Chapter 3. Classification

- Measuring uncertainties and information gains
- Decision trees
- Logistic regression
- Classification in R: `tree`, `glm`

Further readings:

James et al. (2013) Sections 4.1-4.3 & 8.1,

Provost and Fawcett (2013) Chapter 3.

Classification problems occur often, perhaps even more so than regression problems. Examples include:

- An online banking service must be able to determine whether or not a transaction being performed is fraudulent, on the basis of the user's transaction history, IP address, $\cdots$
- Customer Churn: decide if a customer should be offered a special retention deal prior to the expiration of his/her contract
- A patient arrives at a hospital with a set of symptoms which could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
- Spam detection: is a newly arrived email a spam or not?
- Advertising: who among the customers database would be interested a new model of BMW?

Classification is typically a supervised learning problem: use historical data (i.e. training data) to identify the characteristics of different classes, and predict the class of new data based on the identified characteristics.

Key: identify the relevant characteristic variables which carry the information on the classes.

Information is a quantity that reduces uncertainty

Variable selection: choose characteristics/variables which carry most information, or, equivalently, minimize uncertainty.

Tree methods

Particularly useful for analyzing large data sets or the data sets with the number of input variables, denoted by p, greater than the number of individuals (i.e. the number of observations), denoted by n.

Training data: data with both input variables, denoted by $\mathbf{X}$, and outcome (i.e. label of classes) known, denoted by Y.

Basic idea: partition the feature space (i.e. $\mathbf{X}$-space) into a set of rectangles, then assign each rectangle to a particular class, (or fit a simple model, such as a constant, in each rectangle for a regression tree)

Conceptually simple yet powerful, and computationally demanding when the feature variable $\mathbf{X}$ is high dimensional

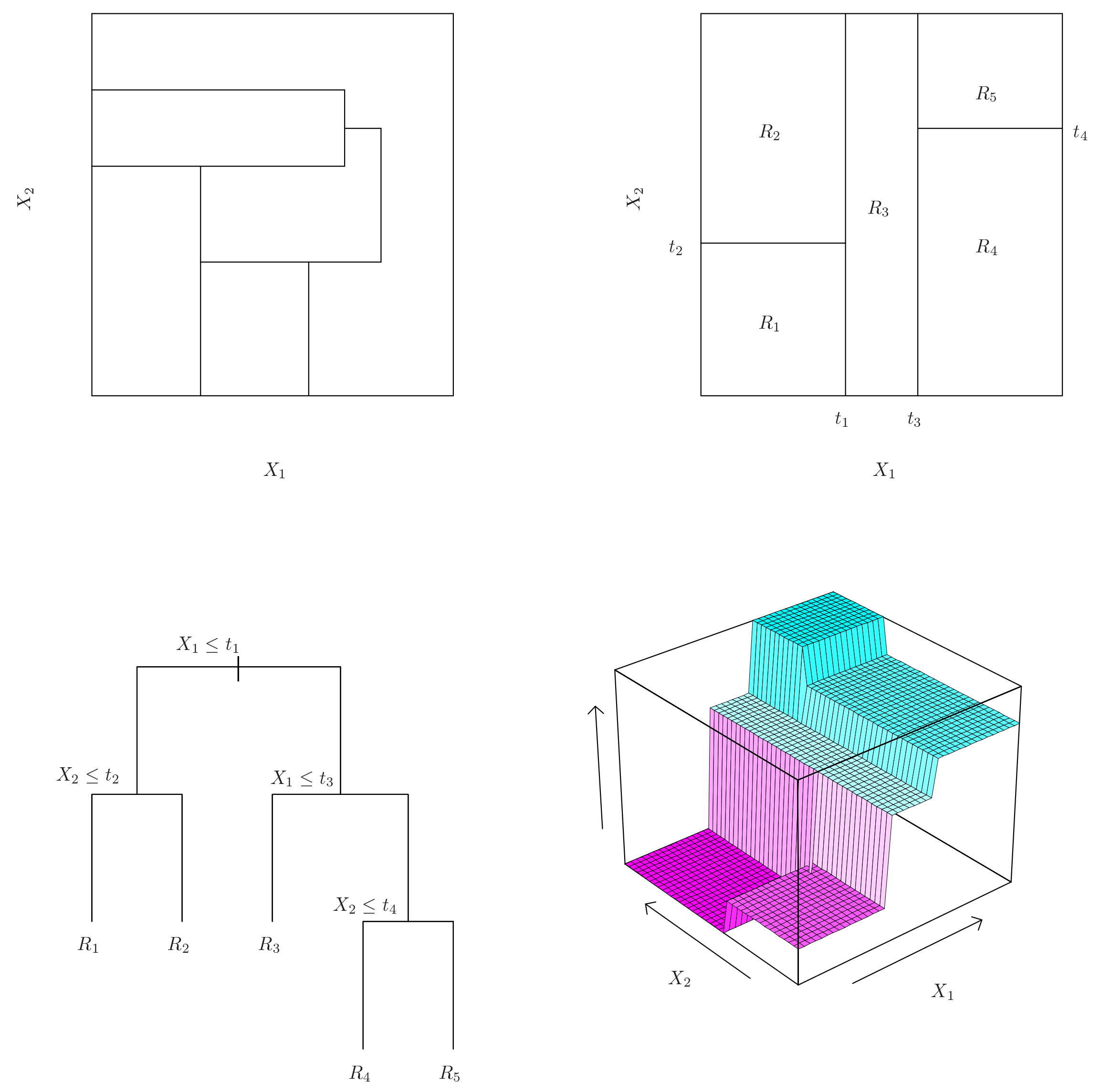
When p is large, the partition is done by a subset of components of $\mathbf{X}$.

An illustration by example: $Y = f(X_1, X_2) + \varepsilon$.

Fitting via *recursive binary splitting*:

$$\widehat{f}(X_1, X_2) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}$$

where the partition $\{R_1, \cdots, R_5\}$ is obtained by several simple splits along either X_1 or X_2 as illustrated in the figure below.



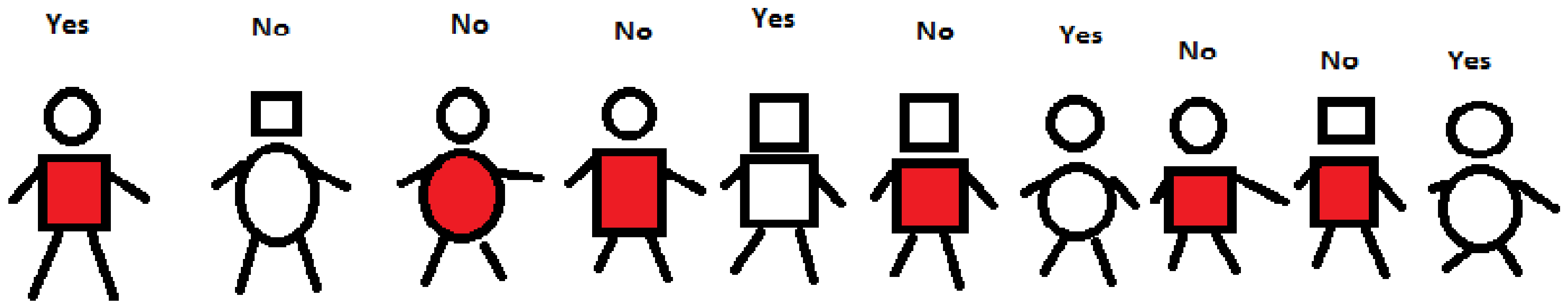
Top Right: Output of recursive binary splitting

Bottom Left: Tree corresponding to the partition

Bottom Right: Perspective plot of the predictive surface corresponding to the tree

Top Left: A partition cannot be resulted from recursive binary splitting.

Churn or not?



$$p_1 \equiv P(\text{Churn}) = 0.4, \qquad p_0 \equiv P(\text{non-Churn}) = 0.6.$$

Entropy — a measure for uncertainty: $-\sum_i p_i \log p_i$

For this example, Entropy$= -0.4\log(0.4) - 0.6\log(0.6) = 0.6730$

Note. Entropy is 0, when one of p_i is 1 — No uncertainty!

