Machine Learning and Data Mining

Classification - I

Claudio Sartori

DISI
Department of Computer Science and Engineering – University of Bologna, Italy
claudio.sartori@unibo.it

Unsupervised Classification

- the unsupervised mining techniques which can be in some way related to classification are usually known in literature with names different from classification
- for this reason in this course with the term classification we will always mean supervised classification

Supervised Classification 1/2

In the following, simply classification

Consider the "soybean" example shown in the introduction (link to the dataset)

- The data set $\mathcal{X}$ contains N individuals described by D attribute values each
- We have also a $\mathcal{Y}$ vector which, for each individual x contains the class value $y(x)$
- The class allows a finite set of different values (e.g. the diseases), say C
- The class values are provided by experts: the supervisors

Supervised Classification 2/2

- We want to learn how to guess the value of the $y(x)$ for individuals which have not been examined by the experts
- We want to learn a classification model

Classification model

- An algorithm which, given an individual for which the class is not known, computes the class
- The algorithm is parametrized in order to optimize the results for the specific problem at hand
- Developing a classification model requires
  - choose the learning algorithm
  - let the algorithm learn its parametrization
  - assess the quality of the classification model
- The classification model is used by a run–time classification algorithm with the developed parametrization

Classification model or, shortly, classifier

A bit of formality

- a decision function which, given a data element x whose class label $y(x)$ is unknown, makes a prediction as

$$\mathcal{M}(x, \theta) = y(x)_{pred}$$

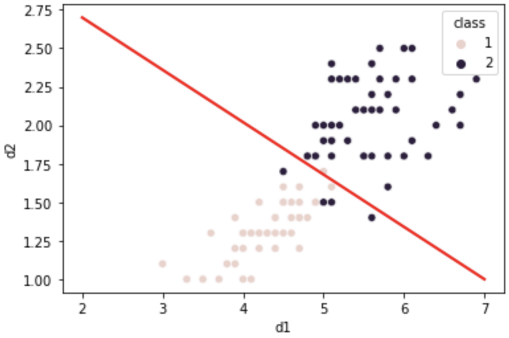
  where θ is a set of values of the parameters of the decision function
  - the prediction can be true or false
- the learning process for a given classifier $\mathcal{M}(.,.)$, given the dataset $\mathcal{X}$ and the set of supervised class labels $\mathcal{Y}$ determines θ in order to reduce the prediction error as much as possible

Example of decision function

- supervised dataset with two dimensions, two classes
- use as decision function a straight line

$$\theta_1 * d_1 + \theta_2 * d_2 + \theta_3 \geqslant 0 \Rightarrow c_1$$

$$\theta_1 * d_1 + \theta_2 * d_2 + \theta_3 < 0 \Rightarrow c_2$$



All models are wrong, but some are useful

George Box

- The model (decision function) of the previous page makes some errors
  - even the best choice of parameters cannot avoid errors
- Different models can have different power to shatter the dataset into subsets with homogeneous classes
  - e.g. what about a quadratic function?
    $\theta_1 * d_1^2 + \theta_2 * d_2^2 + \theta_3 * d_1 * d_2 + \theta_4 * d_1 + \theta_5 d_2 + \theta_6$

Vapnik-Chervonenkis Dimension

The shattering power of a classification model[1]

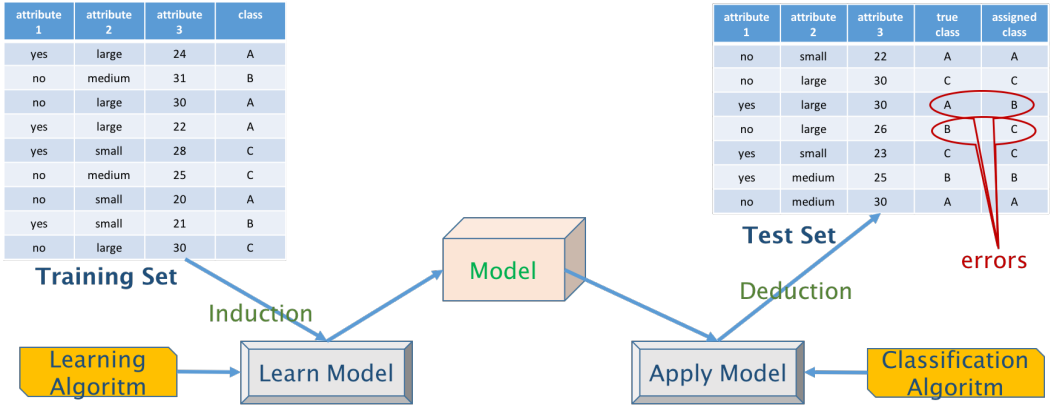
- Given a dataset with N elements there are 2^N possible different learning problems
- If a model $\mathcal{M}(.,.)$ is able to shatter all the possible learning problems with N elements, we say that it has Vapnik-Chervonenkis Dimension equal to N
- The straight line has VC dimension 3
  - don't worry, frequently, in real cases, data are arranged in such a way that also a straight line is not so bad

1 For simplicity, we mention here only the binary case

A workflow for classification

1. Learning the model for the given set of classes
   - 1.1 a training set is available, containing a number of individuals
   - 1.2 for each individual the value of the class label is available (also named ground truth)
   - 1.3 the training set should be representative as much as possible
     - 1.3.1 the training set should be obtained by a random process
   - 1.4 the model is fit learning from data the best parameter setting
2. Estimate the accuracy of the model
   - 2.1 a test set is available, for which the class labels are known
   - 2.2 the model is run by a classification algorithm to assign the labels to the individuals
     - 2.2.1 the classification algorithm implements the model with the parameters
   - 2.3 the labels assigned by the model are compared with the true ones, to estimate the accuracy
3. The model is used to label new individuals
   - 3.1 possibly, after the labeling, the true labels may become available and the true accuracy can be compared with the estimated one

A workflow for Learning and Estimation



Two flavors for classification

Crisp

- the classifier assigns to each individual one label

Probabilistic

- the classifier assigns a probability for each of the possible labels

Decision Trees

C4.5 and beyond [Buntine(1992)]

- Good compromise: decent performance, fast to train and execute, easy to understand
- History
  - 1966 – ID3 [Hunt et al.(1966)Hunt, Marin, and Stone]
  - 1979 – CLS [Quinlan(1979)]
  - 1993 – C4.5 [Quinlan(1979)]
- Generate classifiers structured as decision trees

Using a Decision Tree 1/2[2]

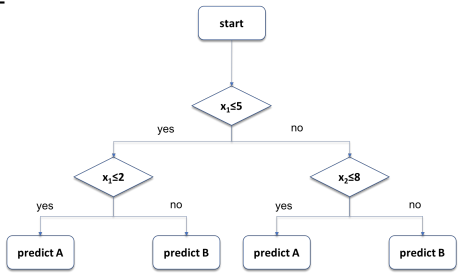
- A run–time classifier structured as a decision tree is a tree–shaped set of tests
- the decision tree has inner nodes and leaf nodes

2 Syntetic description in [Wu et al.(2008)Wu, Kumar, Quinlan, Ghosh, Yang, Motoda, McLachlan, Ng, Liu, Yu, Zhou, Steinbach, Hand, and Steinberg]

Using a Decision Tree 2/2

Algorithm 1: Decision Tree Node Execution

```
if the node is inner then
    if test on attribute d of element x then
        execute node_yes
    else
        execute node_no
else
    // leaf node
    predict class of element x as c
```



Learning a decision tree – Model generation

Given a set $\mathcal{X}$ of elements for which the class is known, grow a decision tree as follows

- if all the elements belong to class c or $\mathcal{X}$ is small generate a leaf node with label c
- otherwise
  - choose a test based on a single attribute with two or more outcomes
  - make this test the root of a tree with one branch for each of the outcomes of the test
  - partition $\mathcal{X}$ into subsets corresponding to the outcomes and apply recursively the procedures to the subsets

Learning a decision tree

Problems to solve:

1. which attribute should we test?
2. which kind of test?
   2.1 binary, multi–way, . . . , depends also on the domain of the attribute
3. what does it mean $\mathcal{X}$ is small, in order to choose if a leaf node is to be generated also if the class in $\mathcal{X}$ is not unique?

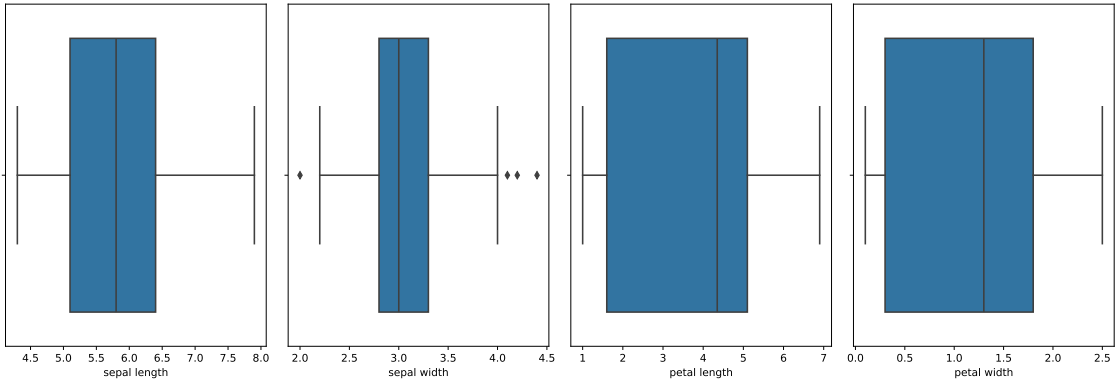
A supervised dataset: Iris

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
6.2	2.2	4.5	1.5	1
5.2	3.5	1.5	0.2	0
5.6	3.0	4.5	1.5	1
6.0	2.9	4.5	1.5	1
7.7	3.0	6.1	2.3	2
5.1	3.8	1.5	0.3	0
5.9	3.2	4.8	1.8	1
5.7	4.4	1.5	0.4	0
6.7	3.1	5.6	2.4	2
6.5	3.2	5.1	2.0	2
...	...	...	...	...

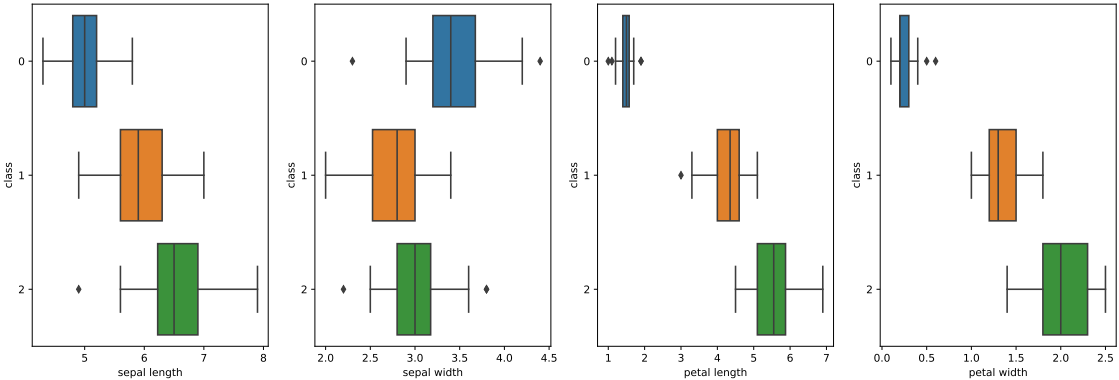
Dataset description

- 150 examples of iris flowers
- 4 attributes describing sizes of petals and sepals, class is the target
  - class has three values
- we could be interested in predicting the class for a new individual, given the measures

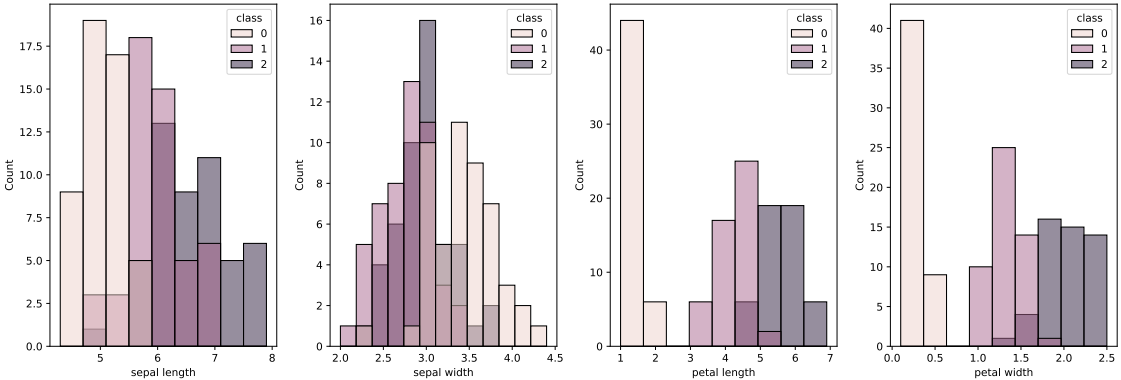
Exploration of the dataset - Boxplot - General



