Data Mining III – Lesson 6

Tamara B. Sipes, Ph.D.

Last Time: Lesson 5

- Continued with more complex data mining tasks
- Introduced to several of the commonly encountered difficulties when mining real-world datasets, such as very unclean, noisy dataset
- Focused on modeling difficult datasets
- Got more practical experience

Lesson 5: The Lesson Learned

- Real life datasets are unclean and often difficult to model
- One might spend anywhere from 60-80% of the whole mining effort cleaning and preparing the data for mining
- Sometimes you just need to wait for more data, or ask for a different protocol for collecting the data, more input variables, etc.
- Sometime it is useful to utilize the attribute selection methods
- We built up your experience and intuition

Lesson 6 Overview

- Ensemble Modeling:
  - Why and How
  - Common methods
- Hands on experience with ensemble modeling in weka:
  - Bagging
  - Boosting
  - Stacking

Instructions

- Hands-on lesson #5 and #6
- First part: Ensemble Modeling – what we need to know
- Second part: hands on (please be ready to have both the Lesson 6 and the weka Explorer running)
- Follow the step by step instructions, and perform the modeling of the dataset yourself as well
- The hands-on part will lead to Assignment III

- Let's get started!

Ensemble Modeling - Synonyms

- Building ensemble classifiers
- Combining multiple classifiers
- Meta learning schemas
- Hybrid modeling

Combining Different Methods

- Basic idea of "meta" learning schemes
  - build different "experts" and let them vote
  - Using more than one "head" is better than one
- Advantage
  - often improves predictive performance
- Disadvantage
  - produces output that is more difficult to analyze

Main Meta Schemes

- Schemes
  - Basic: Bagging, boosting, stacking
  - Many other methods
  - Weka offers 32 different meta learning methods
  - Can be applied to both classification and numeric prediction problems

Combining Classifiers

- Given
  - Training data set *D* for supervised learning
  - Collection of inductive learning algorithms
  - Return: new prediction algorithm that combines outputs from collection of prediction algorithms
- Desired Properties
  - Guarantees of performance of combined prediction
    - ability to improve weak classifiers

Mixtures of Experts

- What Is A Weak Classifier?
  - One not guaranteed to do better than random guessing
  - Goal
    - combine multiple weak classifiers
    - get one at least as accurate as strongest
- Mixtures of Experts
  - “experts” express hypotheses
    - drawn from a hypothesis space

Bootstrap Aggregating or Bagging

- Employs simplest way of combining predictions:
  - voting/averaging
- Each model receives equal weight
- "Idealized" version of bagging
  - Sample several training sets of size n
    - instead of just having one training set of size n
  - Build a classifier for each training set
  - Combine the classifier's predictions

Bagging

- "Two heads are better than one"
- Produce multiple classifiers from one data set
- Bagging
  - Create *k* bootstrap samples *S*[1], *S*[2], …, *S*[*k*]
  - Train distinct model on each *S*[*i*] to produce *k* classifiers
  - Classify new instance by classifier vote (equal weights)

More on bagging

- Bagging reduces variance by voting/averaging
  - reducing the overall expected error
  - In the case of classification there are pathological situations where the overall error might increase
  - Usually the more classifiers the better
- Problem
  - What if we only have one dataset?
- Solution
  - generate new datasets of size n by sampling with replacement from original dataset
- Can be very helpful if data is noisy

Bagging classifiers

model generation

```
Let n be the number of instances in the training data.
For each of t iterations:
  Sample n instances with replacement from training set.
  Apply the learning algorithm to the sample.
  Store the resulting model.
```

classification

```
For each of the t models:
  Predict class of instance using model.
Return class that has been predicted most often.
```

When Should This Help?

- When learner is unstable
  - Small change to training set causes large change in output hypothesis
  - True for decision trees, neural networks; not true for *k*-nearest neighbor
- Experimentally, bagging can help substantially for unstable learners
  - Can possibly somewhat degrade results for stable learners

In Weka: meta.bagging

Class for bagging a classifier to reduce variance. Can do classification and regression depending on the base learner.

OPTIONS

- bagSizePercent -- Size of each bag, as a percentage of the training set size.
- calcOutOfBag -- Whether the out-of-bag error is calculated.
- classifier -- The base classifier to be used.
- debug -- If set to true, classifier may output additional info to the console.
- numIterations -- The number of iterations to be performed.
- seed -- The random number seed to be used.

Boosting

- Also uses voting/averaging but models are weighted according to their performance
- Iterative procedure: new models are influenced by performance of previously built ones
  - New model is encouraged to become expert for instances classified incorrectly by earlier models
  - Intuitive justification - models should be experts that complement each other
- There are several variants of this algorithm

Boosting

- Can be applied without weights using resampling with probability determined by the weights
  - Disadvantage: not all instances are used
  - Advantage: resampling can be repeated if error exceeds 0.5
- Emerged from computational learning theory
- Theoretical result
  - training error decreases exponentially
- Works if base classifiers not too complex and their error doesn't become too large too quickly

Boosting

- Boosting often produces classifiers that are significantly more accurate on fresh data than bagging
  - But sometimes fails in practical situations
    - It can generate a classifier that is significantly less accurate than a single classifier build from the same data
      - Combined classifier overfits the data
- Boosting works with weak learners
  - only condition is that error doesn't exceed 0.5
- LogitBoost: more sophisticated boosting scheme

In Weka: meta.AdaBoostM1

Class for boosting a nominal class classifier using the Adaboost M1 method. Only nominal class problems can be tackled. Often dramatically improves performance, but sometimes overfits.

- OPTIONS
- classifier -- The base classifier to be used.
- debug -- If set to true, classifier may output additional info to the console.
- numIterations -- The number of iterations to be performed.
- seed -- The random number seed to be used.
- useResampling -- Whether resampling is used instead of reweighting.
- weightThreshold -- Weight threshold for weight pruning.

In Weka: meta.LogitBoost

Class for performing additive logistic regression. Performs classification using a regression scheme as the base learner, and can handle multi-class problems.

OPTIONS

- classifier -- The base classifier to be used.
- debug -- If set to true, classifier may output additional info to the console.
- likelihoodThreshold -- Threshold on improvement in likelihood.
- numFolds -- Number of folds for internal cross-validation
- numIterations -- The number of iterations to be performed.
- numRuns -- Number of runs for internal cross-validation.
- seed -- The random number seed to be used.
- shrinkage -- Shrinkage parameter (use small value like 0.1 to reduce overfitting)
- useResampling -- Whether resampling is used instead of reweighting.
- weightThreshold -- Weight threshold for weight pruning (reduce to 90 for speeding up learning process).

