User Guide for Auto-WEKA version 2.5

Lars Kotthoff, Chris Thornton, Frank Hutter
{larsko,cwthornt,hutter}@cs.ubc.ca

April 6, 2017

Contents

1 Introduction

Auto-WEKA is a tool that performs combined algorithm selection and hyperparameter optimisation over the classification and regression algorithms implements in WEKA. More specifically, given a specific dataset, Auto-WEKA explores hyperparameter settings for many algorithms and recommends to a user which method will likely have good generalisation performance, using model based optimisation techniques.

1.1 Availability

Auto-WEKA is available as a WEKA package through the WEKA package manager (WEKA version 3.7.13 and later). The source code is available at `https://github.com/automl/autoweka`, where bugs can be reported as well.

1.2 License

Auto-WEKA is open source software issued under the GNU General Public License. Note that the included SMAC optimisation method is licensed under the AGPLv3 license.

1.3 Requirements

Auto-WEKA does not have any additional requirements compared to WEKA. If you can run WEKA, you should be able to run Auto-WEKA.

2 User Documentation

Auto-WEKA is used much like any other WEKA classifier. After loading a dataset into WEKA, it can be run on it to automatically determine the best WEKA model and its parameters. It does so by intelligently exploring the space of classifiers and parameters using the SMAC tool (`http://www.cs.ubc.ca/labs/beta/Projects/SMAC/`).

A full list of all classifiers and attribute selectors considered along with their hyperparameters can be found in the appendix.

2.1 Using the GUI

There are two different ways of using Auto-WEKA through the WEKA GUI. The easiest way is to use the Auto-WEKA panel, which allows you to run Auto-WEKA directly on a loaded dataset. Figure 1 shows a screenshot of a completed run.

Alternatively, Auto-WEKA can be run through the normal "Classify" panel by selecting it from the list of classifiers (Figure 2).

When using Auto-WEKA like a normal classifier, it is important to select the Test option "Use training set". Auto-WEKA performs a statistically rigorous evaluation internally (10 fold cross-validation) and does not require the external split into training and test sets that WEKA provides. Selecting another option
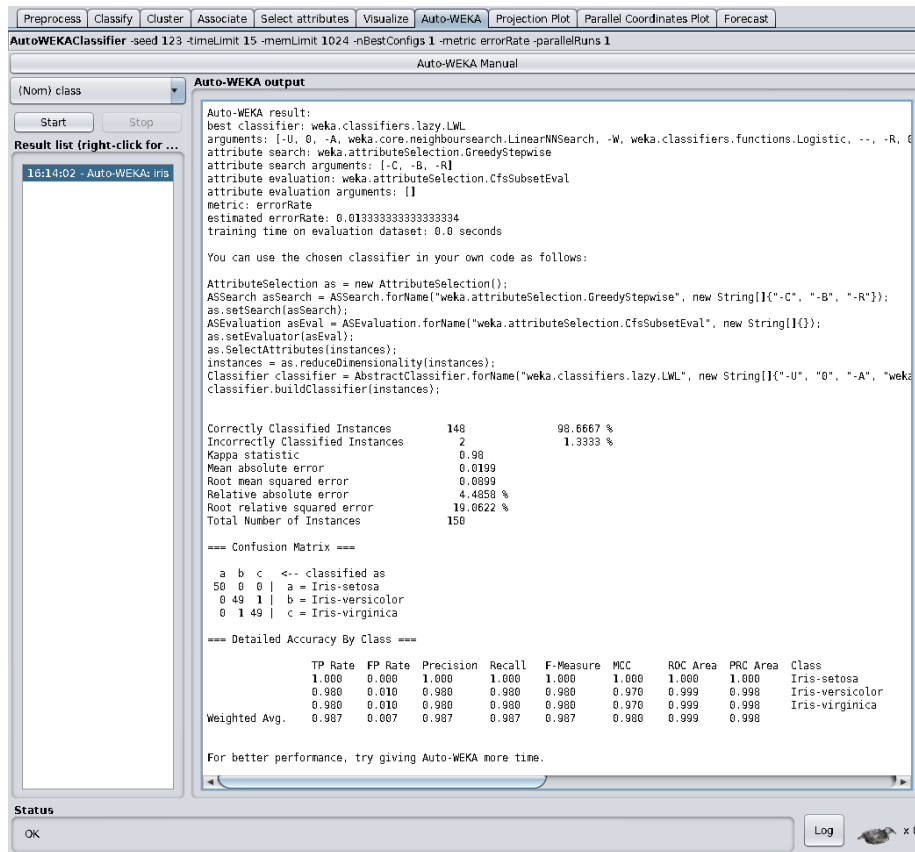
Figure 1: Auto-WEKA tab showing the output of a completed run.

