Multi-Label Dataset Analyzer
User Guide

Contents

Chapter 1

Introduction

The objective of data mining is to extract relevant and potentially useful knowledge under large amounts of information. Classification is one of the main tasks of data mining. Given a set of objects, each one labeled with a pre-defined class, for each new object whose class is unknown, tries to predict which is its class.

An example of classic classification is the Iris dataset [6]. In this problem, depending on the size of petals and sepals of Iris plant, they are classified in one of its three types: setosa, virginica or versicolor. Each instance can be classified only in one class, since each Iris plant can be only of one type (Figure 1.1).

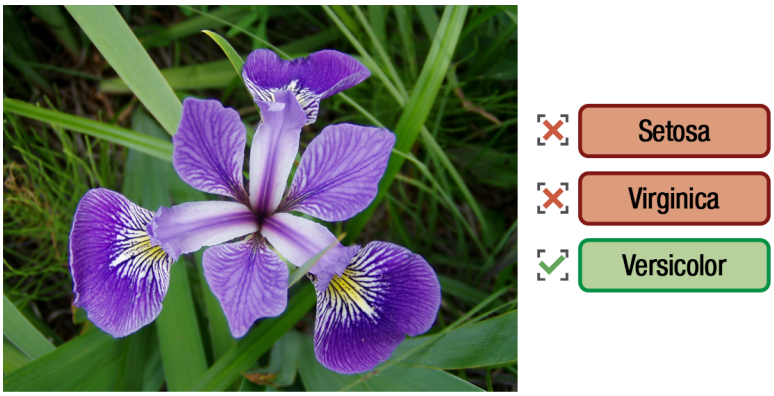

FIGURE 1.1: Classification example with Iris dataset

This definition corresponds to the classic version of classification, which implies the restriction of one class per pattern. However, there are a lot of real-world problems where objects can be associated with more than one class simultaneously. For example, in image annotation, as in Figure 1.2 having 4 labels previously defined, the image could be associated to the "Mountain", "Forest" and "River" labels. In classic classification, for the same example, the image could only be classified in one class or label, so the output would not be complete.

Thereby, they appear new paradigms in machine learning and data mining, which represents the information in a more flexible way. In Multi-Label Learning (MLL), unlike classic learning, an instance could have several classes (labels) simultaneously associated. This kind of learning has experimented a big growth in the last years, due to its success in problems as text categorization [10], social networks mining [16], medical diagnosis [14] or direct marketing [20].

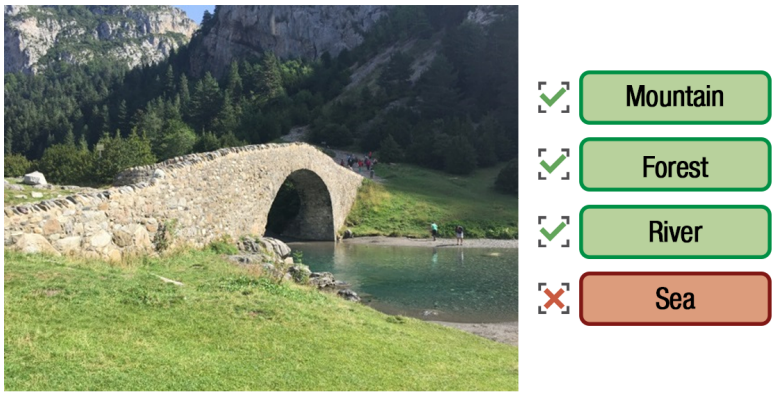

FIGURE 1.2: Example object with several labels

On the other hand, in many fields of application, an object has different representations and is described by several sets of features (views), which have been obtained from different data sources or feature extractors. As a result, the descriptor variables (input space) can be partitioned. For instance, in image annotation (Figure 1.3), many times a single representation is not able to characterize all the classes. With different representations, each view would characterize a different concept (label). For instance, colour information (RGB), shape cue (SIFT) or global structure (GIST) could represent natural substances (e.g. sky or clouds), man-made objects (e.g. plane or motorbike) or scenes (e.g. seaside) respectively [11].

