Symbols, Patterns and Signals: Classification II

Dr. Iván Palomares Carrascosa, MVB 3.16

Email: i.palomares@bristol.ac.uk

In this lecture…

- **Introducing decision trees**
- **Splitting criteria for constructing decision trees → Entropy**
- **Tree pruning to overcome the overfitting problem**
- **Some real-world problems solved with Classification**

Classification II

In this lecture, we look at a classification technique called decision trees

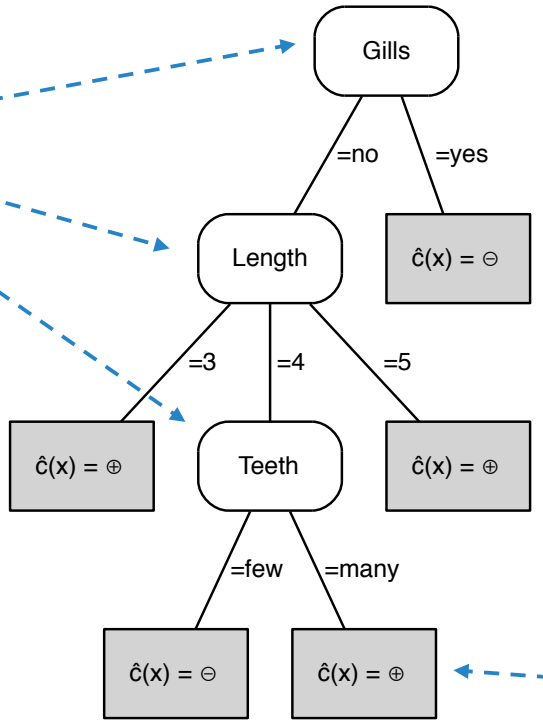
- For probabilistic Naïve-Bayes, we did not build a classification model → ML or MAP decision rules
- Here, we deterministically build a **model** that separates classes on **training set**
- then evaluate and adapt the model on a separate **test set** to avoid overfitting and improve generalization

Decision trees

A decision tree partitions the instance space into regions of (nearly) homogeneous class observations per region

- internal nodes are labelled with features (aka splits)
- leaves are labelled with classes

features



p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few
p4: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p5: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

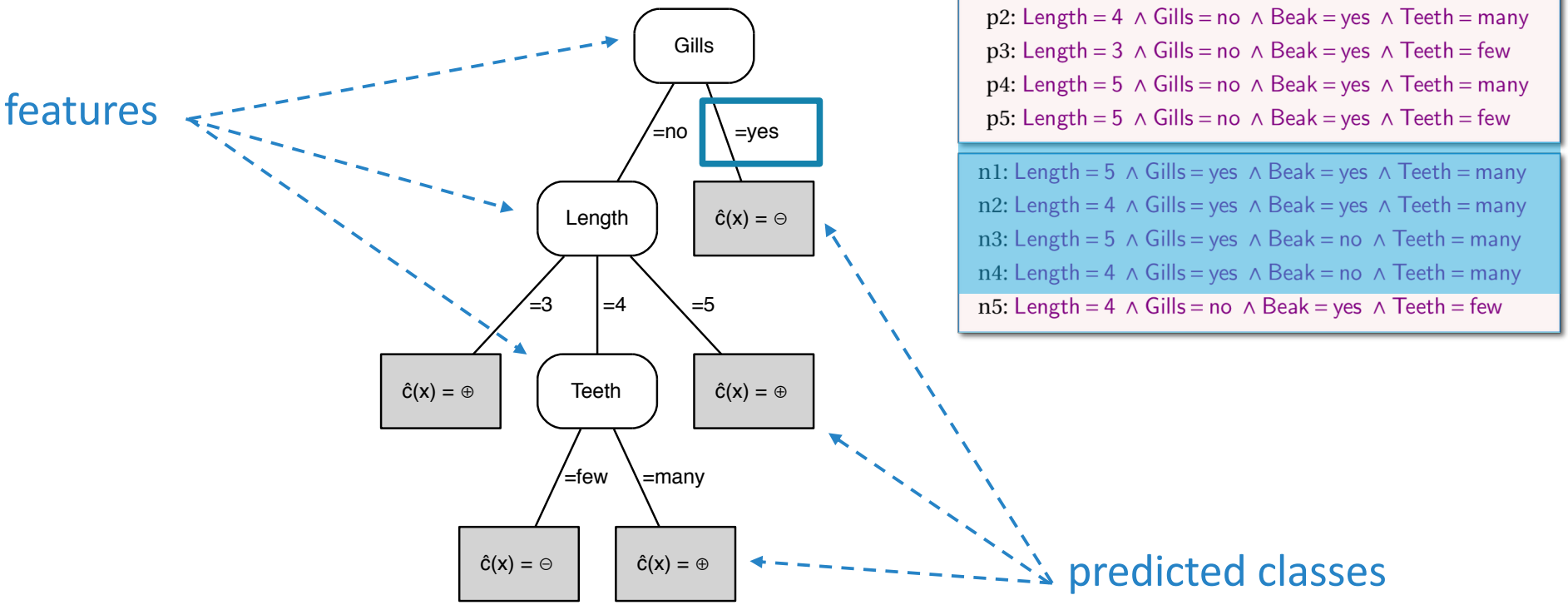
n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
n2: Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
n3: Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
n4: Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
n5: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