will not improve the quality of the result and cause Auto-WEKA to take much longer. Figure 3 shows the recommended setting.

Auto-WEKA has only a few options. Figure 4 shows them. Usually, you can leave them at their default values. For most users, only two options are relevant:

timeLimit The time in minutes Auto-WEKA will take to determine the best classifier and configuration. If you get bad results, try increasing this value.

memLimit The memory limit in megabytes for running classifiers. If you have a very large dataset, you may need to increase this value.

While Auto-WEKA is running, it will provide the number of configurations evaluated so far and the estimated error of the best configuration in the status bar.

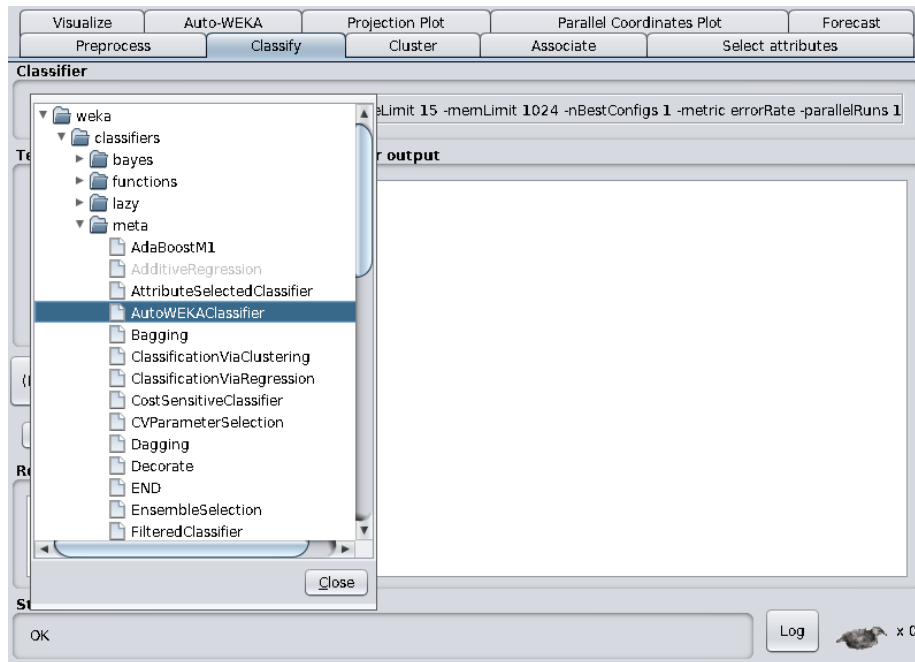Note that the time limit is *approximate* and Auto-WEKA may not take *exactly*

Figure 2: Location of the Auto-WEKA classifier in the list of classifiers.

as long as requested.

2.2 Running Experiments Using the Command Line Interface (CLI)

Auto-WEKA can be run from the CLI like any other WEKA classifier; for example:

```
java -cp autoweka.jar weka.classifiers.meta.AutoWEKAClassifier \
    -t iris.arff -timeLimit 15 -no-cv
```

Note that we specify the flag `-no-cv` to prevent WEKA from splitting the data into training and test sets in addition to what Auto-WEKA does internally.