Let $Y = I(\text{churn})$. Then $Y = 1$ indicates a churn, and $Y = 0$ indicates non-churn, and the entropy of Y is

$$H(Y) = -\sum_i P(Y = i) \log P(Y = i) = 0.6730.$$

Now we try to reduce the uncertainly of Y by predicting Y based on the three characteristics:

- $X_1 = 1$ (round head), or 0 (square head)
- $X_2 = 1$ (solid body), or 0 (light body)
- $X_3 = 1$ (round body), or 0 (square body)

Rule to grow tree: choose X_i to maximize information gain (IG)

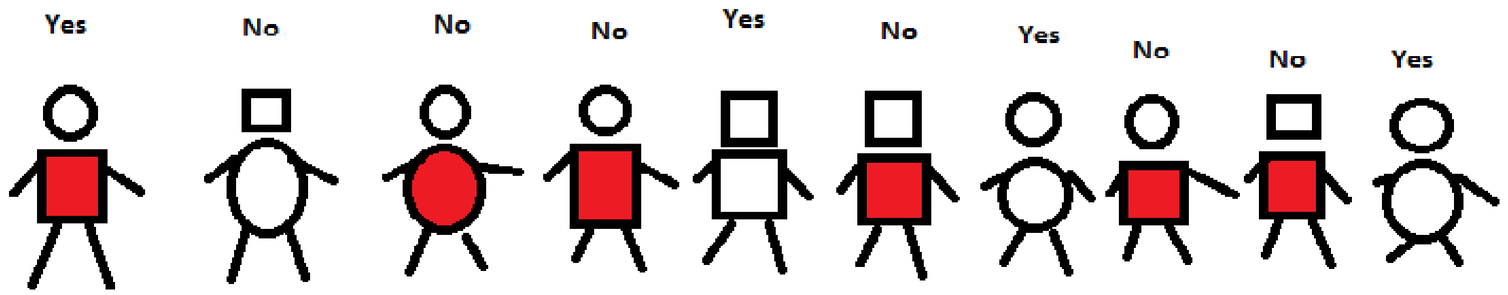
The conditional entropy of Y given X is $H(Y|X)$ defined as

$$H(Y|X) = \sum_k P(X = k) h(Y|X = k)$$

where $h(Y|X = k) = -\sum_i P(Y = i|X = k) \log\{P(Y = i|X = k)\}$.

$$IG(Y|X) = H(Y) - H(Y|X)$$

Churn or not?



Let $p(j|X_i = k) = P(Y = j|X_i = k)$

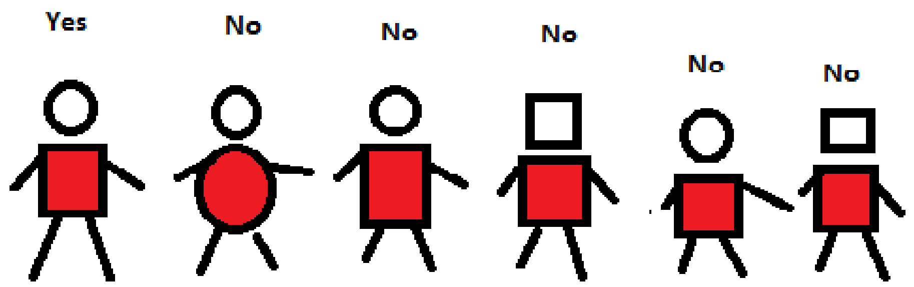
i	$P(X_i = 1)$	$P(1\|X_i = 1)$	$P(0\|X_i = 1)$	$P(X_i = 0)$	$P(1\|X_i = 0)$	$P(0\|X_i = 0)$
1	0.6	0.5	0.5	0.4	0.25	0.75
2	0.6	1/6	5/6	0.4	0.75	0.25
3	0.4	0.5	0.5	0.6	1/3	2/3

$$h(Y|X_1 = 1) = .6931, \quad h(Y|X_1 = 0) = .5623, \quad H(Y|X_1) = .6408$$
$$h(Y|X_2 = 1) = .4506, \quad h(Y|X_2 = 0) = .5623, \quad H(Y|X_2) = .4953$$
$$h(Y|X_3 = 1) = .6931, \quad h(Y|X_3 = 0) = .6365, \quad H(Y|X_3) = .6591$$

Thus, $IG(Y|X_2)$ is the biggest: 1st branch is generated by X_2.

Note. $H(Y) \geq H(Y|X)$ for any Y and X.

$X_2 = 1$ (solid body)



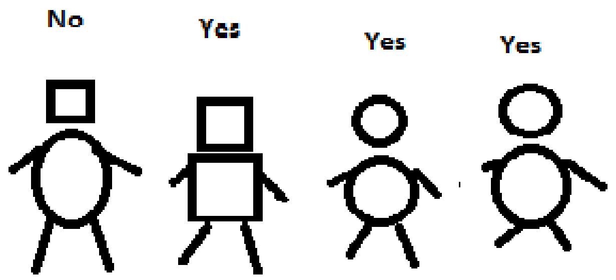
$H(Y) = -(\frac{1}{6}\log\frac{1}{6} + \frac{5}{6}\log\frac{5}{6}) = 0.4505$

$H(Y|X_1) = \frac{4}{6}h(Y|X_1 = 1) + \frac{2}{6}h(Y|X_1 = 0)$
$= \frac{4}{6} \times 0.5623 = 0.3748$

$H(Y|X_3) = \frac{1}{6}h(Y|X_3 = 1) + \frac{5}{6}h(Y|X_3 = 0)$
$= \frac{5}{6} \times 0.5004 = 0.4170$

A further branch grows out of X_1.

$X_2 = 0$ (light body)



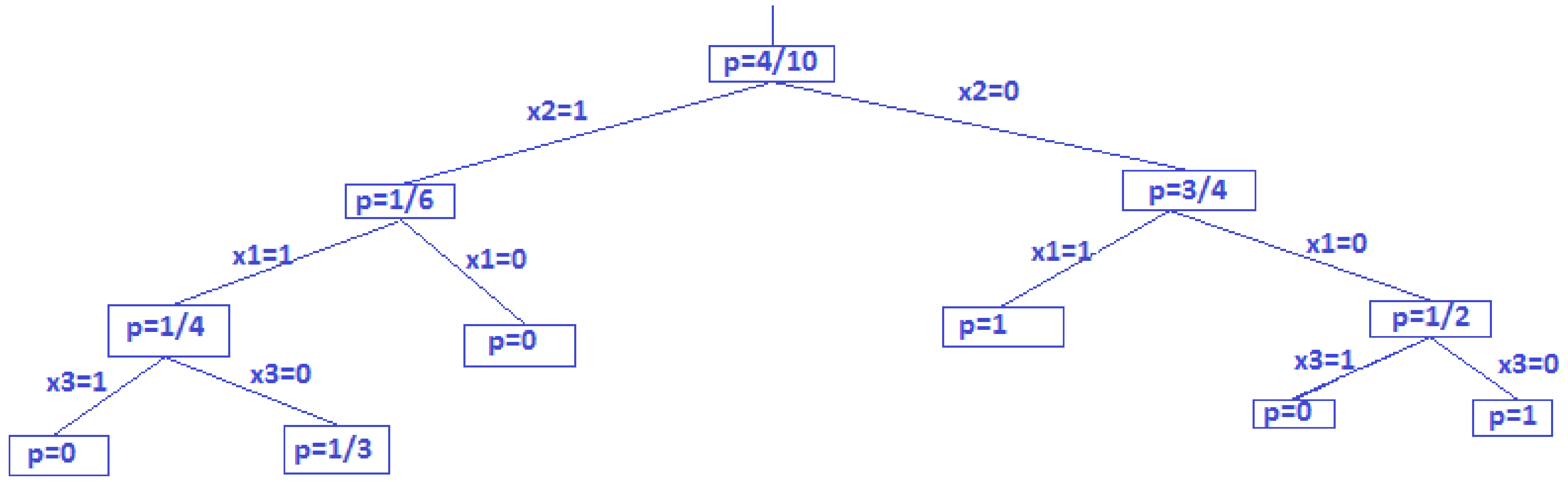
$H(Y) = -(\frac{1}{4}\log\frac{1}{4} + \frac{3}{4}\log\frac{3}{4}) = 0.5623$

$H(Y|X_1) = \frac{2}{4}h(Y|X_1 = 1) + \frac{2}{4}h(Y|X_1 = 0)$
$\frac{2}{4} \times 0.6931 = 0.3466$

$H(Y|X_3) = \frac{3}{4}h(Y|X_3 = 1) + \frac{1}{4}h(Y|X_3 = 0)$
$\frac{3}{4} \times 0.6365 = 0.4774$

A further branch grows out of X_1.

Notation: On each note $p = P(\text{Churn})$



- Computing intensity: more so with more X's or/and X's take multiple/continuous values.

- Stepwise searching: exhausting searching only at each step

- Uncertainty may still exist at some terminal nodes (e.g. along branches $X_2 = 1, X_1 = 1, X_3 = 0$). We then classify the node into the class according to the majority of the individuals in this node (i.e. non-Churn, as $p = 1/3$).

- Overfitting – should be avoided!

  Common wisdom: Data consist of signal (i.e. relevant information) plus noise. A good modelling should extract relevant information only.

  There are sound statistical procedures for preventing or detecting overfitting.

  A good data-mining/statistical modelling should always make judicious use of subject knowledge, creativity and common sense.

