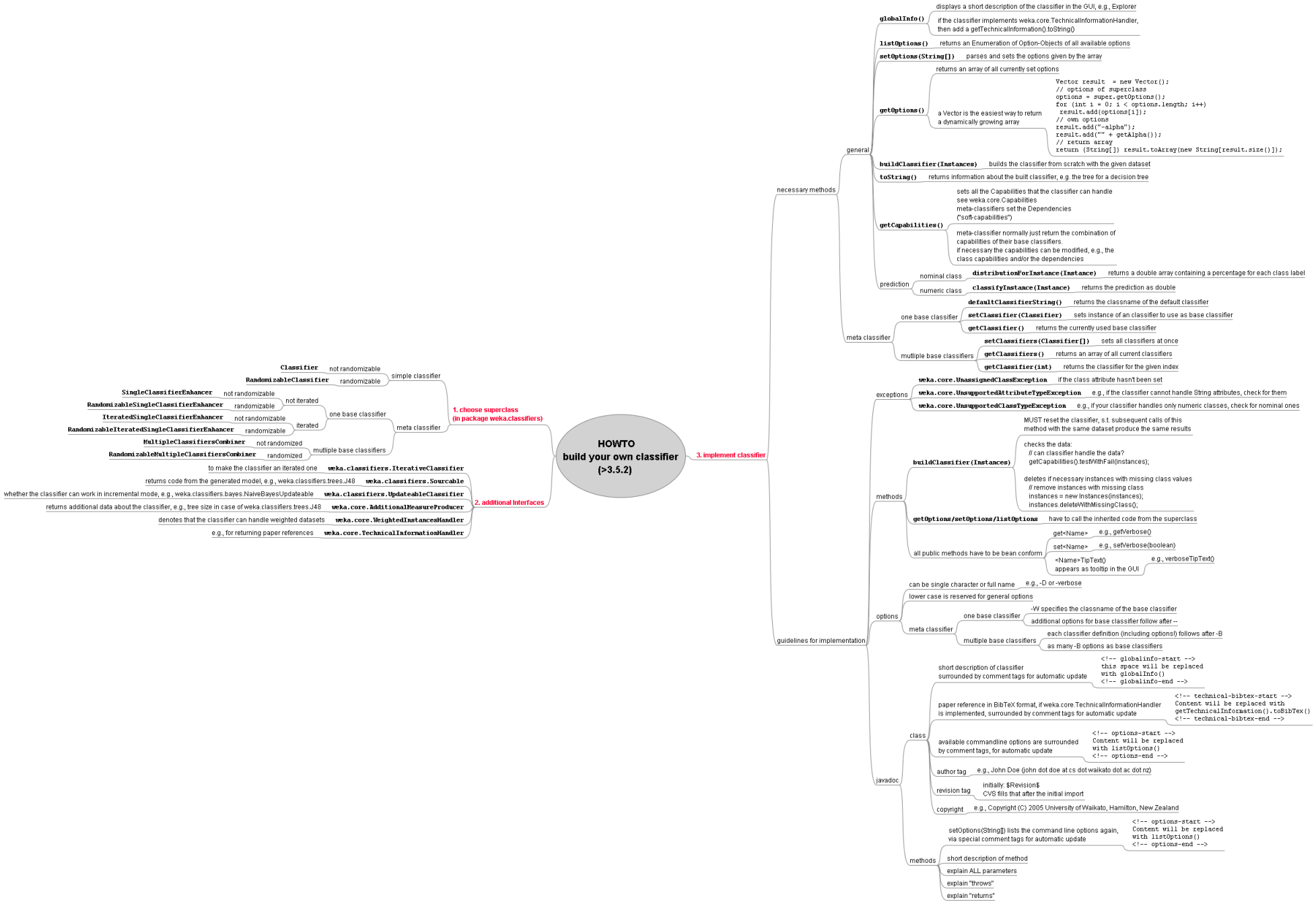
HOWTO
build your own classifier
(>3.5.2)
1. choose superclass
(in package weka.classifiers)
simple classifier
Classifier
not randomizable
RandomizableClassifier
randomizable
meta classifier
one base classifier
not iterated
SingleClassifierEnhancer
not randomizable
RandomizableSingleClassifierEnhancer
randomizable
iterated
IteratedSingleClassifierEnhancer
not randomizable
RandomizableIteratedSingleClassifierEnhancer
randomizable
mutliple base classifiers
MultipleClassifiersCombiner
not randomized
RandomizableMultipleClassifiersCombiner
randomized
2. additional Interfaces
to make the classifier an iterated one
weka.classifiers.IterativeClassifier
returns code from the generated model, e.g., weka.classifiers.trees.J48
weka.classifiers.Sourcable
whether the classifier can work in incremental mode, e.g., weka.classifiers.bayes.NaiveBayesUpdateable
weka.classifiers.UpdateableClassifier
returns additional data about the classifier, e.g., tree size in case of weka.classifiers.trees.J48
weka.core.AdditionalMeasureProducer
denotes that the classifier can handle weighted datasets
weka.core.WeightedInstancesHandler
e.g., for returning paper references
weka.core.TechnicalInformationHandler
3. implement classifier
necessary methods
general
globalInfo()
displays a short description of the classifier in the GUI, e.g., Explorer
if the classifier implements weka.core.TechnicalInformationHandler,
then add a getTechnicalInformation().toString()
listOptions()
returns an Enumeration of Option-Objects of all available options
setOptions(String[])
parses and sets the options given by the array
getOptions()
returns an array of all currently set options
a Vector is the easiest way to return
a dynamically growing array
Vector result = new Vector();
// options of superclass
options = super.getOptions();
for (int i = 0; i < options.length; i++)
result.add(options[i]);
// own options
result.add("-alpha");
result.add("" + getAlpha());
// return array
return (String[]) result.toArray(new String[result.size()]);
buildClassifier(Instances)
builds the classifier from scratch with the given dataset
toString()
returns information about the built classifier, e.g. the tree for a decision tree
getCapabilities()
sets all the Capabilities that the classifier can handle
see weka.core.Capabilities
meta-classifiers set the Dependencies
("soft-capabilities")
meta-classifier normally just return the combination of
capabilities of their base classifiers.
if necessary the capabilities can be modified, e.g., the
class capabilities and/or the dependencies
prediction
nominal class
distributionForInstance(Instance)
returns a double array containing a percentage for each class label
numeric class
classifyInstance(Instance)
returns the prediction as double
meta classifier
defaultClassifierString()
returns the classname of the default classifier
one base classifier
