1 Evaluating Classifiers

1.1 Metrics

There are several common metrics compute to assess performance. It often useful to look at them per class value (e.g. in our dog data set to look at the “Jack Russell” class). To be general, we will call the class $a$.

- True Positives (TP): the number of data points in class $a$ that are classified as class $a$
- True Negatives (TN): the number of data points not in class $a$ that are classified as some class other than $a$
- False Positives (FP): the number of data points not in class $a$ that are classified as class $a$
- False Negatives (FN): the number of data points in class $a$ that are classified as some class other than $a$
- TP rate: $TP/(TP+FN)$
- FP rate: $FP/(FP+TN)$
- precision: $TP/(TP+FP)$
- recall: same as TP rate
- F-Measure: \[
\frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} = \frac{2TP}{2TP + FP + FN}
\]

1.2 Testing data

It is important to reserve some data for testing. Suppose we use all of our data to construct a tree, and that we have all pure leaves in the tree. Then we test our tree on the training data. We compute the error and find that all data points are correctly classified. Does that tell us anything about how well the tree will classify new data points? No.

There are two basic strategies:
1. Use separate data for validation and testing.

2. Use cross-validation. If you don’t have a large data set, then you probably don’t want to leave any data out of your training set. So, the solution is to separate your data into partitions (folds) and let each fold take a turn at being a testing set.
   - Separate your data into $D$ folds
   - For each fold
     - Use the remaining data to train a classifier (e.g. build a tree)
     - Test the classifier using the data in the fold. This involves classifying the data point according to this tree.

Note that this means that each data point is used as a test point exactly once. So you have a predicted class for each data point. You also have the actual values of the class for each data point. So you can compute a confusion matrix and all of the other performance metrics listed above.

Regardless of which method you choose, you will want to be smart about how you partition the data. It is important to **strategy** the data – keep the proportion of data points from each class consistent with its proportion in the entire data set. So if 50% of your dogs are Jack Russells, then each partition should be 50% Jack Russell.

2  **Weka**

In Weka, there are many classifiers. In class, we have talked most about NaiveBayes (under the Bayes subtree), OneR (under the rules subtree), and J48 (under the trees subtree). Today, I also mentioned BFTrees, which are similar to J48 tress, but are constructed in a different order, and use a different pruning method.

There are multiple options for testing the classifier:

1. “Use training set”: This means the classifier’s performance is evaluated using the same data used to train it (build it). This doesn’t tell you must about how well it is likely to perform on new data.

2. “Supplied test set”: Open a new file and test with the new file’s contents.

3. “Cross-validation”: cross-validation as described above. You can pick the number of folds. 10 is standard.

4. “Percentage split”: Split the data into training and testing sets. The default is to use 66% of the data for training (the number in the box) and the remaining 37% for testing. It isn’t clear to Stephanie if this is done once or many times.