In reality, variables affecting mobile-customer churns include:

- *College*: is the customer college-educated?
- *Income*: annual income
- *Overage*: average overcharges per month
- *Leftover*: average number of leftover minutes per month
- *House*: estimated value of house
- *Handset price*: cost of phone
- *Average call duration*: average duration of calls
- *Long call per month*: average number of long calls ($\geq$ 15mins) per month
- *Reported satisfaction*: reported level of satisfaction
- *Reported usage level*: self-reported usage level

For a historical data set of 20,000 customers who either has stayed the company or has churned, we fit a tree model for predicting future churns.

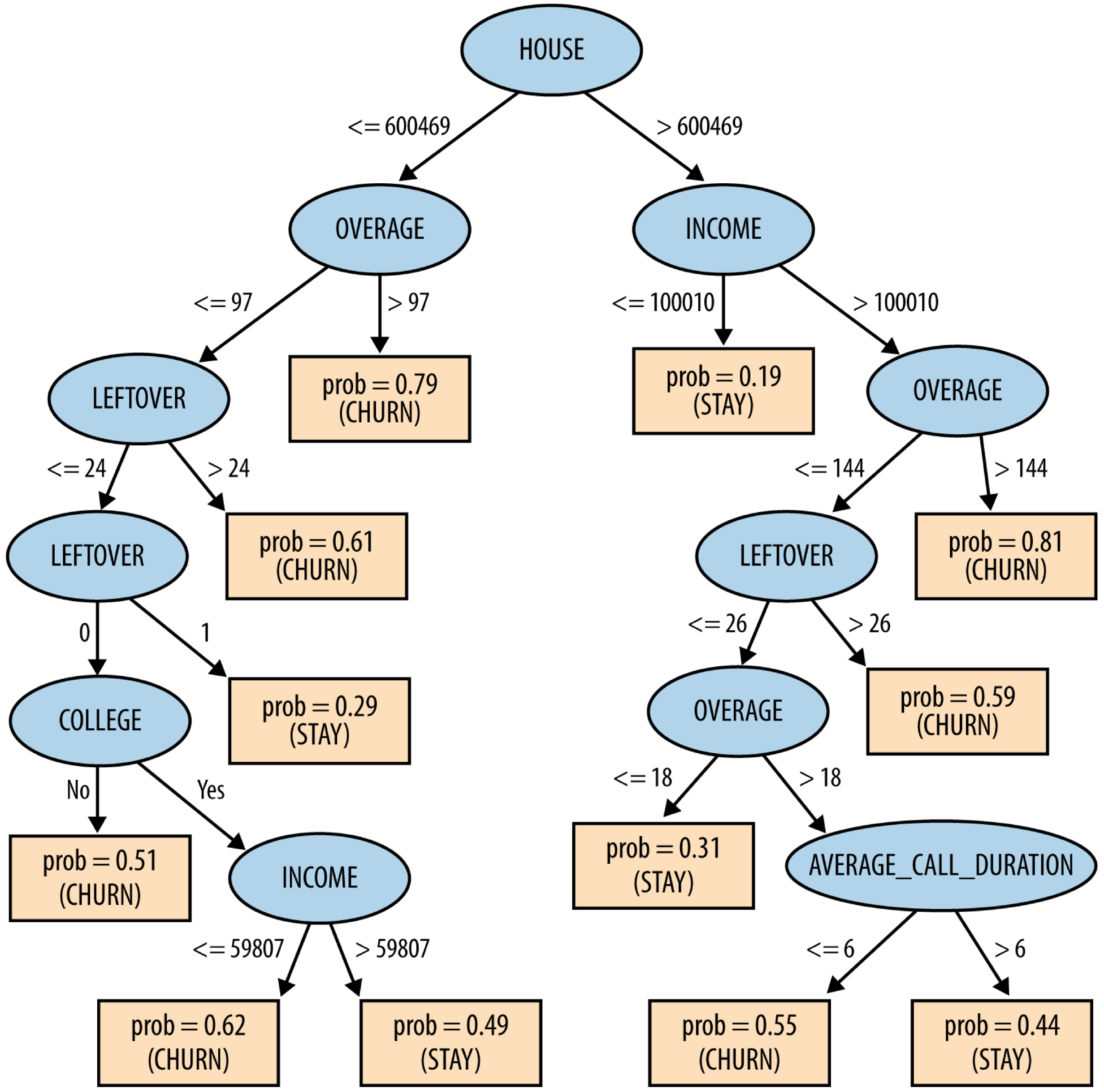
Information gains for each individual variables are: *House* .0461, *Overage* 0.0436, *Long call per month* .0356, *leftover* .0101 and etc

In the tree, terminal nodes are in square with the churn-probability and the classified class printed in square.

The tree achieved 73% accuracy of its decision for the training data.

1. Do we trust this number in the sense that the tree will produce 73% accuracy for a different data set?

2. If we do trust this number, is the model with 73% accuracy worth using?



HOUSE
<= 600469
> 600469
OVERAGE
<= 97
> 97
prob = 0.79 (CHURN)
LEFTOVER
<= 24
> 24
prob = 0.61 (CHURN)
LEFTOVER
0
1
prob = 0.29 (STAY)
COLLEGE
No
Yes
prob = 0.51 (CHURN)
INCOME
<= 59807
> 59807
prob = 0.62 (CHURN)
prob = 0.49 (STAY)
INCOME
<= 100010
> 100010
prob = 0.19 (STAY)
OVERAGE
<= 144
> 144
prob = 0.81 (CHURN)
LEFTOVER
<= 26
> 26
prob = 0.59 (CHURN)
OVERAGE
<= 18
> 18
prob = 0.31 (STAY)
AVERAGE_CALL_DURATION
<= 6
> 6
prob = 0.55 (CHURN)
prob = 0.44 (STAY)

A general strategy: grow up a big tree according to, e.g. the entropy criterion, then prune the tree by removing some internal nodes according to some criteria.

It is not a good idea to use a threshold to control the growth of the initial big tree, as an uninteresting branch may lead to some interesting branches later on.

A pruning criterion: for a given penalty constant $\lambda > 0$, search for the tree which minimizes

$$(\text{misclassification rate}) + \lambda \times (\text{number of terminal nodes})$$

Note. The 1st term measures the goodness of fit to the training data, and the 2nd term penalizes the complexity of model (i.e. the size of tree).

The role of the tuning parameter λ:

- $\lambda = 0$ leads to a big tree with the minimum misclassification on training data. But the resulting model typically performs badly in predicting non-training data

- too large λ leads to a tree with few branches, which underfits the data

For computational efficiency, the pruning is often carried out in step-wise fashion: at each step collapse one internal node which leads to the minimum increase in the misclassification rate.

Choose λ: 5-fold or 10-fold cross-validation.

Predicting Email Spam

Binary indicator Y: 1 – spam, 0 – (genuine) email

57 predictive variables:

- 48 quantitative variables — percentages of the given 48 words including *business, address, internet, free, george*

- 6 quantitative variables — percentages of the 6 characters ; ([! \$ # (among all characters)

- The average length of uninterrupted sequences of capital letters: CAPAVE

- The length of the longest uninterrupted sequence of capital letters: CAPMAN

- The sum of the length of uninterrupted sequences of capital letters: CAPTOT

Use 3065 data points (i.e. emails) for estimation, and randomly selected 1536 points for testing.

The entropy measure was used to grow the tree, then the misspecification rate was used to prune the tree.

Figure 9.4 shows that the misclassification rate flattens out at around 17 terminal nodes, giving the pruned tree in Figure 9.5.