FIGURE 1.3: Example of object with multiple views

Despite the fact that many real-world data are are stored in a distributed and partitioned way and with an heterogeneous organization in multiple data sources, traditional algorithms infer models from a single homogeneous data set, which combines or merges different set of features. This methodology does not perform properly in every problem and may complicate the learning process. Multi-View Learning (MVL) tries to solve this type of problems, combining heterogeneous and complementary information from distinct views [19].

As a result of the hybridization of this two techniques, it appears the Multi-View Multi-Label learning paradigm (MVML). This more flexible learning approach allows to each pattern being represented by several sets of attributes and also being associated with several labels. Figure 1.4 shows an example of the image as an object with multiple views and multiple labels.

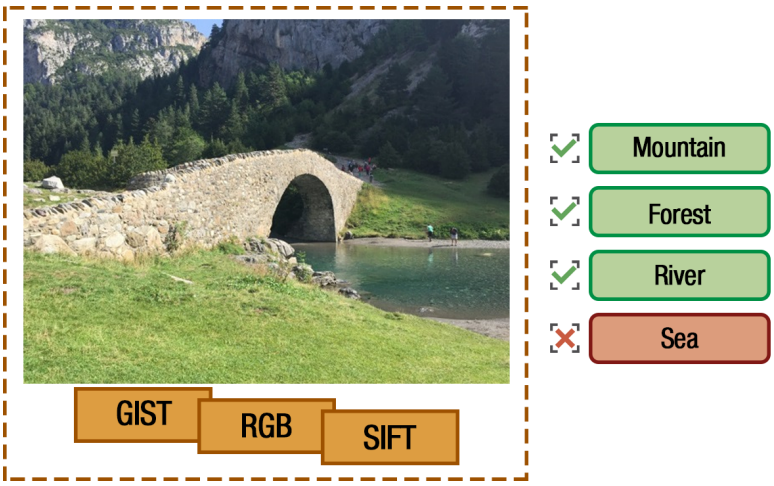

FIGURE 1.4: Example of object with multiple views and multiple labels

In MLL and MVML problems, many datasets have characteristics as imbalance (some labels or label combinations are very frequent while other are barely present), high dimensionality (in number of instances, attributes and labels) or relationship among labels [7]. Also, these characteristics could have direct effect on algorithms performance, hence the importance on characterizing and analyzing data [4]. This has led some authors to go further from previous exploratory analysis, developing meta-learning models based on these metrics in order to determine which algorithm is more appropriate according to the characteristics of each dataset. Furthermore, there are two libraries for multi-label learning, Mulan [18] and Meka [12], with different algorithms implemented and different dataset formats. For that, it could be interesting to convert the format of the datasets. By last, for MVML datasets, as input space is partitioned, it is interesting to characterize each view separately or filter some of them.

The main objective of MLDA (Multi-Label and multi-view Datasets Analyzer) is to offer a tool for data exploration and analysis of multi-label and multi-view multi-label datasets, which allows to represent its more representative characteristics and save its values for further analysis. Moreover, it allows to perform an analysis of imbalance and relationship among labels. MLDA also includes preprocessing tasks as data partitioning, feature and instance selection, transformations and dataset format conversion. The tool allows to load some datasets in order to calculate its characteristics simultaneously. In case of MVML datasets, it also allows to get some specific metrics for multi-view multi-label learning, shows the attribute list of each view and let save a new dataset with a set of user-selected views. Finally, it is implemented a Java API for multi-label dataset characterization, including the full set of metrics of MLDA. All this functionality allows to analyze

problems in order to adapt better the technique selected for its solution, as well as transform and preprocess them in an easy way.

