Assignment 3: Using meta learning schemes with a strong and a weak learner for classification

Hector Lopez CAP6673 FAU SPRING 2018

Introduction

A class of meta learning schemes, or ensemble methods, can be used to increase accuracy and decrease misclassification rates. This is accomplished simply by combining several general classifiers and interpreting the results, thus getting an overall best, composite classifier from this pool of classifiers. Assignment 3 uses models from previous assignments employing cost sensitive classification with two meta learning schemes: bagging and boosting.

Models and Results: Part 1

There were several models created to help evaluate the benefits of using meta learning schemes. Each of the models was created by using 10-fold cross-validation on the training set, used an adjusted cost ratio to reduce overall Type II misclassification (keeping Type I error set to 1), and was evaluated and validated on a test dataset. Weka was used for all the runs assuming most of the defaults given by the software package.

Hypothesis

Combining classifiers is more beneficial for some models. In general, a strong classifier will see less meaningful gains in accuracy than a weak classifier. We saw that the decision stump was a weak classifier in previous assignments. So, in this assignment, the decision stump learning algorithm should see the biggest gain with J48 showing minimal, if any, improvement.

Simulations

For each model run with all classifiers, the FNR, lower left corner of the cost matrix, is adjusted leaving the FPR set to one. The evaluation results for the required models, both training and test, are shown in Figure 1 and Figure 2 leaving the bagging or boosting iterations set to 10. This means that that model is using 10 training datasets.

1. Cost sensitive classifier combined with bagging and J48
2. Cost sensitive classifier combined with bagging and Decision Stump
3. Cost sensitive classifier combined with boosting (AdaBoostM1) and J48
4. Cost sensitive classifier combined with boosting (AdaBoostM1) and Decision Stump

(See Appendix for Weka Command Line Inputs)

		TRAINING DATASET, 10-FOLD CROSS-VALIDATION WITH COST RATIO ADJUSTMENTS 10 ITERATIONS								TEST DATASET WITH CHOSEN COST RATIO 10 ITERATIONS	TEST DATASET WITH CHOSEN COST RATIO (ALT) 10 ITERATIONS	TEST DATASET WITH CHOSEN COST RATIO (ALT) 10 ITERATIONS
BAGGING & J48	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.35**		
	FNR	0.527	0.255	0.236	0.200	0.164	0.127	0.109	0.109	*0.143*		
	FPR	0.038	0.098	0.098	0.128	0.113	0.143	0.189	0.256	*0.136*		
	RMSE	0.366	0.323	0.319	0.319	0.319	0.316	0.339	0.392	*0.323*		
BAGGING	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.6**	**0.8**	**1**

	FNR	0.873	0.182	0.145	0.109	0.073	0.073	0.055	0.018	*0.036*	*0.179*	*0.214*
	FPR	0.008	0.165	0.188	0.218	0.218	0.226	0.233	0.263	*0.270*	*0.212*	*0.212*
	RMSE	0.393	0.332	0.336	0.340	0.348	0.368	0.383	0.411	*0.357*	*0.328*	*0.335*
BOOSTING & J48	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.2**	**0.1**	**0.5**
	FNR	0.236	0.291	0.218	0.255	0.182	0.218	0.127	0.127	*0.357*	*0.036*	*0.179*
	FPR	0.098	0.098	0.120	0.090	0.120	0.113	0.113	0.113	*0.061*	*0.258*	*0.212*
	RMSE	0.359	0.380	0.372	0.353	0.361	0.376	0.341	0.343	*0.386*	*0.394*	*0.341*
BOOSTING & DECISIONSTUMP	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.5**	**0.6**	**0.8**
	FNR	0.455	0.218	0.164	0.182	0.164	0.109	0.127	0.091	*0.179*	*0.036*	*0.179*
	FPR	0.038	0.158	0.173	0.180	0.173	0.218	0.218	0.218	*0.212*	*0.227*	*0.212*
	RMSE	0.358	0.351	0.349	0.355	0.345	0.365	0.374	0.388	*0.341*	*0.335*	*0.328*