		Predicted	Class
		email	spam
True	email	57.3%	4.0%
Class	spam	5.3%	33.4%

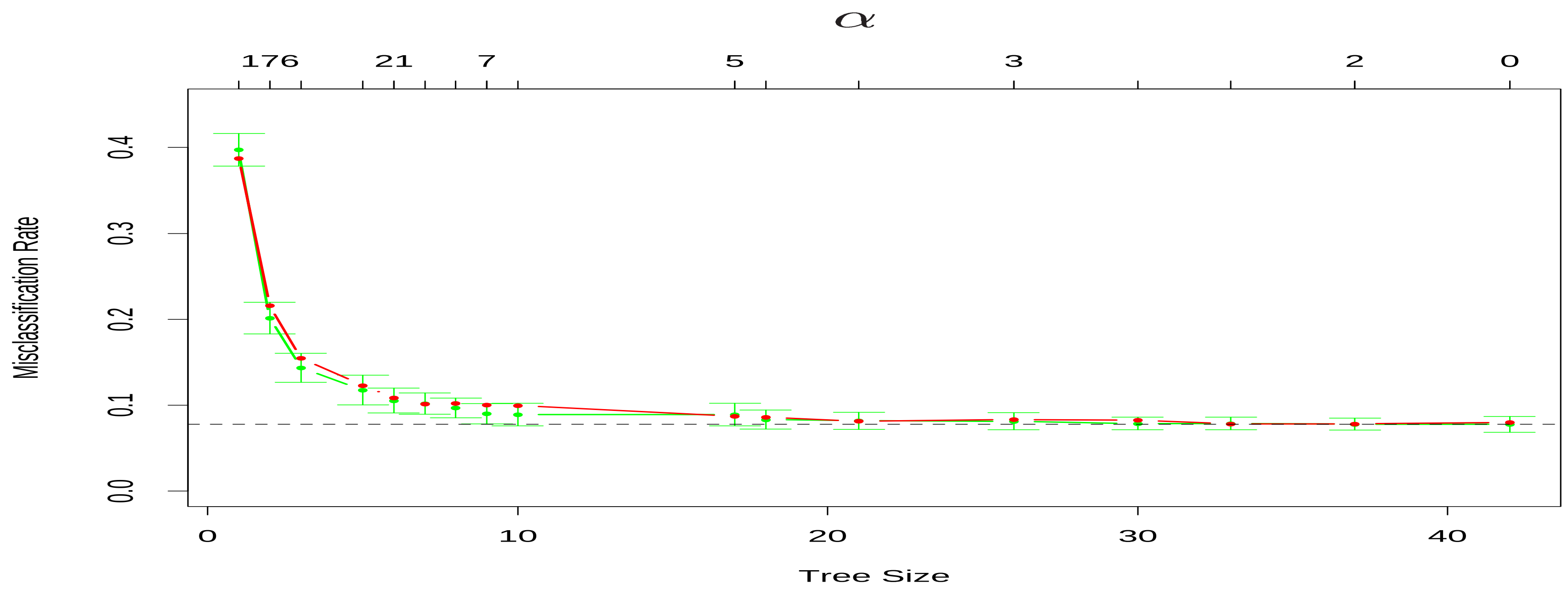
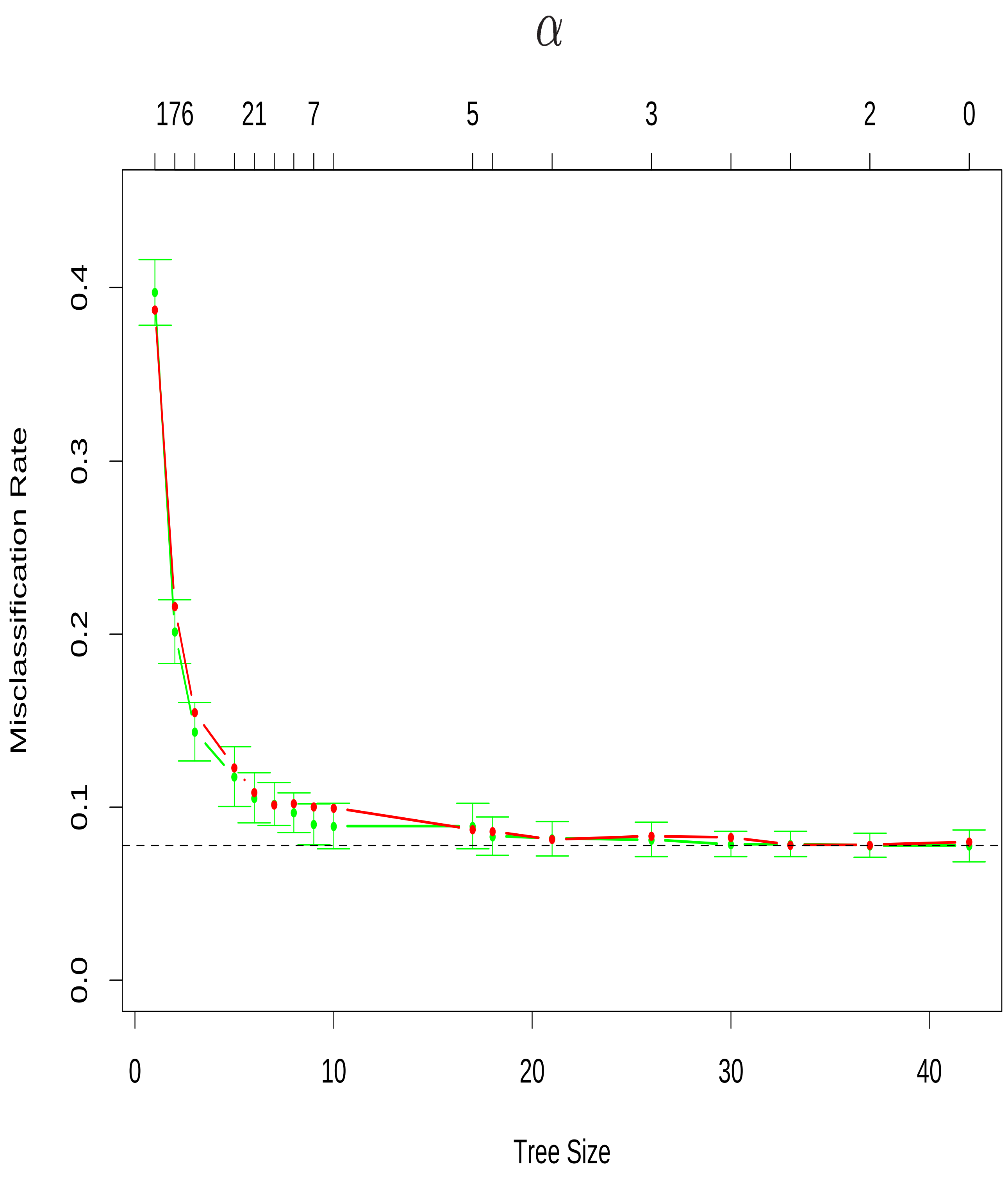


Figure 9.4: *Results for* **spam** *example. The green curve is the tenfold cross-validation estimate of misclassification rate as a function of tree size, with $\pm$ two standard error bars. The minimum occurs at a tree size with about 17 terminal nodes. The red curve is the test error, which tracks the CV error quite closely. The cross-validation was indexed by values of α, shown above. The tree sizes shown below refer to $|T_\alpha|$, the size of the original tree indexed by α.*


α
176
21
7
5
3
2
0
Misclassification Rate
0.0
0.1
0.2
0.3
0.4
0
10
20
30
40
Tree Size

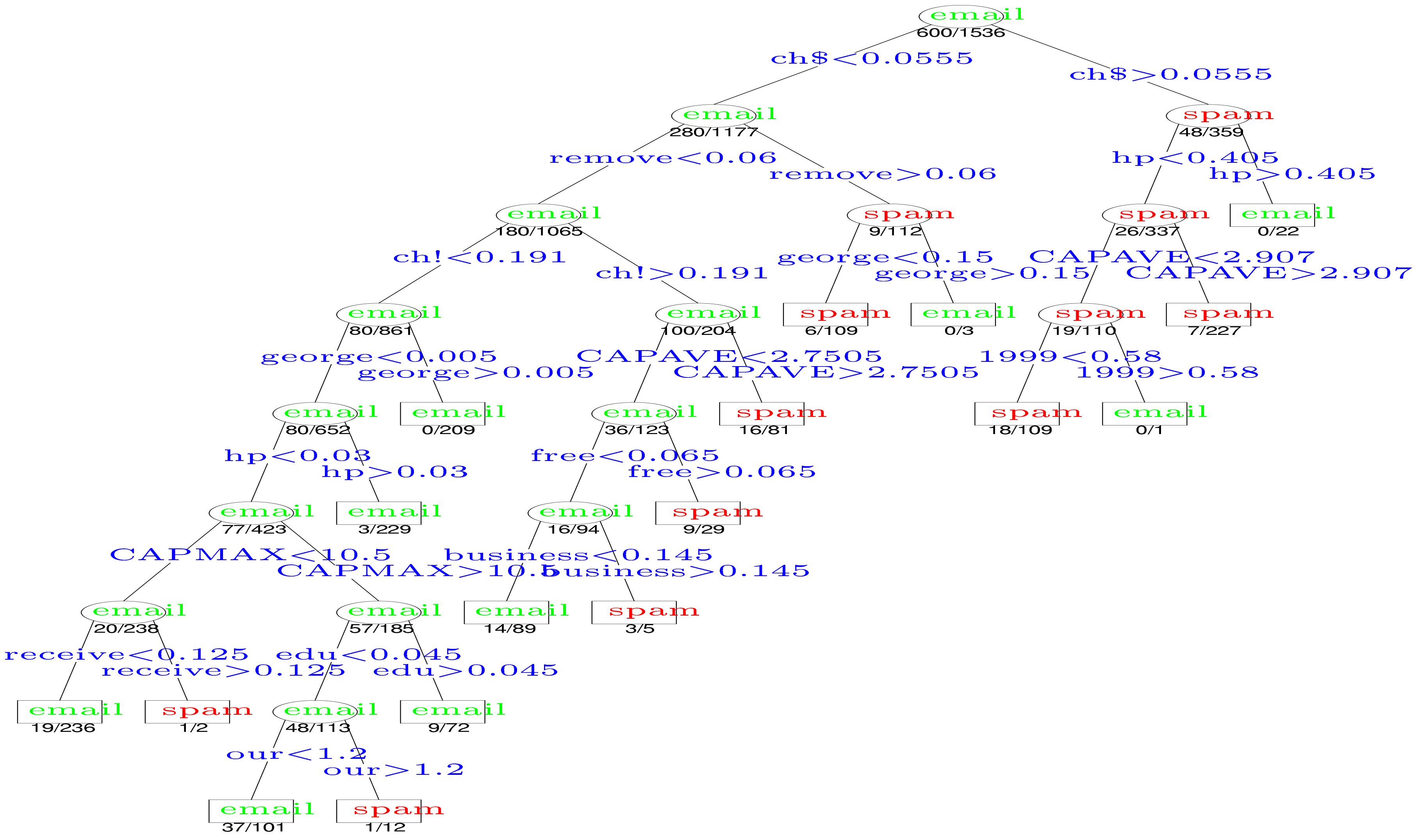


Figure 9.5: *The pruned tree for the* spam *example. The split variables are shown in blue on the branches, and the classification is shown in every node. The numbers under the terminal nodes indicate misclassification rates on the test data.*

