Dataset Metafeatures

By: Noureddin Sadawi

Nov 2014

1 Calling Java Functions from `R`

This section provides information on how to setup the environment so we can use the java package (which I implemented) from within `R` using the package `rJava`.

1.1 Settings (exports are normally done in the .bashrc file):

1. Weka needs to exist on the system
2. Setup `WEKA_HOME` environment variable. Example:
   ```
   # export WEKA_HOME=/home/likewise-open/ACADEMIC/csstnns/binaries/weka-3-7-11/
   ```
3. Setup `JAVA_HOME` environment variable. Example:
   ```
   # export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64/jre/
   ```
4. The `CLASSPATH` needs to point to `weka.jar`. Example:
   ```
   # export CLASSPATH=$WEKA_HOME/weka.jar
   ```

1.2 Calling java from R using package "rJava":

Execute the following inside `R`:

1. Initializes the Java Virtual Machine (JVM) with:
   ```
   # .jinit()
   ```
2. Add the JAR file to the classpath with:
   ```
   # .jaddClassPath("/full/path/to/ds-metafeatures.jar")
   ```

3. Add your current directory to the classpath with:
   ```
   # .jaddClassPath(".")
   ```

4. Create a new Java object with:
   ```
   # obj=.jnew("dsmeta.DatasetMetaFeatures")
   ```

5. Calls the Java methods (supplying their arguments) with:
   ```
   # result=.jcall(obj,"D","computeDimensionality","/full/path/to/csv/file.csv")
   ```
   Notice the `D` is for type `double`, `computeDimensionality` is the function name and `/full/path/to/csv/file.csv` is an argument.

1.3 A List of all functions in the Java Package:

The following is a detailed list of methods in the java package. Observe that each of them returns one real value. Also, please observe that the methods coloured in blue require a categorical class (I can discretize if needed):

1. computeEquivNumOfAttr(String datasetPath): computes the equivalent number of attributes. From Expose: The equivalent number of attributes is a quick estimate of the number of attributes required, on average, to describe the class.
   $$EN - atrr = \frac{H(C)}{MI(C,X)} \tag{1}$$

2. computeNumAttributes(String datasetPath): computes the number of attributes

3. computeNumInstances(String datasetPath): computes the number of instances

4. computeDimensionality(String datasetPath): computes the dimensionality of the data, which is the ratio of number of attributes to number of instances

5. computeNumNominalAttributes(String datasetPath): computes the number of nomial attributes

6. computeNumNumericAttributes(String datasetPath): computes the number of numeric attributes

7. computeNumBinaryAttributes(String datasetPath): computes the number of binary attributes (all attributes in a dataset are checked and those who have two unique values are counted)

8. computeProportionOfNumericAttributes(datasetPath): the ratio of the number of numeric attributes to the total number of attributes

9. computeProportionOfNominalAttributes(datasetPath): the ratio of the number of nominal attributes to the total number of attributes

10. computeProportionOfBinaryAttributes(datasetPath): the ratio of the number of binary attributes to the total number of attributes

11. computeDefaultAccuracy(String datasetPath): computes the default accuracy (The percentage of instances which classes are the same as the mean—mode class value)

12. computeMeanMeansOfNumericAtts(String datasetPath): computes the mean of means of numeric attributes

13. computeMeanStdDevOfNumericAtts(String datasetPath): computes the mean std deviation of numeric attributes

14. computeMeanKurtosisOfNumericAtts(String datasetPath): computes the mean kurtosis of numeric attributes

15. computeClassEntropy(String datasetPath): computes the entropy of the class variable

16. computeMeanAttributeEntropy(String datasetPath): computes the mean entropy of attributes

17. computeMutualInformation(String datasetPath): From Expose: $MI(Y, X)$, the *mutual information* between nominal attributes X and Y describes the reduction in uncertainty of Y due to the knowledge of X, and leans on the conditional entropy $H(Y|X)$. It is also the underlying measure of the information gain metric used in decision tree learners.

$$\begin{aligned} MI(Y,X) &= H(Y) - H(Y|X) \quad (2)\\ H(Y|X) &= \sum_i p(X = x_i) H(Y|X = x_i) \quad (3)\\ &= -\sum_i \pi_{i+} \sum_j \pi_{j|i} log_2(\pi_{j|i}) \quad (4) \end{aligned}$$

18. computeNoise2SignalRatio(String datasetPath): from Expose: The noise to signal ratio is an estimate of the amount of non-useful information in the attributes regarding the class. $\overline{H(X)}$ is the average information (useful or not) of the attributes.

$$NS - ratio = \frac{\overline{H(X)} - \overline{MI(C,X)}}{\overline{MI(C,X)}} \quad (5)$$

19. computeMeanSkewnessOfNumericAtts(String datasetPath): computes the mean skewness of numeric attributes

20. computeGenericDiversityIndex(String datasetPath,int startFP,int endFP): computes a generic diversity index for the dataset. Requires the zero-based index of the first and last FP columns (assuming all in-between are FP's)

21. computeExplicitDiversityIndex(String datasetPath, int startFP, int endFP, int cs): computes the explicit diversity index for the dataset. Requires the zero-based index of the first and last FP columns (assuming all in-between are FP's). Also requires a CS value which should be looked up in a special table using the target ID (table provided by Dundee group).

2 A List of all functions in the `R` script:

The following is a detailed list of functions in the `dataset-meta.R` script which can be imported/imported to `R` with `source('/path/to/dataset-meta.R')`. Observe that each of them returns one real value. Also, please observe that here I perform *equal-width* discretization for the class data if it is numerical and the function requires categorical class data. Please observe that the `data` object is a *data frame*.

1. get_discretized_class_entropy(data): computes the class entropy. The class variable is discretized first if it is numerical.

2. get_normalized_class_entropy(data): computes the normalized class entropy which is the class entropy divided by $log(n)$ where n is the number of categories. The class variable is discretized first if it is numerical.

3. get_attribute_entropy(data): computes the average entropy of all attributes (sums entropy of all attribues and divides by number of attributes)

4. get_normalized_attribute_entropy(data): computes the normalized average entropy of all attributes (sums normalized entropy of all attribues and divides by number of attributes). The normalized entropy of each attribute is the entropy of this attribute divided by $log(n)$ where n is the number of categories. The attribute is discretized first if it is numerical.

5. get_mutual_information(data): computes the mutual information for each column with respect to all categories of the class variable (the class variable is discretized first if it is numerical) then computes the average mutual information for all columns

6. get_equivalent_number_of_attributes(data): computes the equivalent number of attributes by dividing the discretized class entropy by the mutual information

7. get_noise_signal_ratio(data): computes the noise to signal ratio via: $(a - b)/b$ where a is the attribute entropy and b is the mutual information

8. get_multiinformation(data): computes the multiinformation (also called total correlation) among the random variables in the dataset.