Running Auto-WEKA with the `-h` flag or without any options lists the options that can be set by the user. Note that the general WEKA options appear first, followed by Auto-WEKA-specific options.

```
java -cp autoweka.jar weka.classifiers.meta.AutoWEKAClassifier -h
```
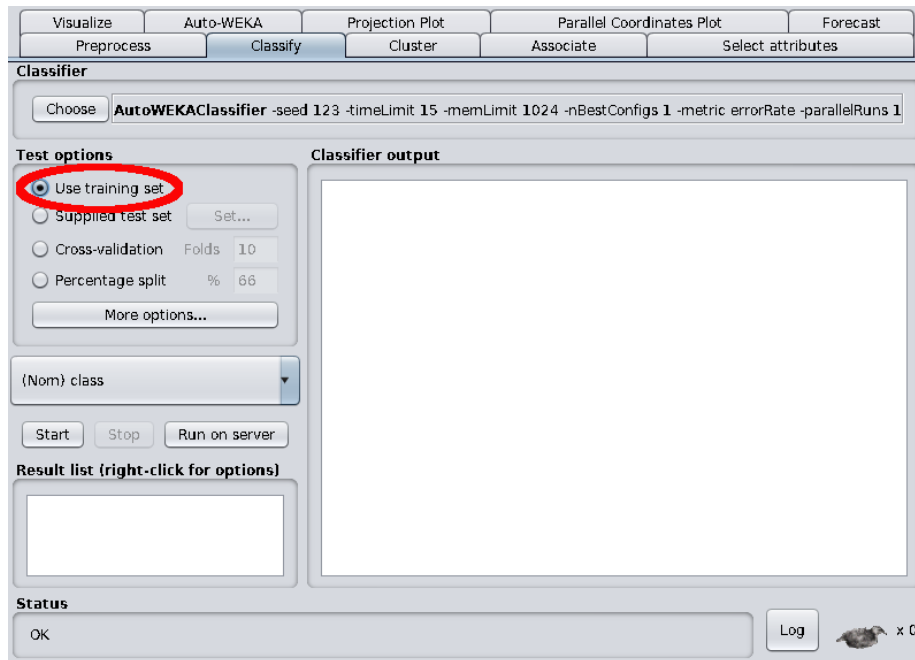
```
Help requested.
```

