Machine Learning for Business

Module 9: Decision tree

Day 5, 9.00 – 12.00

Asst. Prof. Dr. Santitham Prom-on

Department of Computer Engineering, Faculty of Engineering
King Mongkut's University of Technology Thonburi

Predictive Modeling as Supervised Segmentation

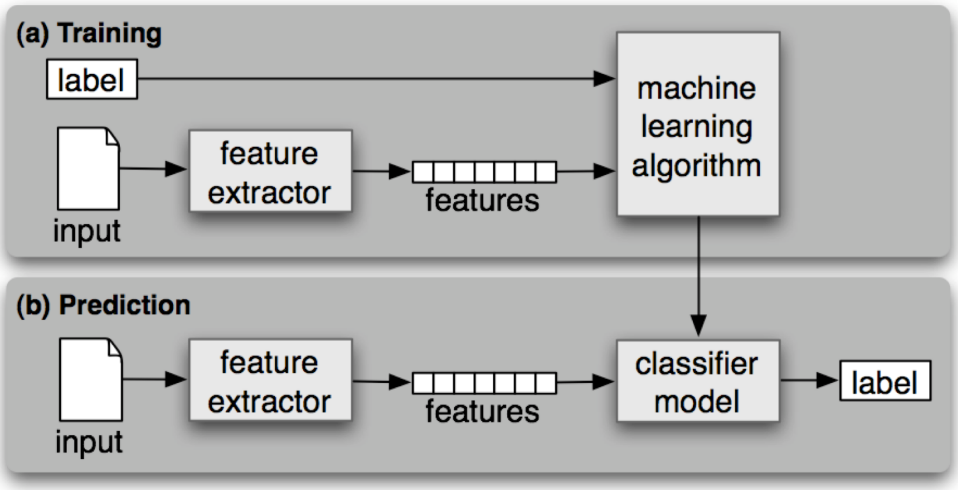
- How can we segment the population into groups that differ from each other with respect to some quantity of interest?

Quantity of interest
=
Things we would like to predict or estimate

Goals of Prediction

- To avoid, such as
  - which customers are likely to leave the company when their contracts expire.
  - Potential customers are likely to write-off (default)
  - Which web pages contain questionable contents
- To target, such as
  - Which consumers are likely to respond to an ads.
  - Which web pages are most appropriate for the search query

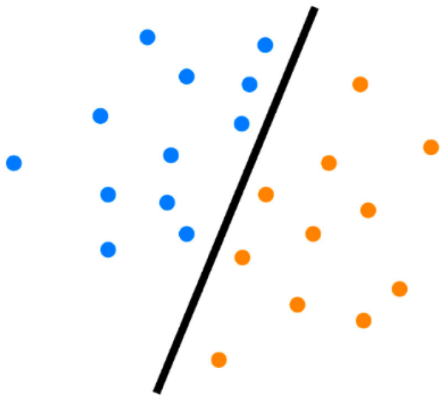
Classification model



BIG DATA EXPERIENCE

Applications

- Churn prediction
- Targeted marketing
- Risk prediction
- Failure prediction
- Sentiment analysis
- Speech recognition
- Image recognition

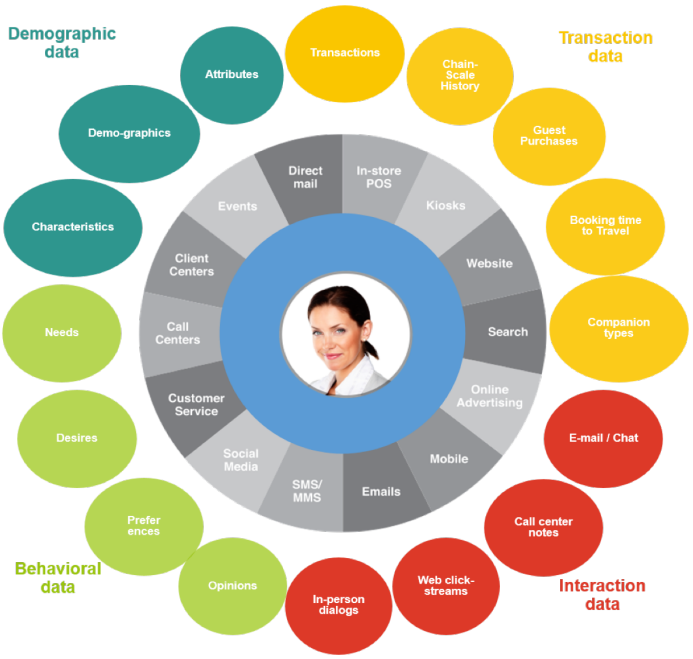


g·able

BIG DATA EXPERIENCE

Training data

- Training data contains
  - Label
  - Attribute data
- Usually they are historical records
- Determining right labels is very crucial

