# CLASSIFICATION

## Introducción a la Ciencia de Datos

Some of the figures in this presentation are taken from: An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

Some slides are based on Abbass Al Sharif's slides for his course DSO 530: Applied Modern Statistical Learning Techniques.

# Overview of Classification

- Examples:
  ① A person arrives at the emergency room with a set of symptoms that could possibly be attributed to one of three medical conditions. Which of the three conditions does the individual have?
  ② An online banking service must be able to determine whether or not a transaction being performed on the site is fraudulent, on the basis of the user's IP address, past transaction history, and so on.
  ③ On the basis of DNA sequence data for a number of patients with and without a given disease, a biologist would like to figure out which DNA mutations are deleterious (disease-causing) and which are not.

# Overview of Classification

- We will always assume that we have observed a set of $n$ different data points. These observations are called the **training data** because we will use these observations to train, or teach, our method how to estimate $f$.

- Let $x_{ij}$ represent the value of the $j^{th}$ predictor, or input, for observation $i$, where $i = 1,2,...,n$ and $j = 1,2,...,p$.
- Correspondingly, let $y_i$ represent the response variable for the $i^{th}$ observation. Then our training data consist of $\{(x_1,y_1),(x_2,y_2),...,(x_n,y_n)\}$ where $x_i = (x_{i1},x_{i2},...,x_{ip})^T$.

# Overview of Classification

- Our goal is to apply a **classification method** to the training data in order to estimate the unknown function *f*.
- In other words, we want to find a function f such that $Y \approx f(X)$ for any observation $(X, Y)$.
- In general, we do not really care how well the classification method works training on the training data. Rather, we are interested in the accuracy of the predictions that we obtain when we apply our method to previously unseen **test data**.

# K-NN

## Classification Methods

# Understanding classification using NN

- **Nearest Neighbor (NN) Classifiers** are defined by their characteristic of classifying unlabeled examples by assigning them the class of the most similar labeled examples.
- NN classifiers are well-suited for classification tasks where relationships among the features and the target classes are difficult to understand, yet the items of similar class type tend to be fairly homogeneous.
- If there is not a clear distinction among the groups, the algorithm is by and large not well-suited for identifying the boundary.

# The k-NN algorithm

- The k-NN algorithm begins with a training dataset made up of examples that are classified into several categories, as labeled by a nominal variable.
- Assume that we have a *test dataset* containing unlabeled examples that otherwise have the same features as the *training data*.
- For each record in the test dataset, k-NN identifies *k* records in the training data that are the "nearest" in similarity, where *k* is an integer specified in advance.
- The unlabeled test instance is assigned the class of the majority of the *k* nearest neighbors.

