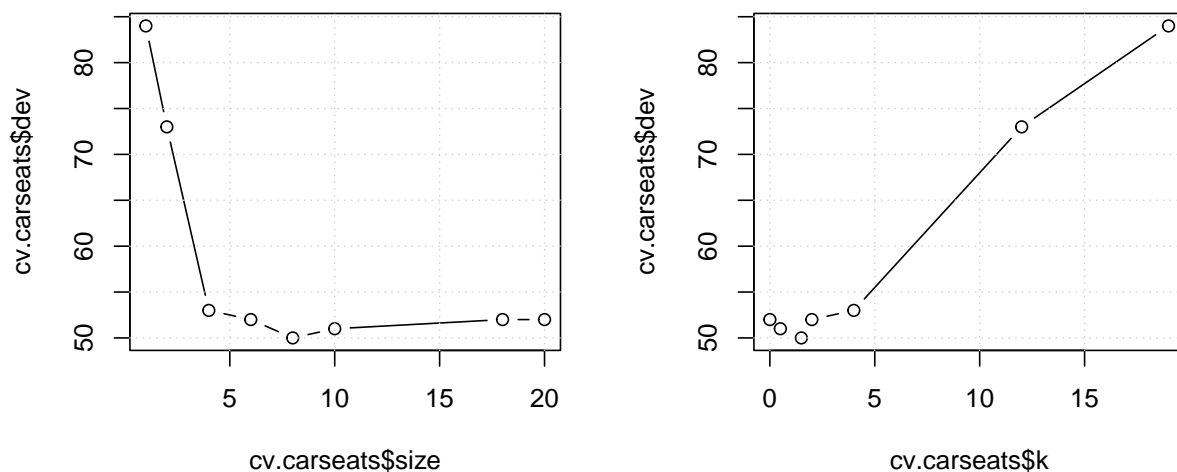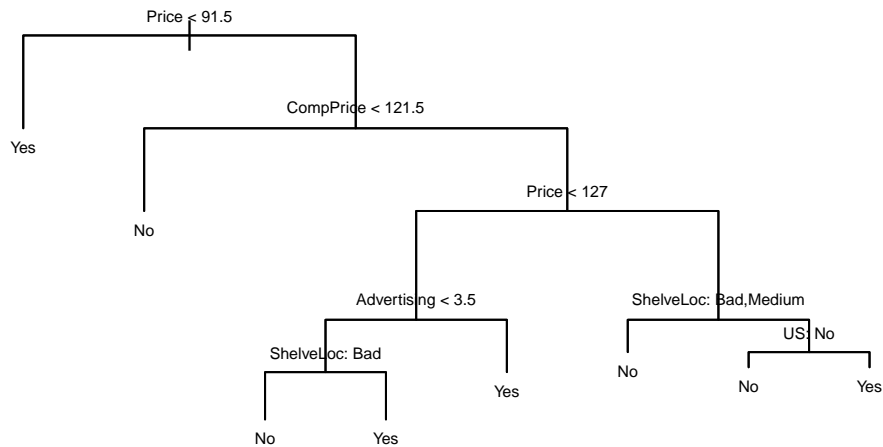

Pruning the tree

We apply the `prune.misclass()` function to prune the tree to obtain the 8-node tree:

```
prune.carseats = prune.misclass(tree.carseats, best = 8)
```

```
plot(prune.carseats)
text(prune.carseats, pretty = 0, cex = .5)
```

```
                              Price < 91.5

                                    CompPrice < 121.5
        Yes

                        No                              Price < 127

                               Advertising < 3.5              ShelveLoc: Bad,Medium
                                                                               US: No
                        ShelveLoc: Bad                     No
                                          Yes                       No         Yes
                   No         Yes
```

Making predictions using the pruned tree

```
tree.pred = predict(prune.carseats, newdata = Carseats[-train,], type='class')
table(tree.pred, High[-train])
```

```
##
## tree.pred No Yes
##       No  83  24
##       Yes 36  57
```

```
mean(tree.pred == High[-train])
```

```
## [1] 0.7
```

Now the tree correctly labels the 70% of the testing data.

2. Fitting Regression Trees