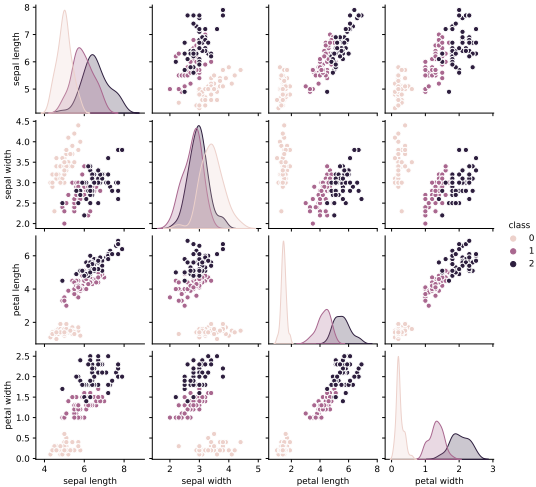
Exploration of the dataset - Boxplot - Inside classes



Exploration of the dataset - Histograms



Exploration of the dataset - Pairplots



Supervised learning goals

- design an algorithm able to forecast the values of an attribute given the values of other attributes
  - in our case, guess the class given the other values
- distinguish real patterns from illusions
- choose useful patterns
- in real life, we could have millions of rows and thousands of columns
  - looking at plots could be very hard

Evaluate how much a pattern is interesting

- several methods, one of them is based on information theory
  - information theory is primarily used in telecommunications
  - it is based on the concept of entropy
    - information content, surprise, ...
    - Claude Shannon, "A Mathematical Theory of Communication", 1948

The bit transmission example

- given a variable with 4 possible values and a given probability distribution
  $P(A) = 0.25, P(B) = 0.25, P(C) = 0.25, P(D) = 0.25$
- an observation of the data stream could return
  BAACBADCDADDDA ...
- the observation could be transmitted on a serial digital line with a two–bit coding
  $A = 00, B = 01, C = 10, D = 11$
- the transmission will be
  01000010010011101100111111100 ...

Less bits

- What if the probability distributions are uneven?
  $P(A) = 0.5, P(B) = 0.25, P(C) = 0.125, P(D) = 0.125$
- of course, the coding shown above is possible, requiring two bits per symbol
- is there a coding requiring only 1.75 bit per symbol, on the average?

Less bits

- What if the probability distributions are uneven?
  $P(A) = 0.5, P(B) = 0.25, P(C) = 0.125, P(D) = 0.125$
- of course, the coding shown above is possible, requiring two bits per symbol
- is there a coding requiring only 1.75 bit per symbol, on the average?

$$A = 0, B = 10, C = 110, D = 111$$

Even less bits

- What if there are only three symbols with equal probability? $P(A) = 1/3, P(B) = 1/3, P(C) = 1/3$
- of course, the two–bit coding shown above is still possible
- is there a coding requiring less than 1.6 bit per symbol, on the average?

Even less bits

- What if there are only three symbols with equal probability?
  $P(A) = 1/3, P(B) = 1/3, P(C) = 1/3$
- of course, the two–bit coding shown above is still possible
- is there a coding requiring less than 1.6 bit per symbol, on the average?

  $A = 0, B = 10, C = 11$ or any permutation of the assignment

General case

- Given a source X with V possible values, with probability distribution

$$P(v_1) = p_1, P(v_2) = p_2, \ldots, P(v_V) = p_V$$

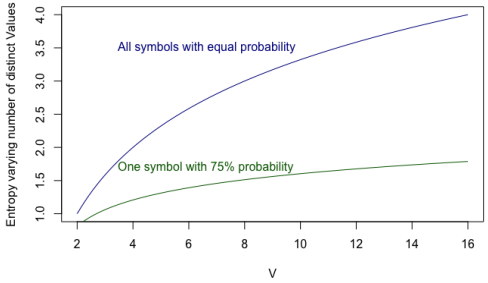
- the best coding allows the transmission with an average number of bits given by

$$H(X) = -\sum_j p_j \log_2(p_j)$$

$H(X)$ is the entropy of the information source X

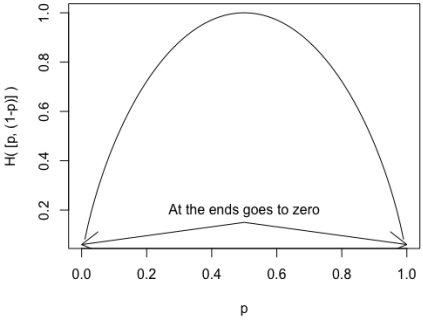
Meaning of entropy of an information source

- high entropy means that the probabilities are mostly similar
  - the histogram would be flat
- low entropy means that some symbols have much higher probability
  - the histogram would have peaks
- higher number of allowed symbols (i.e. of distinct values in an attribute) gives higher entropy



Entropy of a binary source

In a binary source with symbol probabilities p and (1-p) when p is 0 or 1 the entropy goes to 0



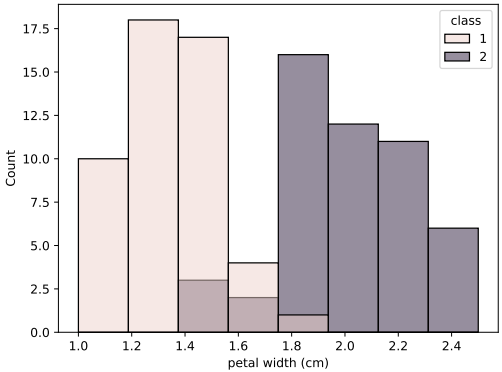
2 Teacher–only note: ML-03-01-Entropy

From entropy in transmission to entropy in classification

- In transmission, information sources with lower entropy can be represented (and transmitted) with shorter codes
  - b.t.w. this is the basis of information compression, i.e. the basis of modern audio and video telecommunications
- In classification, low entropy of the class labels of a dataset means that it is there is low diversity in the labels (i.e. the dataset has high purity, there is a majority class)
- We look for criteria that allow to split a dataset into subsets with higher purity
  - with criteria we mean logical formulas to be used as decision function to partition the set elements into the subsets

Entropy for the target column **class** in the reduced Iris dataset

A subset: only the fourth data column and the target, only the rows with classes 1 or 2



petal width (cm)	class
2	2
1.7	1
1.3	1
2.2	2
1.5	1
1.5	2
2.3	2
2	2
2.5	2
1.7	2
...	...

$$N = 100 \qquad p_{class=1} = 0.5, p_{class=2} = 0.5$$

$$H_{class} = -(p_{class=1} * log_2(p_{class=1}) + p_{class=2} * log_2(p_{class=2})) = 1$$

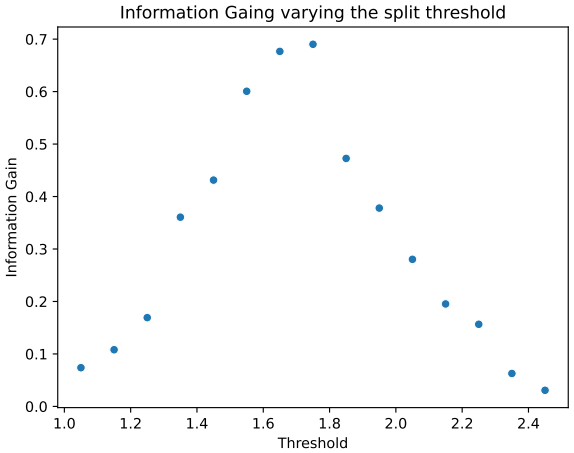
Entropy after a threshold–based split

- Splitting the dataset in two parts according to a threshold on a numeric attribute the entropy changes, and becomes the weighted sum of the entropies of the two parts
  - the weights are the relative sizes of the two parts
- Let $d \in \mathcal{D}$ be a real–valued attribute, let t be a value of the domain of d, let c be the class attribute
- We define the entropy of c w.r.t. d with threshold t as $H(c|d:t) = H(c|d < t) * P(d < t) + H(c|d \geqslant t) * P(d \geqslant t)$

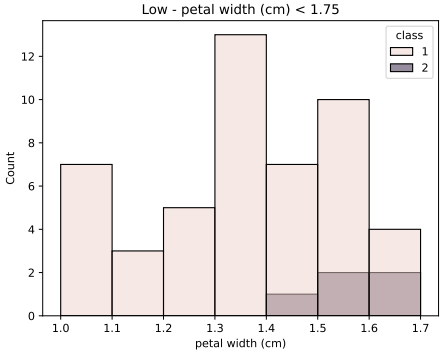
Information Gain for binary split

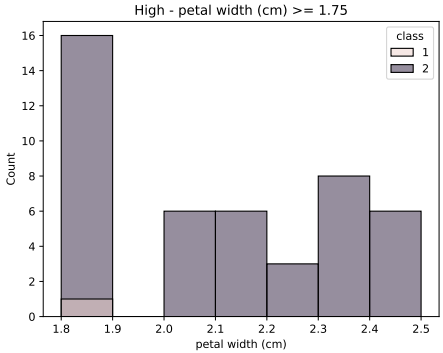
- It is the reduction of the entropy of a target class obtained with a split of the dataset based on a threshold for a given attribute
- We define $IG(c|d:t) = H(c) - H(c|d:t)$
  - it is the information gain provided when we know if, for an individual, d exceeds the threshold t in order to forecast the class value
- We define $IG(c|d) = \max_t IG(c|d:t)$

Change the threshold to find the best split



Let's split the reduced Iris with threshold 1.75





	low	high
1	49	1
2	5	45
	54	46

$$H(class|petalwidth : 1.75) = 0.31 =$$
$$- \quad (49/54 * log_2(49/54) + 5/54 * log_2(5/54)) * 0.54 +$$
$$(1/46 * log_2(1/46) + 45/46 * log_2(45/46)) * 0.46$$

How can we use the information gain?

Predict the probability of long life given some historical data on person characteristics and life style

- $IG(LongLife|HairColor) = 0.01$
- $IG(LongLife|Smoker) = 0.2$
- $IG(LongLife|Gender) = 0.25$
- $IG(LongLife|LastDigitSSN) = 0.00001$

Correlations between attributes is an important issue: it is not considered here

Back to DT generation

Choosing the attribute to test

A decision tree is a tree–structured plan generating a sequence of tests on the known attributes (predicting attributes) to predict the values of an unknown attribute.

Consider question 1 of page 20: which attribute should we test?