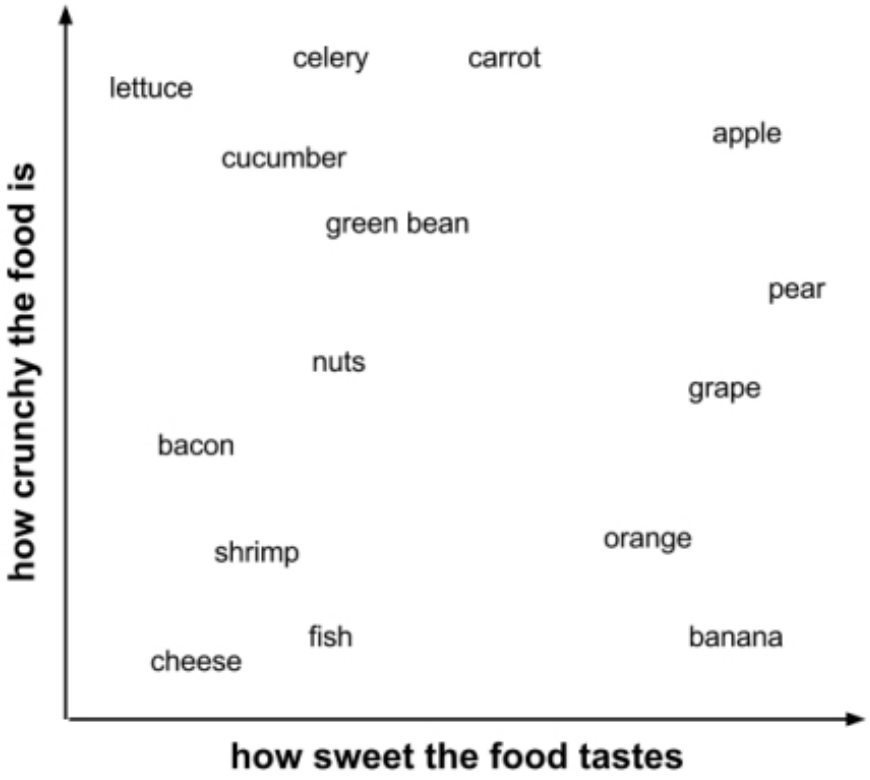
# The k-NN algorithm

- Dataset: Blind tasting experience:
  - Only two features of each ingredient is recorded: (1) a measure from 1 to 10 of how crunchy the ingredient is, and (2) a measure from 1 to 10 score of how sweet the ingredient tastes.
  - We then labeled each ingredient as one of three types of food: fruits, vegetables, or proteins.

| ingredient | sweetness | crunchiness | food type |
|---|---|---|---|
| apple | 10 | 9 | fruit |
| bacon | 1 | 4 | protein |
| banana | 10 | 1 | fruit |
| carrot | 7 | 10 | vegetable |
| celery | 3 | 10 | vegetable |
| cheese | 1 | 1 | protein |

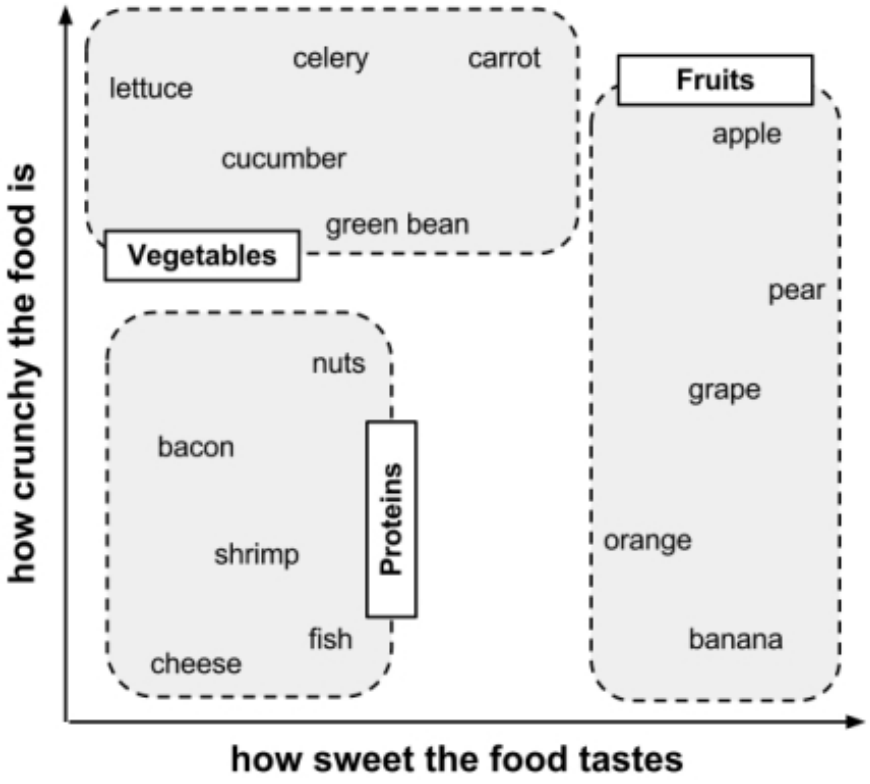
# The k-NN algorithm

- The k-NN algorithm treats the features as coordinates in a multidimensional **feature space**:



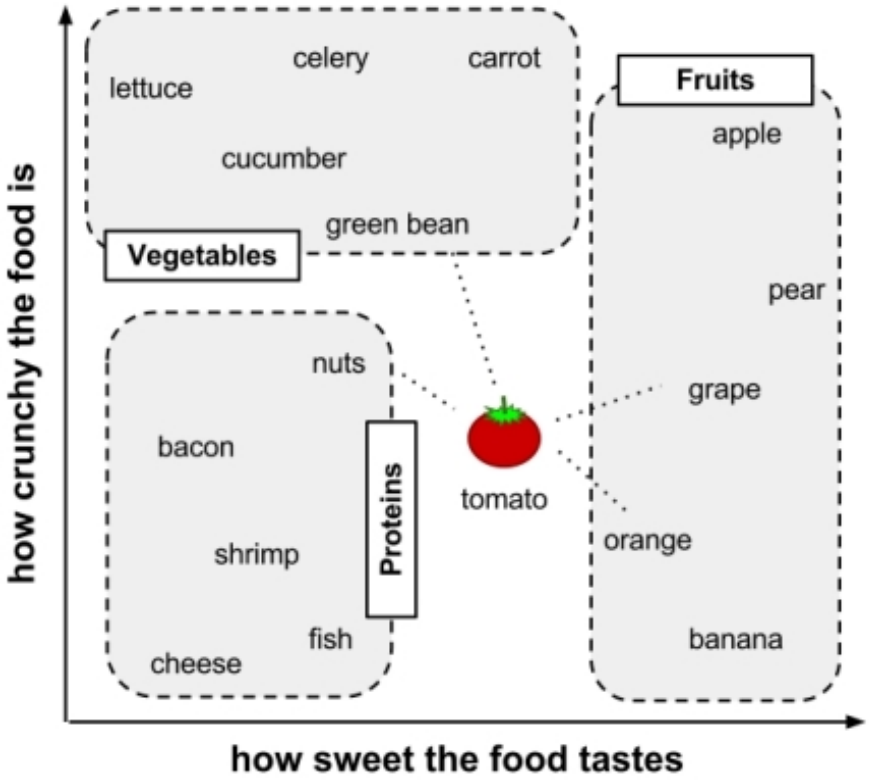
# The k-NN algorithm

- Similar types of food tend to be grouped closely together:



# The k-NN algorithm

- Use k-NN to settle the age-old question: is a tomato a fruit or a vegetable?



# The k-NN algorithm

- Locating the tomato's nearest neighbors requires a **distance function**, or a formula that measures the similarity between two instances.
- Traditionally, the k-NN algorithm uses **Euclidean distance**.

$$\text{dist}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2}$$

- where *p* and *q* are the examples to be compared, each having *n* features. The term $p_1$ refers to the value of the first feature of example *p*, while $q_1$ refers to the value of the first feature of example *q*.

# The k-NN algorithm

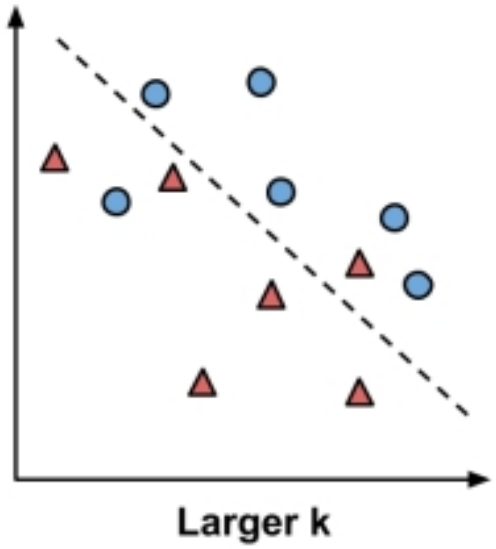
- For example, to calculate the distance between the tomato (sweetness = 6, crunchiness = 4), and the green bean (sweetness = 3, crunchiness = 7), we can use the formula as follows:

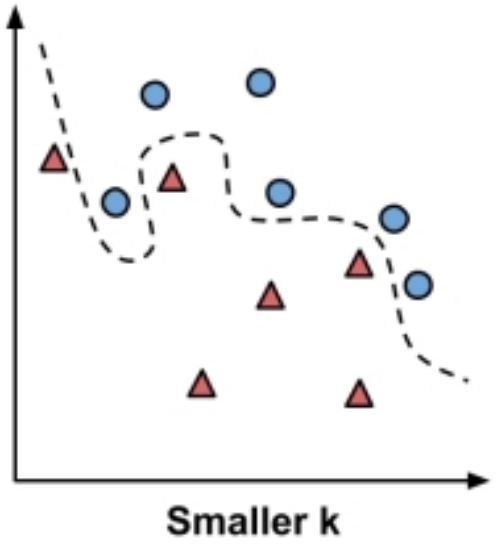
$$\text{dist}(tomato, green\ bean) = \sqrt{(6-3)^2 + (4-7)^2} = 4.2$$