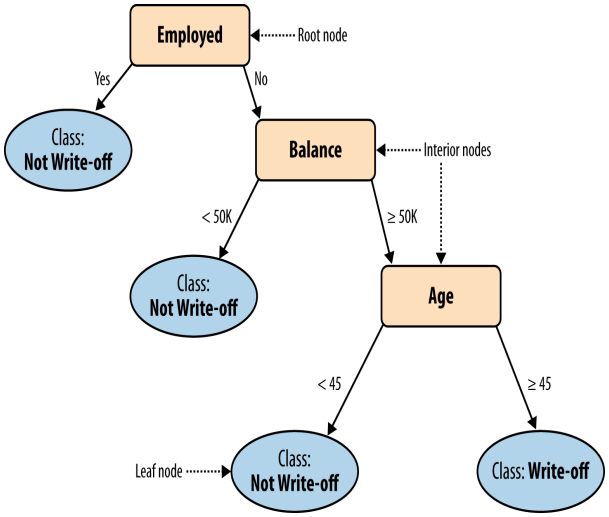


g·able

BIG DATA EXPERIENCE

Basic classifier: decision tree

- A tree consists of nodes: interior and terminal
- Interior node contains a test of an attribute
- Terminal node is a segment



g·able

BIG DATA EXPERIENCE

Which attribute should be used to segment?

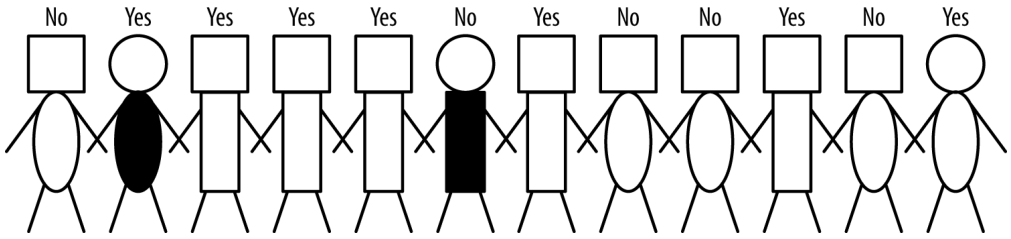
- **Fundamental concept**: how can we judge whether a variable contains important information about the target variable? How much?
- **Aims**: automatic selection, ranking
- We will start with considering the selection of the single most informative attribute
- In our example, what variable gives us the most information: Being professional? Age? Place of resident? Income?

g·able

BIG DATA EXPERIENCE

g·able

Example: Write-off



- Attributes
  - head-shape: square, circular
  - body-shape: rectangular, oval
  - body-color: black, white
- Target variable
  - write-off: Yes, No

Which attribute should be best to segment these people into groups, in a way to distinguish write-off from non-write-off?

BIG DATA EXPERIENCE

g·able

Purity

- Technically, we would like the group to be as pure as possible.
- Pure means homogeneous with respect to the target variable
- If some member in the group has a different target then the group is impure
- Comparing
  - G1 = {Y, Y, Y, Y}
  - G2 = {Y, N, N, Y}

In real data, however, we rarely find pure segments.

BIG DATA EXPERIENCE

g·able

Complications

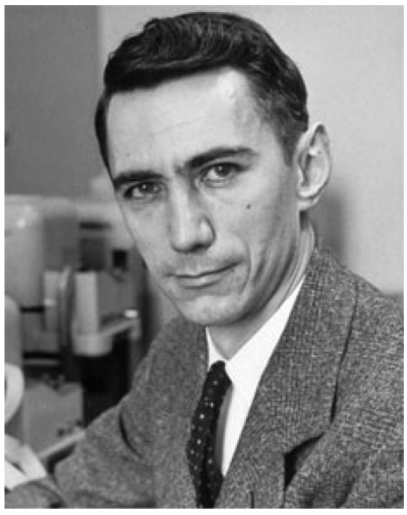
- Attributes rarely split a group perfectly. Even if one may be pure, the others may not.
- Not all attributes are binary. How do we compare them?
- Some attributes are numeric values. How should we create supervised segmentation using numeric attributes?

BIG DATA EXPERIENCE

g·able

Information Gain and Entropy

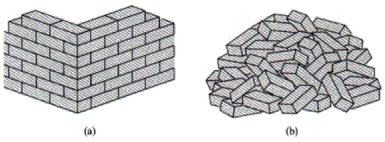
- The most common splitting criterion is called information gain, and it is based on a purity measure called entropy
- Both concepts were invented by the pioneer in information theory, Claude Shannon (1948).