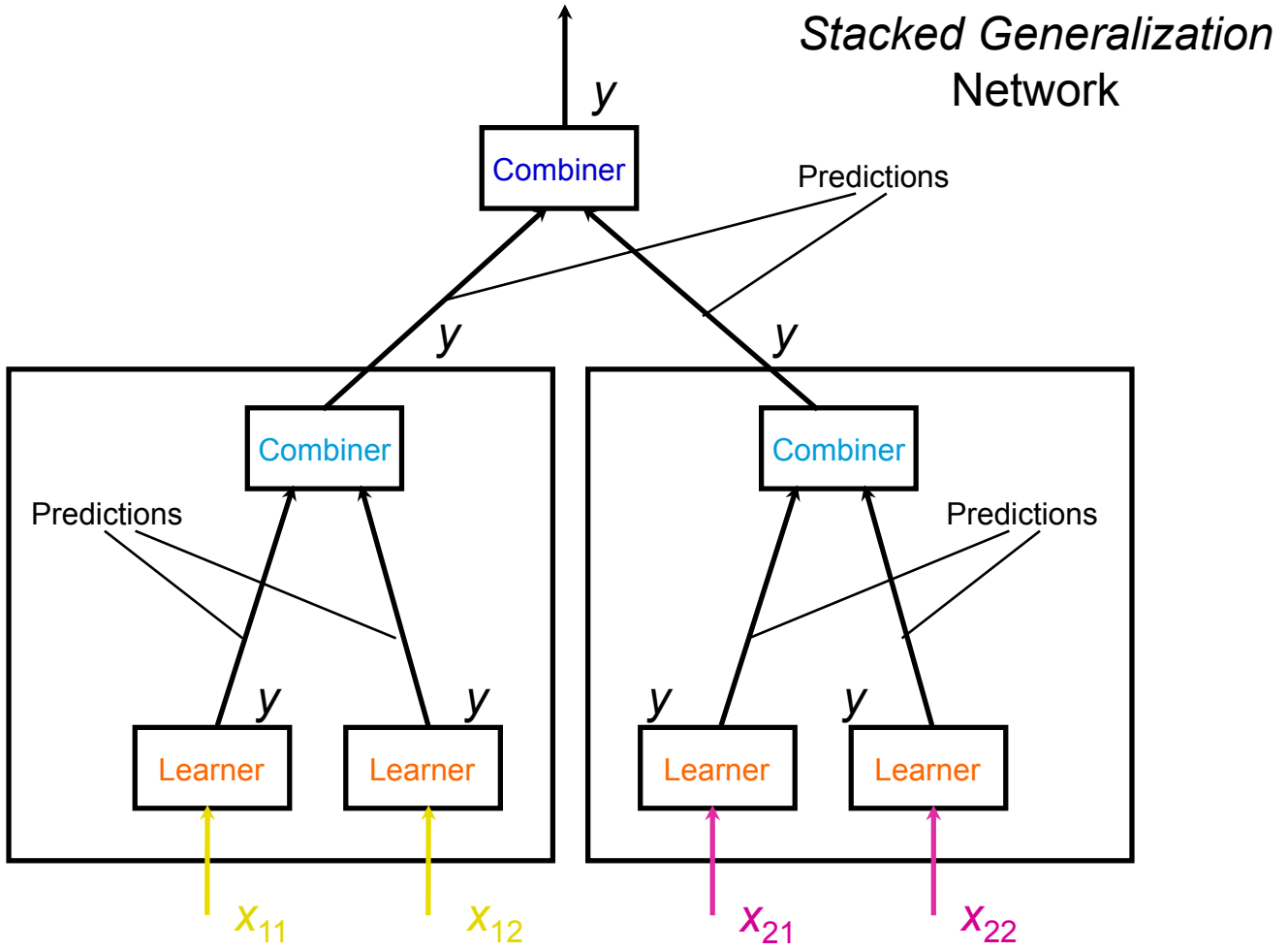
Stacked Generalization

- Stacked Generalization - Stacking
- Intuitive Idea
  - Train multiple learners
    - Each uses subsample of *D*
    - May be ANN, decision tree, etc.
  - Train 'combiner' on validation segment

Stacked Generalization Stacking

- Used to combine models of different types
  - DT, Naïve Bayes and k-nearest neighbor
- With several algorithms available, instead of performing cross validation and selecting the best one it is better to combine them
- Use un-weighted voting (as in bagging)?
  - makes sense only if the learning algorithms perform comparably well
  - it is not clear which classifier to trust
- Stacking uses a meta learner instead of voting – the goal of the meta learner is to learn which classifiers are the reliable ones

Stacking


Stacked Generalization
Network
y
Combiner
Predictions
y
y
Combiner
Combiner
Predictions
Predictions
y
y
y
y
Learner
Learner
Learner
Learner
x_{11}
x_{12}
x_{21}
x_{22}

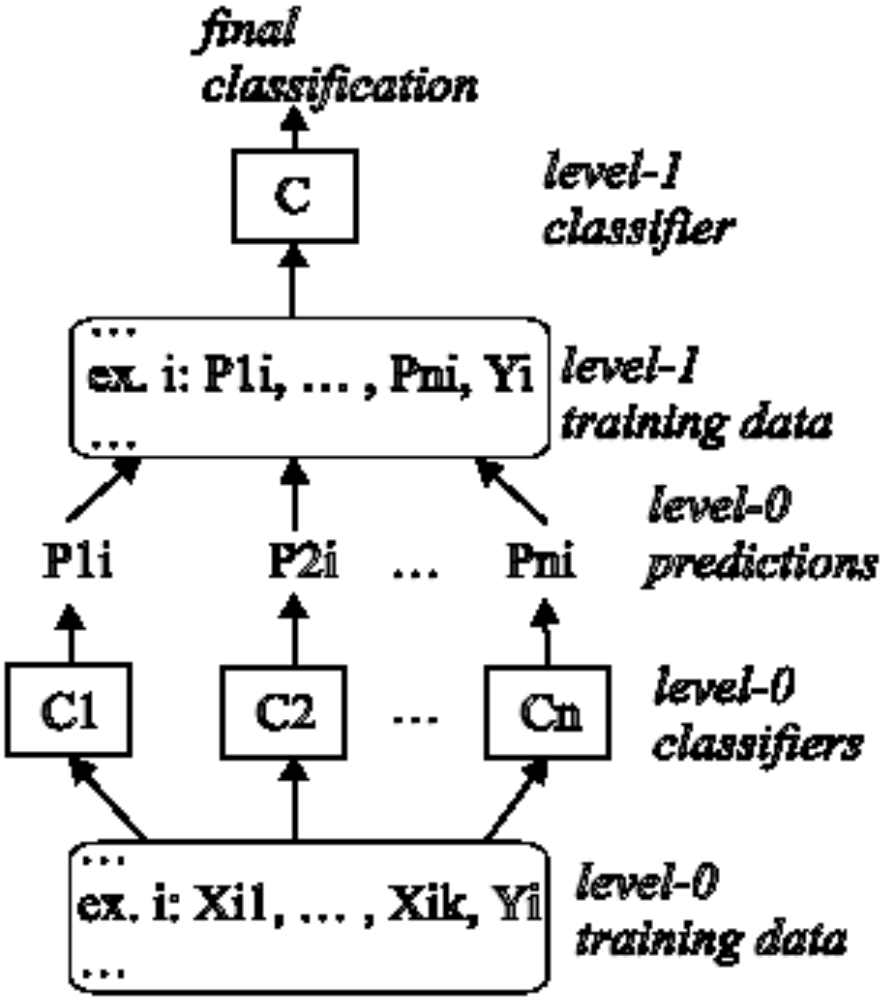
Stacking

- Hard to analyze theoretically - “black magic”
- Uses meta learner instead of voting to combine predictions of base learners
  - Predictions of base learners (level-0 models) are used as input for meta learner (level-1 model)
- Base learners usually different learning schemes
- Predictions on training data can’t be used to generate data for level-1 model!
  - Cross-validation-like scheme is employed

Combining Level-0 and Level-1 Models

- Meta learner is called *level–1 model*
- Base classifiers - *level-0 models*
- Outputs of the level-0 models are inputs to the level-1 model
- How to train the level-1 model?
- Idea 1: let each level-0 model classify a training instance, and attach to their predictions the actual classification as additional attribute, to form a training instance for the level-1 model
- Doesn't work well in practice
  - prefers classifiers overfitting the training data