setClassifier(Classifier)
sets instance of an classifier to use as base classifier
getClassifier()
returns the currently used base classifier
mutliple base classifiers
setClassifiers(Classifier[])
sets all classifiers at once
getClassifiers()
returns an array of all current classifiers
getClassifier(int)
returns the classifier for the given index
guidelines for implementation
exceptions
weka.core.UnassignedClassException
if the class attribute hasn't been set
weka.core.UnsupportedAttributeTypeException
e.g., if the classifier cannot handle String attributes, check for them
weka.core.UnsupportedClassTypeException
e.g., if your classifier handles only numeric classes, check for nominal ones
methods
buildClassifier(Instances)
MUST reset the classifier, s.t. subsequent calls of this
method with the same dataset produce the same results
checks the data:
// can classifier handle the data?
getCapabilities().testWithFail(instances);
deletes if necessary instances with missing class values
// remove instances with missing class
instances = new Instances(instances);
instances.deleteWithMissingClass();
getOptions/setOptions/listOptions
have to call the inherited code from the superclass
all public methods have to be bean conform
get<Name>
e.g., getVerbose()
set<Name>
e.g., setVerbose(boolean)
<Name>TipText()
appears as tooltip in the GUI
e.g., verboseTipText()
options
can be single character or full name
e.g., -D or -verbose
lower case is reserved for general options
meta classifier
one base classifier
-W specifies the classname of the base classifier
additional options for base classifier follow after --
multiple base classifiers
each classifier definition (including options!) follows after -B
as many -B options as base classifiers
javadoc
class
short description of classifier
surrounded by comment tags for automatic update
<!-- globalinfo-start -->
this space will be replaced
with globalInfo()
<!-- globalinfo-end -->
paper reference in BibTeX format, if weka.core.TechnicalInformationHandler
is implemented, surrounded by comment tags for automatic update
<!-- technical-bibtex-start -->
Content will be replaced with
getTechnicalInformation().toBibTex()
<!-- technical-bibtex-end -->
available commandline options are surrounded
by comment tags, for automatic update
<!-- options-start -->
Content will be replaced
with listOptions()
<!-- options-end -->
author tag
e.g., John Doe (john dot doe at cs dot waikato dot ac dot nz)
revision tag
initially: $Revision$
CVS fills that after the initial import
copyright
e.g., Copyright (C) 2005 University of Waikato, Hamilton, New Zealand
methods
setOptions(String[]) lists the command line options again,
via special comment tags for automatic update
<!-- options-start -->
Content will be replaced
with listOptions()
<!-- options-end -->
short description of method
explain ALL parameters
explain "throws"
explain "returns"