BIG DATA EXPERIENCE

Entropy

- Entropy is a measure of disorder that can be applied to a set, such as one of our individual segments
- Disorder corresponds to how mixed (impure) the segment is with respect to the target
- For example, a mixed up segment with lots of write-offs and lots off non-write-offs would have high entropy



g·able

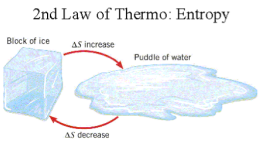
BIG DATA EXPERIENCE

Entropy Formula

- More technically, entropy is defined as

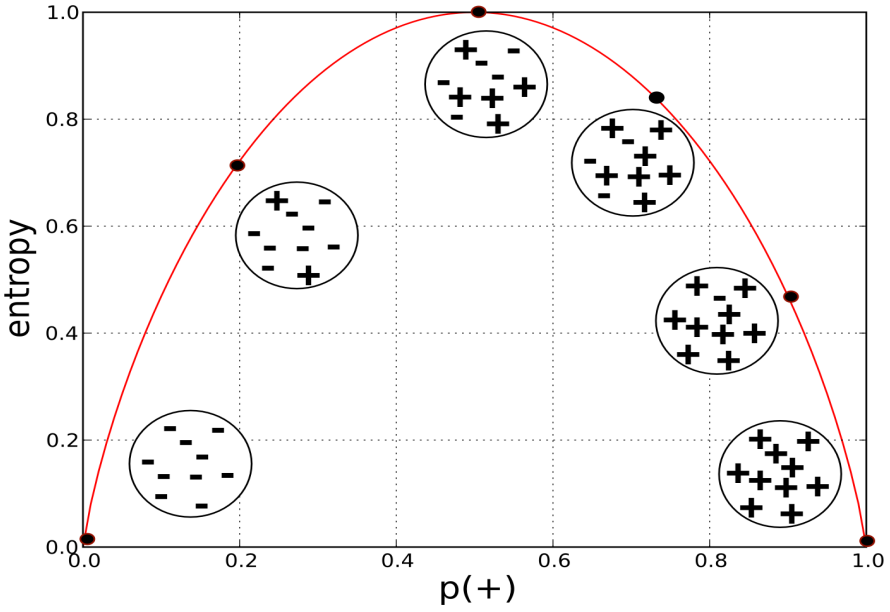
$$entropy = -p_1\log(p_1) - p_2\log(p_2) - \ldots$$

- This is based on Gibbs entropy in thermodynamics
- Each p_i is the probability (the relative percentage) of property i (e.g. write-offs/non-write-offs) of the target.



g·able

Entropy of a two-class set



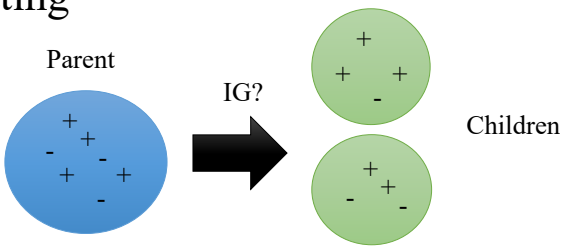
Example

- Consider a set S of 10 people with 7 non-write-off and 3 write-off classes. So:

$$p(\text{non-write-off}) = 7/10 = 0.7$$
$$p(\text{write-off}) = 3/10 = 0.3$$
$$\text{entropy} = -[0.7 \times \log_2(0.7) + 0.3 \times \log_2(0.3)]$$
$$\approx -[0.7 \times -0.51 + 0.3 \times -1.74]$$
$$\approx 0.88$$

Information Gain

- Entropy is only part of the story. We would like to measure how informative an attribute is with respect to target: how much gain in information it gives us about the target?
- Information gain (IG) measure how much attribute improves (decreases) entropy over the whole segmentation it creates.
- In our context, IG measures the change in entropy due to further splitting



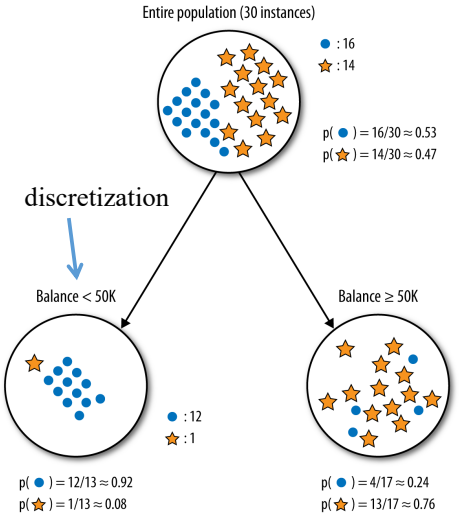
Information Gain Formula

IG(*parent*, *children*)

$$= \text{entropy}(parent) - [p(c_1)\times\text{entropy}(c_1) + p(c_2)\times\text{entropy}(c_2) + \ldots]$$

- The entropy for each child (c_i) is weighted by the proportion of instances belonging to that child, $p(c_i)$.