On the rightmost branches of the tree, we have a spam warning if more than 5.5% of the characters are ‘$’. However if in addition the phrase hp occurs frequently, it is likely to be company business and we classify as email. All of the 22 cases in the test set satisfying these criteria were correctly classified. If the second condition is not met, and in addition the average length of capital letters CAPAVE is larger than 2.9, we classify as spam. Of the 227 test cases, only 7 were misspecified.

Note. A classifier may take the same value on the two sub-regions from one splitting.

Advantage: easy to explain and interpret graphically, easy to handle qualitative predictors

Disadvantage: less accurate than other classification methods.

However improvements are possible by aggregating many decision trees using *bagging, random forests* and *boosting*.

Fitting a decision tree with package `tree`

We use a data set `Carseats` from the package `ISLR` for the illustration.

```
> install.packages("tree"); install.packages("ISLR")
> library(tree); library(ISLR)
> Carseats
    Sales CompPrice Income Advertising Population Price ShelveLoc Age Education Urban  US
1    9.50       138     73          11        276   120       Bad  42        17   Yes Yes
2   11.22       111     48          16        260    83      Good  65        10   Yes Yes
3   10.06       113     35          10        269    80    Medium  59        12   Yes Yes
4    7.40       117    100           4        466    97    Medium  55        14   Yes Yes
... ...
```

The data set contains info on car seat sales in 400 stores with 11 variables:

Sales: Unit sales (in thousands) at each location

CompPrice: Price charged by competitor at each location

Income: Community income level (in thousands of dollars)

Advertising: Local advertising budget for company at each location (in thousands of dollars)

Population: Population size in region (in thousands)

Price: Price company charges for car seats at each site

ShelveLoc: A factor with levels Bad, Good and Medium indicating the quality of the shelving location for the car seats at each site

Age: Average age of the local population

Education: Education level at each location

Urban: A factor with levels No and Yes to indicate whether the store is in an urban or rural location

US: A factor with levels No and Yes to indicate whether the store is in the US or not

```
> attach(Carseats)
> High=ifelse(Sales<=8, "No", "Yes")    # Define the label High iff Sales >8
> Carseats2=data.frame(Carseats, High)   # combine the label into the data set
> tree.carseats=tree(High~.-Sales, Carseats2) # . indicates using all the predictors,
                                             # -Sales: exclude Sales
> summary(tree.carseats)

Classification tree:
tree(formula = High ~ . - Sales, data = Carseats2)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Income"      "CompPrice"   "Population"
[6] "Advertising" "Age" "US"
Number of terminal nodes:  27
Residual mean deviance:  0.4575 = 170.7 / 373
Misclassification error rate: 0.09 = 36 / 400
```

The error rate for the training data is 9%.

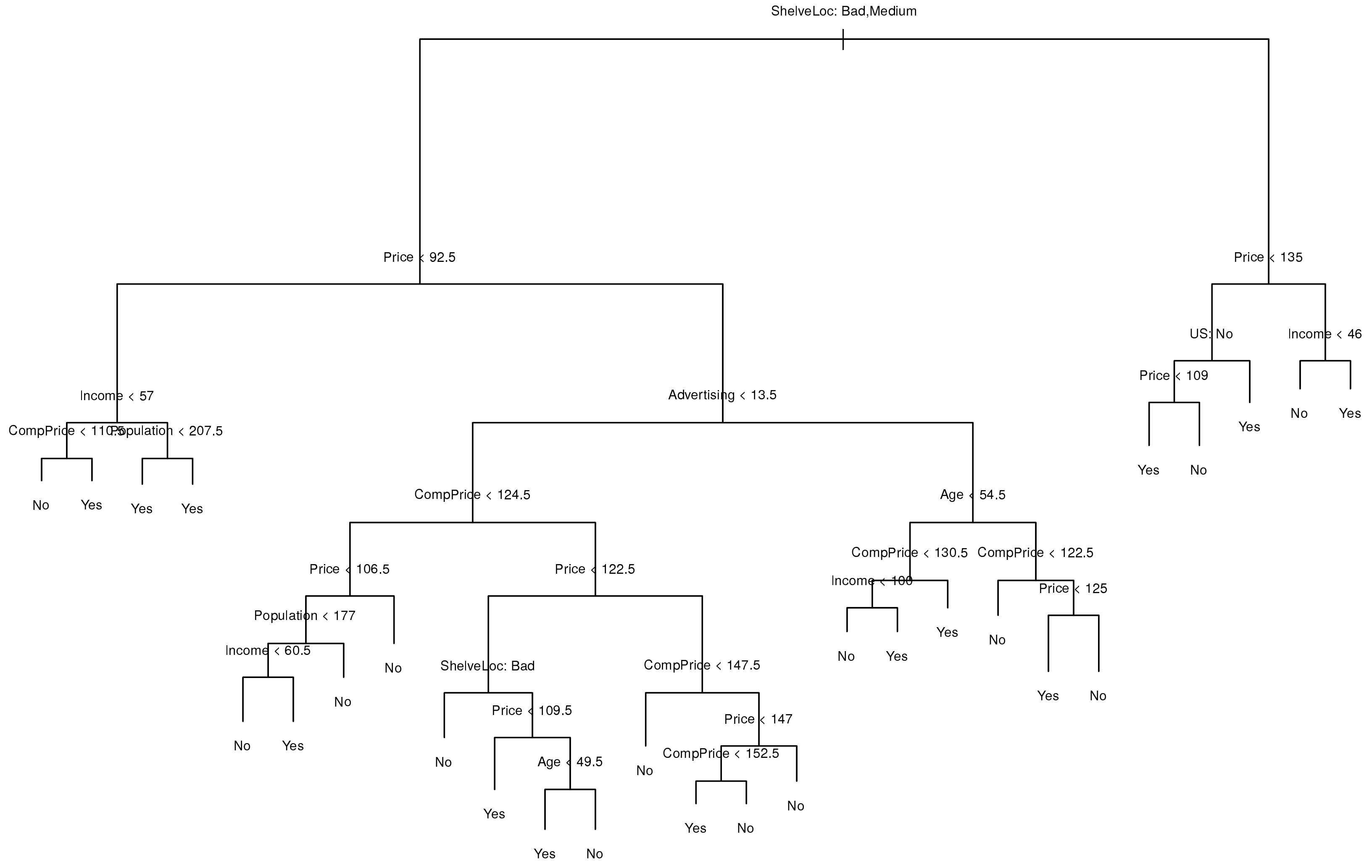
The deviance is defined as

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

where n_{mk} is the number of observations in the m-th terminal node that belong to the k-th class, $\widehat{p}_{mk} = n_{mk}/n_m$, and n_m is the total number of observations in the m-th terminal node. The smaller the deviance is, the better fit for the (training) data.

```
> plot(tree.carseats)
> text(tree.carseats, pretty=0, cex=0.6)
```

The most important indicator is shelving location!



ShelveLoc: Bad,Medium
Price < 92.5
Price < 135
US: No
Income < 46
Price < 109
Income < 57
Advertising < 13.5
CompPrice < 110.5
Population < 207.5
CompPrice < 124.5
Age < 54.5
Price < 106.5
Price < 122.5
CompPrice < 130.5
CompPrice < 122.5
Income < 100
Price < 125
Population < 177
Income < 60.5
ShelveLoc: Bad
CompPrice < 147.5
Price < 109.5
Price < 147
Age < 49.5
CompPrice < 152.5
No
Yes

To assess the performance of the fitted tree to new data, we split the observations into two parts: training set and testing set.

```
> train=sample(1:nrow(Carseats2), 200) # randomly select 200 numbers between 1
                                       # and nrow(Carseats2)
> testData=Carseats2[-train,]      # test data for checking performance
> High.test=High[-train]
> tree2=tree(High~.-Sales, Carseats2, subset=train)
> tree.pred=predict(tree2, testData, type="class") # type="vector" returns
                             # predictive probabilities, check ?predict.tree
> table(tree.pred, High.test)
         High.test
tree.pred No Yes
      No  81  23
      Yes 35  61
> (23+35)/(23+35+81+61)
[1] 0.29   # Misclassfication rate for the testing data
```

We expect that the accuracy for classifying new data from this fitted tree would be about 1-0.29=71%.