- test the attribute which guarantees the maximum IG for the class attribute in the current data set $\mathcal{X}$
- partition $\mathcal{X}$ according to the test outcomes
- recursion on the partitioned data

Train/Test split[4]

- The supervised data set will be split in (at least) two parts:
  - Training set used to learn the model
  - Test set used to evaluate the learned model on fresh data
- The split is done randomly
- Assumption: the parts have similar characteristics
- The proportion of the split is decided by the experimenter
  - Common solutions: 80-20, 67-33, 50-50
- The following slides consider a 50-50 split of the Iris dataset[3]
  - For this specific split, entropies for the class column in training and test turns out to be both 1.58

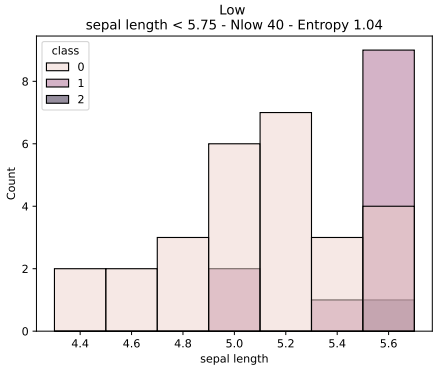
3 In the example, the split has been done using sklearn.model_selection_train_test_split and random_state = 10

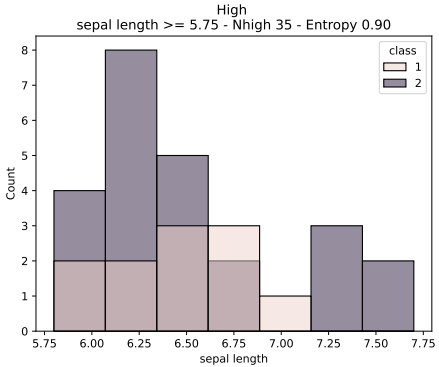
4 You can read this link for a short discussion

Iris Dataset - Predicting attribute: Sepal Length

Best threshold: 5.75

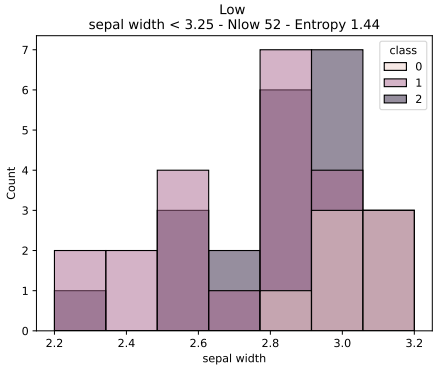
Information Gain: $1.58 - (40 * 1.04 + 35 * 0.90)/75 = 0.61$

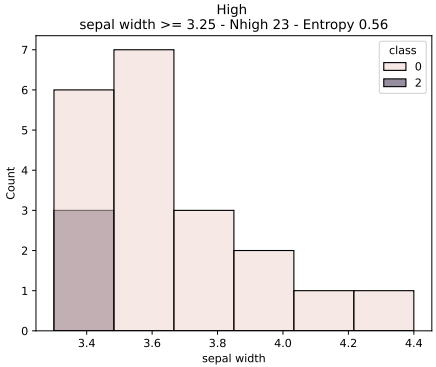




Iris Dataset - Predicting attribute: Sepal Width

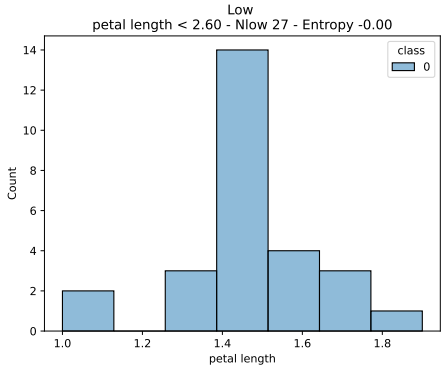
Best threshold: 3.25 – Information Gain: 1.58 -(52*1.44+23*0.56)/75 = 0.41

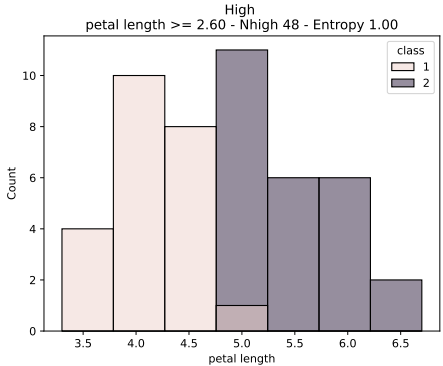




Iris Dataset - Predicting attribute: Petal Length

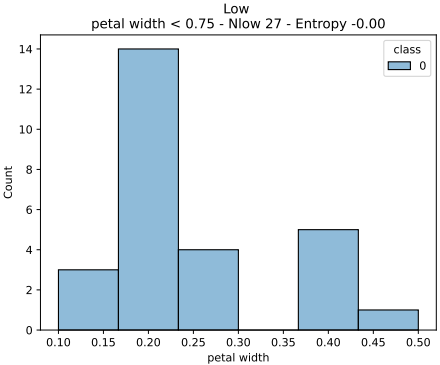
Best threshold: 2.6 – Information Gain: 0.94

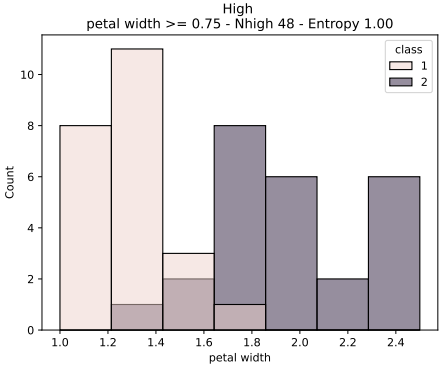




Iris Dataset - Predicting attribute: Petal Width

Best threshold: 0.75 – Information Gain: 0.94

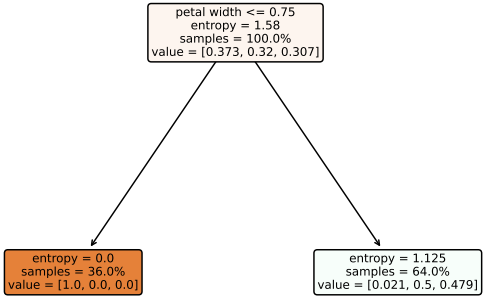




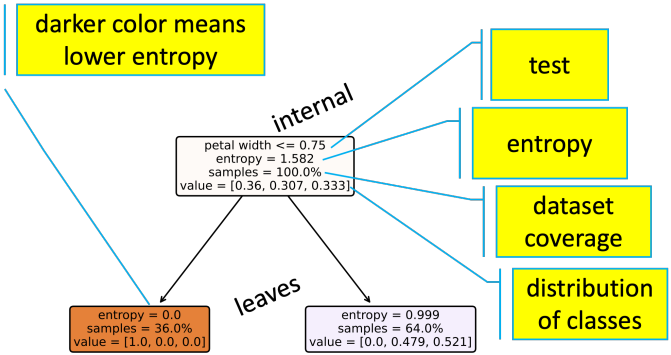
One–stump decision

Now on the entire training set with the three classes

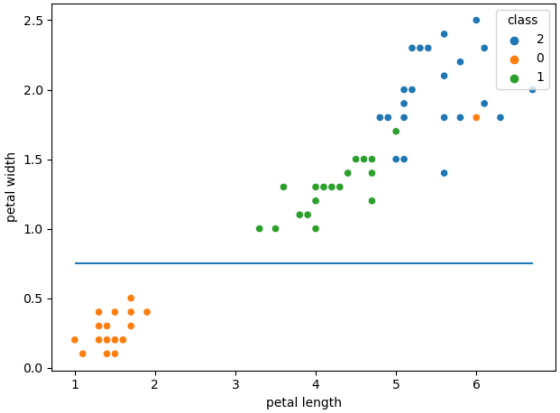
- choose the attribute giving the highest IG
- partition the dataset according to the chosen attribute
- choose as class label of each partition the majority



What's in a node

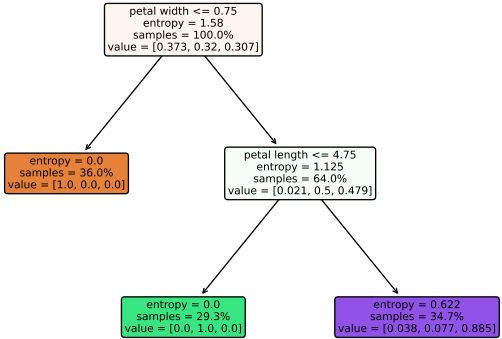


First split

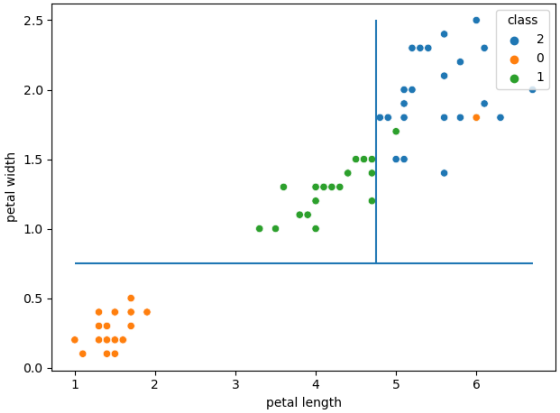


Recursion step

Build a new tree starting from each subset where the minority is non–empty

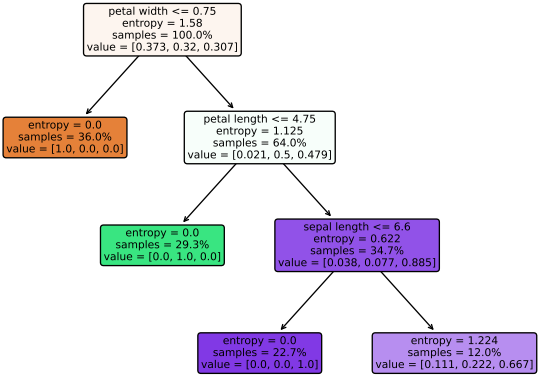


Second split



Recursion step

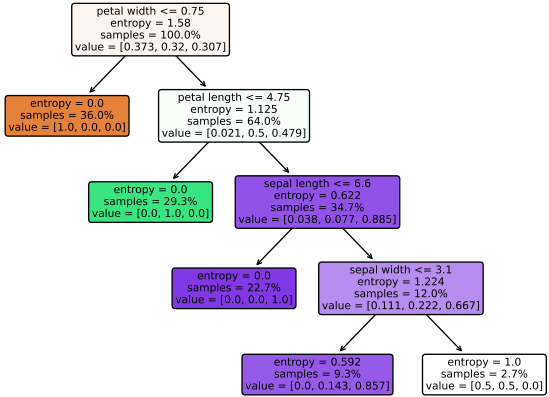
Build a new tree starting from each subset where the minority is non–empty



Recursion step

Observation

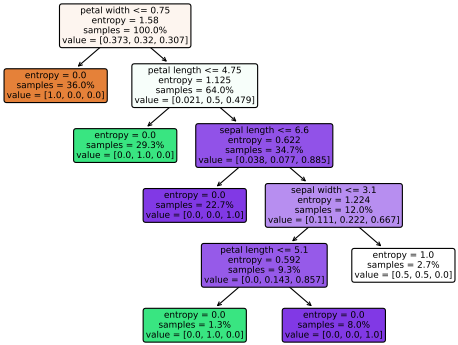
- The weighted sum of the entropy of the descendant nodes is always smaller than the entropy in the ancestor node, even if one of the descendant has higher entropy.
- Consider the three bottom right nodes (a=ancestor, ld=left descendant, rd=right descendant)



$$ent_a = 0.622 > (ent_{ld} * samp_{ld} + ent_{rd} * samp_{rd}) / samp_a = (0 * 22.7 + 1.224 * 12) / 34.7 = 0.39$$

Recursion ends

- Most of the leaves are pure, recursion impossible
- One of the leaves is not pure, but no more tests are able to give positive information gain, recursion impossible
  - it is labelled with the majority class, or, in case of tie, with one of the non–empty classes
- The error rate on the training set is 1.35%
  - 1 of the 75 examples in the training set is not correctly classified by the learned decision tree
  - it is one of the two items in the rightmost leaf



Building a Decision Tree with binary splits

Procedure BUILDTREE(dataset $\mathcal{X}$, node p)

- **if** all the class values of $\mathcal{X}$ are c **then**
  - **return** node p as a leaf, label of p is c;
- **if** no attribute can give a positive information gain in $\mathcal{X}$ **then**
  - compute the most frequent class in $\mathcal{X}$ let's say it is c;
  - **return** node p as a leaf, label of p is c;
- find the attribute d and threshold t giving maximum information gain in $\mathcal{X}$;
- create two internal nodes descendant of p, say p_{left} and p_{right};
- let $\mathcal{X}_{left} =$ selection on $\mathcal{X}$ with $d < t$;
- BUILDTREE($\mathcal{X}_{left}$, p_{left});
- let $\mathcal{X}_{right} =$ selection on $\mathcal{X}$ with $d \geqslant t$;
- BUILDTREE($\mathcal{X}_{right}$, p_{right});

Decision tree for the Iris classifier

Internal representation

	ChLeft	ChRight	Feature	Threshold	NNodeSamples	Impurity
0	1	2	petal width	0.750000	75	1.579659
1	-	-	-	nan	27	0.000000
2	3	4	petal length	4.750000	48	1.124941
3	-	-	-	nan	22	0.000000
4	5	6	sepal length	6.600000	26	0.621904
5	-	-	-	nan	17	0.000000
6	7	10	sepal width	3.100000	9	1.224394
7	8	9	petal length	5.100000	7	0.591673
8	-	-	-	nan	1	0.000000
9	-	-	-	nan	6	0.000000
10	-	-	-	nan	2	1.000000

Training Set Error

- execute the generated decision tree on the training set itself
  - obviously the class attribute is hidden
- count the number of discordances between the true and the predicted class
- this is the training set error

Causes of non–zero training set error

https://app.wooclap.com/COFEKT

The training set error can be greater than zero because of...