Example 1



$$\begin{aligned} entropy(parent) &= -[p(\bullet)\times\log_2 p(\bullet)+p(\star)\times\log_2 p(\star)] \\ &\approx -[0.53\times -0.9+0.47\times -1.1] \\ &\approx 0.99 \quad (\text{very impure}) \end{aligned}$$

Entropy of the left child is

$$\begin{aligned} entropy(Balance<50K) &= -[p(\bullet)\times\log_2 p(\bullet)+p(\star)\times\log_2 p(\star)] \\ &\approx -[0.92\times(-0.12)+0.08\times(-3.7)] \\ &\approx 0.39 \end{aligned}$$

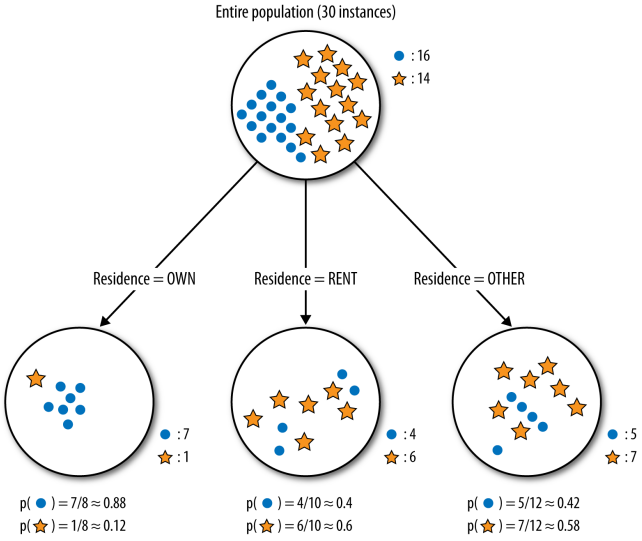
Entropy of the right child is

$$\begin{aligned} entropy(Balance\geq 50K) &= -[p(\bullet)\times\log_2 p(\bullet)+p(\star)\times\log_2 p(\star)] \\ &\approx -[0.24\times(-2.1)+0.76\times(-0.39)] \\ &\approx 0.79 \end{aligned}$$

Information gain is

$$\begin{aligned} IG &= entropy(parent)-[p(\text{Balance}<50\text{K})\times entropy(\text{Balance}<50\text{K}) \\ &\quad +p(\text{Balance}\geq 50\text{K})\times entropy(\text{Balance}\geq 50\text{K})] \\ &\approx 0.99-[0.43\times 0.39+0.57\times 0.79] \\ &\approx 0.37 \end{aligned}$$

Example 2



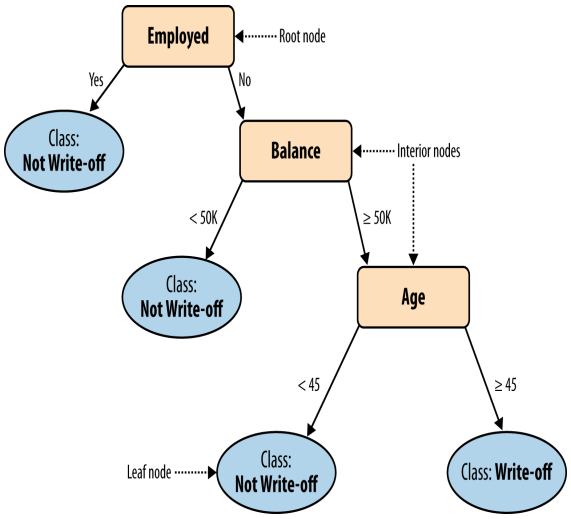
Calculations are omitted

entropy(*parent*) ≈ 0.99
entropy(Residence=OWN) ≈ 0.54
entropy(Residence=RENT) ≈ 0.97
entropy(Residence=OTHER) ≈ 0.98
IG ≈ 0.13

Residence variable is less informative than Balance.

Decision Tree

- A tree consists of nodes: interior and terminal
- Interior node contains a test of an attribute
- Terminal node is a segment

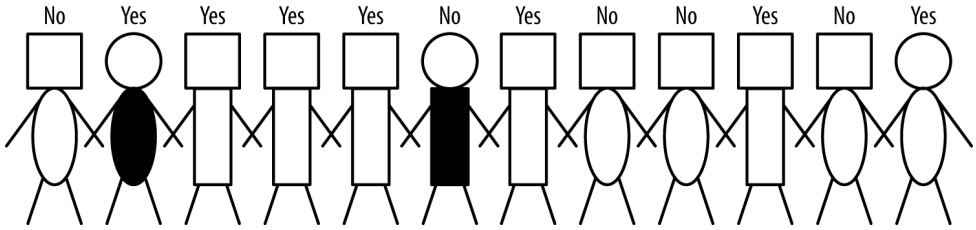


<Claudio, 115000, 40, No>?

Tree Induction

- How do we create a decision tree from data?
- Tree induction takes a divide-and-conquer approach,
  1. starting with the whole dataset
  2. applying variable selection to create subgroups
  3. Recursively repeating step 2 for each subgroup
- We will illustrate this using the write-off example