Stacking


final classification
C
level-1 classifier
ex. i: P1i, … , Pni, Yi
level-1 training data
level-0 predictions
P1i
P2i
…
Pni
C1
C2
…
Cn
level-0 classifiers
ex. i: Xi1, … , Xik, Yi
level-0 training data

Combining Level-0 and Level-1 Models

- Idea 2: separate the training data into training and validation set
- use the training data to train the level-0 classifiers
- apply the validation set to these classifiers and use the predictions to build training data for the level-1 model e
  - Even better: instead of training and validation set use cross validation - slow but allows the level-0 models to make full use of the data
  - Learning schemes output probabilities for each class label – use them instead of the single categorical prediction
- To classify a new example, the level-0 models are used to produce a vector which is an input to the level-1 model producing the final classification

Stacking

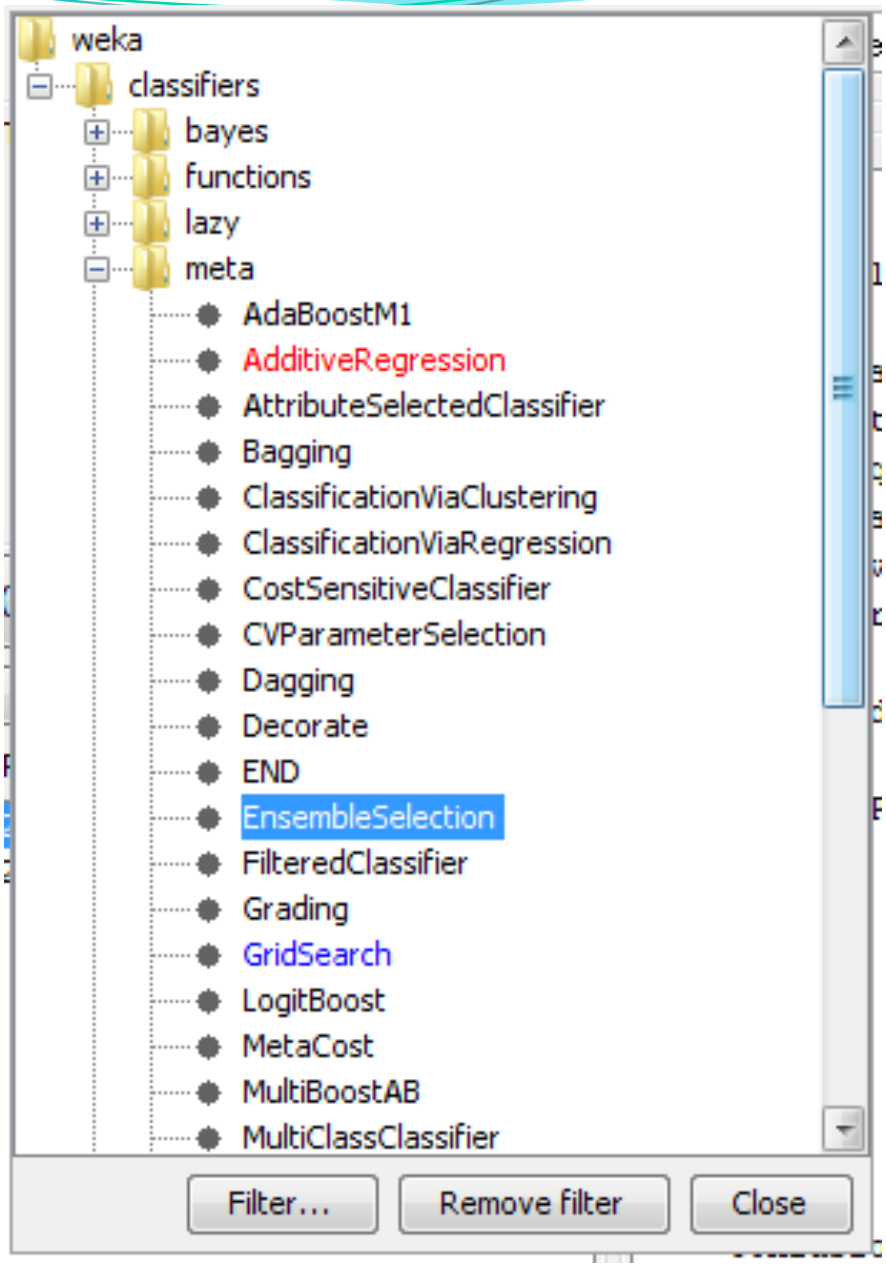
- If base learners can output probabilities it's better to use those as input to meta learner
- Which algorithm to use to generate meta learner?
  - In principle, any learning scheme can be applied
  - "relatively global, smooth" model
  - Base learners do most of the work
  - Reduces risk of overfitting
- Stacking can also be applied to numeric prediction

In Weka: meta.Stacking

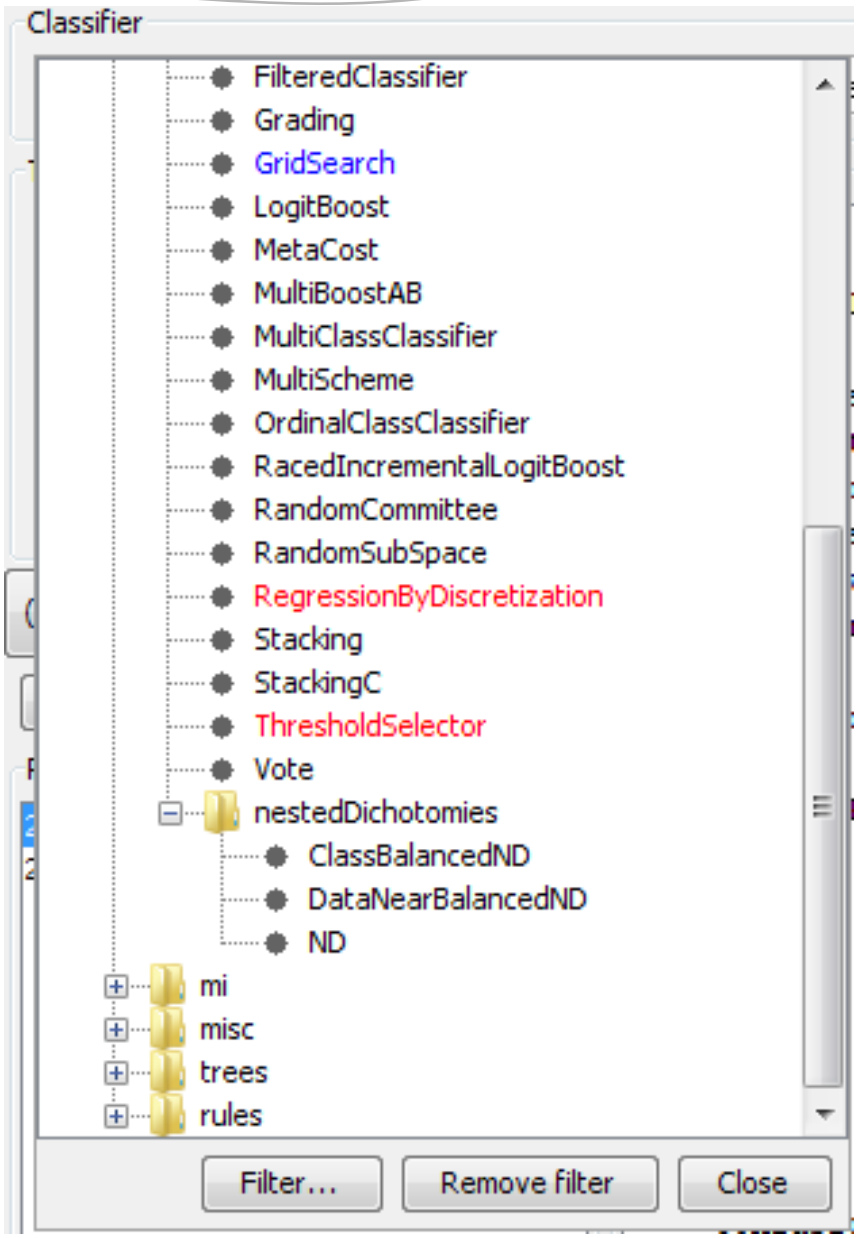
Combines several classifiers using the stacking method. Can do classification or regression.