| ingredient | sweetness | crunchiness | food type | distance to the tomato |
|---|---|---|---|---|
| grape | 8 | 5 | fruit | 2.2 |
| green bean | 3 | 7 | vegetable | 4.2 |
| nuts | 3 | 6 | protein | 3.6 |
| orange | 7 | 3 | fruit | 1.4 |

# Choosing an appropriate k

- The balance between *underfitting* and *overfitting* the training data is a problem known as the **bias-variance tradeoff**.



# Preparing data for use with k-NN

- Features are typically transformed to a standard range (normalized) prior to applying the k-NN algorithm.
- The rationale for this step is that the distance formula is dependent on how features are measured.
- In particular, if certain features have much larger values than others, the distance measurements will be strongly dominated by the larger values.

# Preparing data for use with k-NN

- Nominal features
  - Dummy coding
  - Transform to numeric keeping ordering...

# The k-NN algorithm

## STRENGTHS

- Simple and effective
- Makes no assumptions about the underlying data distribution
- Fast training phase

## WEAKNESSES

- Does not produce a model, which limits the ability to find novel insights in relationships among features
- Slow classification phase
- Requires a large amount of memory
- Nominal features and missing data require additional processing

# K-NN

R session

# Diagnosing breast cancer with k-NN

- We will use the "Breast Cancer Wisconsin Diagnostic" dataset from the *UCI Machine Learning Repository*, which is available at http://archive.ics.uci.edu/ml.
- The breast cancer data includes 569 examples of cancer biopsies, each with 32 features. One feature is an identification number, another is the cancer diagnosis, and 30 are numeric-valued laboratory measurements. The diagnosis is coded as M to indicate malignant or B to indicate benign.

# Diagnosing breast cancer with k-NN

```
# Local directory - use your own!!!!!!
setwd("/Users/rcrzarg/Documents/Docencia/MasterCienciaDatos/2017/")

# Load data
wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
# Examine the structure of the wbcd data frame
str(wbcd)
wbcd

# Drop the id feature
wbcd <- wbcd[,-1]

# Table of diagnosis
table(wbcd$diagnosis)

# Recode diagnosis as a factor
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"), labels = c("Benign", "Malignant"))
table(wbcd$diagnosis)

# Table or proportions with more informative labels
round(prop.table(table(wbcd$diagnosis)) * 100, digits = 1)

# Summarize three numeric features
summary(wbcd[,c("radius_mean", "area_mean", "smoothness_mean")])

# Normalize the wbcd data
wbcd_n <- as.data.frame(lapply(wbcd[,2:31], scale, center = TRUE, scale = TRUE))
# Confirm that normalization worked
summary(wbcd_n[,c("radius_mean", "area_mean", "smoothness_mean")])
boxplot(wbcd_n[,c("radius_mean", "area_mean", "smoothness_mean")])

plot(wbcd[,2:5])

plot(wbcd_n[,1:4], col=wbcd[,1])

cor(wbcd[,2:5])

cor(wbcd_n[,1:4])
```

# Diagnosing breast cancer with k-NN

```
# Create training and test data
shuffle_ds <- sample(dim(wbcd_n)[1])
eightypct <- (dim(wbcd_n)[1] * 80) %/% 100
wbcd_train <- wbcd_n[shuffle_ds[1:eightypct], ]
wbcd_test <- wbcd_n[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], ]

# Create labels for training and test data
wbcd_train_labels <- wbcd[shuffle_ds[1:eightypct], 1]
wbcd_test_labels <- wbcd[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], 1]

# Load the "class" library
library(class)
wbcd_test_pred <- knn(train = wbcd_train, test = wbcd_test, cl = wbcd_train_labels, k=21)
wbcd_test_pred

# Evaluating model performance
table(wbcd_test_pred,wbcd_test_labels)

require(caret)
knnModel <- train(x = wbcd_train, y = wbcd_train_labels, method = "knn")
class(knnModel)
knnModel

knnModel <- train(wbcd_train, wbcd_train_labels, method="knn", metric="Accuracy", tuneGrid = data.frame(.k=1:15))
knnModel

require(caret)
knnModel <- train(x = wbcd[shuffle_ds[1:eightypct],-1], y = wbcd[shuffle_ds[1:eightypct],1], method = "knn", preProc = c("center",
"scale"))
class(knnModel)
knnModel

knnPred <- predict(knnModel, newdata = wbcd[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], -1])
knnPred

postResample(pred = knnPred, obs = wbcd[shuffle_ds[(eightypct+1):dim(wbcd_n)[1]], 1])
```