predicted classes

Decision trees

A decision tree partitions the instance space into regions of (nearly) homogeneous class observations per region

- internal nodes are labelled with features (aka splits)
- leaves are labelled with classes



p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few
p4: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p5: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

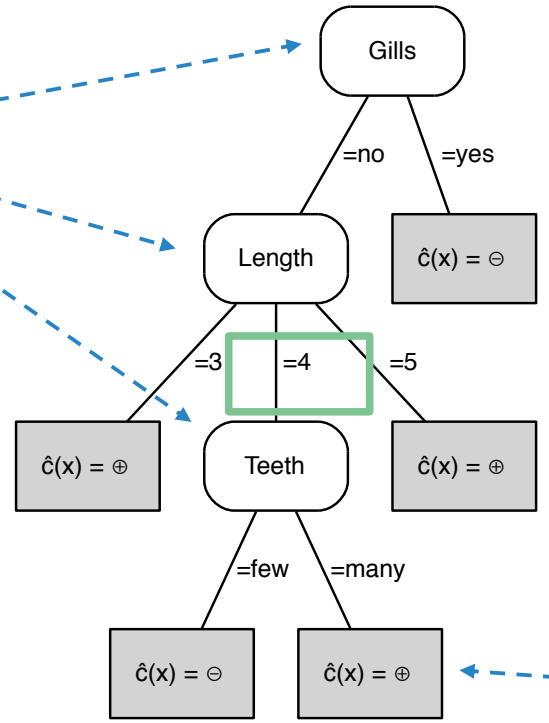
n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
n2: Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
n3: Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
n4: Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
n5: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

Decision trees

A decision tree partitions the instance space into regions of (nearly) homogeneous class observations per region

- internal nodes are labelled with features (aka splits)
- leaves are labelled with classes

features



p1: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p2: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p3: Length = 3 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few
p4: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = many
p5: Length = 5 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

n1: Length = 5 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
n2: Length = 4 ∧ Gills = yes ∧ Beak = yes ∧ Teeth = many
n3: Length = 5 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
n4: Length = 4 ∧ Gills = yes ∧ Beak = no ∧ Teeth = many
n5: Length = 4 ∧ Gills = no ∧ Beak = yes ∧ Teeth = few