- OPTIONS
- classifiers -- The base classifiers to be used.
- debug -- If set to true, classifier may output additional info to the console.
- metaClassifier -- The meta classifiers to be used.
- numFolds -- The number of folds used for cross-validation.
- seed -- The random number seed to be used.

Other Meta Methods



Continued


Classifier
FilteredClassifier
Grading
GridSearch
LogitBoost
MetaCost
MultiBoostAB
MultiClassClassifier
MultiScheme
OrdinalClassClassifier
RacedIncrementalLogitBoost
RandomCommittee
RandomSubSpace
RegressionByDiscretization
Stacking
StackingC
ThresholdSelector
Vote
nestedDichotomies
ClassBalancedND
DataNearBalancedND
ND
mi
misc
trees
rules
Filter...
Remove filter
Close

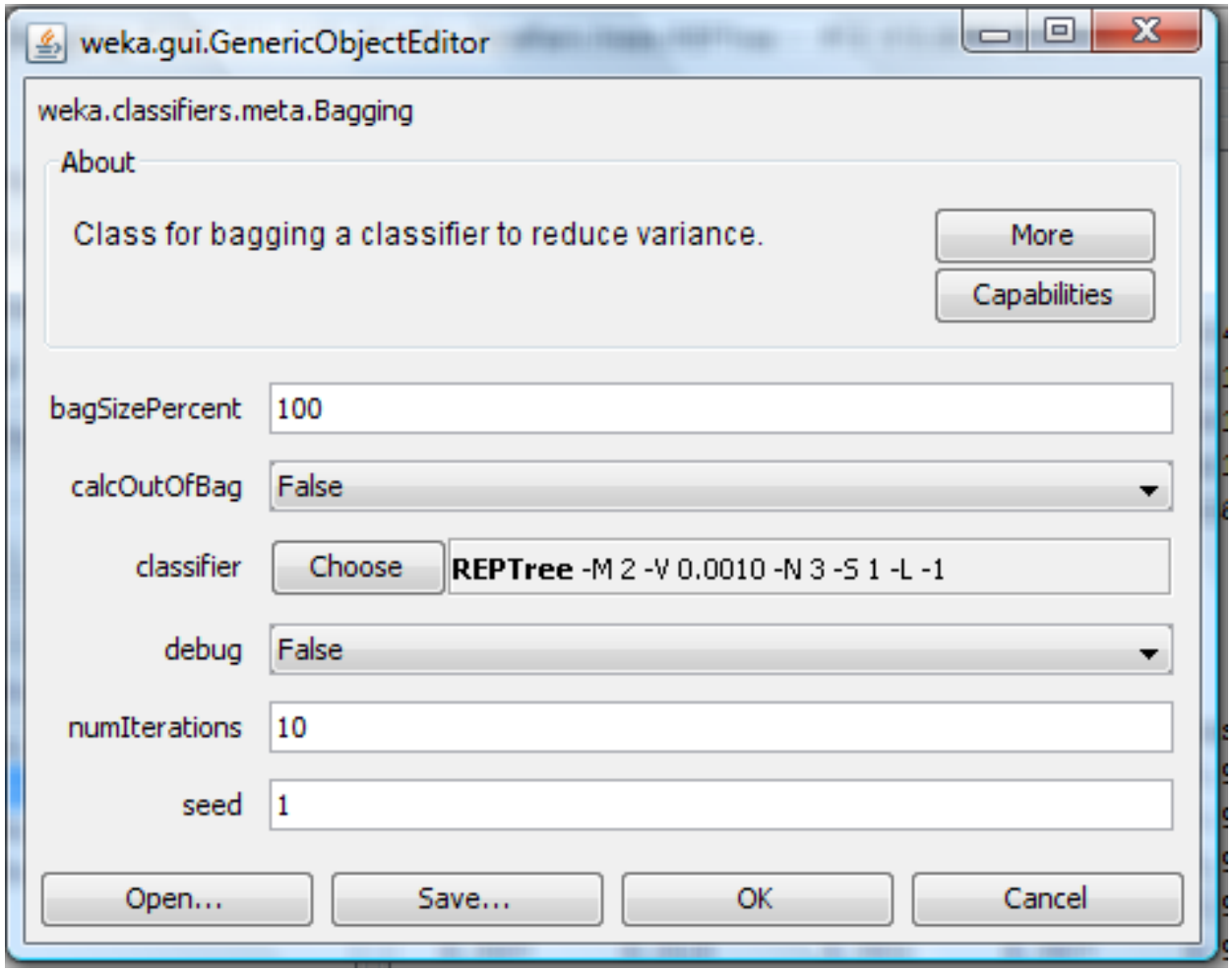
Hands-On Meta Learners

- Open IMAGE_AssignmentII.arff
- TO DO:
  - Open weka 3.5.7 or similar
  - Choose Explorer from the Applications menu
  - “Open file” IMAGE_AssignmentII.arff
- Run RepresentationsTree method:
  - Size of the tree : 37
  - Correctly Classified Instances 1414 94.2667 %
  - Incorrectly Classified Instances 86 5.7333 %

RepTree Model

```
region-centroid-row < 155.5
|   rawred-mean < 27.28
|   |   hue-mean < -1.89
|   |   |   hue-mean < -2.21
|   |   |   |   region-centroid-row < 146.5 : foliage (72/1) [38/1]
|   |   |   |   region-centroid-row >= 146.5 : cement (2/0) [2/1]
|   |   |   hue-mean >= -2.21
|   |   |   |   rawred-mean < 2.61
|   |   |   |   |   hue-mean < -2.09
|   |   |   |   |   |   region-centroid-row < 132.5 : foliage (33/0) [18/1]
|   |   |   |   |   |   region-centroid-row >= 132.5
|   |   |   |   |   |   |   region-centroid-col < 128 : foliage (21/7) [7/1]
|   |   |   |   |   |   |   region-centroid-col >= 128 : window (11/2) [5/0]
|   |   |   |   |   hue-mean >= -2.09 : window (29/1) [9/0]
|   |   |   |   rawred-mean >= 2.61...
```

Bagging


weka.gui.GenericObjectEditor
weka.classifiers.meta.Bagging
About
Class for bagging a classifier to reduce variance.
More
Capabilities
bagSizePercent
100
calcOutOfBag
False
classifier
Choose
REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1
debug
False
numIterations
10
seed
1
Open...
Save...
OK
Cancel