Causes of non–zero training set error

https://app.wooclap.com/COFEKT

The training set error can be greater than zero because of...

- the limits of decision trees in general:
  - a decision tree based on tests on attribute values can fail
- insufficient information in the predicting attributes

Training Set Error

- Is this 1.35% interesting? What is its meaning?

Training Set Error

- Is this 1.35% interesting? What is its meaning?
- It is the error we make on the data we used to generate the classification model
- It is probably the lower limit of the error we can expect when classifying new data
- We are much more interested to an upper limit, or to a more significant value

Test set error

- The test set error is more indicative of the expected behaviour with new data
- Additional statistic reasoning can be used to infer error bounds given the test set error
- We have available 75 additional labelled records in the Iris dataset

Iris classification error

	Num Errors	Set Size	%*Wrong*
Training Set	1	75	1.35
Test Set	13	75	17.33

Iris classification error

	Num Errors	Set Size	%*Wrong*
Training Set	1	75	1.35
Test Set	13	75	17.33

Why the test set error is so much worse?

Overfitting 1/2

Definition: overfitting happens when the learning is affected by noise

When a learning algorithm is affected by noise, the performance on the test set is (much) worse than that on the training set

Overfitting 2/2

More formally

A decision tree is a hypothesis of the relationship between the predictor attributes and the class. Some definitions:

- h = hypothesis
- $error_{train}(h)$ = error of the hypothesis on the training set
- $error_{\mathcal{X}}(h)$ = error of the hypothesis on the entire dataset

h overfits the training set if there is an alternative hypothesis h' such that

$$\begin{aligned} error_{train}(h) &< error_{train}(h') \\ error_{\mathcal{X}}(h) &> error_{\mathcal{X}}(h') \end{aligned}$$

Causes for overfitting

1. Presence of noise
   - individuals in the training set can have bad values in the predicting attributes and/or in the class label, or can represent unusual cases
   - in this case, the model is influenced from partly wrong or unusual training data
2. Lack of representative instances
   - some situations of the real world can be underrepresented, or not represented at all, in the training set
   - this situation is quite common

A good model has low generalization error i.e. it works well on examples different from those used in training

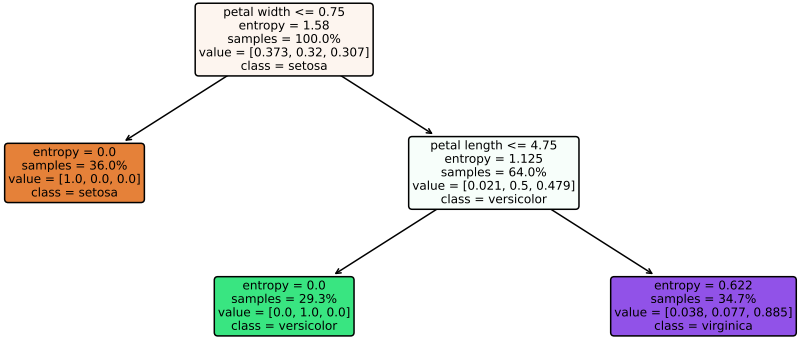
Occam's Razor [5]

Everything should be made
as simple as possibile,
but not simpler

- all other things being equal, simple theories are preferable to complex ones
- a long hypothesis that fits the data is more likely to be a coincidence
- pruning a decision tree is a way to simplify it
  - we need to find precise, quantitative guidelines for effective pruning

5 William of Ockham, an english franciscan philosopher of the 14-th century

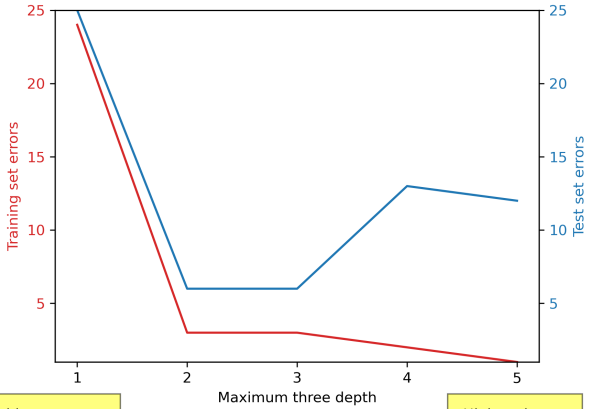
Example: Iris classification with pruned tree



	Num Errors	Set Size	%*Wrong*
Training Set	3	75	4.00
Test Set	6	75	8.00

Effect of adjusting pruning

Pruning is the way to simplify the model when you are using a decision tree



Hyperparameters

- Every model generation algorithm can be adjusted by setting specific hyperparameters
- Each model has its own hyperparameters
- One of the hyperparameters of decision tree generation is the maximum tree depth

Choice of the attribute to split the dataset

- Looking for the split generating the maximum purity
- We need a measure for the purity of a node
  - a node with two classes in the same proportion has low purity
  - a node with only one class has highest purity

Impurity functions

Measures of the impurity of a node

Entropy – already seen [6]

Gini Index

Misclassification Error

OPTIONAL

6 it is available in Scikit-Learn

Gini Index[7] – Intuition

- Consider a node p with C_p classes
- Which is the frequency of the wrong classification in class j given by a random assignment based only on the class frequencies in the current node?
- For class j
  - frequency $f_{p,j}$
  - frequency of the other classes $1 - f_{p,j}$
  - probability of wrong assignment $f_{p,j} * (1 - f_{p,j})$
- the Gini Index is the total probability of wrong classification

$$\sum_j f_{p,j} * (1 - f_{p,j}) = \sum_j f_{p,j} - \sum_j f_{p,j}^2 = 1 - \sum_j f_{p,j}^2$$

7 This is the default impurity measure in Scikit-Learn

Gini Index – Discussion

- the maximum value is when all the records are uniformly distributed over all the classes: $\mathbf{1 - 1/C_p}$
- the minimum value is when all the records belong to the same class: $\mathbf{0}$

Splitting based on the Gini Index

- Used by CART, SLIQ, SPRINT
- When a node p is split into ds descendants, say $p_1, \ldots, p_{ds}$
- Let $N_{p,i}$ and N_p be the number of records in the i-th descendant node and in the root, respectively
- We choose the split giving the maximum reduction of the Gini Index

$$GINI_{split} = GINI_p - \sum_{i=1}^{ds} \frac{N_{p,i}}{N_p} GINI(p_i)$$

Misclassification Error

OPTIONAL

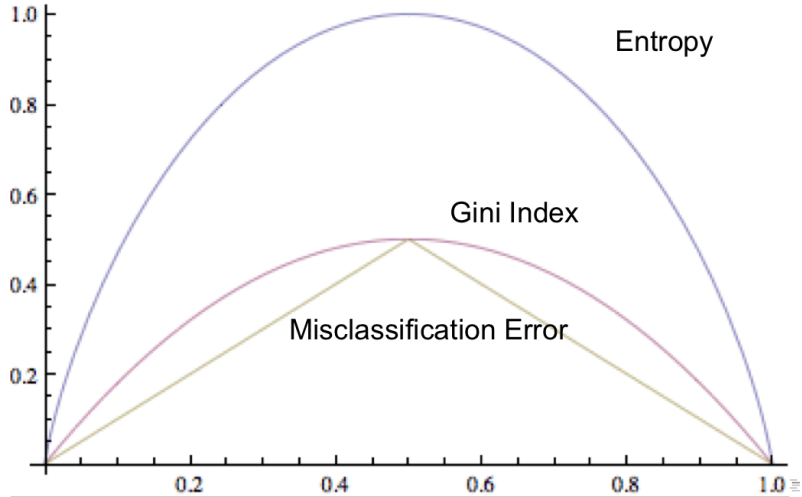
- If a node is a leaf, we find the highest label frequency; this frequency is the accuracy of the node and this label is the output of the node
- The misclassification error is the complement to 1 of the accuracy
- Since the most frequent class determines the node label, the complement is the error
  - The maximum value is when all the records are uniformly distributed over all the classes: $1 - 1/C_p$
  - The minimum value is when all the records belong to the same class: 0
- The choice of the split is done in the same way as for the Gini index

$$ME(p) = 1 - \max_j f_{p,j}$$

Comparison of the impurity functions

OPTIONAL

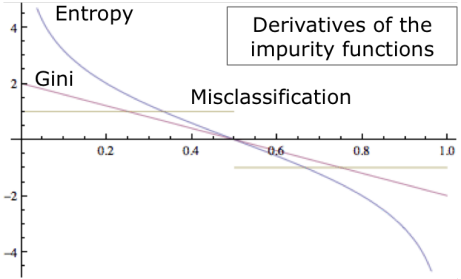
For two classes with frequencies f and $1 - f$



Comparison of the impurity functions – Discussion

Optional

- The behavior of ME is linear, therefore an error in the frequency is completely transferred into the impurity computation
- Entropy and Gini have varying derivative, with the minimum around the center
  - they are more robust w.r.t. errors in the frequency, when the frequencies of the two classes are similar



Complexity of DT induction (C4.5 algorithm) I

- N instances and D attributes in $\mathcal{X}$
  - tree height is $\mathcal{O}(\log N)$
- Each level of the tree requires the consideration of all the dataset (considering all the nodes)
- Each node requires the consideration of all the attributes
  - overall cost is $\mathcal{O}(DN \log N)$

Complexity of DT induction (C4.5 algorithm) II

- In addition
  - binary split of numeric attributes costs $\mathcal{O}(N \log N)$, without increment of complexity
  - pruning requires to consider globally all instances at each level, generating an additional $\mathcal{O}(N \log N)$, which does not increase complexity.

Characteristics of DT Induction I

1. It is a non–parametric approach to build classification models
   - it does not require any assumption on the probability distributions of classes and attribute values
2. Finding the best DT is NP–complete, the heuristic algorithms allow to find sub–optimal solutions in reasonable times
3. The run–time use of a DT to classify new instances is extremely efficient: $\mathcal{O}(h)$, where h is the height of the tree
4. Robust w.r.t. noise in the training set (i.e. wrong class labels), if the overfitting is avoided with appropriate pruning
5. Redundant attributes do not cause any difficulty

Characteristics of DT Induction II

- In case of strong correlation between two attributes, if one is chosen for a split, most likely the other will never provide a good increment of node purity, and will never be chosen

6. The nodes at a high depth are easily irrelevant (and therefore pruned), due to the low number of training records they cover
7. In practice, the impurity measure has low impact on the final result
8. In practice, the pruning strategy has high impact on the final result

Algorithms for building DTs

- Several variants, depending on
  - tree construction strategy
  - partition strategy
  - pruning strategy
- Tests based on linear combinations of numeric attributes
- Multivariate tests (e.g. a = x and b = y)
- ...

Conclusion

- Decision trees are usually the best starting point to learn supervised machine learning
  - easy to understand
  - easy to implement
  - easy to use
- Overfitting can be controlled by adjusting the maximum tree depth
  - other adjustments are possible, depending on the implementation
    - e.g. lookup the `scikit-learn` description of the DecisionTreeClassifier
- Are able to predict discrete values (the class) on the basis of continuous or discrete predictor attributes[8]

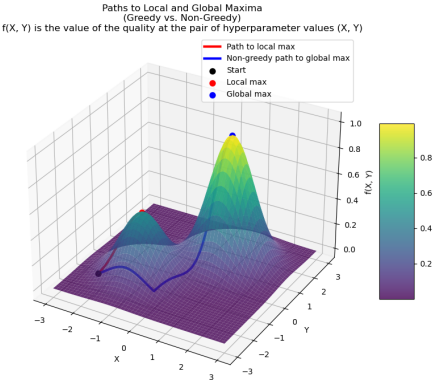
8 The Scikit-Learn implementation of Decision Trees do not allow discrete attributes, therefore in these cases a data transformation is necessary

Greedy algorithms

DT building is a greedy algorithm

Greedy algorithms

- An approximate family of algorithms, belonging to the broader family of hill–climbing algorithms
  - they do not guarantee reaching the best theoretically possible result
- At every decision point, the choice is made that guarantees the maximum immediate gain
  - it does not take into account that a sequence of decisions may lead to the best final result even if individually they do not pass through the points of the best partial improvement



Important concepts

- Impurity functions: entropy, Gini, misclassification
- The recursive greedy algorithm for building a decision tree
- Training error and test error
- Why the test error can be much greater than the training error
- Why the pruning can improve the performance
- How to deal with continuous attributes

Questions

- Why do we prefer a greedy algorithm instead of trying all the possibile trees?
- Consider the Adult dataset. If the decision tree to predict wealth has the marital status near to the top, can we say that the marital status is a major cause for wealth?
- Can we say that the attributes which are not mentioned in the tree are not a cause for wealth?