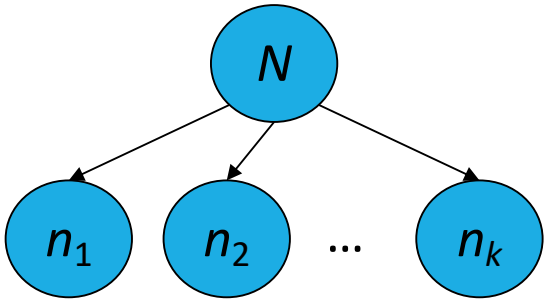
predicted classes

Growing trees – Basic Algorithm

1. **Assign all training instances (*D*) to the root of the tree. Set current node to root node.**
2. **For each attribute**
    i. Partition observations in the current node by attribute value.
    ii. Compute the information gain ratio from the partitioning.
3. **FIND BEST SPLIT: Identify feature leading to greatest information gain**
    i. If the best information gain is 0, tag the current node as a leaf and return.
4. Keep the partition (from step 2) associated to the "best split attribute"
5. Denote each partition as a child node of the current node.
6. For each child node:
    i. If the child node is "pure" (has instances from only one class) tag it as a leaf and return.
    ii. Else, set the child node as the current node and **make recursion to step 2.**

Finding the best split

Suppose a node with **N** instances is split into **k** nodes, with $\mathbf{n_i}$ instances each
($1 \leq i \leq k$, $\Sigma i\ n_i = N$)



Information gain of a split can be calculated as the decrease in impurity going from parent to children:

$$Imp(Parent) - \sum_{i=1}^{k} \frac{n_i}{N} Imp(Child_i)$$

A node has less **impurity** when the instances contained in it are "closer" from all belonging to the same class

If we have **c** different classes and the proportion of class **j** in a (sub)set of instances associated to a node is $\mathbf{p_j}$, then its impurity can be measured by using entropy:

$$\sum_{j=1}^{c} -p_j \log_2 p_j$$

What is Entropy?

$-log_2\ p_j$ measures the information content (in bits) of an event occurring with probability p_j

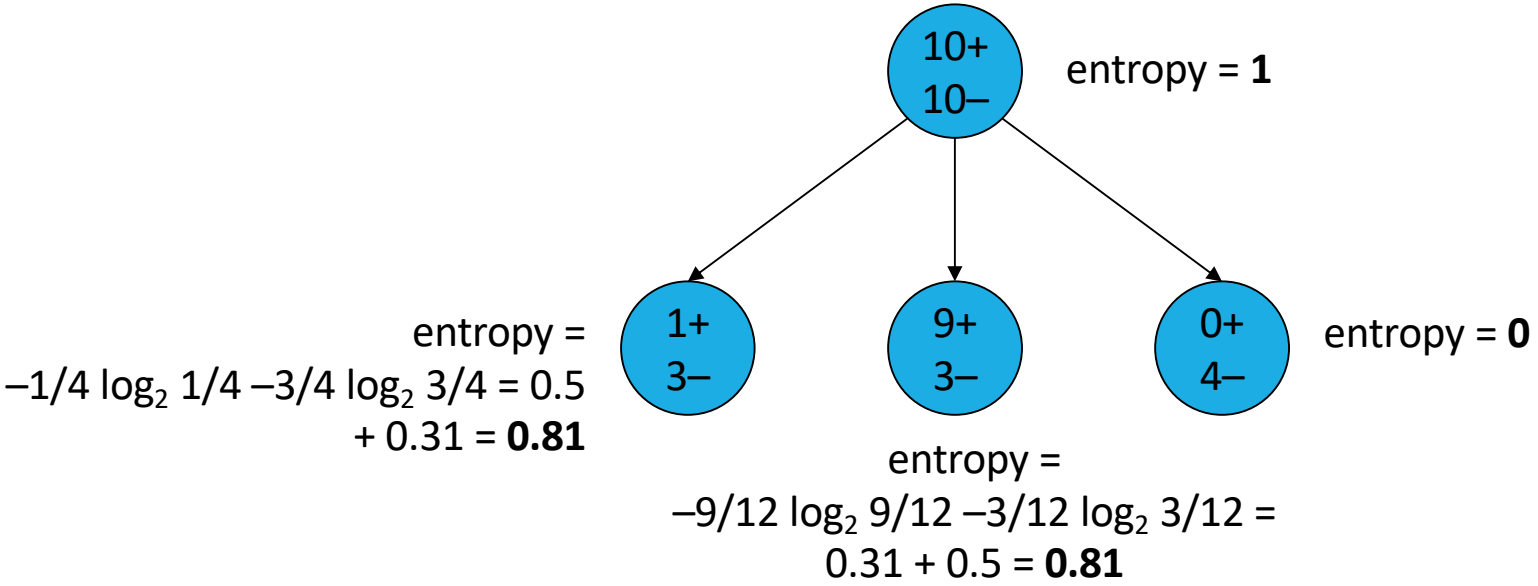
$\Sigma_j\ -p_j\ log_2\ p_j$ is the average information content per event