Chapter 2

MLDA user guide

2.1 Downloading and executing

Before downloading the tool, it is necessary to have Java version 1.8 or higher installed. In order to download and install Java, you have to access to `https://www.java.com/es/download/`, download the version which corresponds to your operative system and install it.

The tool is available for download at `http://www.uco.es/grupos/kdis/kdiswiki/MLLResources/Software/MLDA_v1_1.zip`. Once downloaded, the file have to be decompressed, and it contains the executable *.jar* file. To execute the tool, you have to open a command-line interface, access to the path where the *.jar* file is located and type the command `java -jar MLDA.jar`.

2.2 Format of the datasets

2.2.1 Multi-label datasets

The tool is able to read two multi-label dataset formats, which are Mulan [18] and Meka [12] formats. Both of them are based on the *.arff* format from Weka [9] Weka's *.arff* format has the following characteristics:

- The relation name goes in the first line, following the statement "*@relation*". If the relation name has spaces, it must be between quotes.
- Each attribute must be in a different line, and its syntax is: "*@attribute name type*" .The attribute name must be between quotes if it has spaces. The attribute type could be:
  - numeric: integer and real are treated as numeric.
  - <nominal-values>: between braces and separated by commas all the possible values.
  - string.
  - date[<format>].
- Instances start with the statement "*@data*", and each one will be in a different line. Each attribute value is separated by commas, and they will be in the same order they were declared. Also there is a shortly way to write the instances, where attributes with zero value are not included. In this case, instances will be in braces, and separated by commas each pair composed by attribute and value.
- Line comments are inserted with the character "%".

Mulan and Meka use this format but they are different in how they define the labels. Mulan uses two different files: an *.arff* file and a *.xml* file. In the *.arff* file the full set of attributes and labels are exposed, without distinguishing among them. The only constraint is that labels may be binary attributes (values 0 and 1). The *.xml* file indicates which are the attributes that act like labels in the dataset. This *.xml* file is necessary because nowhere in the *.arff* file is indicated which the labels are. Figures 2.1 and 2.2 shows the *.arff* and *.xml* files respectively from a dataset in Mulan format.

```
@relation emotions_test

@attribute att1 numeric
@attribute att2 numeric
@attribute att3 numeric
   ...
@attribute att72 numeric
@attribute amazed-suprised {0,1}
@attribute happy-pleased {0,1}
@attribute relaxing-calm {0,1}
@attribute quiet-still {0,1}
@attribute sad-lonely {0,1}
@attribute angry-aggresive {0,1}

%Starting with the data
@data
0.094829,0.204498,0.082824, ..., 0.335371,1,0,0,0,1,1
0.065248,0.117975,0.08597, ..., 0.442898,0,0,0,1,0,0
0.101287,0.23254,0.078028, ..., 1.183461,1,1,0,0,0,0
   ...
0.172427,0.378696,0.081777, ..., 1.294949,1,1,1,0,0,0
```

FIGURE 2.1: Mulan *.arff* file

```
<?xml version="1.0" encoding="utf-8"?>
  <labels xmlns="http://mulan.sourceforge.net/labels">
   <label name="amazed-suprised"></label>
   <label name="happy-pleased"></label>
   <label name="relaxing-calm"></label>
   <label name="quiet-still"></label>
   <label name="sad-lonely"></label>
   <label name="angry-aggresive"></label>
</labels>
```

FIGURE 2.2: Mulan *.xml* file

On the other hand, Meka format only needs the *.arff* file. In this case, the separation between features and labels is done in the *.arff* file. Meka's *.arff* format is similar to Mulan's *.arff*, excepting the relation name line, where Meka indicates which attributes are the labels. Following the relation name, and separated by a colon, it is indicated with parameter "-C" and a integer positive number if the first q attributes are labels, or with an integer negative number if the labels are the last q attributes. Figure 2.3 shows the *.arff* file from a Meka dataset.