Model Selection
Evaluation and optimisation of a Classifier
Questions to be answered

- Which of the available models for classification is the best one?
- Which of the available algorithms is the best one?
- Which is the best parameters configuration?

$\Rightarrow$ Evaluation

The Oil Slick example I

- Detect oil slick (failures, illegal dumping) from satellite images, for early alarm
- Radar satellite images
  - Dark regions whose size and shape depend on weather and sea conditions
  - Look–alike dark regions can also be caused by local weather conditions, such as high winds
  - Manual detection by experts is definitely expensive and slow
- Scarcity of training data: oil spills are, fortunately, rare
- Unbalanced nature of data: the negative examples (non–spills) are predominant over the positive ones

The Oil Slick example II

- An automatic hazard detection system has been developed and marketed
  - Pre–selection of images for final manual processing
  - Necessary a tradeoff between undetected spills and false alarms
  - Evaluation of performance guides the tradeoff

The training set

- In supervised learning the training set performance is overoptimistic
- We need a lower bound for performance obtained by independent tests
- Supervised data are usually scarce, we need to balance the use of them between:
  - train
  - validation, to tune the parameter (sometimes it is omitted)
  - test
- Evaluate how much the theory fits the data
- Evaluate the cost generated by prediction errors

Learning and evaluation

- Empirically (and intuitively) the more training data we use the best performance we should expect
  - Statistically, we should expect a larger covering of the situation that can occur when classifying new data
- We must consider the effect of random changes
- The evaluation is independent from the algorithm used to generate the model

Evaluation of a classifier

A first measure

The meaning of the test error

- let us suppose that the test set is a good representation, on the average, of the entire dataset $\mathcal{X}$ (i.e. run–time)
- the relationship between the training set and $\mathcal{X}$ will be subject to probabilistic variability
- the evaluation can be done either at different levels
  - general – the whole performance of the classifier
  - local – the local performance of a component of the model, i.e. a node of a decision tree
- if the test set error ratio is x, we should expect a run–time error $x \pm$???

$\Rightarrow$ confidence interval

Confidence interval in error estimation

Bernoulli process

- forecasting each element of the test set is like one experiment of a Bernoulli process
  - good prediction $\Rightarrow$ success
  - bad prediction $\Rightarrow$ error
- the same as N independent binary random events of the same type
- $f = S/N =$ empirical frequency of error
- which is the probability p of error?

Empirical frequency and true frequency

- Deviations of the empirical frequency from the true frequency are due to noise
- Usually, noise is assumed to have a normal distribution around the true probability (for $N \geqslant 30$)
- We need statistical reasoning to determine a pessimistic evaluation of p given f, N, and the amount of risk that we can tolerate
  - we can estimate the interval centred in f such that p will be in that interval with confidence $1 - \alpha$, where α is typically .05 or 0.01

$$P\left(z_{\alpha/2} \leqslant \frac{f-p}{\sqrt{p(1-p)/N}} \leqslant z_{1-\alpha/2}\right) = 1 - \alpha$$

Range error estimate

Wilson score interval

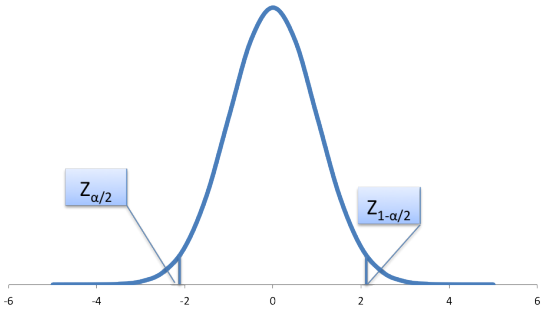
- z depends on the desired confidence level $1 - \alpha$
- it is the abscissa delimiting the area $1 - \alpha$ for a normal distribution
  - i.e. the inverse of the standard cumulative normal distribution
- with a little of algebra

$$\frac{1}{1 + \frac{1}{N}z^2}\left[f + \frac{1}{2N}z^2 \pm z\sqrt{\frac{1}{N}f(1-f) + \frac{1}{4N^2}z^2}\right]$$

The pessimistic error is obtained by substituting $\pm$ with $+$

Confidence level in error estimation

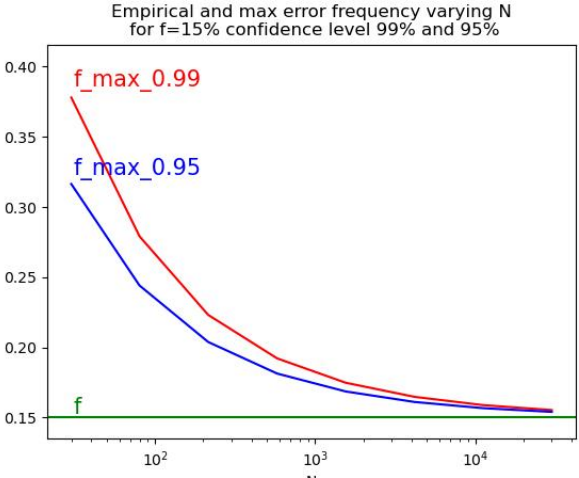
The confidence level $1 - \alpha$ is the probability that the estimation of the frequency of good predictions is good, i.e. the probability that the true frequency is inside the confidence interval around the empirical frequency



$1 - \alpha$	Z
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65
0.75	1.04
0.50	0.67

Confidence interval in error estimation

Increasing N, with constant empirical frequency, the uncertainty for p narrows.

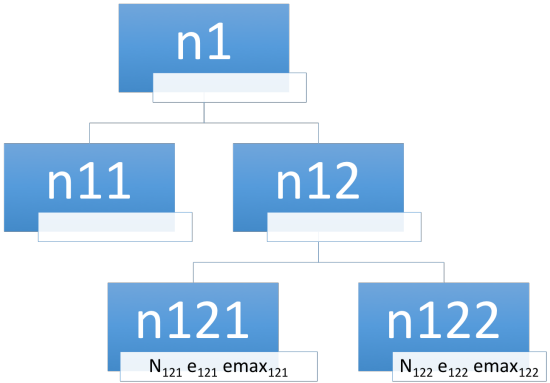


Statistical pruning of DT with error estimation

The C4.5 strategy

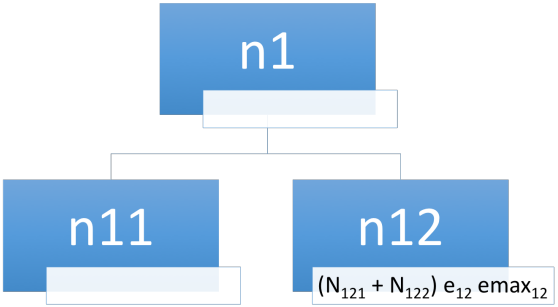
- Consider a subtree near the leaves
- Compute its maximum error e_l as a weighted sum of the maximum error of the leaves
- Compute the maximum error e_r of the root of the subtree transformed into a leaf
- Prune if $e_r \leqslant e_l$
- With pruning, the error frequency increases, but the number of records in node also increases, therefore the maximum error can decrease

DT before pruning



N_x = Records in node f_x = Error frequency in node e_x = Maximum error frequency in node

DT after pruning



$e_x - f_x$ increases as N_x decreases

Pruning is done if $(N_{121} + N_{121}) * e_{12} < N_{121} * e_{121} + N_{122} * e_{122}$

Other pruning techniques - examples

Scikit-Learn allows to adjust pruning with one or more of the hyperparameters below

- `max_depth` - the maximum depth allowed for the tree; it is a horizontal cut, pruning all the branches below a given depth
  - this is the most important
- `min_samples_split` - either the minimum absolute number of samples or the minimum fraction of samples (with respect to the entire population in the dataset) in a node to make a split, if the threshold is not exceeded the node becomes a leaf
- `min_samples_leaf` - the minimum number of samples (or fraction, as above) required to be at a leaf node
- `min_impurity_decrease` - a node will be split if this split induces a decrease of the impurity greater than or equal to this value; if the weighted sum of the descendant leaves do not decrease from the node under consideration more than this threshold then the node becomes a leaf

Pruning and other

- pruning is a typical method to reduce overfitting and optimise a Decision tree
- we will see several other classification models
- each of them has several hyperparameters that can be tuned to reach the same objective

Accuracy of a classifier I

- The error frequency is the simplest indicator of the quality of a classifier
  - it is the sum of errors on any class divided by the number of tested records
- From now on, for simplicity, we will use the empirical error frequencies
  - remember that in real cases the maximum error frequencies should be used instead

Accuracy of a classifier II

- Accuracy and other more sophisticated indicators are used to:
  - compare different classifiers or hyperparameter settings
  - estimate the run–time performance we can expect, and therefore the cost of errors
    - with run–time performance we mean the performance observed when we will use the trained classification model with a classifier to produce value in some activity

The hyperparameters – a.k.a. model selection I

Optimising the model learned

- Every machine learning algorithm has one or more parameters that influence its behaviour
  - they are usually called hyperparameters
- It is crucial to obtain a highly reliable estimate of the run–time performance

The hyperparameters – a.k.a. model selection II

Optimising the model learned

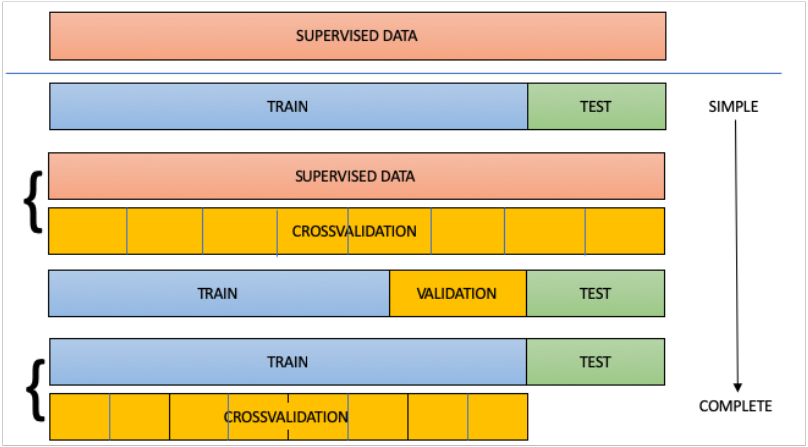
- Several train/test loops are in general necessary to find the best set of values for the hyperparameters
  - to be more general, the model selection includes the selection of the learning algorithm and its optimisation
- Sometimes it is necessary to find the best compromise between the optimisation step and the quality of the result
  - some learning algorithms require long computation times

Testing strategies

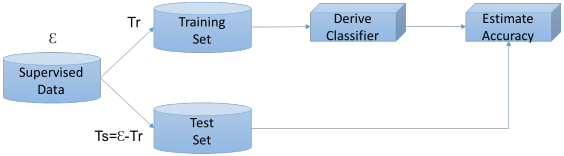
Getting the most out of the available supervised data

- as pointed out earlier, train and test should be done using different portions of the supervised data available
- in every step the data should be representative of the data that will be classified run–time
- a good strategy should
  - make the optimal usage of the supervised data
  - be compatible with the time constraints for the model selection

The train/test process: some alternatives



Holdout



- The split should be as random as possible
  - It may happen that the proportion of classes in the supervised dataset $\mathcal{X}$ is altered in the Training and Test sets, to prevent such cases the statistical sampling technique of stratification ensures the maintenance of the proportion of classes
- In this setting, the test set is used to obtain an estimation of the performance measures with new data

Dataset Size

- **Small Datasets**:
  - it's often beneficial to allocate more data for training to ensure the model has enough examples to learn from.
  - Typical splits might be 80/20 or even 90/10.
- **Large Datasets**:
  - you can afford to reserve more data for testing without compromising the training process.
  - Common splits are 70/30 or 80/20.

Model Complexity

- **Simple Models**:
  - require less data to train effectively, then you can afford a larger test set.
- **Complex Models**:
  - e.g. deep learning models often require more training data, so a split like 90/10 or 85/15 might be more appropriate.

