LAB ASSIGNMENT - 4

1. You are required to work on the "diabetes.arff" data set. You have to apply the J48 classification algorithm with -split-percentage as 80. This option will create a training data set with 80% of the points, and use the remaining points for testing.

The focus of this exercise is to visualise the effect of the parameter called the 'Confidence Factor'. Once the tree has been built (nodes with fewer than minNumObj data points are not split), a further pass of pruning is performed by applying the confidence factor. The confidence factor may be viewed as how confident the algorithm is about the *training* data. (confidence factor represents a threshold of allowed inherent error in data while pruning the decision tree.) A low value leads to more pruning; a high value keeps the model close to the original tree built from the training data (the parameter is used in estimating error probabilities at leaves).

Run experiments with the following values of the confidence factor (default 0.25, maximum 0.5), with the train/test split as specified.

- Confidence Factor: 0.002 , 0.008 , 0.02 , 0.08 , 0.1 , 0.2 , 0.3 , 0.4 , 0.5.

For these values, record both the training and test accuracy.

2. **Comparing decision trees and neural networks**

Apply both J48 and the MultilayerPerceptron to the following data sets: "bank-data.arff". Note down the accuracy (5-fold cross-validation) while keeping the number of epochs for the Multilayer Perceptron as 3. Keep all the other parameters at their default settings.

What accuracy do these methods achieve on "bank-data.arff" ?

Can you think of reasons to explain their relative accuracy on these data sets?

What are the properties of the data sets?

Describe the decision tree model that is learned in each case.

3. **Analysis of training time**

Apply both J48 and MultilayerPerceptron on the following data sets.

- iris.arff
- vote.arff
- soybean.arff
- unbalanced.arff
- segment-challenge.arff
- supermarket.arff

The main object of interest is the ***time*** taken to build the decision tree and neural network models in each case. Since the experiments could take time, keep the split percentage as 80. Plot a common graph of time (for building the model, not testing) versus the size of the data set, with one curve for each classifier. Also plot a second, similar graph of test accuracy versus the size of the data set.

Write down your observations from the two graphs. Is either J48 or MultilayerPerceptron always the method of choice? If yes, which one? If not, why?

4. Boosting Decision Stumps

Decision stumps are degenerate decision trees with only **one** split (sometimes we also grow random trees—in which features are chosen at random to split—to two or some small number of levels, combined in an ensemble called a *random forest*). Naturally the expressive power of decision stumps is very limited. However, decision stumps can be combined into an ensemble, whose collective classification accuracy could be much higher. This task is meant to demonstrate the phenomenon of combining multiple *weak* learners into a *strong* learner, using a meta-learning algorithm called Boosting.

For this final task, work with DecisionStump and LogitBoost on the "musk.arff" data set. LogitBoost constructs an additive model, starting with a single decision stump, and then training each subsequent decision stump to optimally remedy the errors made by the current group of decision stumps (or the ensemble). Members of the ensemble are also weighted; the prediction made for a test query is the weighted majority of the predictions made by the ensemble. While running LogitBoost, specify DecisionStump as the base learner.

How many iterations does LogitBoost need to reach 90% accuracy?