Figure 3: Recommended evaluation setting when using Auto-WEKA like a normal classifier.

```
General options:

-h or -help
Output help information.
-synopsis or -info
Output synopsis for classifier (use in conjunction  with -h)
-t <name of training file>
Sets training file.
-T <name of test file>
Sets test file. If missing, a cross-validation will be performed
on the training data.
-c <class index>
Sets index of class attribute (default: last).
-x <number of folds>
Sets number of folds for cross-validation (default: 10).
-no-cv
Do not perform any cross validation.
-force-batch-training
Always train classifier in batch mode, never incrementally.
-split-percentage <percentage>
```
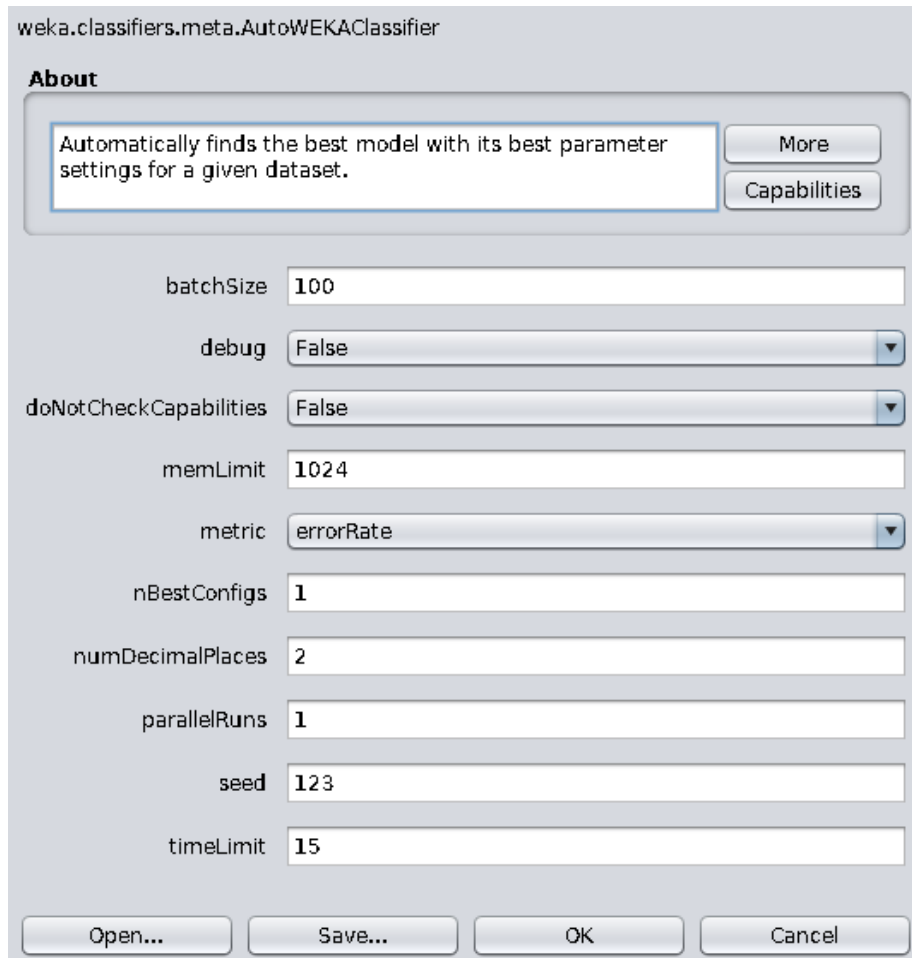
Figure 4: Auto-WEKA options.

```
Sets the percentage for the train/test set split, e.g., 66.
-preserve-order
Preserves the order in the percentage split.
-s <random number seed>
Sets random number seed for cross-validation or percentage split
(default: 1).
-m <name of file with cost matrix>
Sets file with cost matrix.
-toggle <comma-separated list of evaluation metric names>
Comma separated list of metric names to toggle in the output.
All metrics are output by default with the exception of 'Coverage' and 'Region size'.
Available metrics:
```

```
Correct,Incorrect,Kappa,Total cost,Average cost,KB relative,KB information,
Correlation,Complexity 0,Complexity scheme,Complexity improvement,
MAE,RMSE,RAE,RRSE,Coverage,Region size,TP rate,FP rate,Precision,Recall,
F-measure,MCC,ROC area,PRC area
-l <name of input file>
Sets model input file. In case the filename ends with '.xml',
a PMML file is loaded or, if that fails, options are loaded
from the XML file.
-d <name of output file>
Sets model output file. In case the filename ends with '.xml',
only the options are saved to the XML file, not the model.
-v
Outputs no statistics for training data.
-o
Outputs statistics only, not the classifier.
-do-not-output-per-class-statistics
Do not output statistics for each class.
-k
Outputs information-theoretic statistics.
-classifications "weka.classifiers.evaluation.output.prediction.AbstractOutput + options"
Uses the specified class for generating the classification output.
E.g.: weka.classifiers.evaluation.output.prediction.PlainText
-p range
Outputs predictions for test instances (or the train instances if
no test instances provided and -no-cv is used), along with the
attributes in the specified range (and nothing else).
Use '-p 0' if no attributes are desired.
Deprecated: use "-classifications ..." instead.
-distribution
Outputs the distribution instead of only the prediction
in conjunction with the '-p' option (only nominal classes).
Deprecated: use "-classifications ..." instead.
-r
Only outputs cumulative margin distribution.
-xml filename | xml-string
Retrieves the options from the XML-data instead of the command line.
-threshold-file <file>
The file to save the threshold data to.
The format is determined by the extensions, e.g., '.arff' for ARFF
format or '.csv' for CSV.
-threshold-label <label>
The class label to determine the threshold data for
(default is the first label)
-no-predictions
Turns off the collection of predictions in order to conserve memory.
```

```
Options specific to weka.classifiers.meta.AutoWEKAClassifier:

-seed <seed>
The seed for the random number generator.
(default: 123)
-timeLimit <limit>
The time limit for tuning in minutes (approximately).
(default: 15)
-memLimit <limit>
The memory limit for runs in MiB.
(default: 1024)
-nBestConfigs <limit>
The amount of best configurations to output.
(default: 1)
-metric <metric>
The metric to optimise.
(default: errorRate)
-parallelRuns <runs>
The number of parallel runs. EXPERIMENTAL.
(default: 1)
-output-debug-info
If set, classifier is run in debug mode and
may output additional info to the console
-do-not-check-capabilities
If set, classifier capabilities are not checked before classifier is built
(use with caution).
-num-decimal-places
The number of decimal places for the output of numbers in the model (default 2).
-batch-size
The desired batch size for batch prediction  (default 100).
```