Figure 1 – Evaluation table for 10-iteration meta learning schemes

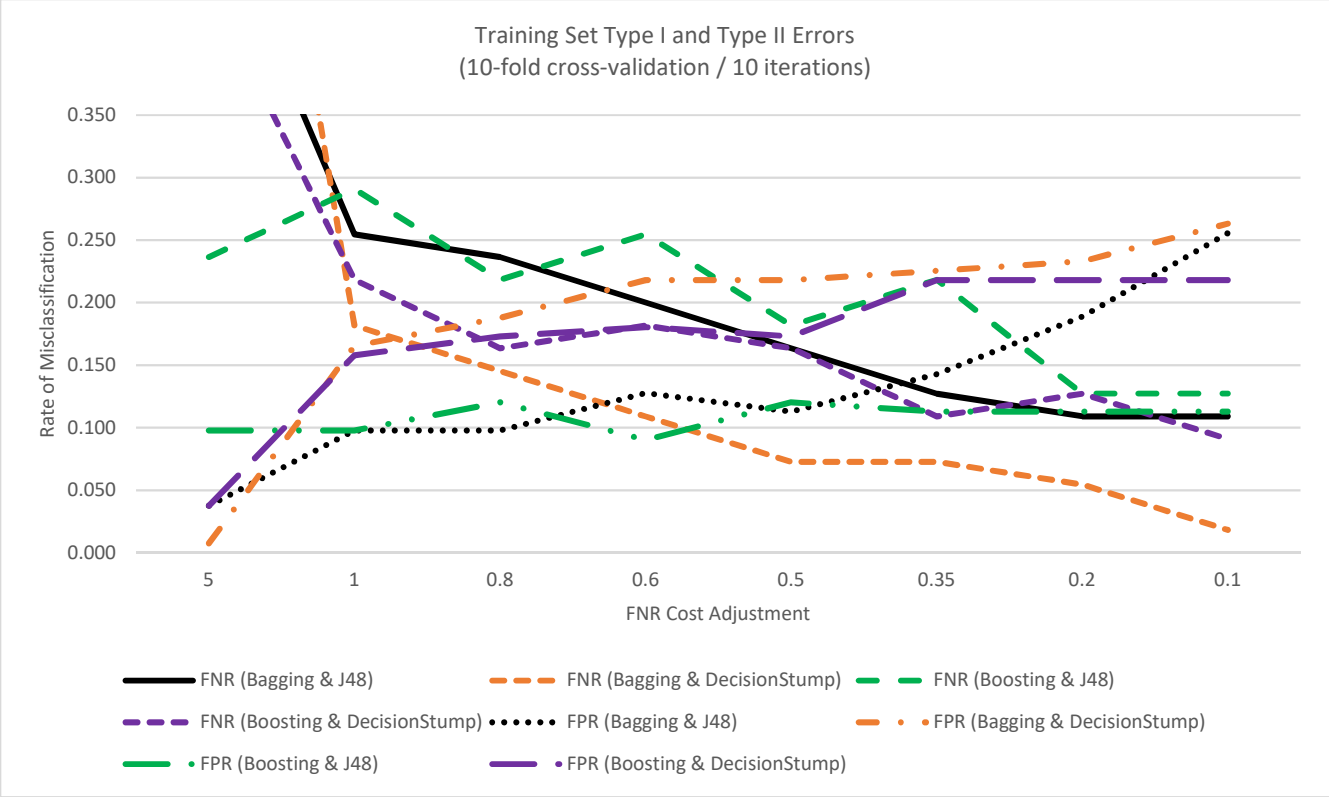

Figure 2 – Evaluation plot of FNR and FPR (10 iterations)

The goal of finding the right model is to adjust the cost ratio, changing FNR and keeping FPR at one, to find the FNR and FPR being close to equal with the FNR as low as possible; thus trying to obtain an optimal cost ratio.The point at which FNR equals FPR can be seen in Figure 2 when both lines cross. The bagging with J48 lines show a crossing at a cost of ~0.35 which, when cross referenced with the table in Figure 1, indeed indicates the best FNR cost adjustment and model with near equal FPR and FNR with lower FNR and the lowest RMSE. The RMSE shows the average distance of the model from the mean with smaller values indicating the better results. Therefore, in the case of bagging with J48, a cost adjustment of 0.35 is found to be near-optimal. All models seem to cross at some point or come close to crossing near, what might be considered, the optimal cost ratio.