Bagging Results

- Size of the tree : 45

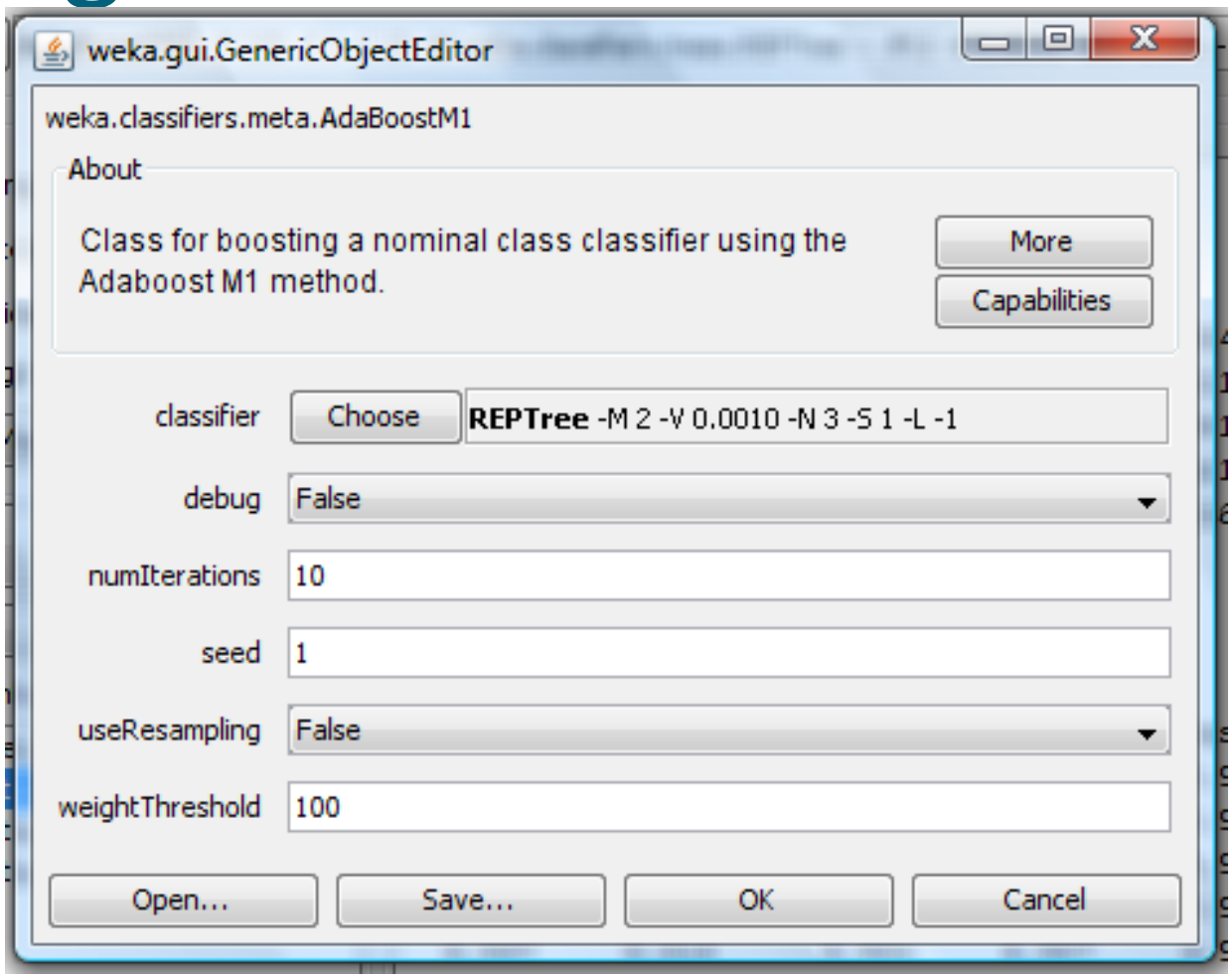
- === Stratified cross-validation ===

Correctly Classified Instances	1442	96.1333 %
Incorrectly Classified Instances	58	3.8667 %

Bagging Model

```
region-centroid-row < 155.5
|   rawgreen-mean < 22.28
|   |   hue-mean < -2.01
|   |   |   exgreen-mean < -5.94
|   |   |   |   region-centroid-row < 119.5
|   |   |   |   |   hue-mean < -2.18
|   |   |   |   |   |   region-centroid-col < 144 : foliage (3/0) [3/0]
|   |   |   |   |   |   region-centroid-col >= 144 : window (2/0) [1/0]
|   |   |   |   |   hue-mean >= -2.18 : window (52/0) [24/1]
|   |   |   |   region-centroid-row >= 119.5
|   |   |   |   |   saturation-mean < 0.5 : cement (18/4) [4/0]
|   |   |   |   |   saturation-mean >= 0.5...
```

Boosting


weka.gui.GenericObjectEditor
weka.classifiers.meta.AdaBoostM1
About
Class for boosting a nominal class classifier using the Adaboost M1 method.
More
Capabilities
classifier
Choose
REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1
debug
False
numIterations
10
seed
1
useResampling
False
weightThreshold
100
Open...
Save...
OK
Cancel