As seen, both formats are similar, and while Meka's format is simpler, Mulan's format is more powerful because it does not need to define all the labels at the beginning or the end of the attributes set.

```
@relation "emotions: -C -6"

@attribute att1 numeric
@attribute att2 numeric
@attribute att3 numeric
   ...
@attribute att72 numeric
@attribute amazed-suprised {0,1}
@attribute happy-pleased {0,1}
@attribute relaxing-calm {0,1}
@attribute quiet-still {0,1}
@attribute sad-lonely {0,1}
@attribute angry-aggresive {0,1}

%Starting with the data
@data
0.094829,0.204498,0.082824, ..., 0.335371,1,0,0,0,1,1
0.065248,0.117975,0.08597, ..., 0.442898,0,0,0,1,0,0
0.101287,0.23254,0.078028, ..., 1.183461,1,1,0,0,0,0
   ...
0.172427,0.378696,0.081777, ..., 1.294949,1,1,1,0,0,0
```

FIGURE 2.3: Meka *.arff* file

2.2.2 Multi-view multi-label datasets

The format to define the multiple views consists only in adding this information to the header of the *.arff* file. To define the views, inside the relation name the parameter "*-V*" is included. It is followed by a colon "*:*" and the set of views. Each view is defined with an interval with lower and higher attribute indices of the view, both included, and separated by a hyphen "-". Also, it is allowed to include attributes separated by commas "," if they are not consecutive. Different views are separated by an exclamation mark "*!*". Figure 2.4 shows an example of a multi-view multi-label dataset, where two views are defined, one from attribute 0 to 63 and other from attribute 64 to 71. This declaration of views is applicable to both Mulan and Meka formats.

```
@relation "emotions -V:0-63!64-71"

@attribute att1 numeric
@attribute att2 numeric
@attribute att3 numeric
   ...
@attribute att72 numeric
@attribute amazed-suprised {0,1}
@attribute happy-pleased {0,1}
@attribute relaxing-calm {0,1}
@attribute quiet-still {0,1}
@attribute sad-lonely {0,1}
@attribute angry-aggresive {0,1}

%Starting with the data
@data
0.094829,0.204498,0.082824, ..., 0.335371,1,0,0,0,1,1
0.065248,0.117975,0.08597, ..., 0.442898,0,0,0,1,0,0
0.101287,0.23254,0.078028, ..., 1.183461,1,1,0,0,0,0
   ...
0.172427,0.378696,0.081777, ..., 1.294949,1,1,1,0,0,0
```

FIGURE 2.4: MVML dataset *.arff* file

2.3 Summary tab

Once started, the first tab is the summary, where main metrics for dataset characterization can be calculated. For loading a dataset, you have to click on the button in the upper right corner (Figure 2.5), and select an *.arff* file in Mulan or Meka formats. As it can be seen, this tab shows in its top a set of basic metrics for characterizing datasets, while in the bottom it shows a larger set of metrics, from which you can select a subset of them, and calculate with the *Calculate* button (Figure 2.6).

In order to select the metrics, there also exist four extra buttons: *All*, to select all the metrics, *None*, to unselect all of them, *Invert*, to invert the selection, and *Clear*, which unselect and clear all the calculated values. The *Save* button allows to save all the metric values in four formats: *.txt*, *.csv*, *.arff* y *.tex*.