Results

Meta Learning: Bagging

Bagging with J48 indicates a lower RMSE than without bagging with the FPR and FNR being more equal having a 0.16 absolute delta between FNR and FPR, versus the non-bagging approach having a 0.19 delta. The RMSE is also lower with bagging indicating less variability around the mean (smaller residuals). Based on the training set the model created using bagging with a cost adjustment of 0.35 is marginally better than the J48 model without bagging. The validation run shows the bagging model to be a little better than the non-bagging model with lower misclassification rates and RMSE. Of course, the models are chosen based on the training set so further test datasets should be used for validation to mitigate the chance of have a poor quality, or one off, test dataset. As mentioned previously, using an ensemble approach with a strong learner provides little benefit and this seems to be the case based on these sample models using bagging.

Meta Learning: Boosting

Boosting with J48 shows the FNR and FPR misclassifications being closer together at a 0.14 versus 0.19. The FNR though is higher than the FPR when boosting was employed, but it has the lowest FNR over the various cost adjustments over this training set. The RMSE is also higher with boosting. Given the evaluation results from the training set, it seems that boosting creates a worse model than without boosting. This could be attributed to boosting increasing the chance of overfitting the data. This though is not a certainty and additional evaluation on different training and validation sets should be done to see how boosting affects these additional classifiers. In this instance, not using boosting creates a better classifier.

Meta Learning: None

Assignment 2 part 4, Figures 3 and 4, used a cost sensitive model to create a classifier, J48, but did not include any meta learning schemes. Its performance was improved slightly with meta learning.

Training dataset, 10-fold cross-validation with cost ratio adjustments								Test dataset with chosen cost ratio	Test dataset with chosen cost ratio (alt)
(Assignment #2, part 4)								(Assignment #2, part 4)	(Assignment #2, part 4)
FNR Cost	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.4**	**0.3**	**0.5**	**0.4**
FNR	0.564	0.2	0.164	0.127	0.109	0.145	0.145	0.286	0.286
FPR	0.03	0.09	0.113	0.135	0.128	0.128	0.135	0.152	0.121
RMSE	0.4	0.334	0.334	0.33	0.322	0.349	0.354	0.414	0.401

Figure 3 – Evaluation results from Assignment 2, part 4 decision tree

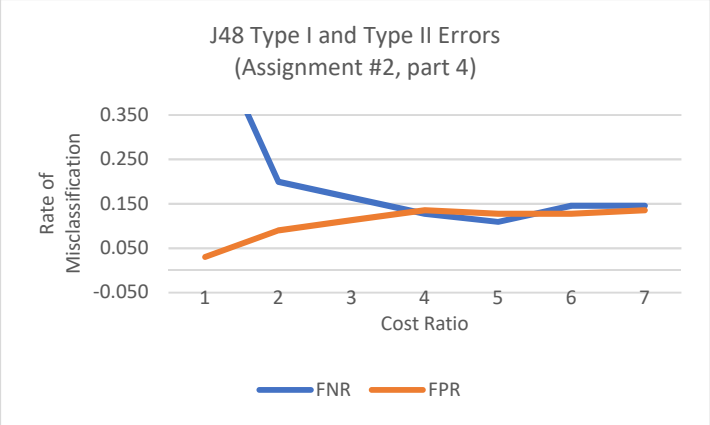

Figure 4 – Evaluation plot of FNR and FPR from Assignment 2, part 4 decision tree

Summary

To summarize the above observations with the strong learner, bagging and boosting create different classifiers but the difference between using the meta learning schemes and not using them is marginal in terms of overall model evaluation results. Unlike with the strong learners, using meta learning schemes on weak learners has a larger and more material benefit to the final, composite classifier. We compare the improvements in J48 with no meta learner applied as in the previous assignment.