- or the 'randomness' of the distribution
- entropy is maximal for uniform distributions, (are classes equally distributed)
- entropy is minimum (zero) if all but one $\mathbf{p_j}$ are zero (all instances in same class)
- e.g., for two mutually exclusive events (classes) entropy is:

$$-p_1\ log_2\ p_1 \text{ — } (1-p_1)\ log_2\ (1-p_1)$$

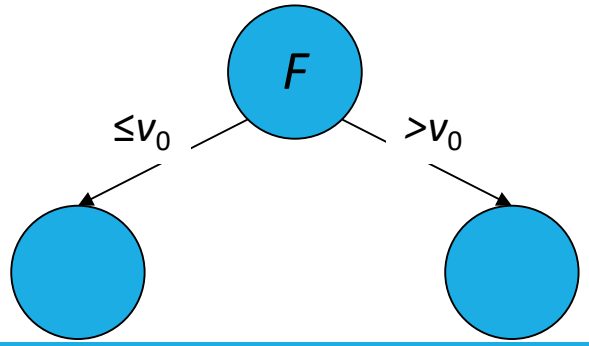
Here, we use entropy to measure how much additional information a particular split provides us about the class

An example

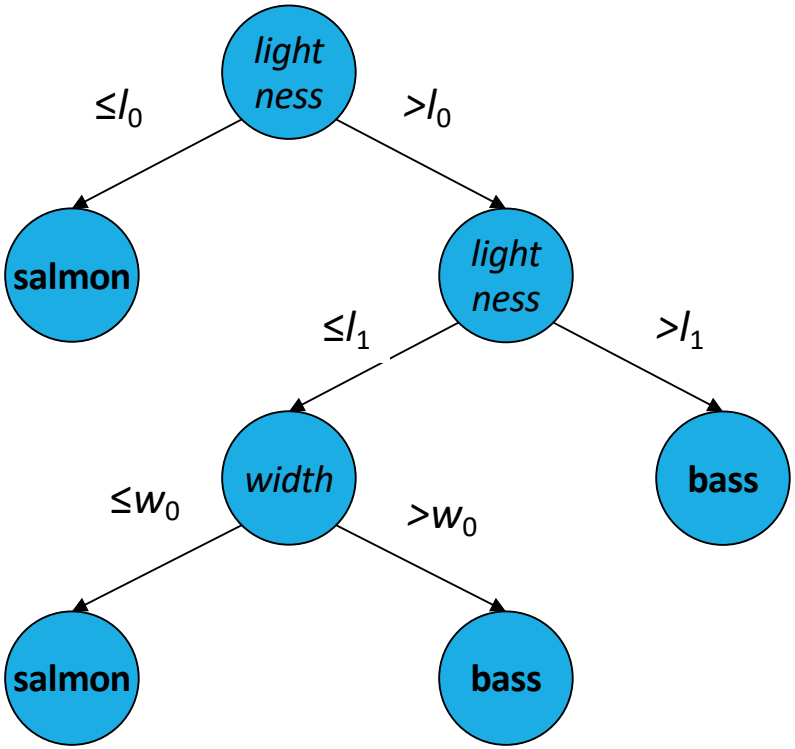


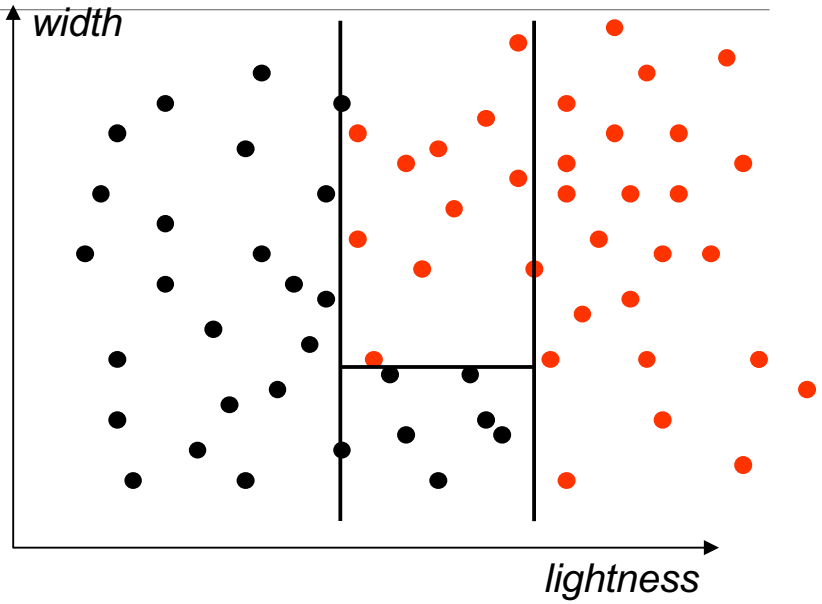
Information gain = 1 – (4/20)0.81 – (12/20)0.81 – (4/20)0 = 0.35

For numerical features we create binary splits by setting a threshold that maximizes information gain