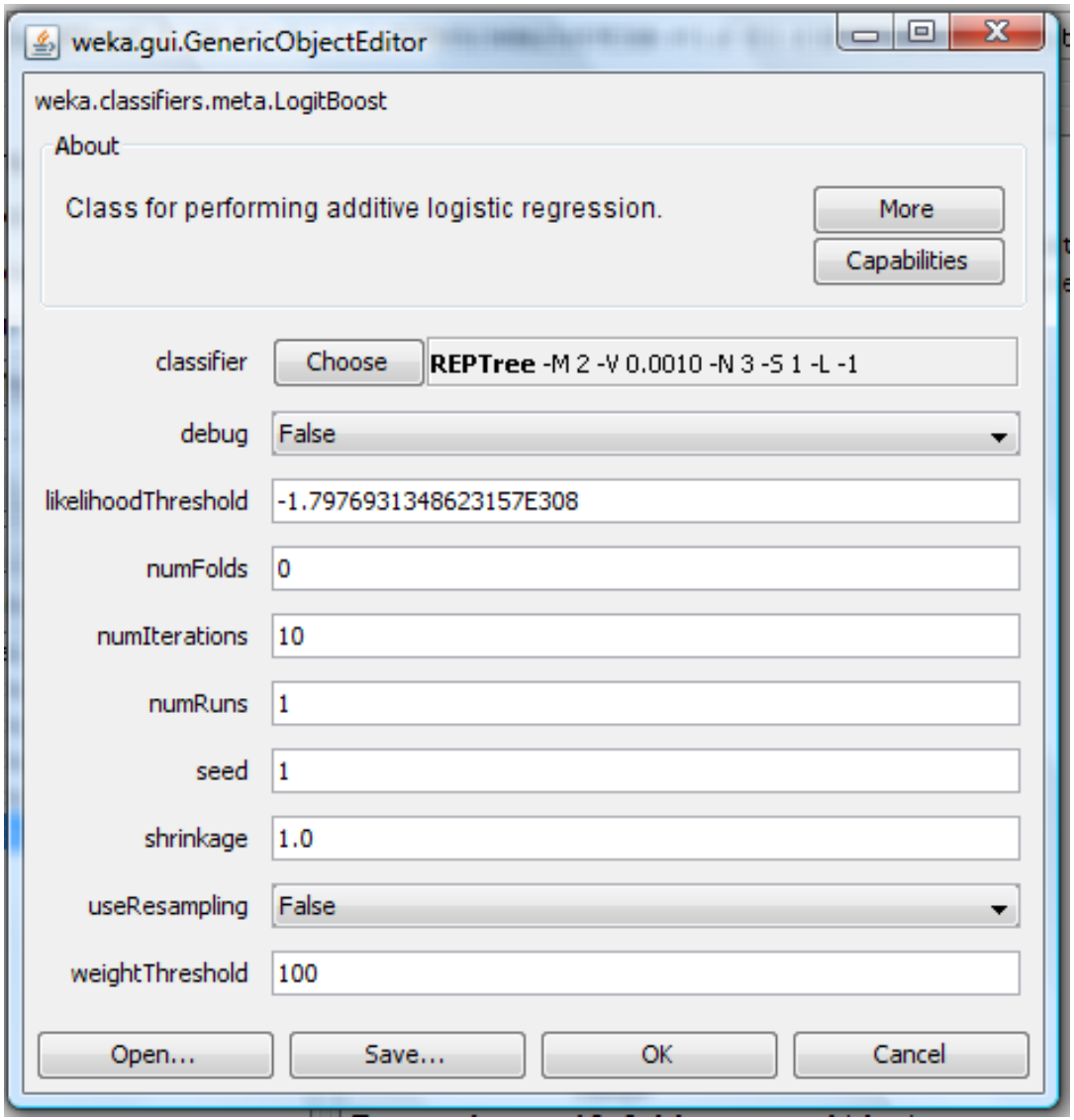
Boosting Results

- Size of the tree : 49
- Weight: 0.4
- Number of performed Iterations: 10
- === Stratified cross-validation ===
- Correctly Classified Instances 1439 95.9333 %
- Incorrectly Classified Instances 61 4.0667 %

Boosting Model

```
value-mean < 101.11
|   exgreen-mean < -5.5
|   |   saturation-mean < 0.35
|   |   |   region-centroid-row < 159.5
|   |   |   |   hue-mean < -2.15
|   |   |   |   |   exred-mean < -14.61 : foliage (2.21/0.89) [76.36/0.44]
|   |   |   |   |   exred-mean >= -14.61 : window (5.08/0) [0.69/0]
|   |   |   |   hue-mean >= -2.15 : cement (47.28/0) [9.97/0]
|   |   |   region-centroid-row >= 159.5 : path (13.75/0) [7.87/0]
|   |   saturation-mean >= 0.35
|   |   |   region-centroid-row < 122.5
|   |   |   |   hue-mean < -1.96
|   |   |   |   |   hedge-sd < 1.8
...
```

LogitBoost


weka.gui.GenericObjectEditor
weka.classifiers.meta.LogitBoost
About
Class for performing additive logistic regression.
More
Capabilities
classifier
Choose
REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1
debug
False
likelihoodThreshold
-1.7976931348623157E308
numFolds
0
numIterations
10
numRuns
1
seed
1
shrinkage
1.0
useResampling
False
weightThreshold
100
Open...
Save...
OK
Cancel

LogitBoost Results

- === Stratified cross-validation ===

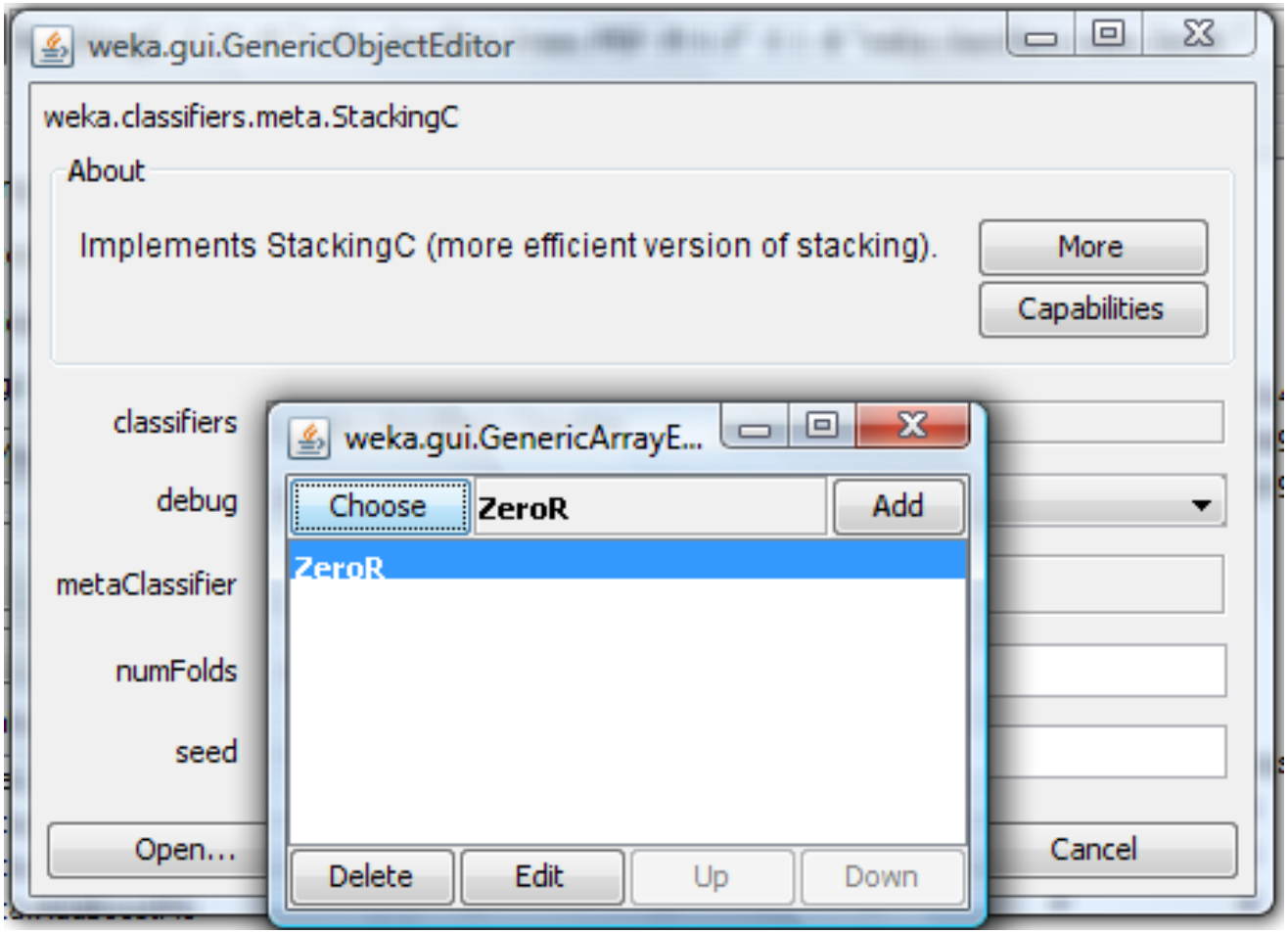
Correctly Classified Instances	1452	96.8	%
Incorrectly Classified Instances	48	3.2	%