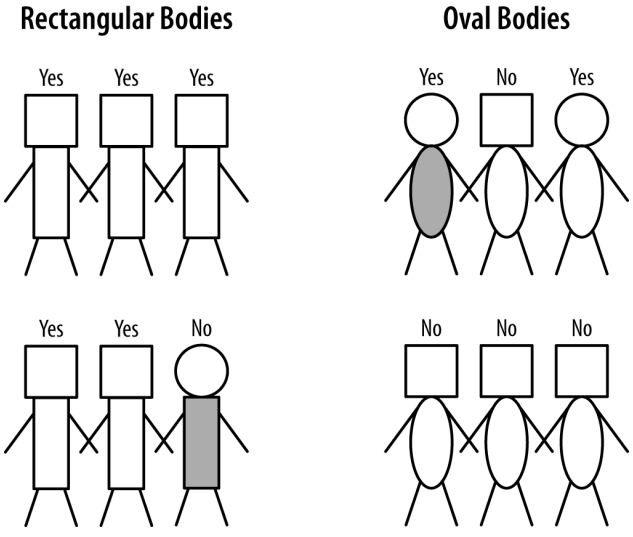
Example



Attributes
- head-shape: square, circular
- body-shape: rectangular, oval
- body-color: gray, white

Target variable
- write-off: Yes, No

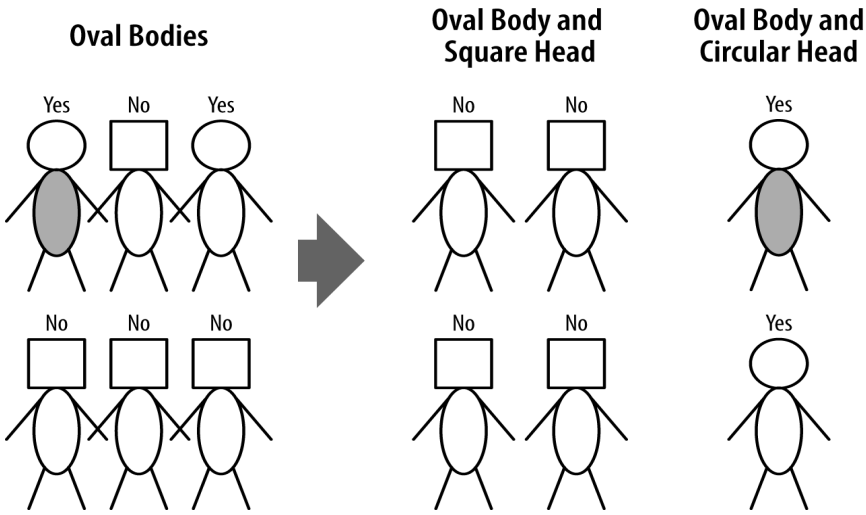
First Partitioning: body-shape



body-shape has the highest IG, so it is selected as the first attribute

BIG DATA EXPERIENCE

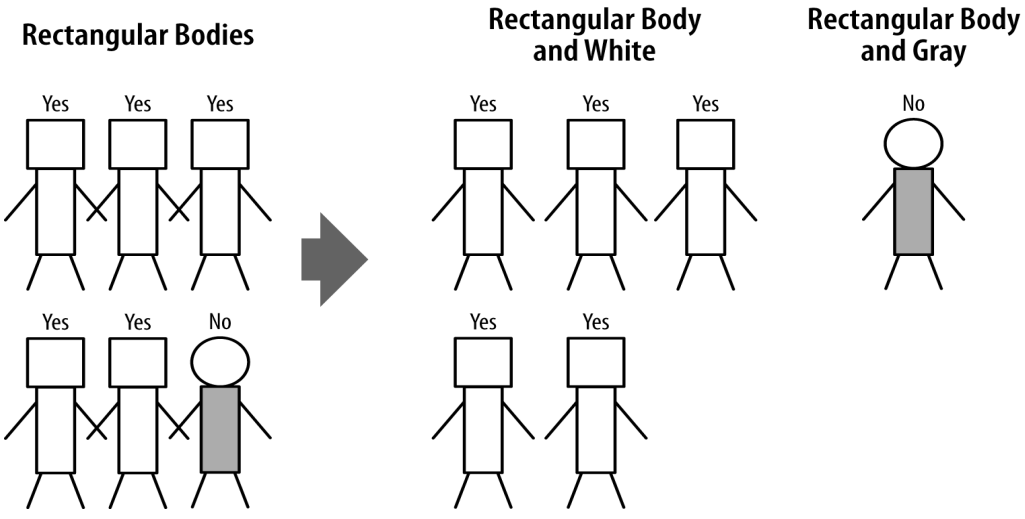
2nd partitioning: oval-body, head-type



g·able

BIG DATA EXPERIENCE

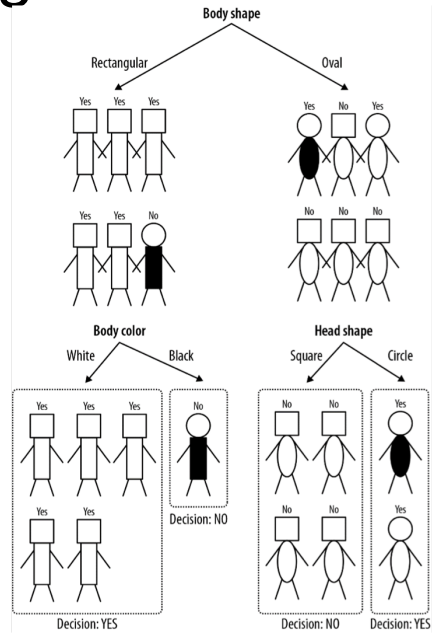
3rd partitioning: rectangular-body, body-color