# Exercise 1

- Try with different k choices and do a quick comparison.
  - You can draw a plot to show the results.

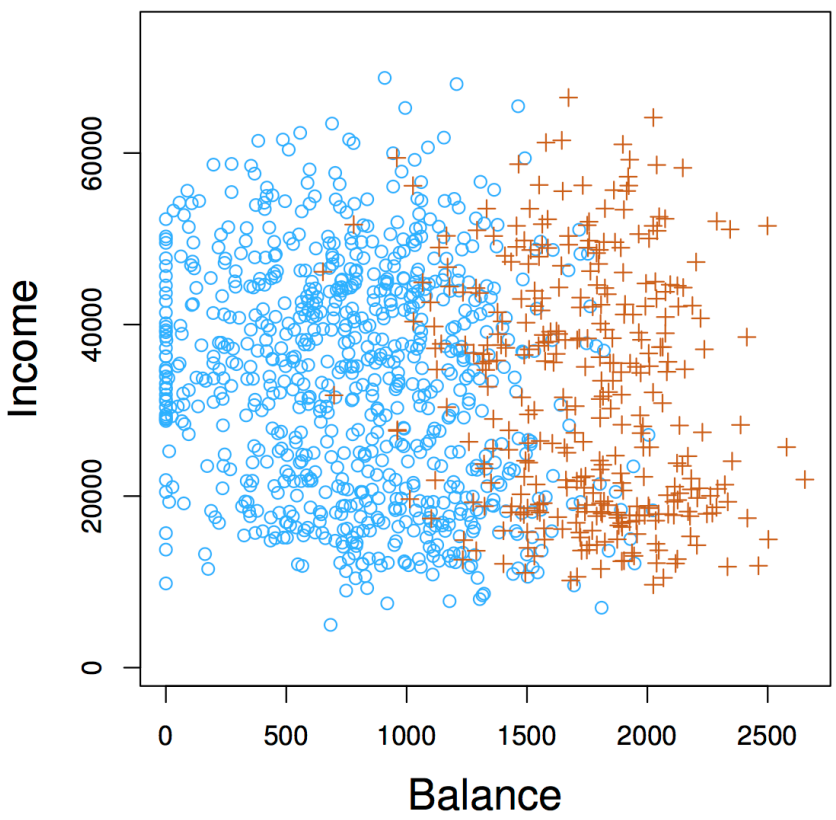
# LOGISTIC REGRESSION

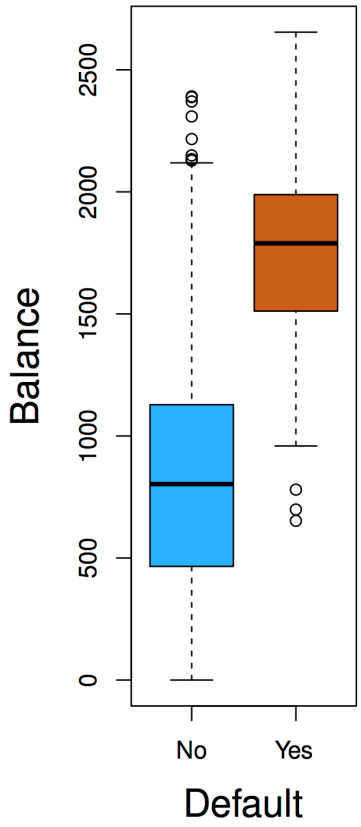
Classification Methods

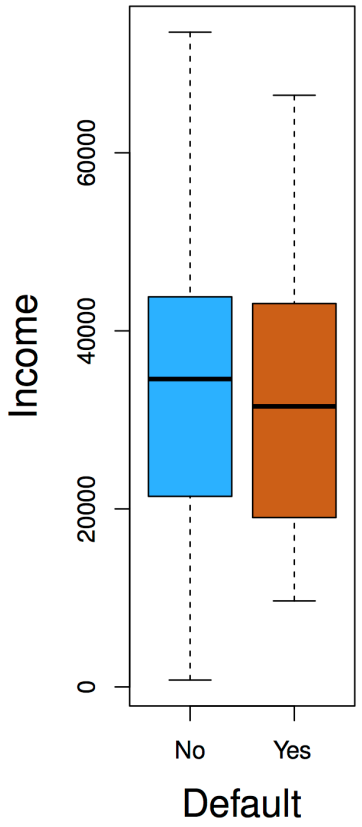
# Credit Card Default Data

- We would like to be able to predict customers that are likely to default.
- X variables:
  - Income
  - Balance
  - Student
- The Y variable (default) is categorical: Yes or No
- How do we check the relationship between Y and X?

# The Default Dataset

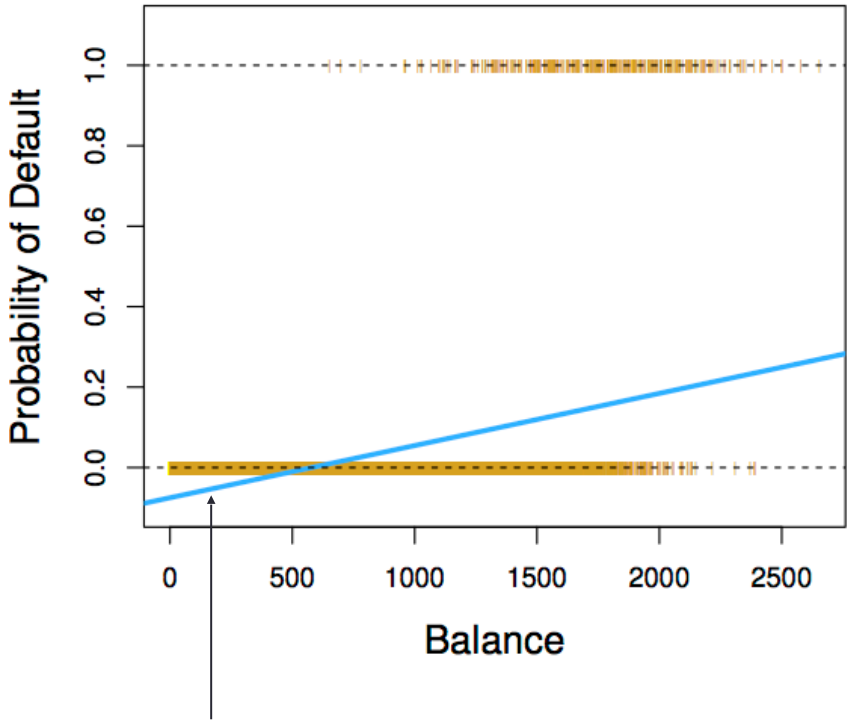


# Why not Linear Regression?

- If we fit a linear regression to the Default data, then for very low balances we predict a negative probability, and for high balances we predict a probability above 1!




When Balance < 500,

Pr(default) is negative!

# Logistic Regression

- Rather than modeling the response *Y* (Yes or No) directly, logistic regression models the **probability** that *Y* belongs to a particular category.
- The probability of default given balance:

$$P(default = Yes \mid balance)$$

- For any given value of balance, a prediction can be made for default.