Trees, branches and leaves





The instance space is divided into axis-parallel rectangles

- or hyper-rectangles in **d**-dimensional space

Other impurity functions

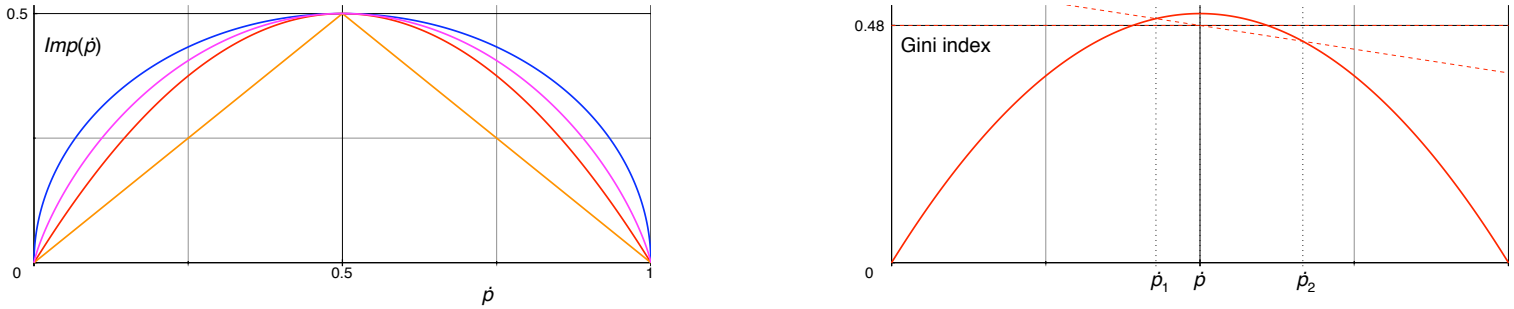


Figure 5.2. **(left)** Impurity functions plotted against the empirical probability of the positive class. From the bottom: the relative size of the minority class, $\min(\dot{p}, 1-\dot{p})$; the Gini index, $2\dot{p}(1-\dot{p})$; entropy, $-\dot{p}\log_2\dot{p} - (1-\dot{p})\log_2(1-\dot{p})$ (divided by 2 so that it reaches its maximum in the same point as the others); and the (rescaled) square root of the Gini index, $\sqrt{\dot{p}(1-\dot{p})}$ – notice that this last function describes a semi-circle. **(right)** Geometric construction to determine the impurity of a split (Teeth = [many, few] from Example 5.1): $\dot{p}$ is the empirical probability of the parent, and $\dot{p}_1$ and $\dot{p}_2$ are the empirical probabilities of the children.

Pruning trees

The *Growing Tree* algorithm will achieve very high accuracy on the training set

- 100% accuracy if features can be re-used
- but many of the leaves will contain only a few instances —> overfitting!

In order to improve generalization, we prune the tree back from leaves to root, using a separate test set

i. Let **N** be the parent of a current leaf

ii. Let α be the test set accuracy of the current tree *(% of test unlabeled obs. correctly classified by the tree*)

iii. Let β be the test set accuracy if **N** and its children are replaced by a leaf, labelled with the most frequent class among **N**'s training instances ('majority class')

iv. If $\alpha < \beta$ then prune (i.e., replace the subtree with the leaf)

v. If all parents of leaves have been tried then stop else go to 1.

Trees and probabilities

Consider a descending path of a decision tree, from root to a leaf: let *x* be the conjunction of conditions on that branch, and let p_j be the relative frequency of class *j* instances in the leaf.

Then $\mathbf{p_j}$ can be interpreted as an estimate of $\mathbf{P(\omega_j|x)}$, i.e., the posterior probability of class ω_j given **x**