Auto-WEKA internally uses the value 10^{100} to denote infinity, used for the worst possible metric result. In specific cases, the value does not need to be this high; for example when optimizing accuracy for a classification problem. The value can be changed through the `autoweka.infinity` system property:

```
java -Dautoweka.infinity=2 -cp autoweka.jar \
  weka.classifiers.meta.AutoWEKAClassifier -t iris.arff \
  -timeLimit 15 -no-cv
```

2.3   Saving and Loading Models

Once Auto-WEKA has been run, you can save the trained model to get predictions on test data. In the WEKA GUI, right-click on a run in the output list window and select 'Save model'. On the CLI, use the -d flag. More information on saving and loading models can be found on the WEKA wiki at

`https://weka.wikispaces.com/Saving+and+loading+models`.

The Auto-WEKA output also contains the Java code for creating an instance of the classifier and attribute selection method that it found. This allows to retrain the model on different data.

3 Developer Documentation

Auto-WEKA's internal structure is complex and comprises more than 10,000 lines of code. A detailed explanation is beyond the scope of this document; more details can be found in the Javadoc, available at `https://automl.github.io/autoweka/`. A high-level overview of the internal structure of Auto-WEKA is shown in Figure 5.
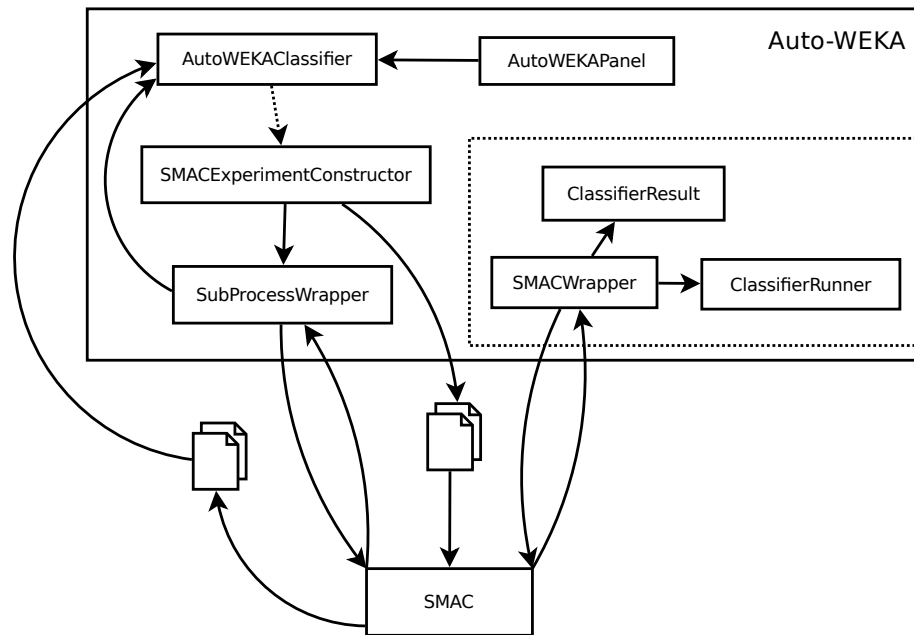
Figure 5: High-level overview of Auto-WEKA internal structure.