LogitBoost Model

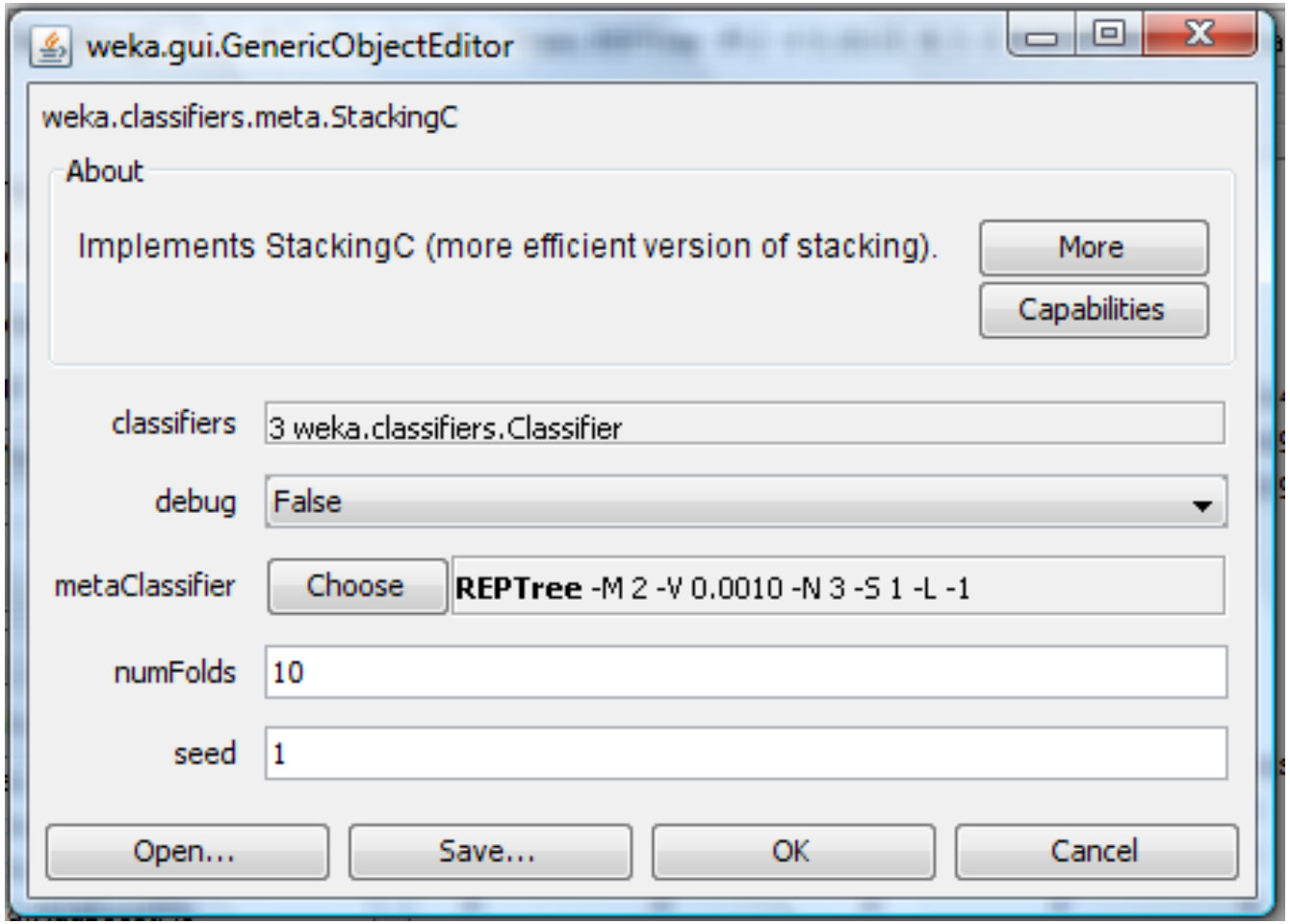
- Not easy to read:
- 10 iterations
- 7 models (one for each class distribution value) per each iteration

Stacking

- Click on classifiers
- Delete ZeroR
- Add j48, RepTree, j48graft



Stacking Parameters



Stacking Results

• Correctly Classified Instances	1438	95.8667 %
• Incorrectly Classified Instances	62	4.1333 %

Stacking Model

- 3 (three classifiers) x 7 (seven different class labels) different models

Analysis

- Using the 10-fold cross validation, the best performing model is:

LogitBoost model

With 96.8 % correctly classified instances

Assignment III

- If you choose so, you may work on the problem 1 and problem 2 of your Assignment III now,

or

- Complete Assignment III at the end of this lecture

New Dataset

- Open breast-cancer-wisconsin.arff (in class Resources)
- This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison (from Dr. William H. Wolberg)

- Number of Instances: 699 (as of 15 July 1992)
- Number of Attributes: 10 plus the class attribute

Description

- Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.
- They describe characteristics of the cell nuclei present in the image.

Attributes

- 1. Sample code number id number
- 2. Clump Thickness 1 - 10
- 3. Uniformity of Cell Size 1 - 10
- 4. Uniformity of Cell Shape 1 - 10
- 5. Marginal Adhesion 1 - 10
- 6. Single Epithelial Cell Size 1 - 10
- 7. Bare Nuclei 1 - 10
- 8. Bland Chromatin 1 - 10
- 9. Normal Nucleoli 1 - 10
- 10. Mitoses 1 - 10 11.
- Class: (2 for benign, 4 for malignant)

Data History

- Samples arrive periodically as Dr. Wolberg reports his clinical cases.
- The database therefore reflects this chronological grouping of the data.

- Group 1: 367 instances (January 1989)
- Group 2: 70 instances (October 1989)
- Group 3: 31 instances (February 1990)
- Group 4: 17 instances (April 1990)
- Group 5: 48 instances (August 1990)
- Group 6: 49 instances (Updated January 1991)
- Group 7: 31 instances (June 1991)
- Group 8: 86 instances (November 1991)
- ----------
- Total: 699 points (as of the donated database on 15 July 1992)

Data Quality

- Missing attribute values: 16
- There are 16 instances in Groups 1 to 6 that contain a single missing (i.e., unavailable) attribute value, denoted by "?".

- Class distribution:
- Benign: 458 (65.5%)
- Malignant: 241 (34.5%)

Problem Defition

- Model which features that are computed from a digitized image of a fine needle aspirate of a breast mass are more likely to describe benign vs. malignant cell nuclei
- Predict CLASS

Data Preparation

- Clean dataset
- ID not suitable for mining, as it only has unique values
- Remove ID
- Make class nominal:
  - weka.filters.unsupervised.attribute.NumericToNominal -R last