# The Logistic Model

- In logistic regression we use the logistic function:

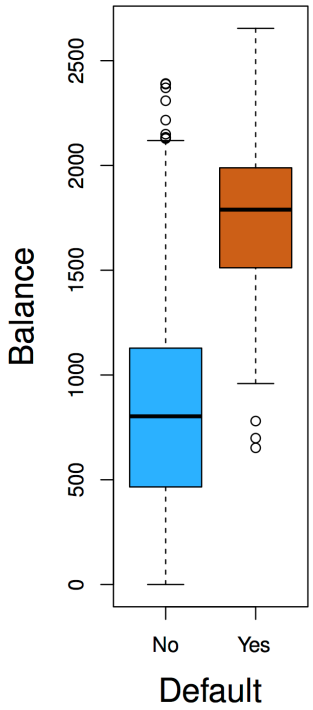
$$p(X) = P(Y = 1) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}$$

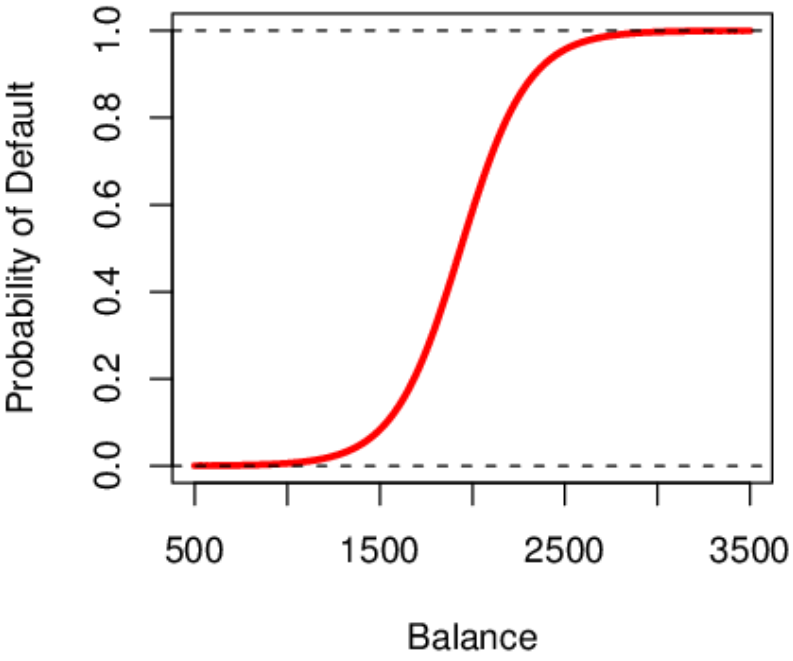
- We come up with $b_0$ and $b_1$ to estimate $\beta_0$ and $\beta_1$.

- How sure are we about our guesses for $\beta_0$ and $\beta_1$?

# Logistic Function on Default Data

- The probability of default is close to, but not less than zero for low balances. And close to but not above 1 for high balances.



# Interpreting $\beta_1$

- Interpreting what $\beta_1$ means is not very easy with logistic regression, simply because we are predicting *P(Y)* and not *Y*.
  - If $\beta_1 = 0$, this means that there is no relationship between *Y* and *X*.
  - If $\beta_1 > 0$, this means that when *X* gets larger so does the probability that *Y* = 1.
  - If $\beta_1 < 0$, this means that when *X* gets larger, the probability that *Y* = 1 gets smaller.
- But how much bigger or smaller depends on where we are on the slope.

# Are the coefficients significant?

- We still want to perform a **hypothesis test** to see whether we can be sure that are $\beta_0$ and $\beta_1$ significantly different from zero.
- We use a Z test instead of a T test, but of course that doesn't change the way we interpret the p-value.
- Here the p-value for balance is very small, and $\beta_1$ is positive, so we are sure that if the balance increase, then the probability of default will increase as well.

```
Call:
glm(formula = default ~ balance, family = "binomial", data = Default)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Making Prediction

- Suppose an individual has an average balance of $1000. What is their probability of default?

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0+\hat{\beta}_1 X}}{1+e^{\hat{\beta}_0+\hat{\beta}_1 X}} = \frac{e^{-10.6513+0.0055\times 1000}}{1+e^{-10.6513+0.0055\times 1000}} = 0.00576$$

- The predicted probability of default for an individual with a balance of $1000 is less than 1%.

- For a balance of $2000, the probability is much higher, and equals to 0.586 (58.6%).

# Qualitative Predictors in Logistic Regression

- We can predict if an individual default by checking if he/she is a student or not. Thus we can use a qualitative variable "Student" coded as (Student = 1, Non-student = 0)
- $b_1$ is positive: This indicates students tend to have higher default probabilities than non-students

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
studentYes   0.40489    0.11502    3.52 0.000431 ***
```