The user interface of Auto-WEKA is defined in the `AutoWEKAClassifier` and `AutoWEKAPanel` classes, where the latter defines the elements of the custom Auto-WEKA panel and calls `AutoWEKAClassifier`. New user options and additional arguments should be specified there.

The SMAC optimization tool is called through the `SMACExperimentConstructor` and `SubProcessWrapper` classes. The former prepares a set of files that contain the data, the definition of the parameter space, the training budget, and other

information necessary for SMAC to run. These files are created in the system's temporary directory in a folder named `autoweka<random number>`. The `SubProcessWrapper` calls SMAC as an external process and points it to these files.

SMAC then calls Auto-WEKA's `SMACWrapper` class during the optimization process to evaluate the performance of a classifier and its parameters. The `SMACWrapper` trains and evaluates the classifier with given parameters with the `ClassifierRunner` class, which creates a `ClassifierResult`. This information is passed back to SMAC.

After SMAC's optimization completes, control is passed back to Auto-WEKA, which takes the information communicated directly from SMAC through its output and parses result files that SMAC generated. This information is used to determine the best classifier and its parameters, which is finally presented to the user.

4   Known Issues

In order to use Auto-WEKA through the WEKA GUI on OSX, WEKA must be run from the command line instead of the Mac application. For example,

```
cd /Applications/weka-3-8-1-oracle-jvm.app/Contents/Java/
java -jar weka.jar
```

Note that Java needs to be installed on your system for this to work. This is because Auto-WEKA needs a `java` executable to spawn worker processes, and the JRE bundled with the Mac application does not contain one.

A   Auto-WEKA Configuration Space

A.1   Classifiers, Parameters, and Parameter Ranges

Classifier	Parameter	Value Range	Default
J48	O	true, false	false
	U	true, false	false
	B	true, false	false
	J	true, false	false
	A	true, false	false
	S	true, false	false
	M	1, 64	2

<div align="right">continued...</div>

Classifier	Parameter	Value Range	Default
	C	0, 1	25
DecisionTable	E	acc, rmse, mae, auc	acc
	I	true, false	false
	S	BestFirst, GreedyStepwise	BestFirst
	X	1, 2, 3, 4	1
GaussianProcesses	L	0001, 1	1
	N	0, 1, 2	0
	K	NormalizedPolyKernel, PolyKernel, Puk, RBFKernel	NormalizedPolyKernel
	E	2, 5	0
	L	true, false	false
	E	2, 5	0
	L	true, false	false
	S	1, 10	0
	O	1, 1	0
	C	0001, 1	01
M5P	N	true, false	false
	M	1, 64	4
	U	true, false	false
	R	true, false	false
KStar	B	1, 100	20
	E	true, false	false
	M	a, d, m, n	a
LMT	B	true, false	false
	R	true, false	false
	C	true, false	false
	P	true, false	false
	M	1, 64	15
	W	0	0
	W	0, 1	0
	A	true, false	false
PART	N	2, 5	3
	M	1, 64	2
	R	true, false	false
	B	true, false	false
SMO	C	5, 5	0
	N	0, 1, 2	0
	M	true, false	false