Decision stump was used as the weak learner. In Figure 1, decision stump evaluates favorably with J48 using either bagging or boosting ensemble approaches. In fact, with boosting, the decision stump performs better than the J48 with boosting. This is a drastic improvement over decision stump alone and further indicates the power of using meta learning schemes. Overall, J48 with bagging is probably the best classifier but using the meta learning schemes, decision stump is a viable option. The validation results show the decision stump as having generally better results with lower FNR though slightly higher RMSE. This is notable given decision stump's use of a single attribute to classify an entire dataset. This again reaffirms the benefits of ensemble approaches with unstable, weak machine learning algorithms.

Models and Results: Part 2

The final task was to increase the number of iterations used in each meta learning scheme from 10 to 25. Overall the results were similar with bagging, but the decision stump evaluation results are better as are the boosting results. Figures 5 and 6 display the summarized results when using 25 iterations. These are in the same format as the figures used for the 10 iteration models. The decision stump performance increase is expected given the increase in decision possibilities going from one, with a single attribute decision with decision stump, to 10 possible attributes to decide with, to finally 25 possible decision points. Decision stump, over more iterations, is better able to model the training set and provide more accurate predictions. Boosting also performed better on both training and fit datasets as suspected though is still worse than bagging.

		TRAINING DATASET, 10-FOLD CROSS-VALIDATION WITH COST RATIO ADJUSTMENTS 25 ITERATIONS								TEST DATASET WITH CHOSEN COST RATIO 25 ITERATIONS	TEST DATASET WITH CHOSEN COST RATIO (ALT) 25 ITERATIONS	TEST DATASET WITH CHOSEN COST RATIO (ALT) 25 ITERATIONS
BAGGING & J48	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.35**	**0.5**	**0.6**
	FNR	0.564	0.236	0.200	0.127	0.145	0.127	0.073	0.073	*0.143*	*0.179*	*0.179*
	FPR	0.023	0.083	0.090	0.120	0.143	0.135	0.180	0.256	*0.121*	*0.091*	*0.121*
	RMSE	0.364	0.314	0.310	0.310	0.311	0.321	0.339	0.386	*0.321*	*0.325*	*0.319*
BAGGING & DECISIONSTUMP	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.8**		
	FNR	0.836	0.200	0.109	0.091	0.091	0.073	0.055	0.018	*0.071*		
	FPR	0.015	0.150	0.188	0.211	0.211	0.233	0.233	0.263	*0.212*		
	RMSE	0.393	0.333	0.334	0.341	0.346	0.364	0.378	0.408	*0.340*		
BOOSTING & J48	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.5**	**0.1**	
	FNR	0.473	0.236	0.255	0.291	0.182	0.236	0.200	0.182	*0.357*	*0.214*	
	FPR	0.053	0.083	0.090	0.083	0.113	0.113	0.098	0.083	*0.045*	*0.061*	
	RMSE	0.354	0.359	0.359	0.374	0.366	0.381	0.358	0.326	*0.372*	*0.326*	
BOOSTING & DECISIONSTUMP	**FNR Cost**	**5**	**1**	**0.8**	**0.6**	**0.5**	**0.35**	**0.2**	**0.1**	**0.35**	**0.2**	
	FNR	0.473	0.273	0.236	0.164	0.182	0.145	0.109	0.091	*0.179*	*0.115*	
	FPR	0.053	0.113	0.105	0.120	0.143	0.188	0.180	0.218	*0.167*	*0.212*	
	RMSE	0.354	0.334	0.329	0.326	0.339	0.341	0.341	0.377	*0.338*	*0.356*	