g·able

BIG DATA EXPERIENCE

Resulting Decision Tree



g·able

BIG DATA EXPERIENCE

R: Decision Tree

g·able

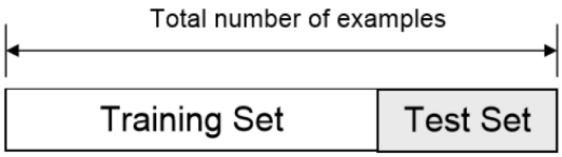
R: load data

```
bankData <- read.csv("bank-data.csv", sep = ";")
head(bankData)
```

```
  age          job marital education default balance housing loan contact day
1  58   management married  tertiary      no    2143     yes   no unknown   5
2  44   technician  single secondary      no      29     yes   no unknown   5
3  33 entrepreneur married secondary      no       2     yes  yes unknown   5
4  47  blue-collar married   unknown      no    1506     yes   no unknown   5
5  33      unknown  single   unknown      no       1      no   no unknown   5
6  35   management married  tertiary      no     231     yes   no unknown   5
  month duration campaign pdays previous poutcome  y
1   may      261        1    -1        0  unknown no
2   may      151        1    -1        0  unknown no
3   may       76        1    -1        0  unknown no
4   may       92        1    -1        0  unknown no
5   may      198        1    -1        0  unknown no
6   may      139        1    -1        0  unknown no
```

Training and Testing

- Hold-out method
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier



- We will sample 60% of the data to train and use the remaining 40% to test

BIG DATA EXPERIENCE

R: create training and testing set

```
library(dplyr)
bankData.train <- sample_frac(bankData, 0.6)
training.ind <- as.integer(rownames(bankData.train))
bankData.test <- bankData[-training.ind,]
```

Data	
bankData	45211 obs. of 17 variables
bankData.test	18084 obs. of 17 variables
bankData.train	27127 obs. of 17 variables

g·able

BIG DATA EXPERIENCE

R: check the distribution Training data

```
summary(bankData.train)
# Pr(yes) = 3187/(3187+23940) = 0.117

summary(bankData.test)
# Pr(yes) = 2102/(2102+15982) = 0.116
```

g·able

BIG DATA EXPERIENCE

Splitting criteria

Regression: residual sum of squares

$$RSS = \sum_{left} (y_i - y_L^*)^2 + \sum_{right} (y_i - y_R^*)^2$$

where y_L^* = mean y-value for left node

y_R^* = mean y-value for right node

Classification: Gini criterion (Similar to Information Gain)

$$Gini = N_L \sum_{k=1,\ldots,K} p_{kL} (1- p_{kL}) + N_R \sum_{k=1,\ldots,K} p_{kR} (1- p_{kR})$$

where p_{kL} = proportion of class k in left node

p_{kR} = proportion of class k in right node

g·able

BIG DATA EXPERIENCE

Classification and Regression Trees

- Grow a binary tree
- At each node, "split" the data into two child nodes
- Splits are chosen using a splitting criterion (e.g. information gain)
- Bottom nodes are terminal nodes
- For regression, the predicted value at a node is the average response variable for all observation in the node
- For classification, the predicted class is the most common class