Bagging: a bootstrap aggregation method

Decision tree suffers from *high variance*, i.e. the tree depends on training data sensitively.

Basic idea to reduce variance: average a set of observations (such as a sample mean)

Create B sets of training data by bootstrapping from the original data set. For each bootstrap sample, create a decision tree.

Those trees are grown deep, and are not pruned. Hence each individual tree has high variance but low bias.

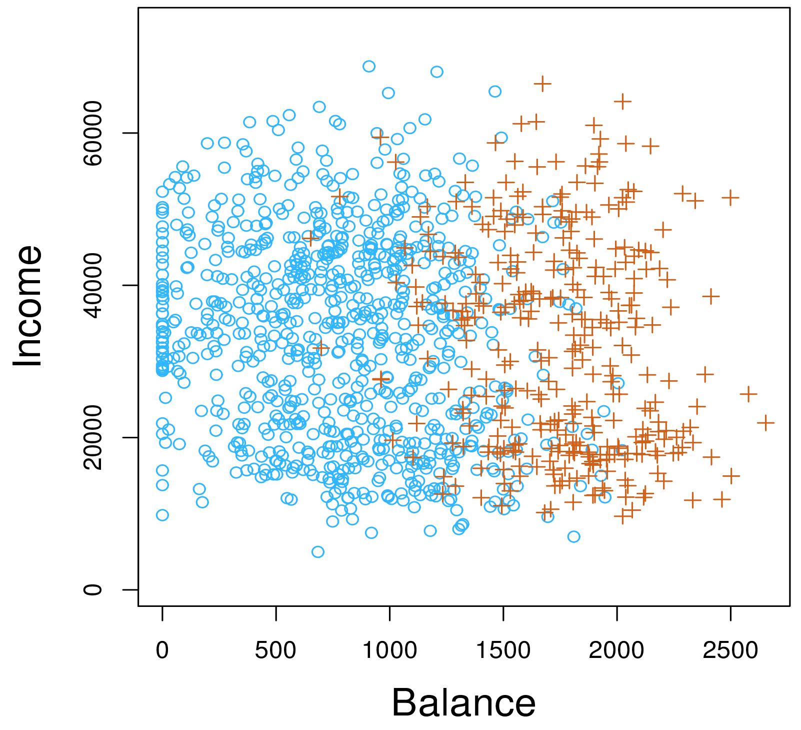
'Averaging' those B trees reduces the variance.

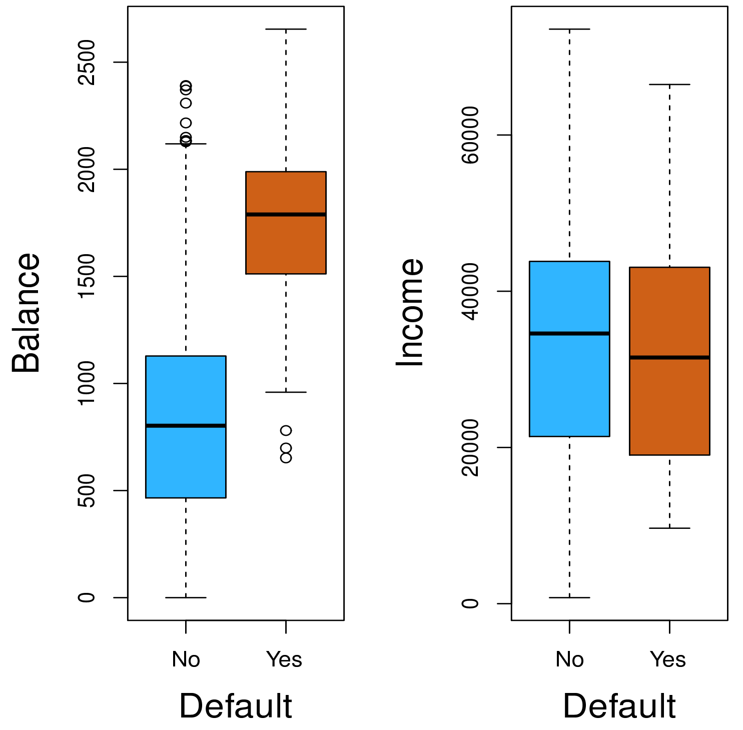
For decision trees, 'averaging' means taking the majority votes of the B trees.

Random forests: similar to Bagging, but at each split only choose from randomly selected $\sqrt{p}$ input variables

Boosting: create a strong learner from a set of weak learners.

Logistic Regression: model the probability that Y belongs to a particular class.





Goal: predict if an individual will default on his/her credit card payment, using the credit card *balance* and the annual *income*.

Balance is a more effective predictor!

Let

$$P(\text{ default } = \text{ Yes} \,|\, \text{balance}) \equiv P(Y = 1|X) \equiv p(X).$$

Then $p(X) \in [0, 1]$, it varies wrt *balance*.

Logistic Regression:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

Consequently,

$$1 - p(X) = P(Y = 0|X) = \frac{1}{1 + e^{\beta_0+\beta_1 X}},$$

and log-odds or logit is

$$\log \frac{p(X)}{1 - p(X)} = \beta_0 + \beta_1 X.$$

Estimation for logistic regression: Suppose we have training data $(Y_i, X_i), i = 1, \cdots n$.

Likelihood function:

$$L(\beta_0, \beta_1) = \prod_{i:Y_i=1} p(X_i) \prod_{j:Y_j=0} (1 - p(X_j))$$

Log-likelihood function:

$$l(\beta_0, \beta_1) = \sum_{i:Y_i=1} \log p(X_i) + \sum_{j:Y_j=0} \log(1 - p(X_j)).$$

The maximum likelihood estimators (MLE):

$$(\widehat{\beta}_0, \widehat{\beta}_1) = \arg\max_{\beta_0,\beta_1} l(\beta_0, \beta_1) = \arg\max_{\beta_0,\beta_1} L(\beta_0, \beta_1).$$

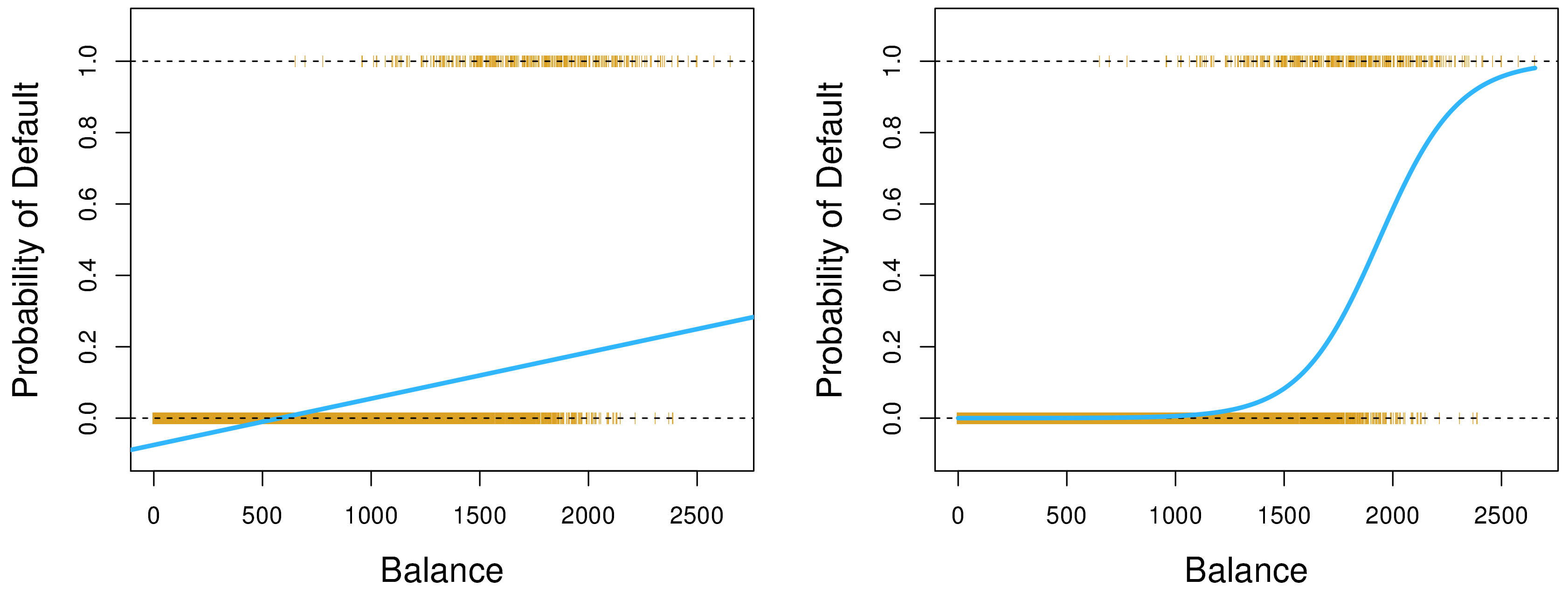
	Coefficient	Std error	Z-statistic	P-value
Intercept β_0	-10.6513	0.3612	-29.5	<0.0001
balance β_1	0.0055	0.0002	24.9	<0.0001

A unit increase in balance leads to an increase of 0.0055 in the log odds of default

Prediction: For an individual with balance \$1000, the predicted default probability is

$$\widehat{p}(1000) = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.00576,$$

i.e. the probability that this individual will default is less than 1%. However, for individual with balance \$2000, the predicted default probability is 58.6%.



