CS6762 Assignment 2 Tutorial

Introduction:

For parts 1, 2 and 3 of this assignment 2, you should use all the features and the decision tree classifier (parts 1 and 2 will have 6 features and part 3 will have 12 features). See the code for the class MyWekaUtils.java. Use its *readCSV* method to read the csv file(s), and its *csvToArff* method to convert it to arff with **all** the features selected. Then use its *classify* method with the arff data and option 1 (the decision tree) to find the classification accuracy. Using a program this way avoids needing to manually use the GUI to work with the decision tree classifier.

For parts 4 and 5, you will choose only the best features via a Sequential Feature Selection method. To do this you need to iterate and for each iteration find the accuracy for a subset of the features. For this purpose use the *csvToArff* method on each iteration (will be further explained in class), but for different subsets of features. Note that the *csvData* (input for *csvToArff* method) is not changed.

Information about the Sequential (Iterative) Feature Selection Method

1. Create an array (FeatureList) which is empty at the beginning that will hold the important features, one by one as they are determined. This is a standard procedure for trying to find the most informative features out of a large set.

2. First find the classification accuracies for each of the features individually. Select the feature that provides the best result, and add it to the **FeatureList[0]**. After this step, the list contains only one feature.

2. For each of the remaining features that are not selected i.e., not in the **FeatureList[],** add the remaining features to the previously already selected feature(s) one by one, and find the resultant classification accuracies. The feature from the remaining group that provides best accuracy in combination with the previously selected features is added to the **FeatureList[].** For example, if you have 12 features and you already decided that feature 7 is most important and that feature 8 is next most important, then the next loop would assess 7,8,1 and 7,8,2, and 7,8,3 and 7,8,4, and 7,8,5 and 7,8,6, and 7,8,9, and 7,8,10, and 7,8,11, and 7,8,12.

3. Step 2 is repeated until there is less than 1% improvement in the accuracy, or all features are selected.

Programming guidelines:

Add weka.jar as a library to your project. The jar is available from the Weka installation folder.

You are provided with a java class (MyWekaUtils.java) which comes with the following public static methods:

Method	Input(s)	Output
public static double classify(String arffData, int option) : Takes data in arff format, and the classifier option, and returns the classification accuracy.	arffData: The arff file as a String	the accuracy as %
	option: 1 - Decision tree 2 - Random forest 3 - SVM Others - invalid, returns 0 accuracy	
public static String[ ][ ] readCSV (String filePath): Read a csv file, and convert it to a 2d array of strings.	filePath: The path to the csv file	the csv file as a 2d array of strings
public static String csvToArff (String[ ][ ] csvData, int[ ] featureIndices): takes csvData as 2d array of Strings, and the indices of the features that need to be included. Returns arff file containing data of the features.	csvData: from above method	An arff file (as String) that contains only the features that are seleted
	featureIndices: indices of the selected feartures. Index start from 0. Never include index of the class i.e. the last index.	

<u>Notes:</u>

1. The methods throw exceptions, and so appropriate means should be taken. (like try...catch).
2. The code provided is not written for efficiency. Students should modify the code to run efficiently where possible.
3. Students should study the code to understand how to classify using the Weka library.
4. Before using the MyWekaUtils class, test it with some small amount of data. See how the methods work, and what the outputs are.