Variance and Overfitting

- **High Variance Models**:
  - Models that are prone to overfitting may benefit from more training data to capture the underlying patterns better.
- **Cross-Validation**:
  - Using techniques like k-fold cross-validation can help mitigate overfitting and provide a more robust evaluation, reducing the dependence on a single train/test split.

Evaluation Stability

- **Stable Performance**:
  - A larger test set provides a more stable and reliable estimate of model performance, reducing variance in the evaluation metrics.
- **Rare Events**:
  - If the dataset contains rare events or classes, ensuring that these are adequately represented in both the training and test sets is crucial. Stratified sampling can help with this.

Computational Resources

- **Resource Constraints**:
  - Larger test sets mean more computational resources for evaluation. In resource-constrained environments, a smaller test set might be necessary.

Business and Domain Considerations

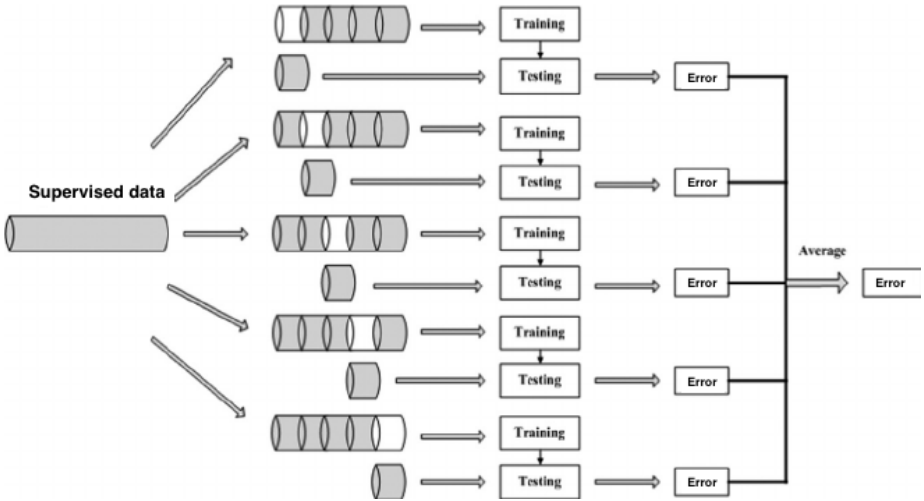
- **Critical Applications**:
  - In high-stakes applications (e.g., medical diagnosis, autonomous driving), ensuring a robust and reliable evaluation is crucial, often requiring a larger test set.
- **Regulatory Requirements**:
  - Some domains may have specific guidelines or standards that dictate the proportion of data to be used for testing and validation.

Cross Validation (k–fold)

A more complex strategy

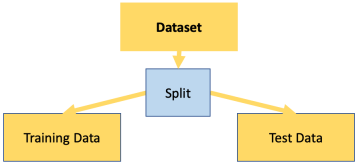
- The training set is randomly partitioned into k subsets
  - If necessary, use stratified partitioning
- k iterations using one of the subsets for test and the others for training
- Combine the result of tests
- Generate the final model using the entire training set
- Optimal use of the supervised data
  - each record is used $k-1$ times for training and once for testing
- Typical value: $k = 10$

Cross Validation

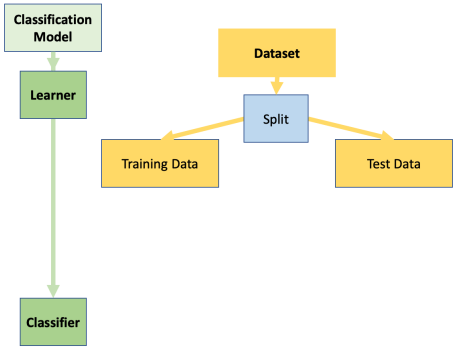


Cross–validation workflow

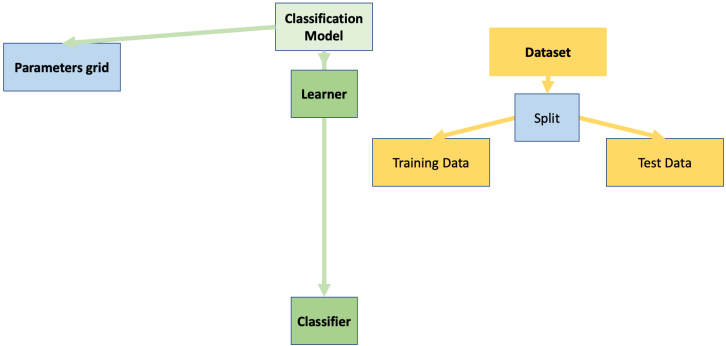
Cross–validation workflow



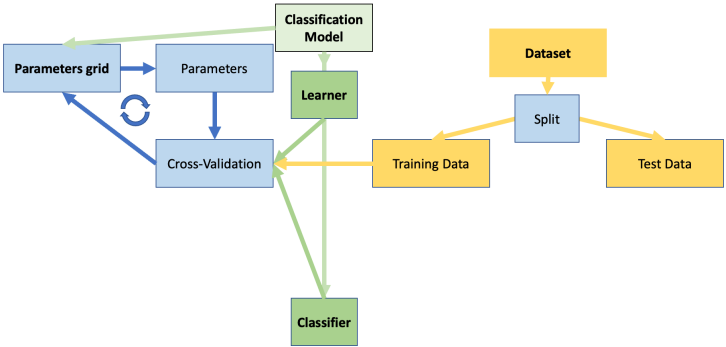
Cross–validation workflow



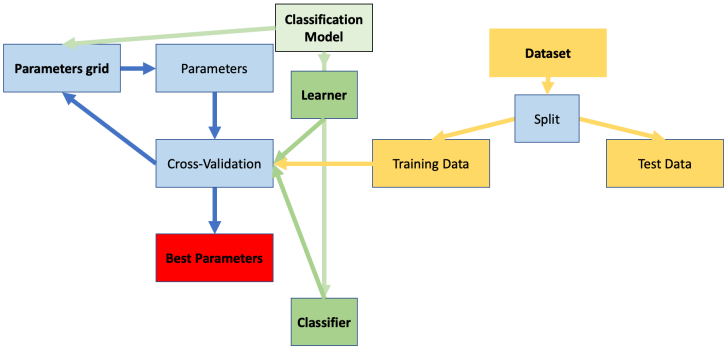
Cross–validation workflow



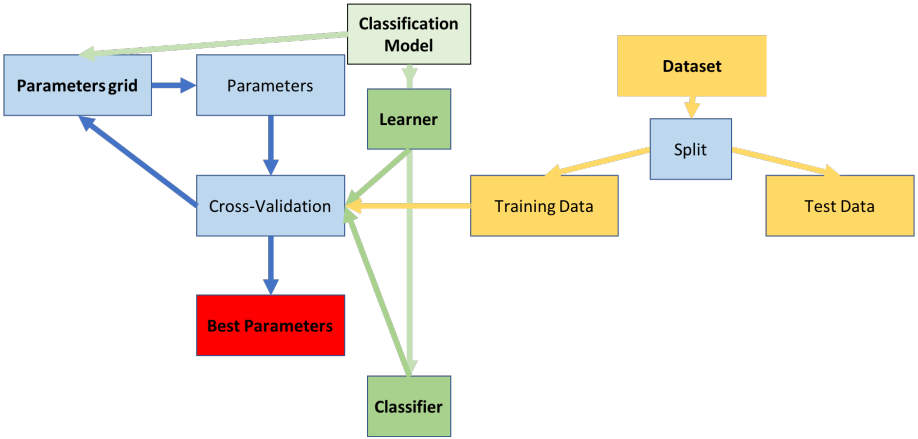
Cross–validation workflow



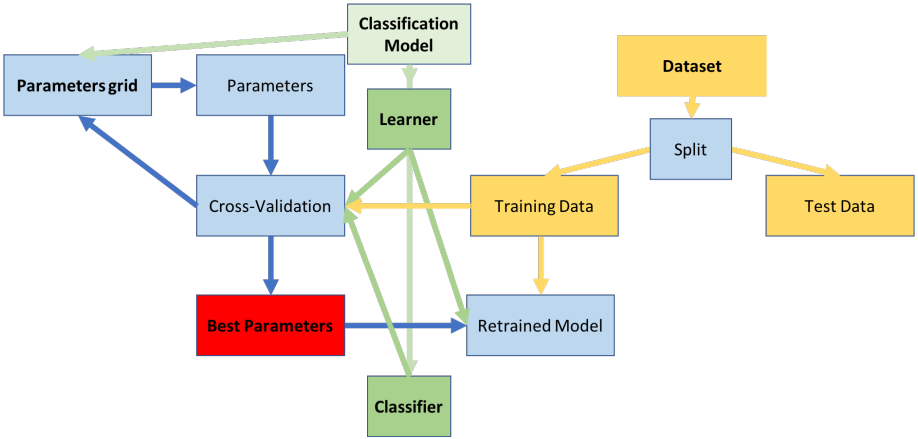
Cross–validation workflow



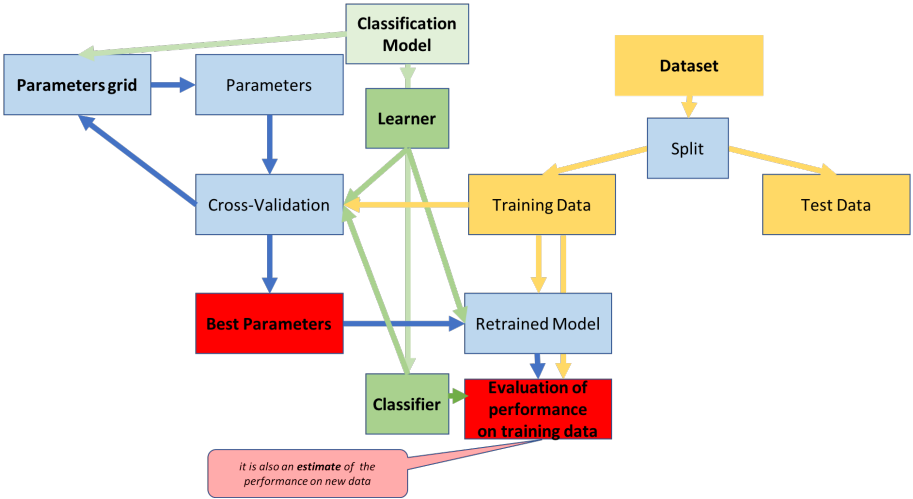
Cross–validation workflow



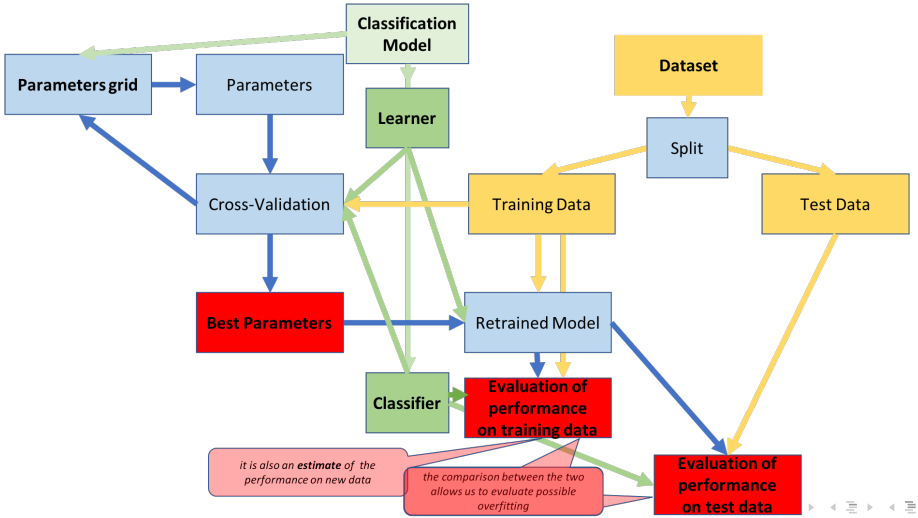
Cross–validation workflow



Cross–validation workflow



Cross–validation workflow



Cross Validation – pros and cons

- ☹ The train/test loop is repeated k times
- ☺ The estimate of the performance is averaged on k runs
  $\Rightarrow$ more reliablity
- ☺ All the examples are used once for testing
- ☺ The final model is obtained using all the examples
  $\Rightarrow$ best use of the examples

Train, Validation, Test – pros and cons

- The train/validation loop is faster than the Cross Validation
- The optimisation of the hyperparameters is done with the validation set, independent from the final evaluation

  $\Rightarrow$ more reliable than the simple holdout
- The test during the hyperparameters optimisation is done on a portion of the examples (the validation set)

  $\Rightarrow$ less reliable with respect to Cross Validation

Model selection – Zero level

- *Use of supervised data*
  - Entire dataset
- *Hyperparameters*
  - defaults
- *Action*
  - Train the model with the entire dataset, make predictions on the entire dataset and compare the predictions with the ground truth
- *Quality of results*
  - probably not the best possible with that model
- *Reliability of evaluation for generalisation*
  - Not reliable
- *Observations*
  - Just a first trial
- *`sklearn` model selection procedures*
  - none

Model selection – Quick attempt

- *Use of supervised data*
  - split dataset into Train and Test
- *Hyperparameters*
  - defaults
- *Action*
  - train the model with the train set, make predictions on the test set and compare the predictions with the ground truth
- *Quality of results*
  - probably not the best possible with that model
- *Reliability of evaluation for generalisation*
  - better estimation of the expected performance
- *Observations*
  - If you are in a hurry and/or the supervised dataset is small
- *`sklearn` model selection procedures*
  - `train_test_split`

Model selection – Standard

- *Use of supervised data*
  - Split dataset into Train and Test
- *Hyperparameters*
  - Find appropriate ranges of hyperparameters
- *Action*
  - For a number of combinations of hyperparameter values train the model with the Train set, predict with the Test set and evaluate the result. Choose the combination of hyperparameters that gives the best result, train the model with the Train set, predict and evaluate with the test set
- *Quality of results*
  - improved with respect to the defaults
- *Reliability of evaluation for generalisation*
  - possible overestimation, because the test data are involved in the optimisation
- *Observations*
  - Not too time consuming: the number of train/validate opeations is the number of combinations of hyperparameter values.