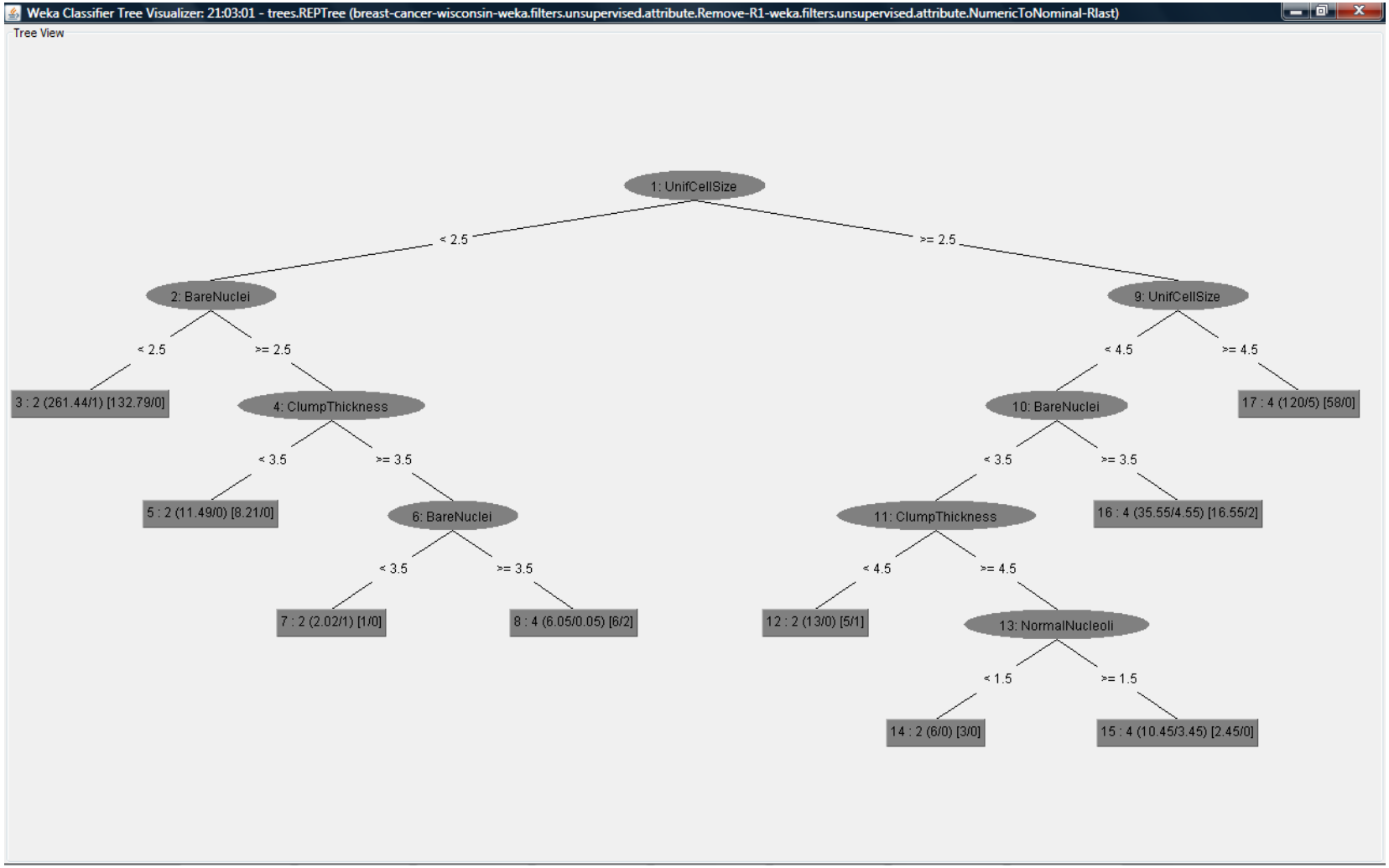
Modeling

- Let's start with:

- RepTree: default parameters:
  weka.classifiers.trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1

- Decision Tree: default parameters:
  weka.classifiers.trees.J48 -C 0.25 -M 2

The RepTree Model



Model Details

```
UnifCellSize < 2.5
|   BareNuclei < 2.5 : 2 (261.44/1) [132.79/0]
|   BareNuclei >= 2.5
|   |   ClumpThickness < 3.5 : 2 (11.49/0) [8.21/0]
|   |   ClumpThickness >= 3.5
|   |   |   BareNuclei < 3.5 : 2 (2.02/1) [1/0]
|   |   |   BareNuclei >= 3.5 : 4 (6.05/0.05) [6/2]
UnifCellSize >= 2.5
|   UnifCellSize < 4.5
|   |   BareNuclei < 3.5
|   |   |   ClumpThickness < 4.5 : 2 (13/0) [5/1]
|   |   |   ClumpThickness >= 4.5
|   |   |   |   NormalNucleoli < 1.5 : 2 (6/0) [3/0]
|   |   |   |   NormalNucleoli >= 1.5 : 4 (10.45/3.45) [2.45/0]
|   |   BareNuclei >= 3.5 : 4 (35.55/4.55) [16.55/2]
|   UnifCellSize >= 4.5 : 4 (120/5) [58/0]
```

RepTree Evaluation

=== Stratified cross-validation ===

Correctly Classified Instances	656	93.8484 %
Incorrectly Classified Instances	43	6.1516 %
Total Number of Instances	699	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.941	0.066	0.964	0.941	0.952	0.968	2
0.934	0.059	0.893	0.934	0.913	0.968	4

=== Confusion Matrix ===

a	b	<-- classified as
431	27	a = 2
16	225	b = 4

Decision Tree

```
UnifCellSize <= 2
|   BareNuclei <= 3: 2
|   BareNuclei > 3
|   |   ClumpThickness <= 3: 2
|   |   ClumpThickness > 3
|   |   |   BlandChromatin <= 2
|   |   |   |   MarginalAdhesion <= 3: 4
|   |   |   |   MarginalAdhesion > 3: 2
|   |   |   BlandChromatin > 2: 4
```

```
UnifCellSize > 2
|   UnifCellShape <= 2
|   |   ClumpThickness <= 5: 2
|   |   ClumpThickness > 5: 4
|   UnifCellShape > 2
|   |   UnifCellSize <= 4
|   |   |   BareNuclei <= 2
|   |   |   |   MarginalAdhesion <= 3: 2
|   |   |   |   MarginalAdhesion > 3: 4
|   |   |   BareNuclei > 2
|   |   |   |   ClumpThickness <= 6
|   |   |   |   |   UnifCellSize <= 3: 4
|   |   |   |   |   UnifCellSize > 3
|   |   |   |   |   |   MarginalAdhesion <= 5: 2
|   |   |   |   |   |   MarginalAdhesion > 5: 4
|   |   |   |   ClumpThickness > 6: 4
|   |   UnifCellSize > 4: 4
```

Decision Tree Evaluation

=== Stratified cross-validation ===

Correctly Classified Instances	661	94.5637 %
Incorrectly Classified Instances	38	5.4363 %

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.956	0.075	0.961	0.956	0.958	0.955	2
0.925	0.044	0.918	0.925	0.921	0.955	4

=== Confusion Matrix ===

a	b	<-- classified as
438	20	a = 2
18	223	b = 4

Meta Classifiers - Bagging

- weka.classifiers.meta.Bagging -P 100 -S 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.0010 -N 3 -S 1 -L -1 → 95.5651% correctly classified instances

- weka.classifiers.meta.Bagging -P 100 -S 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2 → 95.8512%

Meta Classifiers - Boosting

- weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.REPTree -- -M 2 -V 0.0010 -N 3 -S 1 -L -1 → 93.9914%

- weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2 → 95.7082%

Meta Classifiers - Staking

- weka.classifiers.meta.StackingC -X 10 -M "weka.classifiers.functions.LinearRegression -S 1 -R 1.0E-8" -S 1 -B "weka.classifiers.trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1" -B "weka.classifiers.trees.J48 -C 0.25 -M 2“ → 93.8484%

- weka.classifiers.meta.StackingC -X 10 -M "weka.classifiers.trees.M5P -M 4.0" -S 1 -B "weka.classifiers.trees.REPTree -M 2 -V 0.0010 -N 3 -S 1 -L -1" -B "weka.classifiers.trees.J48 -C 0.25 -M 2“ → 93.2761%

Analysis

- Best performer:
- Bagged Decision Tree model with 95.85% correctly classified instances
- Many more options (paths in our search) to explore!

Summary

- Combining Classifiers
  - Problem definition and motivation: improving accuracy in concept learning
  - General framework: collection of weak classifiers to be improved

Summary

- Bootstrap Aggregating (Bagging)
  - Voting system for collection of algorithms
  - Training set for each member: sampled with replacement
  - Works for unstable inducers
- Stacked Generalization (*aka* Stacking)
  - Hierarchical system for combining inducers (ANNs or other inducers)
  - Training sets for "leaves": sampled with replacement; combiner: validation set

Summary

- Bagging and boosting use the same method of aggregating different models together - voting
- Bagging, boosting and stacking can be applied to both classification and numeric prediction
- Bagging and boosting combine models of the same type, stacking – of different types

Conclusion

- Real life datasets are unclean and often difficult to model
- Sometimes you just need to wait for more data, or ask for a different protocol for collecting the data, more input variables, etc.
- Sometimes using a meta learner can help solve the problem

Next

- Complete Assignment III
- Next is Lesson 7: More on Practical Weka, Summary, Conclusions, Tips and Other Helpful Guidance