g·able

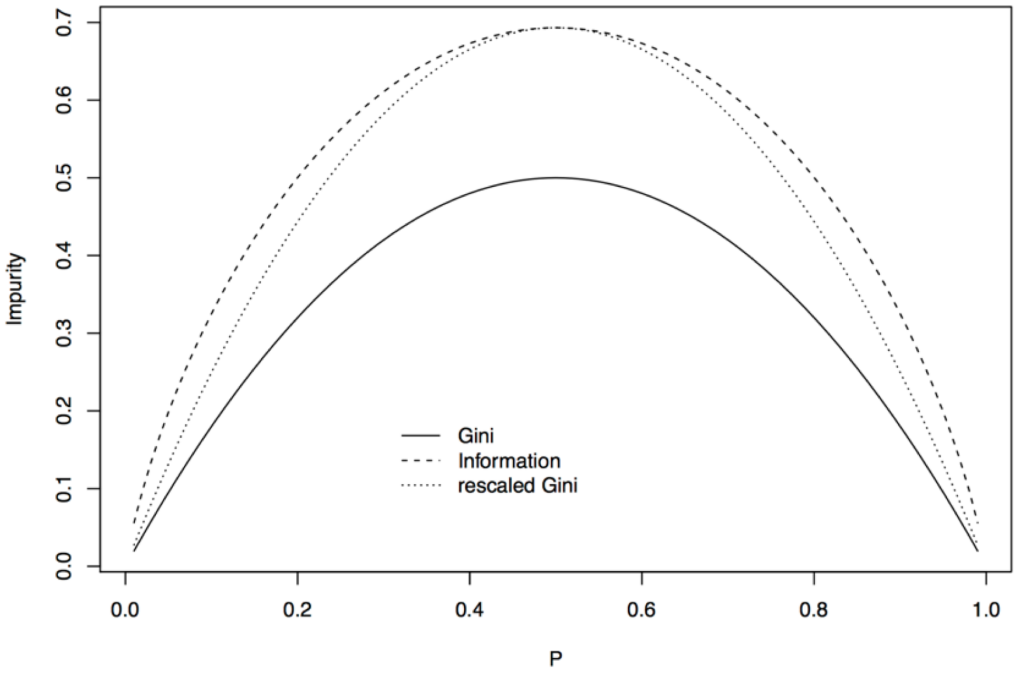
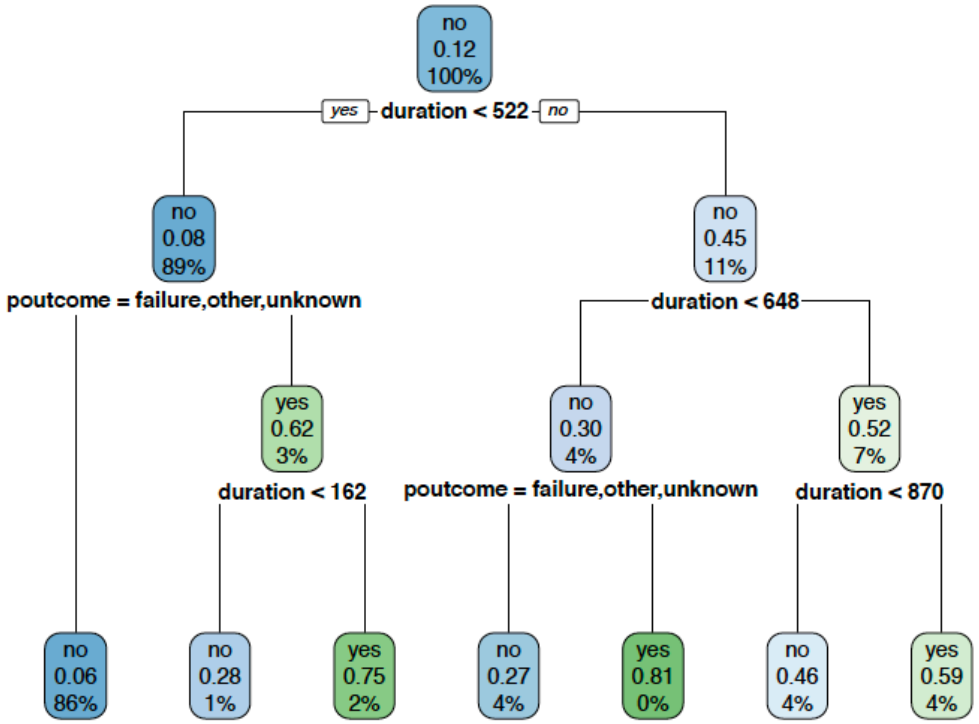


Figure 2: Comparison of Gini and Information impurity for two groups.

R: train model

```
library(rpart)
library(rpart.plot)
tree <- rpart(y ~ ., data = bankData.train)
pdf("bankDataTree.pdf")
rpart.plot(tree)
dev.off()
```

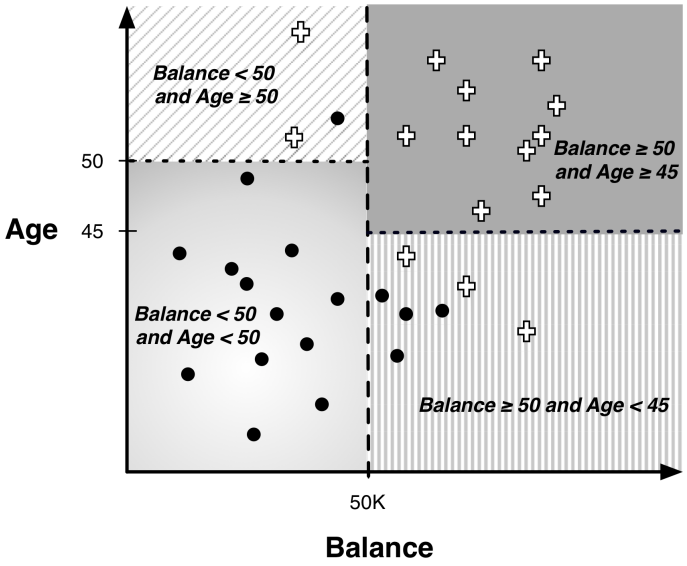


R: variable importance

```
tree$variable.importance
```

```
    duration     poutcome      contact        pdays
870.98080381 507.11560047   1.88811161   1.17674966
     balance     campaign          day        month
  1.11138701   0.77720818   0.70318835   0.42561400
     previous      default          age
  0.32630824   0.31468527   0.06526165
```