Figure 5 – Evaluation table for 25-iteration meta learning schemes

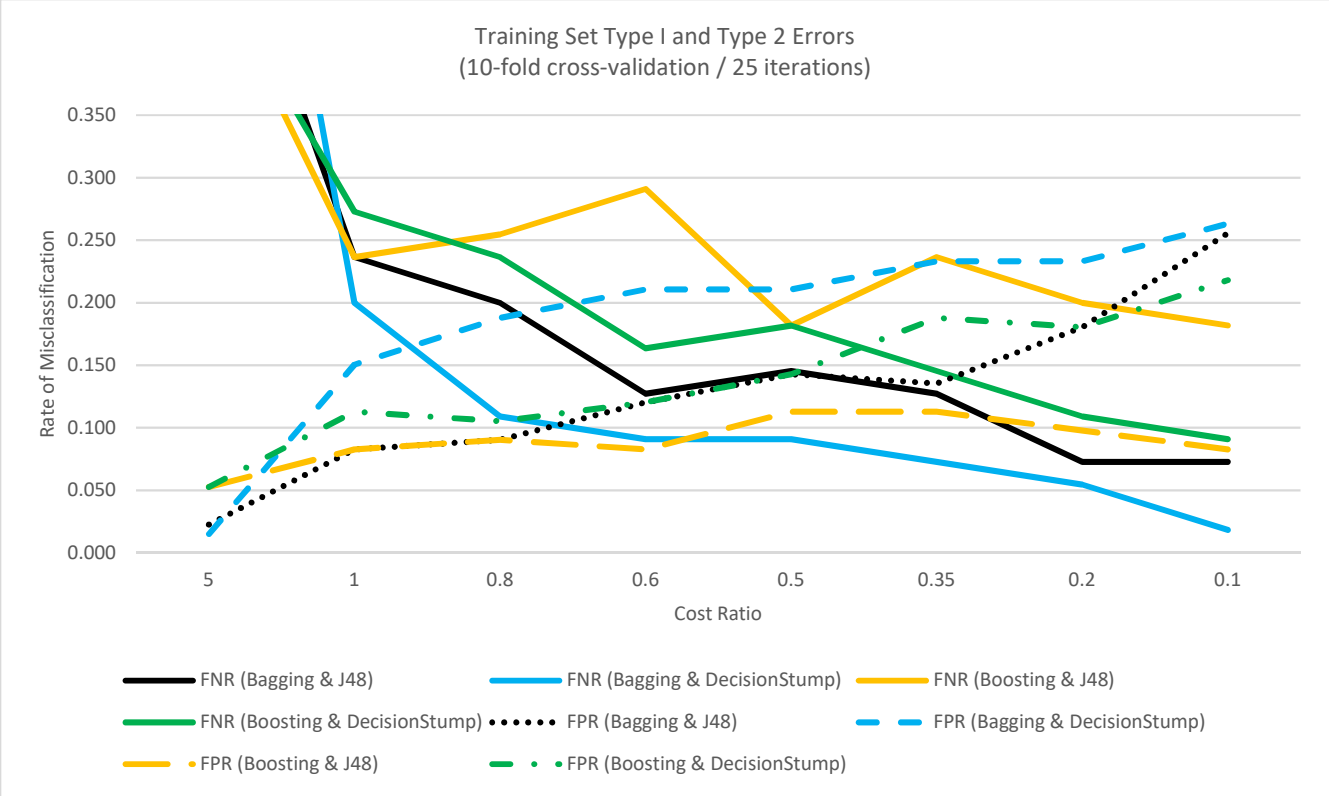

Figure 6 – Evaluation plot of FNR and FPR (25 iterations)

Conclusion

Meta learning schemes provide a hybrid or ensemble approach to machine learning methods generally increasing overall classifier performance. The benefit of ensemble learners comes from applying them on unstable, weak learners such as decision stump. There are less material gains applying meta learning schemes to strong learners, such as J48. And these ensemble approaches may actually decrease performance as was the case with boosting and J48. Bagging is the most consistent in terms of evaluation statistics with better performance over the iterations. Boosting also increased performance with increase iterations though it still performed subpar compared to bagging and using no meta learning scheme. It is surmised that boosting is benefitted by increasing overfitting thus increased iterations leads to better boosting performance.

Data Analysis

Using the decision stump with bagging the number of executable lines of code may show faults due to fatigue or human error which would make sense is less likely to occur than faults during comprehension or doing mental acrobatics.

Appendix

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 5.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .8 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .6 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .35 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .2 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .1 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 5.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .8 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .6 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .35 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .2 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

java weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; .1 0.0]" -S 1 -W weka.classifiers.meta.Bagging -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 -S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.J48 -- -C 0.25 -M 2

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump

weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 1.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -- -P 100 - S 1 -num-slots 1 -I 10 -W weka.classifiers.trees.DecisionStump