Classifier	Parameter	Value Range	Default
	K	NormalizedPolyKernel, PolyKernel, Puk, RBFKernel	NormalizedPolyKernel
	E	2, 5	0
	L	true, false	false
	E	2, 5	0
	L	true, false	false
	S	1, 10	0
	O	1, 1	0
	G	0001, 1	01
BayesNet	D	true, false	false
	Q	K2, HillClimber, LAGDHillClimber, SimulatedAnnealing, TabuSearch, TAN	K2
NaiveBayes	K	true, false	false
	D	true, false	false
JRip	N	1, 5	0
	E	true, false	false
	P	true, false	false
	O	1, 5	2
SimpleLogistic	S	true, false	false
	W	0	0
	W	0, 1	0
	A	true, false	false
LinearRegression	S	0, 1, 2	0
	C	true, false	false
	R	1e-7, 10	1e-7
VotedPerceptron	I	1, 10	1
	M	5000, 50000	10000
	E	2, 5	0
SGD	F	0, 1, 2	0
	L	00001, 1	01
	R	1e-12, 10	1e-4
	N	true, false	false
	M	true, false	false
Logistic	R	1e-12, 10	1e-7
OneR	B	1, 32	6

Classifier	Parameter	Value Range	Default
MultilayerPerceptron	L	1, 1	3
	M	1, 1	2
	B	true, false	false
	H	a, i, o, t	a
	C	true, false	false
	R	true, false	false
	D	true, false	false
	S	1	1
REPTree	M	1, 64	2
	V	1e-5, 1e-1	1e-3
	L	-1	-1
	L	2, 20	2
	P	true, false	false
IBk	E	true, false	false
	K	1, 64	1
	X	true, false	false
	F	true, false	false
	I	true, false	false
M5Rules	N	true, false	false
	M	1, 64	4
	U	true, false	false
	R	true, false	false
RandomForest	I	2, 256	10
	K	0	0
	K	1, 32	2
	depth	0	0
	depth	1, 20	2
RandomTree	M	1, 64	1
	K	0	0
	K	2, 32	2
	depth	0	0
	depth	2, 20	2
	N	0	0
	N	2, 5	3
	U	true, false	false
SMOreg	C	5, 5	0
	N	0, 1, 2	0
	I	RegSMOImproved	RegSMOImproved
	V	true, false	false

Classifier	Parameter	Value Range	Default
	K	NormalizedPolyKernel, PolyKernel, Puk, RBFKernel	NormalizedPolyKernel
	E	2, 5	0
	L	true, false	false
	E	2, 5	0
	L	true, false	false
	S	1, 10	0
	O	1, 1	0
	G	0001, 1	01
Vote	R	AVG, PROD, MAJ, MIN, MAX	AVG
	S	1	1
Stacking	X	10	10
	S	1	1
Bagging	P	10, 100	100
	I	2, 128	10
	S	1	1
	O	true, false	false
	P	100	100
	P	50, 100	100
	I	2, 128	10
	Q	true, false	false
	S	1	1
	S	1	1
	S	0, 0	1
	I	2, 128	10
	K	-1, 10, 30, 60, 90, 120	-1
	U	0, 1, 2, 3, 4	0
	A	LinearNNSearch	LinearNNSearch
RandomSubSpace	I	2, 64	10
	P	1, 0	5
	S	1	1
AttributeSelectedClassifier	S	BestFirst, GreedyStepwise	BestFirst
	E	CfsSubsetEval	CfsSubsetEval
RandomCommittee	I	2, 64	10
	S	1	1

Classifier	Parameter	Value Range	Default

A.2 Attribute Searches, Parameters, and Parameter Ranges

Attribute Search	Parameter	Value Range	Default
BestFirst	D	0, 1, 2	1
	N	2, 10	5
GreedyStepwise	C	true, false	false
	B	true, false	false
	R	true, false	false
	N	10, 1000	30

A.3 Attribute Evaluations, Parameters, and Parameter Ranges

Attribute Evaluation	Parameter	Value Range	Default
CfsSubsetEval	M	true, false	false
	L	true, false	false