Chapter 2: Classification

- Decision trees
- Logistic regression

Classification - Tree-Based Method

A classification tree is used to predict a qualitative response. For a classification tree, we predict that each observation belongs to the *most commonly occurring class* of training observations in the region to which it belongs.

Roughly speaking, there are two steps:

1. We divide the predictor space - that is, the set of possible values for X_1, $X_2,\ldots,X_p$ - into J distinct and non-overlapping regions, R_1, $R_2,\ldots,R_J$.
2. For every observation that falls into the region R_j, we make the same prediction, which is simply the class of the majority of the training observations in R_j.

* For regression tree, the prediction is the mean of the response values for the training observations in R_j.

Classification - Fitting Classification Trees

Building a Classification Tree:

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.
2. Apply cost complexity pruning (error + penalty) to the large tree in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold cross-validation, which will be taught by Prof. Yao later, to choose α and determine the size of the tree.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Question: Why K-fold cross-validation?

Classification - Fitting Classification Trees

The tree library is used to construct classification and regression trees. We first use classification trees to analyse the Carseats data set. In these data, Sales is a continuous variable, and so we begin by recoding it as a binary variable. We use the ifelse() function to create a variable, called High, which takes on a value of Yes if the Sales variable exceeds 8, and takes on a value of No otherwise. We now use the tree() function to fit a classification tree in order to predict High using all variables but Sales.

```
> library(tree)
> High = ifelse(Carseats$Sales > 8, "Yes", "No")
> Carseats = data.frame(Carseats, High)
> treeCarseats = tree(formula = High~.-Sales, data = Carseats)
> summary(treeCarseats)

 Classification tree:
 tree(formula = High ~ . - Sales, data = Carseats)
 Variables actually used in tree construction:
 [1] "ShelveLoc"  "Price"  "Income"  "CompPrice"  "Population"  "
     Advertising"  "Age"  "US"
 Number of terminal nodes:  27
 Residual mean deviance:  0.4575 = 170.7 / 373
 Misclassification error rate: 0.09 = 36 / 400
```

Classification - Fitting Classification Trees

We see that the training error rate is 9%. For classification trees, the deviance reported in the output of summary() 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. A small deviance indicates a tree that provides a good fit to the (training) data. The residual mean deviance reported is simply the deviance divided by $n - |T_0|$, which in this case is $400 - 27 = 373$.

One of the most attractive properties of trees is that they can be graphically displayed. We use the plot() function to display the tree structure, and the text() function to display the node labels.

```
> plot(treeCarseats)
> text(treeCarseats, cex = 0.9) # check it by yourself.
```

The most important indicator of Sales appears to be shelving location, since the first branch differentiates Good locations from Bad and Medium locations.

Classification - Fitting Classification Trees

In order to properly evaluate the performance of a classification tree on these data, we must estimate the test error rather than simply computing the training error. We split the observations into a training set and a test set, build the tree using the training set, and evaluate its performance on the test data. The predict() function can be used for this purpose. In the case of a classification tree, the argument type="class" instructs R to return the actual class prediction. This approach leads to correct predictions for around 71.5% of the locations in the test data set.

```
> set.seed (2)
> train = sample (1:nrow(Carseats), 200) # half of data
> CarseatsTest = Carseats[-train, ]
> HighTest = High[-train]
> treeCarseats = tree(High~.-Sales, Carseats, subset = train )
> treePred = predict(treeCarseats, CarseatsTest, type ="class")
> table(treePred, HighTest)
        HighTest
treePred No Yes
     No  86  27
     Yes 30  57
> (86 + 57)/200
[1] 0.715
```

Classification - Tree Pruning

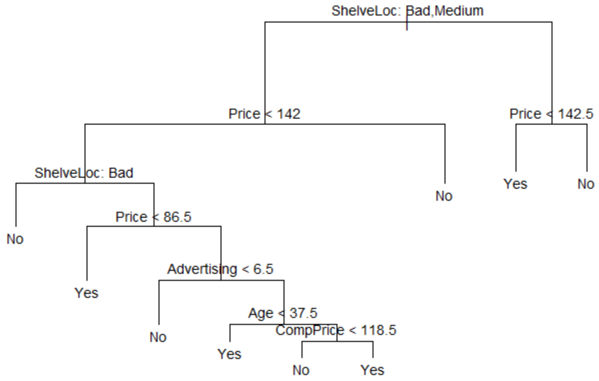
Next, we consider whether pruning the tree might lead to improved results. The function cv.tree() performs cross-validation in order to determine the optimal level of tree complexity; cost complexity pruning is used in order to select a sequence of trees for consideration. We use the argument FUN=prune.misclass in order to indicate that we want the classification error rate to guide the cross-validation and pruning process, rather than the default for the cv.tree() function, which is deviance.

```
> set.seed (3)
> cvCarseats = cv.tree(treeCarseats, FUN = prune.misclass )
> cvCarseats
$size
[1] 19 17 14 13  9  7  3  2  1
$dev
[1] 55 55 53 52 50 56 69 65 80
$k
[1]        -Inf  0.0000000  0.6666667  1.0000000  1.7500000
      2.0000000  4.2500000  5.0000000 23.0000000
$method
[1] "misclass"
```

Classification - Tree Pruning

We now apply the prune.misclass() function in order to prune the tree to obtain the nine-node tree.

```
> pruneCarseats = prune.misclass(treeCarseats, best = 9)
> plot(pruneCarseats)
> text(pruneCarseats, cex = 0.9, pretty = 0)
```



Classification - Tree Pruning

How well does this pruned tree perform on the test data set? Once again, we apply the predict() function.

```
> treePred = predict(pruneCarseats, CarseatsTest, type="class")
> table(treePred, HighTest)
        HighTest
treePred No Yes
     No  94  24
     Yes 22  60
> (94+60)/200
[1] 0.77
```

Now 77% of the test observations are correctly classified, so not only has the pruning process produced a more interpretable tree, but it has also improved the classification accuracy.