Decision boundary



R: Prediction result Probability

```
head(predict(tree,bankData.test))
```

```
           no        yes
2   0.9410507 0.05894935
6   0.9410507 0.05894935
8   0.9410507 0.05894935
15  0.9410507 0.05894935
17  0.9410507 0.05894935
18  0.9410507 0.05894935
```

R: Prediction result Class

```
res <- predict(tree,bankData.test,type = "class")
head(res)
```

```
 2  6  8 15 17 18
no no no no no no
Levels: no yes
```

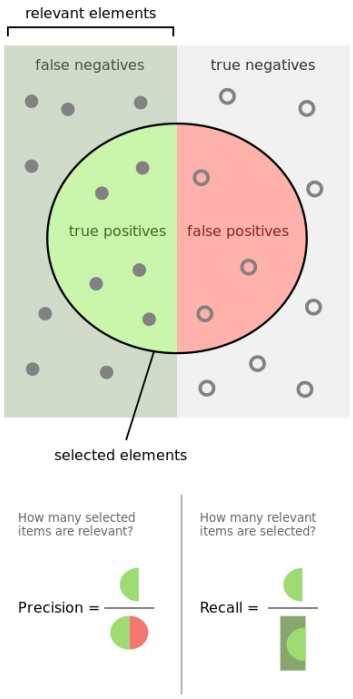
Accuracy, Precision and Recall

	Actual Positive (p)	Actual Negative (n)
The model says "Yes" = positive (y)	True positives	False positives
The model says "No" = not positive (n)	False negatives	True negatives

- Accuracy = (TP + TN)/(TP + FP + TN + FN)
- Recall (Completeness) = true positive rate = TP/(TP + FN)
- Precision (Exactness) = the accuracy over the cases predicted to be positive, TP/(TP + FP)
- F-measure = the harmonic mean of precision and recall

  = the balance between recall and precision

  $= 2 . \frac{precision * recall}{precision + recall}$



Confusion matrix

```
library(caret)
confusionMatrix(res,
bankData.test$y,
positive="yes",
mode="prec_recall")
```

```
          Reference
Prediction    no   yes
       no  15621  1450
       yes   361   652

               Accuracy : 0.8999
                 95% CI : (0.8954, 0.9042)
    No Information Rate : 0.8838
    P-Value [Acc > NIR] : 2.979e-12

                  Kappa : 0.3711
 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.64363
                 Recall : 0.31018
                     F1 : 0.41862
```

Pruning

- If the tree is too big, the lower "branches" are modeling noise in the data ("overfitting").
- The usual paradigm is to grow the trees large and "**prune**" back unnecessary splits.
- Methods for **pruning** trees have been developed. Most use some form of cross validation. Tuning may be necessary.
- The complexity parameter, *cp*, controls whether the tree can be complex (low *cp*) or simple (high *cp*)

Adjusting complexity parameter

```
tree <- rpart(y ~ .,
   data = bankData.train,
   control = rpart.control(cp = 0.001))
res <- predict(tree,bankData.test,
                type = "class")
confusionMatrix(res, bankData.test$y,
    positive="yes", mode="prec_recall")
```

```
          Reference
Prediction    no   yes
       no  15374  1122
       yes   608   980

               Accuracy : 0.9043
                 95% CI : (0.9, 0.9086)
    No Information Rate : 0.8838
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.479
 Mcnemar's Test P-Value : < 2.2e-16

              Precision : 0.61713
                 Recall : 0.46622
                     F1 : 0.53117
```

R: Manually adjusting threshold Boost recall

```
tree <- rpart(y ~ ., data = bankData.train)
res <- predict(tree,bankData.test)
res1 <- factor(
   ifelse(res[,"yes"] > 0.2, "yes","no"),
   levels = c("no","yes"))
confusionMatrix(res1, bankData.test$y,
    positive="yes", mode="prec_recall")
```

```
          Reference
Prediction    no   yes
       no  14441   783
       yes  1501  1359

               Accuracy : 0.8737
                 95% CI : (0.8688, 0.8785)
    No Information Rate : 0.8816
    P-Value [Acc > NIR] : 0.9994

                  Kappa : 0.4718
 Mcnemar's Test P-Value : <2e-16

              Precision : 0.47517
                 Recall : 0.63445
                     F1 : 0.54338
```

BIG DATA EXPERIENCE

g·able

Classification Strategies

- Binary Classifier: 2 classes
  - e.g. churn, not churn
- Multiclass Classifier: > 2 classes
  - e.g. high, medium, low
- One-vs-The-Rest
  - e.g. to build positive, neutral, negative sentiment prediction. We need (1) positive-vs-not-positive, (2) negative-vs-not-negative.
- One-vs-One
  - Each class is compared to another class

BIG DATA EXPERIENCE

g·able

Classification and Regression Trees Disadvantages

- *Accuracy* - current methods, such as support vector machines and ensemble classifiers often have 30% lower error rates than CART.
- *Instability* – if we change the data a little, the tree picture can change a lot. So the interpretation is not as straightforward as it appears.

- Today, we can do better!

Random Forests

BIG DATA
EXPERIENCE

Thank you

Question?

g·able