Therefore, assigning the majority class, $\mathbf{argmax_\omega\ P(\omega_j|x)}$, to the leaf node, corresponds to the MAP decision rule

- **Finally, it is worth noticing that decision trees can represent almost any posterior probability distribution (they have low bias) but tend to be sensitive to small variations in the training data (they have high variance)**

The following tutorial is a nice supplement to what we learnt in this lecture:
http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/

Classification: Application 1

Direct Marketing

- Goal: Reduce cost of mailing by targeting a subset of consumers likely to buy a new cell-phone product.
- Approach:
  - Use the data for a similar product introduced before.
  - We know which customers decided to buy and which decided otherwise. This {buy, don't buy} decision forms the class attribute.
    - Meaning: ground truth is known.
  - BUT: what attributes to use to determine {buy, don't buy}?
  - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
    - Type of business, where they stay, how much they earn, etc.
  - Use this information as input attributes to learn a classifier model.

Classification: Application 2

Fraud Detection

- Goal: Predict fraudulent cases in credit card transactions.
- Approach:
  - Use credit card transactions and the information on its account-holder as attributes.
    - When does a customer buy, what does s/he buy, how often he pays on time, etc
  - Label past transactions as *fraud* or *fair* transactions. This forms the class attribute.
  - Learn a model for the class of the transactions.
  - Use this model to detect fraud by observing credit card transactions on an account.

Classification: Application 3

Sky Survey Cataloging

- Goal: To predict class (*star* or *galaxy*) of sky objects, especially visually faint ones, based on the telescopic survey images (from *Palomar* Observatory, Spain).
  - 3000 images with 23,040 x 23,040 pixels per image.
- Approach:
  - Segment the image.
  - Measure image attributes (features) - 40 of them per object.
  - Build classifier that predicts the class based on these features.
  - Success Story: could find 16 new high red-shift quasars, some of the farthest objects that are difficult to find!