    Not good for very small datasets.
- *`sklearn` model selection procedures*
  - `train_test_split`, `ParameterGrid`

Model selection – Train/Validation/Test

- *Use of supervised data*
  - Split into Train-Validate and Test, then split Train-Validate into Train and Validate
- *Hyperparameters*
  - Find appropriate ranges of hyperparameters
- *Action*
  - For a number of combinations of hyperparameter values train with the Train set, predict with the Validate set and evaluate the result. Choose the combination of hyperparameters that gives the best result, train with the Train-Validate set and evaluate with the Test set
- *Quality of results*
  - Optimised for the model
- *Reliability of evaluation for generalisation*
  - Good reliability of the estimated performance
- *Observations*
  - Not too time consuming: the number of train/validate operations is the number of combinations of hyperparameter values.

    Not good for very small datasets
- *`sklearn` model selection procedures*
  - `train_test_split` two times, `ParameterGrid`

Model selection – Cross Validation

- *Use of supervised data*
  - Split dataset into Train and Test
- *Hyperparameters*
  - Find appropriate ranges of hyperparameters
- *Action*
  - For a number of combinations of hyperparameter values and do Cross Validation. Choose the combination of hyperparameters that gives the best result, train the model with the Train set and evaluate with the Test set
- *Quality of results*
  - Optimised for the model
- *Reliability of evaluation for generalisation*
  - Best reliability of the estimated performance
- *Observations*
  - Time consuming: the number of train/validate operations is the number of combinations of hyperparameter values times the number of cross validation folds.

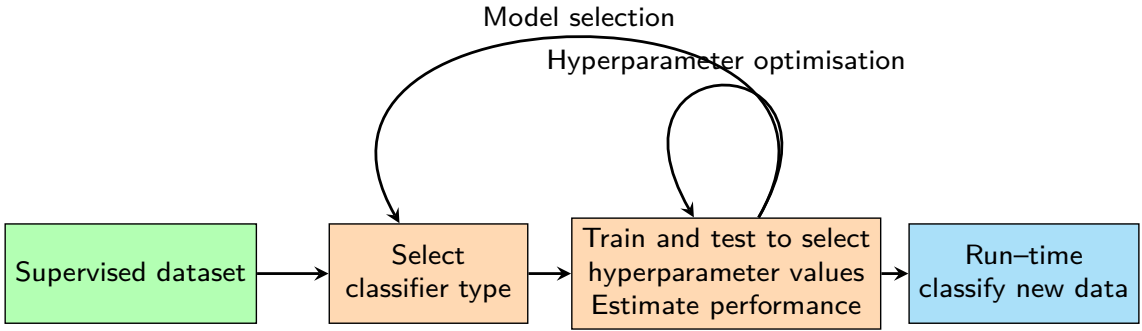
    Not good for very small datasets
- *`sklearn` model selection procedures*
  - `train_test_split`, `GridSearchCV` or `RandomizedSearchCV`

The model–selection process for the development of a classifier – I

A wider selection process

- We can use different models for doing classification
  - decision tree is one of them
- The general process implies to try several models, for each of them find the best hyperparameters and, at the end, use the model giving the best results with the best hyperparameters to classify new (unsupervised) data
  - later we will refer to this "productive" phase as run–time

The model–selection process for the development of a classifier – II



Performance measures of a classifier

What does best in page 140 mean?

Binary prediction

For simplicity: Positive/Negative

- success rate = accuracy

$$\frac{TP + TN}{N_{test}}$$

- error rate

$$1 - \text{success rate}$$

		Predicted class		
		pos	neg	Total
True class	pos	TP	FN	T_{pos}
	neg	FP	TN	T_{neg}
	Total	P_{pos}	P_{neg}	N_{test}

Confusion matrix

Is accuracy enough? I

- Is the accuracy the only performance indicator for a classifier? Other possible indicators:
  - Velocity
  - Robustness w.r.t. noise
    - i.e. training data with bad class label
  - Scalability
  - Interpretability

Is accuracy enough? II

- A classification error can have different consequences, depending on the class of the individual
  - when forecasting an illness a false positive can be less dangerous than a false negative
    - unless the cares or the additional examinations are dangerous or invasive
  - consider the cost of retiring a machinery as damaged, while it is ok (false positive) and the cost of an unpredicted failure (false negative)

A summary of measures I

Precision – $TP/(TP + FP)$

the rate of true positives among the positive classifications

Recall – $TP/(TP + FN)$

the rate of the positives that I can catch (a.k.a. Sensitivity)

Specificity – $TN/(TN + FP)$

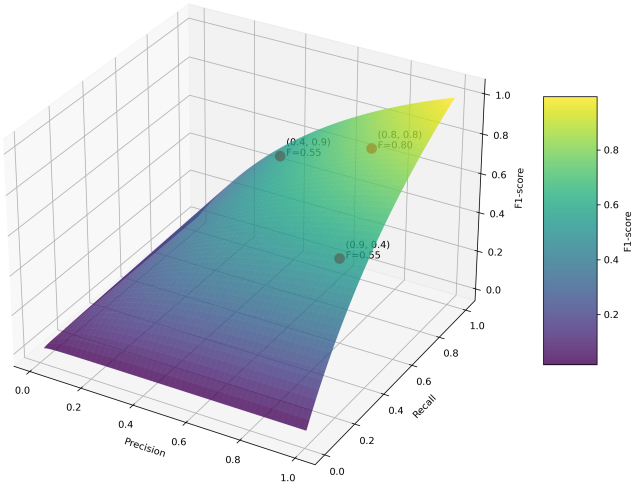
the rate of the negatives that I can catch

A summary of measures II

F1–score – the armonic mean of precision and recall, a.k.a. balanced F-score

$$F = 2\frac{\text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$$

F1–score as a function of Precision and Recall



Which measure should we use?

A rule of thumb

- Accuracy gives an initial feeling of the effectiveness of the classifier, but can be heavily misleading when classes are highly imbalanced
  - it considers globally all the classes, also when $C > 2$ (non-binary case, also called multi–class)
- F1–score is always interesting, because has higher values when precision and recall are reasonably balanced
- if the the costs of errors on positives and negatives are significantly different, then it is necessary to evaluate precision and recall.
- for multi–class problems see page 153.

Multi–class case

- The confusion matrix of page 144 is easily extended when there are more than two classes
- Each cell contains the number of test records of class i and predicted as class j
- The numbers in the main diagonal are the true predictions

Confusion matrix with three classes, say a, b, c

$T_i =$ true number of i labels in the dataset

$P_i =$ total number of i predictions by a given classifier, say $\overline{C}$

$TP_i =$ number of true predictions for class i given by classifier $\overline{C}$

$FP_{i-j} =$ number of false prediction for class i predicted as j

$$\text{accuracy} = \frac{\sum_i TP_i}{N}$$

$$\text{precision}_i = \frac{TP_i}{P_i}$$

$$\text{recall}_i = \frac{TP_i}{T_i}$$

		Predicted class			
		a	b	c	Total
True class	a	TP_a	FP_{a-b}	FP_{a-c}	T_a
	b	FP_{b-a}	TP_b	FP_{b-c}	T_b
	c	FP_{c-a}	FP_{c-b}	TP_c	T_c
	Total	P_a	P_b	P_c	N

Multi-class evaluation I

The number of classes is more than two

- precision, recall and f1–score are intrinsically defined for a single class;
- in binary classification they are commonly referred to the so–called positive class;
- in multi-class cases, the scorers of `scikit-learn` produce an array of values, one for each class
- when a single value is necessary to optimise the hyper parameters, as in `GridSearchCV`, if we need to maximise one of precision, recall, f1–score an average value is required

Multi-class evaluation II

The number of classes is more than two

Averaging of measure f for classes $c_i \in \mathcal{C}$, each class with frequency C_i

- macro average: $f(\mathcal{C}) = \frac{\sum f(c_i)}{C}$
- weighted average: $f(\mathcal{C}) = \frac{\sum f(c_i) * C_i}{C}$
- micro average: the definition differs for the different measures:

$$\text{precision\_micro} = \frac{\sum TP_i}{\sum (TP_i + FP_i)} \qquad \text{recall\_micro} = \frac{\sum TP_i}{\sum (TP_i + FN_i)}$$

$$\text{f1\_micro} = \frac{2 * \text{precision\_micro} * \text{recall\_micro}}{\text{precision\_micro} + \text{recall\_micro}}$$

Multi-class evaluation III

The number of classes is more than two

- with macro average the measure of each class has the same impact on the average value, therefore in case of imbalancing the minority classes have influence bigger than their size
- with wighted average the measure of each class influences the result in proportion with its size, therefore in case of imbalancing the minority classes have smaller influence
- with micro average precision and recall are computed globally across all samples, f1_micro is defined on the basis of precision_micro and recall_micro, in analogy with the binary case

Use cases for micro averaging

- When class imbalance is not a concern
- When evaluating overall system performance
- Search engines, multi-class classification, medical diagnosis (in case of discrimination among many diseases)

Use cases for macro averaging

- When all classes should be treated equally
- When class imbalance exists but all classes must contribute equally
- Sentiment analysis, rare disease detection, multi-class classification

Use cases for weighted averaging

- When class imbalance exists, and larger classes should influence the result more
- Spam filtering, product recommendation, customer churn prediction

Comparison Table

Metric	Handles Imbalance?	Emphasizes Large Classes?	Treats All Classes Equally?
*measure*_micro	No	Yes	No
*measure*_macro	Yes	No	Yes
*measure*_weighted	Yes	Yes	No

Summary on averaging multi–class performance for single–class scores (precision, recall, f1–score)

- **Use score_micro** when overall performance matters
- **Use score_macro** when class importance should be equal
- **Use score_weighted** when larger classes should have more influence

Beyond pure counting

Taking into account the "a priori" information

- Is it likely to obtain a correct prediction by chance?
- Example: early diagnosis
  - let us consider a disease affecting 2% of patients
  - a prediction saying always "no disease" has 98% precision, which, in general would be a good result
  - the evaluation of a prediction should take this as a baseline, and, possibly, look for improvements

Example of confusion matrix

- Confusion matrix of classifier $\overline{C}$ on a given dataset
- 140 correct predictions
- The predicted proportion of classes is 100:60:40

		Predicted class			
		a	b	c	Total
True class	a	88	14	18	120
	b	10	40	10	60
	c	2	6	12	20
	Total	100	60	40	200

Confusion matrix of a random classifier $R_{\overline{C}}$

A virtual experiment

- A random classifier $R_{\overline{C}}$ producing the same proportion of classes as $\overline{C}$
  - the horizontal margin is the same as $\overline{C}$
- The rows have all the same proportion as the horizontal margin
- 82 predictions are exact by chance
  - the sum of the main diagonal of $R_{\overline{C}}$

		Predicted class			
		a	b	c	Total
True class	a	60	36	24	120
	b	30	18	12	60
	c	10	6	4	20
	Total	100	60	40	200

Taking into account the random component

- The improvement of $\overline{C}$ over $R_{\overline{C}}$ is $140 - 82 = 58$
- The improvement of the perfect classifier is $200 - 82 = 118$
- We define $\kappa(\overline{C})$ the improvement of the classifier at hand w.r.t. the improvement of the perfect classifier

$$\kappa(\overline{C}) = 58/118 = 0.492$$

κ statistic [Cohen(1960)]

- Evaluates the concordance between two classifications
  - in our case between the predicted and the true one
- Fraction of concordance observed $\mathbf{Pr}(o) = \frac{TP_a + TP_b + TP_c}{N}$
- Expected fraction of concordance for a random assignment $\mathbf{Pr}(e) = \frac{T_a * P_a + T_b * P_b + T_c * P_c}{N^2}$
- κ is the ratio between the concordance exceeding the random component and the maximum surplus possible

$$-1 \leqslant \kappa = \frac{\mathbf{Pr}(o) - \mathbf{Pr}(e)}{1 - \mathbf{Pr}(e)} \leqslant 1$$

Definition of Cohen's Kappa for Binary Classification

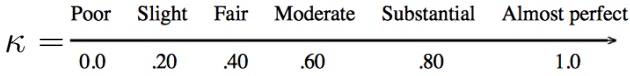
Cohen's Kappa Score can be expressed in terms of the confusion matrix components TP, FP, TN, and FN for binary classification.