Right panel: plot of the estimated $p(X)$ against X. Orange dots are the training data (X_i, Y_i) used in estimation.

Left panel: direct linear regression estimation $Y = \widehat{\beta}_0 + \widehat{\beta}_1 X$

Logistic regression with a discrete predictor

Predict credit card default using student status indicator: $X = 1 -$ students, $X = 0 -$ non-student:

	Coefficient	Std error	Z-statistic	P-value
Intercept	-3.5041	0.0707	-49.55	<0.0001
student	0.4049	0.1150	3.52	0.0004

This leads to the predictive probabilities:

$$P(\text{default}|\text{student}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$
$$P(\text{default}|\text{non-student}) = \frac{e^{-3.5041}}{1+e^{-3.5041}} = 0.0292.$$

Thus students tend to have higher default probabilities than non-students — useful info for credit card companies.

Do they need to know more?

Multiple logistic regression: Let

$$p(X_1, \cdots, X_p) \equiv P(Y = 1 | X_1, \cdots, X_p) = 1 - P(Y = 0 | X_1, \cdots X_p)$$

Then the model is of the form

$$p(\mathbf{X}) = \frac{e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p}},$$

or equivalently, the log odds is linear in $X_1, \cdots, X_p$

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

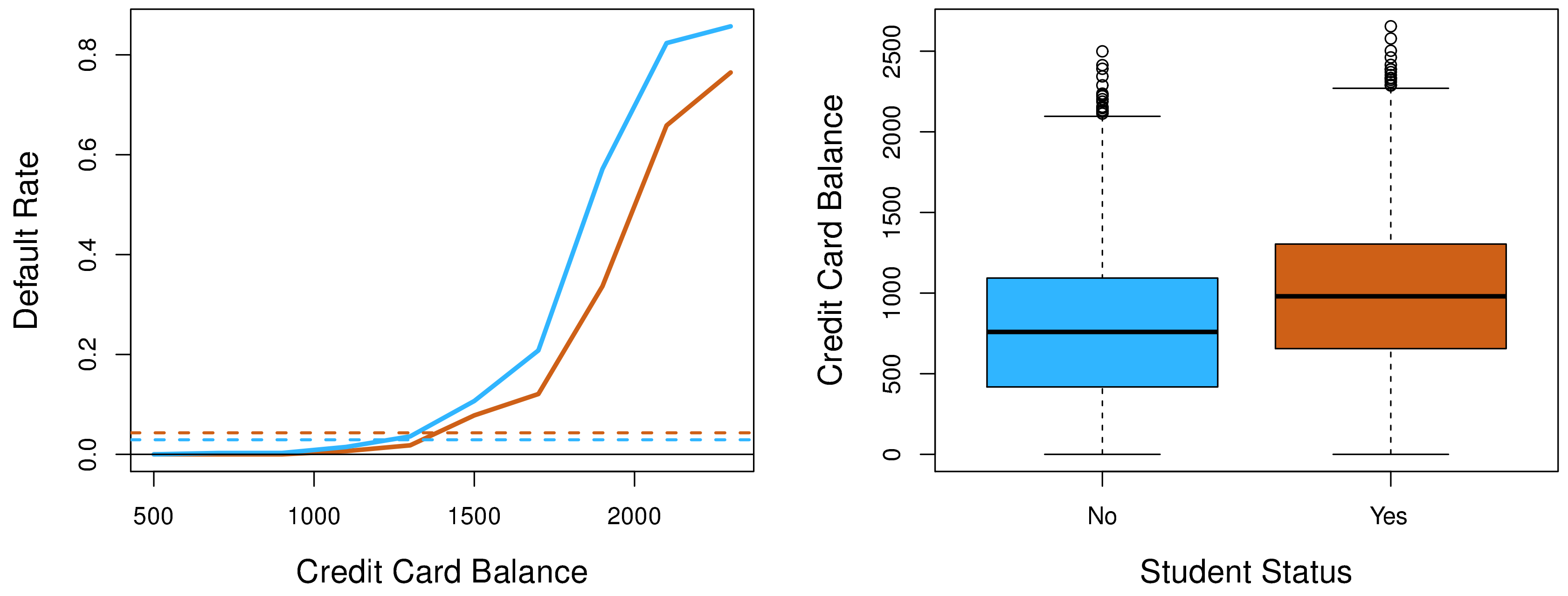
Fitting a logistic model for predicting default using *balance*, *income* and *student status* (i.e. 1 for student, and 0 for non-student):

	Coefficient	Std error	Z-statistic	P-value
Intercept	-10.8690	0.4932	-22.08	<0.0001
balance	0.0057	0.0002	24.74	<0.0001
income	0.0030	0.0082	0.37	0.7115
student	-0.6468	0.2362	-2.74	0.0062

P-value for *Income* is large, indicating that it has no predictive power to the possibility of default. Thus it can and should be removed from the model

Coefficient for student is negative, indicating that students are less likely to default than non-students — Contradictory?

Confounding between *Balance* X_1 and *Student status* X_2



Left panel: plots of default probability $\widehat{p}(X_1, X_2)$ against X_1 with $X_2 = 1$ (student) in orange, and $X_2 = 0$ (non-student) in blue.

With the same balance, students are less likely to default than non-students

Right panel: Boxplots of *Balance* for students in orange, and non-students in blue.

Summary — information for credit card companies

- Individual income has no predictive power on default
- A student is riskier than a non-student in general
- With the same credit card balance, a student is less riskier than a non-student

Logistic regression in R

Consider dataset `Smarket` from package `ISLR`, which contains the daily percentage returns of S&P 500 index over 1,250 days in $2001 - 2005$.

