Section 1: SMO

1. Written by John Platt, the Sequential Minimization Optimization algorithm iteratively solves the optimization problem by breaking the classification down to smallest possible sub-problems, and using those problems to solve the whole problem.

2. We tried to vary all the parameters we could on the SMO algorithm. It turns out that -C, the complexity constant has the most effect on the robustness, followed by - N, the normalization parameter. -L, the tolerance parameter, and -P, the epsilon for round-off error, -W, the random number seed, and -V, the number of folds for the internal CV did not have much effect on the algorithm, and we ended up with the same robustness results as the untuned SMO. The complexity constant, which determines how soft class margins are, affects the SVM significantly since it determines the boundary for classifying each instance. In experimenting with these parameters, we used CVParameterSelection wrapper, and it took from 5 to 10 minutes on my Macbook Air to train the models.

3. We noticed that mostly, -C between 1 and 2 gives the best results, but on the diabetes dataset, -C set to 5.333334 gives a good result. Hence, to keep the compute time under 5 minutes, we chose to add “C 2 12 4” to our CVParameter. Furthermore, we found that most of the times, fixing N to 1 increases accuracy. Therefore, we also added “N 1 1 1” to our CVParameter.

Section 2: BayesNet

1. It implements Bayesian network. There are two stages in the learning. It first learns a network structure and then learns the probability table. There are several different search algorithms can be used to identify a network structure. Once that is done, we can estimate the conditional probability table for each variable.

Weka class name: weka.classifiers.bayes.BayesNet

2. We tried to vary all the parameters for BayesNet. There are not many parameters that can be varied. Two major parameters are search algorithm and estimate algorithm. Search algorithm is used to identify the network structure and estimate algorithm is used to estimate conditional probability table. I think the default search algorithm is K2 and the default estimator is SimpleEstimator. We tried to change the search algorithms and estimate algorithms, but the performance of classifier does not change much. As we change search algorithms to other algorithms like SimulatedAnnealing, it takes much longer to classify data. Error rate stays pretty much the same as untuned BayesNet. The algorithm did not take much time to train.

3. We ran the tuned BayesNet classifier on all 12 data sets. However, we did not notice any improvement. Hence, we stopped the tuning process.

Section 3: AdaBoostM1

1. AdaBoostM1 extends the AdaBoost algorithm written by Yoav Freund and Robert Schapire. AdaBoost, or Adaptive Boosting, generates a new classifier for each round (out of total T rounds), where each round weighs the instances the classifier previously incorrectly identified more so that the classifiers improve in performance each round.

2. We tried to vary all the parameters we could on the AdaBoostM1 algorithm. It turns out that -Q, the variable that indicates whether or not to use resampling for boosting, -I, the number of iterations, and -P, the percentage of weight mass to base training on, all do not improve the robustness significantly. Only around 1% gain was seen. Also, the algorithm did not take much time to train.

There is another parameter which the base classifier that we can change. We tried to change that base classifier to the current best Classifier we have which is SMO. Although this improves error rates for some datasets, for the rest the result either does not change or become worse. I guess it is because AdaBoostM1 tends to overfit data.

3. Even though only little change is seen, adding "I 10 100 6" and "Q 1 1 1” to CVParameter improved the performance.

Section 4: Summary

We decided to use SMO, which was the best untuned classifier, and tune it described in the final settings described above. With tuning, we were able to increase the mean robustness from 0.6984172165761393 (milestone 2) to 0.6947674055571316 (milestone 3), yet the maximum robustness increased a little bit from 0.9838709677419355 to 1.0869565217391304.