Formula

$$\kappa = \frac{2 \times (TP \times TN - FP \times FN)}{(TP + FP)(FP + TN) + (TP + FN)(FN + TN)}$$

Range of κ

- 1 for perfect agreement
  - $TP_a + TP_b + TP_c = N$
- -1 for total disagreement
  - $TP_a + TP_b + TP_c = 0$ and there is a perfect swap between predictions and true labels
    - if all classes have non–zero counts -1 is possible only if the number of labels is two
- 0 for random agreement



Definition of Matthews Correlation Coefficient (MCC)

It is a metric for evaluating binary classifications, especially effective when classes are imbalanced. It measures the correlation between observed and predicted classifications.

Formula

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Properties of MCC

- **Range:** MCC values range from -1 to 1.
  - 1: Perfect prediction
  - 0: Prediction no better than random
  - -1: Complete disagreement between predictions and actuals
- **Balanced Evaluation:** MCC is a balanced metric, particularly useful for imbalanced datasets since it accounts for all components of the confusion matrix.

The cost of errors

- Our decisions are driven by predictions
- Bad predictions imply a cost
- Examples
  - grant a loan to a person who turns out to be a bad payer costs more than denying a loan to a person that could be a good payer
  - a false "oil spill" alarm is less expensive than an undetected spill
  - a wrong "fault prediction" in an industrial plant is in general less expensive than an unexpected fault disabling the plant and creating damages
  - in direct marketing, sending advertisement material without redemption is less harmful than the loss of business if a promising customer is ignored

Cost sensitive learning I

Weight the errors

- Alternative 1: alterate the proportion of classes in the supervised data, duplicating the examples for which the classification error is higher
  - In this way, the classifier will became more able to classify the classes for which the classification error cost is higher
  - This solution is useful also when the classes are imbalanced, that is the frequencies of the class labels in $\mathcal{X}$ are not equal

Cost sensitive learning II

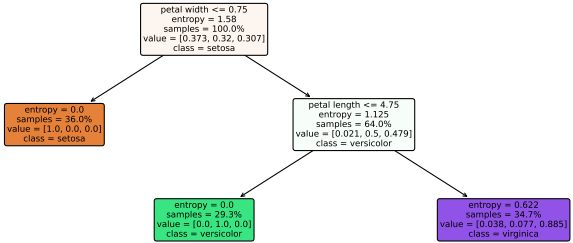
- Alternative 2: some learning schemes allow to add weights to the instances
  - e.g. the `DecisionTreeClassifier` of Scikit-Learn has the hyperparameter **class_weight**: it allows to define a dictionary, with one key per distinct class, specifying the relative weight to be assigned to each class, the optimisation of the fitting will be adjusted accordingly
    - the `balance` option balances the classes automatically

Predicting probabilities of classes I

- Many classifiers produce, rather than a class label (crisp prediction), a tuple of probabilities, one for each possible class, (probabilistic, or soft prediction)
- The adequacy of one output or the other depends on the application domain
  - when an immediate decision is required the crisp output is necessary
  - when the classification is part of a process including several evaluation/action steps the probabilistic output can be more appropriated

Crisp values sometimes hide probabilities

- For example, when a leaf of a decision tree has non–zero counts for the minority classes a less–than–one probability, the probabilities of the examples falling in that leaf can be assigned on the basis of the fractions of the training data elements in that leaf belonging to each class
- Consider the rightmost leaf in the figure (it is the pruned tree of the first module on Classification) [9]



9 Since it is quite common to have leaves with a small number of examples and/or minority classes with frequencies near to zero, smoothing techniques are used to adjust the probabilities

Probabilities to crisp

- Probabilities can be converted to a crisp value with different techniques, depending on the number of classes (binary or multiclass)
  - binary – set a threshold for the positive class
  - multiclass – output the class with the maximum probability

Binary – Lift Chart

- Used to evaluate various scenarios, depending on the application
- Consider a dataset with 1000 positives and apply a probabilistic classification scheme
- Sort all the classified elements for decreasing probability of positive class
- Make a bi-dimensional chart with axes
  x = sample size, y = number of positives in sample
- Only the rank is important, not the specific probability

Lift and Cumulative Gains Chart – I

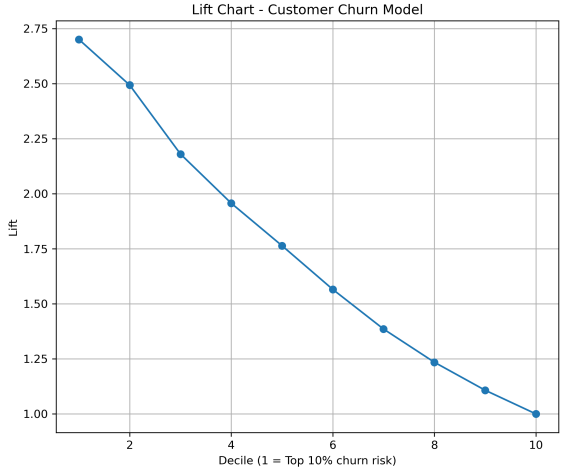
Lift Chart

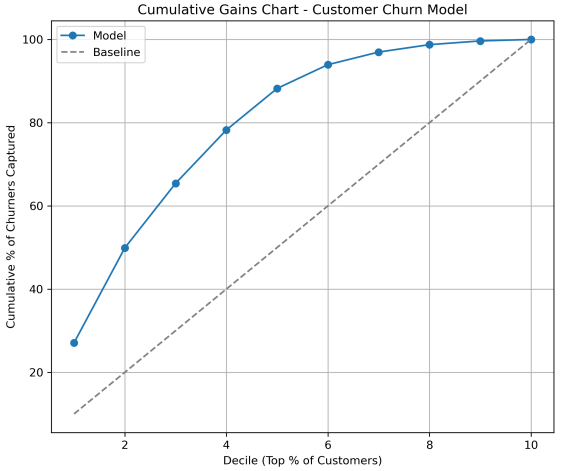
- for each fraction of data, with decreasing probability of positivity, shows a multiplication factor of the likelihood of positivity with respect to random choice

Cumulative Gains Chart

- The straight line plots the number of positives obtained with a random choice of a sample of test data
- The curve plots the number of positives obtained drawing a fraction of test data with decreasing probability
- The larger the area between the two curves, the best the classification model

Lift and Cumulative Gains Chart – II





ROC Curve

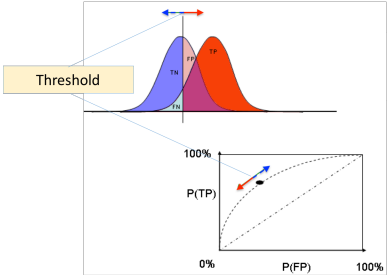
Receiver-Operator Characteristic

History: interpretation of radar signals during WWII

- Tradeoff between hit rate and false alarm in a noisy channel
- The noise can be such that the recognition of the transmission is altered
- The noise alters the two levels according to a gaussian distribution
- Problem: set the positive/negative threshold in order to maximize the tradeoff above, according to application–dependent requirements

ROC Curve

- With less noise the two gaussian curves are better separated
- Moving the threshold towards right increases both the rate of true positives and false positives caught
- The area between the non–discrimination line and the ROC curve is a quality index of the line
- The maximum area is the upper left triangle



a

a Image from Wikipedia
TN = blue + cyan = probability of a negative to be caught
FN = cyan = probability of a negative to be missed
FP = pink + purple = probability of a positive to be missed
TP = red + purple = probability of a positive to be caught

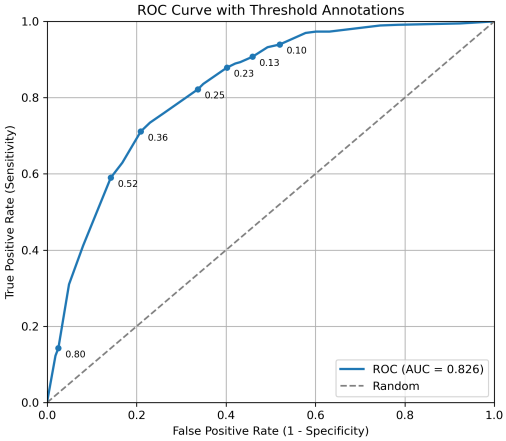
ROC for a soft classifier

- The soft classifier can be converted into a crisp one by choosing a threshold $\Rightarrow$ predict positive if the probability of the test record exceeds the threshold
- Varying the threshold the behavior of the classifier changes, by changing the ratio of TP and FP
- Threshold steps allow to track the ROC curve
  - sort the test elements by decreasing positive probability
  - set the threshold to the highest probability, set TP and FP to zero
  - repeat
    - update the number of TP and FP with probability from the threshold to 1
    - draw a point in the curve
    - move to next top probability of positive
  - end repeat

Drawing the ROC curve for a soft classifier

Sampled records, with label, predicted probability and positive rate

original_index	y	y_pred_prob	positive_rate
2797	1	0.840	1.000
6365	1	0.840	1.000
2708	1	0.795	1.000
5706	1	0.681	1.000
5716	0	0.562	0.800
1446	0	0.562	0.667
2699	0	0.521	0.571
4028	0	0.284	0.500
3762	0	0.227	0.444
1259	1	0.127	0.500
2810	0	0.104	0.455
4552	0	0.091	0.417
2525	0	0.039	0.385
4361	0	0.027	0.357
4886	0	0.027	0.333
1116	0	0.027	0.312
839	0	0.017	0.294
5756	0	0.017	0.278
405	0	0.000	0.263



Glossary and synonyms

- feature, attribute, column, predicting attribute, independent variable
  - vector of values that cooperate in the prediction of the class
- target, class, dependent variable
  - vector of supervised values, used in learning and to be predicted in classification
- hyperparameter
  - value that influences the learning process, their optimisation improves the performance of the classification

Bibliography I

- Wray Buntine.
  Learning classification trees.
  Statistics and Computing, 2(2):63–73, Jun 1992.
  ISSN 1573-1375.
  doi: 10.1007/BF01889584.
  URL https://doi.org/10.1007/BF01889584.
- Jacob Cohen.
  A coefficient of agreement for nominal scales.
  Educational and Psychological Measurement, 20(1):37–46, 1960.
  doi: 10.1177/001316446002000104.
  URL http://dx.doi.org/10.1177/001316446002000104.
- Earl B. Hunt, Janet Marin, and Philip J. Stone.
  Experiments in induction.
  Academic Press, 1966.
  URL https://books.google.it/books?id=sQoLAAAAMAAJ.

Bibliography II

- Ross Quinlan.
  Discovering rules by induction from large collections of examples.
  In Expert systems in the micro electronic age. Edimburgh University Press, 1979.
- Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus F. M. Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg.
  Top 10 algorithms in data mining.
  Knowl. Inf. Syst., 14(1):1–37, 2008.
  doi: 10.1007/s10115-007-0114-2.
  URL http://dx.doi.org/10.1007/s10115-007-0114-2.