Classification - Logistic Regression

How should we model the relationship between $p(X) = \mathbb{P}(Y = 1|X)$ and X? The linear regression model will lead some negative predicted probabilities. To avoid this problem we must model $p(X)$ using a function that gives outputs between 0 and 1 for all values of X.

We now consider the problem of predicting a binary response using logistic regression in multiple predictors. The logit is

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p,$$

where $X = (X_1, \ldots, X_p)$ are p predictors. The above equation can be rewritten as

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}{1 + \exp(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}.$$

Normally we use the maximum likelihood method to estimate $\beta_0, \ldots, \beta_p$ although we can use least squares to estimate.

Classification - Logistic Regression

Consider the Default data set, where the response default falls into one of two categories, Yes or No. Rather than modeling this response Y directly, logistic regression models the probability that Y belongs to a particular category. In the result, a one-unit increase in balance is associated with an increase in the log odds of default by 0.0055 units.

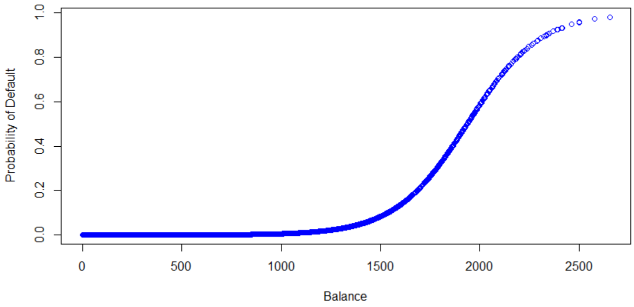
```
> attach(Default)
> lgDefault = glm(formula = default~balance, data = Default, family
    = binomial)
> summary (lgDefault)

Coefficients:
                  Estimate  Std. Error  z value   Pr(>|z|)
(Intercept)     -1.065e+01  3.612e-01  -29.49    <2e-16 ***
Default$balance  5.499e-03  2.204e-04   24.95    <2e-16 ***
```

Classification - Logistic Regression

We can plot the predicted probabilities of default against balance. All of predicated values are located between 0 and 1.

```
> plot(balance, lgDefault$fitted.values, xlab = "Balance", ylab = "
    Probability of Default", col = 4)
```



Classification - Logistic Regression

The predict() function can be used to predict the probability of default, given values of the predictors. The type="response" option tells R to output probabilities of the form $\mathbb{P}(Y = 1|X)$, as opposed to other information such as the logit.

We predict that the default probability for an individual with a balance of 1,000 which is below 1%. In contrast, the predicted probability of default for an individual with a balance of $2,000$ is much higher, and equals 0.586 or 58.6%.

```
> predict(lgDefault, data.frame(balance = c(1000, 2000)), type = "
    response")
          1           2
0.005752145 0.585769370
```

One can use qualitative predictors with the logistic regression model using the dummy variable approach. As an example, the Default data set contains the qualitative variable student. To fit the model we simply create a dummy variable that takes on a value of 1 for students and 0 for non-students.

```
lgDefault = glm(formula = default~student, data = Default, family =
    binomial)
summary (lgDefault)
```

Classification - Multiple Logistic Regression

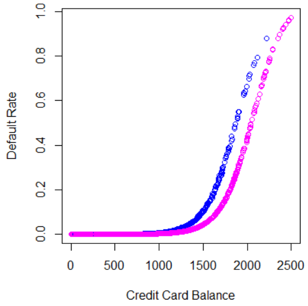
Fitting a logistic model for predicting default using balance, income and student status. The p-value of income is large, indicating that it can be removed from the model. Coefficient for student is negative, indicating that students are less likely to default than non-students - Contradictory?

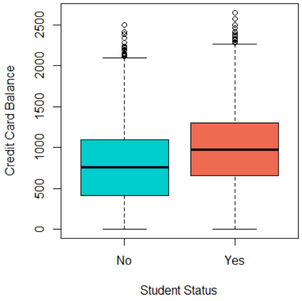
```
> lgDefault = glm(formula = default~., data = Default, family =
    binomial)
> summary (lgDefault)
Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
```

The variables student and balance are correlated. Students tend to hold higher levels of debt, which is in turn associated with higher probability of default. In other words, students are more likely to have large credit card balances tend to be associated with high default rates.

Classification - Multiple Logistic Regression

We can apply R to plot confounding in the Default data. The left plot illustrates that Default rates are shown for students (purple) and non-students (blue). The right one is a Boxplots of balance for students (orange) and non-students (cyan) are shown. The code for plotting is left to you.





Classification - Multiple Logistic Regression

For example, a student with a credit card balance of $1,500$ and an income of $40,000$ has an estimated probability of default is 0.058. A non-student with the same balance and income has an estimated probability of default 0.105.

```
> predict(lgDefault, data.frame(balance = c(1500), student = c("Yes"
    , "No"), income = c(40000)), type = "response")
         1          2
0.05788194 0.10499192
```

We sometimes wish to classify a response variable that has more than two classes. The two-class logistic regression models discussed in the previous sections have multiple-class extensions, but in practice they tend not to be used all that often. One of the reasons is that the method, discriminant analysis, is popular for multiple-class classification. So we do not go into the details of multiple-class logistic regression here, but simply note that such an approach is possible, and that software for it is available in R.