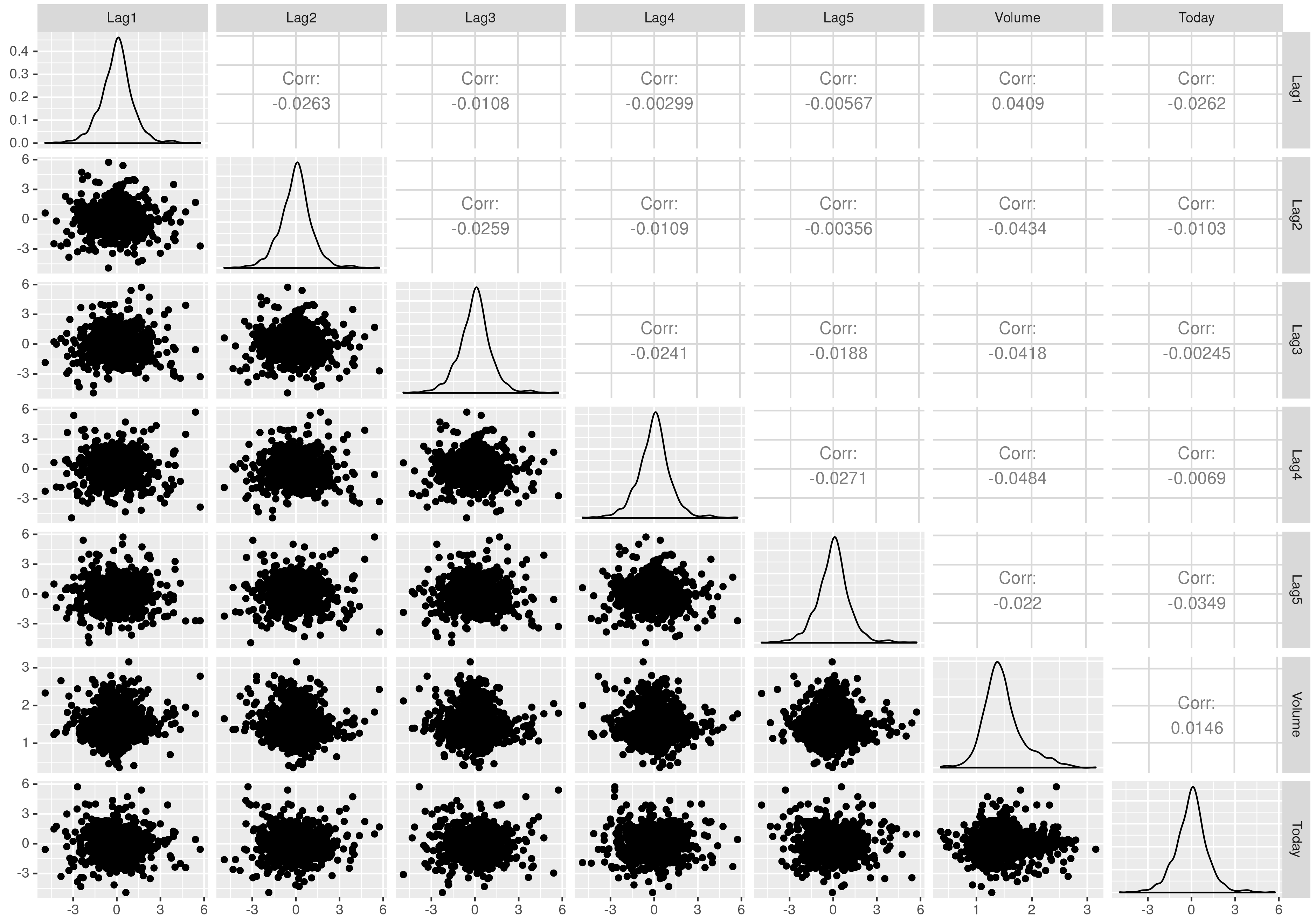
```
> library(ISLR)
> names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"      "Volume"
[8] "Today"     "Direction"
> dim(Smarket)
[1] 1250    9
> summary(Smarket)
      Year           Lag1                Lag2                Lag3                Lag4
 Min.   :2001   Min.   :-4.9220   Min.   :-4.9220   Min.   :-4.9220   Min.   :-4.9220
 1st Qu.:2002   1st Qu.:-0.6395   1st Qu.:-0.6395   1st Qu.:-0.6400   1st Qu.:-0.6400
 Median :2003   Median : 0.0390   Median : 0.0390   Median : 0.0385   Median : 0.0385
 Mean   :2003   Mean   : 0.0038   Mean   : 0.0039   Mean   : 0.0017   Mean   : 0.0016
 3rd Qu.:2004   3rd Qu.: 0.5967   3rd Qu.: 0.5967   3rd Qu.: 0.5967   3rd Qu.: 0.5967
 Max.   :2005   Max.   : 5.7330   Max.   : 5.7330   Max.   : 5.7330   Max.   : 5.7330
      Lag5              Volume            Today            Direction
 Min.   :-4.9220   Min.   :0.3561   Min.   :-4.9220   Down:602
 1st Qu.:-0.6400   1st Qu.:1.2574   1st Qu.:-0.6395   Up  :648
 Median : 0.0385   Median :1.4229   Median : 0.0385
 Mean   : 0.0056   Mean   :1.4783   Mean   : 0.0031
 3rd Qu.: 0.5970   3rd Qu.:1.6417   3rd Qu.: 0.5967
 Max.   : 5.7330   Max.   :3.1525   Max.   : 5.7330

> round(cor(Smarket[-c(1,9)]), digits=4)
          Lag1    Lag2    Lag3    Lag4    Lag5  Volume   Today
Lag1    1.0000 -0.0263 -0.0108 -0.0030 -0.0057  0.0409 -0.0262
Lag2   -0.0263  1.0000 -0.0259 -0.0109 -0.0036 -0.0434 -0.0103
Lag3   -0.0108 -0.0259  1.0000 -0.0241 -0.0188 -0.0418 -0.0024
Lag4   -0.0030 -0.0109 -0.0241  1.0000 -0.0271 -0.0484 -0.0069
Lag5   -0.0057 -0.0036 -0.0188 -0.0271  1.0000 -0.0220 -0.0349
Volume  0.0409 -0.0434 -0.0418 -0.0484 -0.0220  1.0000  0.0146
Today  -0.0262 -0.0103 -0.0024 -0.0069 -0.0349  0.0146  1.0000

> install.packages("GGally")
> library(GGalley)
> ggpairs(Smarket[,-c(1,9)])
```

There are hardly any correlations among today's return and its lagged values – the efficient market hypothesis!

Note. `GGalley` is an added-on package to `ggplot2`. `ggpairs` presents more information than `pairs` – a standard plot in R.


Lag1
Lag2
Lag3
Lag4
Lag5
Volume
Today
Corr:
-0.0263
Corr:
-0.0108
Corr:
-0.00299
Corr:
-0.00567
Corr:
0.0409
Corr:
-0.0262
Corr:
-0.0259
Corr:
-0.0109
Corr:
-0.00356
Corr:
-0.0434
Corr:
-0.0103
Corr:
-0.0241
Corr:
-0.0188
Corr:
-0.0418
Corr:
-0.00245
Corr:
-0.0271
Corr:
-0.0484
Corr:
-0.0069
Corr:
-0.022
Corr:
-0.0349
Corr:
0.0146

Now we fit a logistic model for predicting the direction of market:

```
> logistic.Smarket=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket,
                      family=binomial)
> summary(logistic.Smarket)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.446  -1.203   1.065   1.145   1.326

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.126000   0.240736  -0.523    0.601
Lag1        -0.073074   0.050167  -1.457    0.145
Lag2        -0.042301   0.050086  -0.845    0.398
Lag3         0.011085   0.049939   0.222    0.824
Lag4         0.009359   0.049974   0.187    0.851
Lag5         0.010313   0.049511   0.208    0.835
Volume       0.135441   0.158360   0.855    0.392

AIC: 1741.6
```

```
Number of Fisher Scoring iterations: 3
```

The most significant predictor is Lag1 with a negative coefficient; indicating mean regression. Note the p-value is 0.145 — not very significant.

Function `predict` can be used to predict the probability that the market will go up, given values of the predictors. The type="response" option tells R to output probabilities of the form $P(Y = 1|X)$, as opposed to other information such as the logit. If no data set is supplied, the probabilities are computed for the training data that was used to fit the logistic regression model. Note $Y = 1$ stands for ‘up’, as

```
> contrasts(Direction)
Down  0
Up    1
```

```
> pred.Smarket=predict(logistic.Smarket, type="response")
> pred.SmarketDiction=rep("Down", 1250) # a sequence of "Down" repeated 1250 times
> pred.SmarketDiction[pred.Smarket>.5]="Up"
> table(pred.SmarketDiction, Direction)
                   Direction
pred.SmarketDiction Down  Up
               Down  145 141
               Up    457 507
> (145+507)/1250
[1] 0.5216  # accuracy rate on the training data
```

Given most the predictors are insignificant, we re-run the fitting using only Lag1 and Lag2:

```
>logistic.Smarket=glm(Direction~Lag1+Lag2, data=Smarket, family=binomial)
> pred.Smarket=predict(logistic.Smarket, type="response")
> pred.SmarketDiction[pred.Smarket>.5]="Up"
> table(pred.SmarketDiction, Direction)
                   Direction
pred.SmarketDiction Down  Up
               Down   93  84
               Up    509 564
> (93+564)/1250
[1] 0.5256
```

Those accuracy rates are expected to be greater than the real rates when we use the fitting on new data.

An alternative is to split the sample into training and testing subsets:

```
> Smarket.2005=Smarket[Year>=2005,]
> dim(Smarket.2005)
[1] 252   9
> Direction.2005=Direction[Year==2005]
> logistic.Smarket=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket,
                 family=binomial, subset=(Year <2005))
> Prob.2005=predict(logistic.Smarket, Smarket.2005, type="response")
> PredDirection.2005=rep("Down", 252)
> PredDirection.2005[Prob.2005>0.5]="Up"
> table(PredDirection.2005, Direction.2005)
                  Direction.2005
PredDirection.2005 Down Up
              Down   77 97
              Up     34 44
> (77+44)/252
[1] 0.4801587  # Accuracy rate on testing sample
```

Now we use only two lagged variables:

```
> logistic.Smarket=glm(Direction~Lag1+Lag2, data=Smarket, family=binomial,
              subset=(Year <2005))
> Prob.2005=predict(logistic.Smarket, Smarket.2005, type="response")
> PredDirection.2005=rep("Down", 252)
> PredDirection.2005[Prob.2005>0.5]="Up"
> table(PredDirection.2005, Direction.2005)
                 Direction.2005
PredDirection.2005 Down  Up
              Down   35  35
              Up     76 106
> (35+106)/252
[1] 0.5595238
```

The overfitted model (with 6 predictors) perform about the same on the training data as the model with the 2 predictors. However it performs much worse on the testing data!