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

# Multiple Logistic Regression

- We can fit multiple logistic just like regular regression

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

# Multiple Logistic Regression- Default Data

- Predict Default using:
  - Balance (quantitative)
  - Income (quantitative)
  - Student (qualitative)

```
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
```

# Predictions

- A student with a credit card balance of $1,500 and an income of $40,000 has an estimated probability of default

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times 1500+0.003\times 40-0.6468\times 1}}{1+e^{-10.869+0.00574\times 1500+0.003\times 40-0.6468\times 1}} = 0.058.$$

# An Apparent Contradiction!

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
studentYes   0.40489    0.11502    3.52 0.000431 ***
```

Positive

```
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
```

Negative

# To whom should credit be offered?

- A student is risker than non students if no information about the credit card balance is available

- However, that student is less risky than a non-student with the same credit card balance!

# Summary

- The logistic regression model is very popular due to its simplicity and ability to make inferential statements about model terms

# LOGISTIC REGRESSION

R session

# The Stock Market Data

- We will use the "The Stock Market dataset" from the book "An Introduction to Statistical Learning, with applications in R", G. James, D. Witten, T. Hastie and R. Tibshirani, Springer, 2013. There is a package in R called ISLR with this dataset included.
- Daily percentage returns for the S&P 500 stock index between 2001 and 2005 *(source: raw values of the S&P 500 were obtained from Yahoo Finance and then converted to percentages and lagged)*.
- The stock market data includes 1250 examples of stock market information, each with 9 features: Year, Lag1, Lag2, Lag3, Lag4, Lag5, Volume, Today and Direction. Direction is the class feature with two possible outcomes: up or down.

# The Stock Market Data

```
require(ISLR)
names(Smarket)
summary(Smarket)

?Smarket

pairs(Smarket,col=Smarket$Direction)

cor(Smarket) # This won't work, why)

cor(Smarket[,-9]) # Note that Volume has some correlation with Year...

boxplot(Smarket$Volume~Smarket$Year)

# Direction is derive from Today
cor(as.numeric(Smarket$Direction),Smarket$Today)

glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket, family=binomial)
summary(glm.fit)

glm.probs <- predict(glm.fit,type="response")
glm.probs

glm.pred <- ifelse(glm.probs>0.5,"Up","Down")
glm.pred

table(glm.pred,Smarket$Direction)
mean(glm.pred==Smarket$Direction)

# Make training and test set
train <- (Smarket$Year < 2005)
glm.fit <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket, family=binomial, subset=train)
glm.fit

glm.probs <- predict(glm.fit,newdata=Smarket[!train,], type="response")
glm.pred <- ifelse(glm.probs >0.5,"Up","Down")
Direction.2005 <- Smarket$Direction[!train]
table(glm.pred,Direction.2005) # Overfitting!
mean(glm.pred==Direction.2005)
```

# The Stock Market Data

```
glm.fit <- glm(Direction~Lag1+Lag2, data=Smarket,family=binomial,
subset=train)
glm.fit

glm.probs <- predict(glm.fit,newdata=Smarket[!
train,],type="response")
glm.pred <- ifelse(glm.probs > 0.5,"Up","Down")
table(glm.pred,Direction.2005)
mean(glm.pred==Direction.2005)

require(caret)
glmFit <- train(Smarket[train,-9], y = Smarket[train,9], method =
"glm", preProcess = c("center", "scale"),
                tuneLength = 10, control=glm.control(maxit=500),
trControl = trainControl(method = "cv"))
glmFit
```

# Exercise 2

- Using the Smarket dataset:
  - Perform 10 fold-cv with logistic regression.

# Bibliography


- Machine Learning with R. Brett Lantz. Packt Publishing. 2013.
- DSO 530: Applied Modern Statistical Learning Techniques. Abbass Al Sharif. http://www.alsharif.info/#!iom530/c21o7
- An Introduction to Statistical Learning. Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. http://www-bcf.usc.edu/~gareth/ISL/index.html
- Applied Predictive Modeling. Max Kuhn and Kjell Johnson. 2013th Edition. Springer. http://appliedpredictivemodeling.com