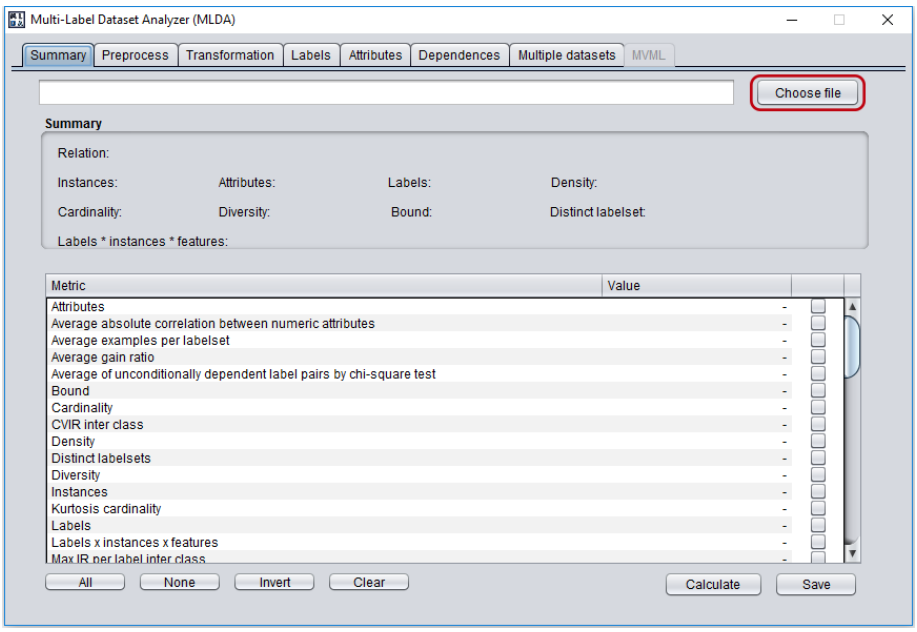

FIGURE 2.5: Selecting a dataset in summary tab

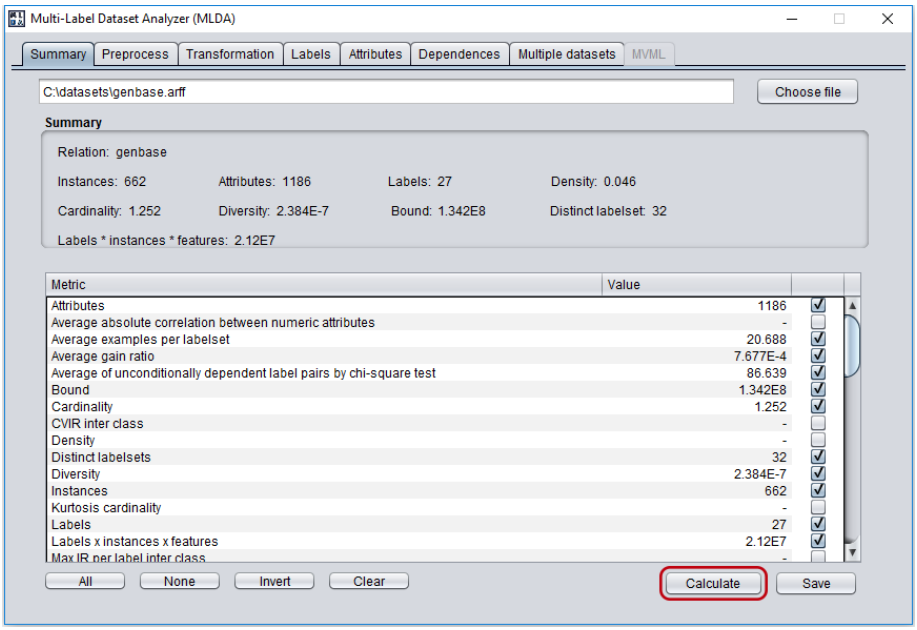

FIGURE 2.6: Calculating metrics in summary tab

2.4 Preprocess

The preprocess tab allows to perform preprocess tasks over the multi-label datasets. It includes instance selection, feature selection, data partitioning and format conversion. All the preprocess types could be done separately or together, where preprocessing is performed in the following: instance selection, feature selection, data partitioning and format conversion. That is to say, in case of selecting all the preprocessing types, first it will be made the instance selection, then, over the modified dataset, it will be made the feature selection, and last the splitting. Finally, the dataset could be saved in Mulan or Meka format.

2.4.1 Instance selection

To perform the random instance selection, only the number of instances to select has to be indicated, which must be less than the number of instances in the original dataset (Figure 2.7).

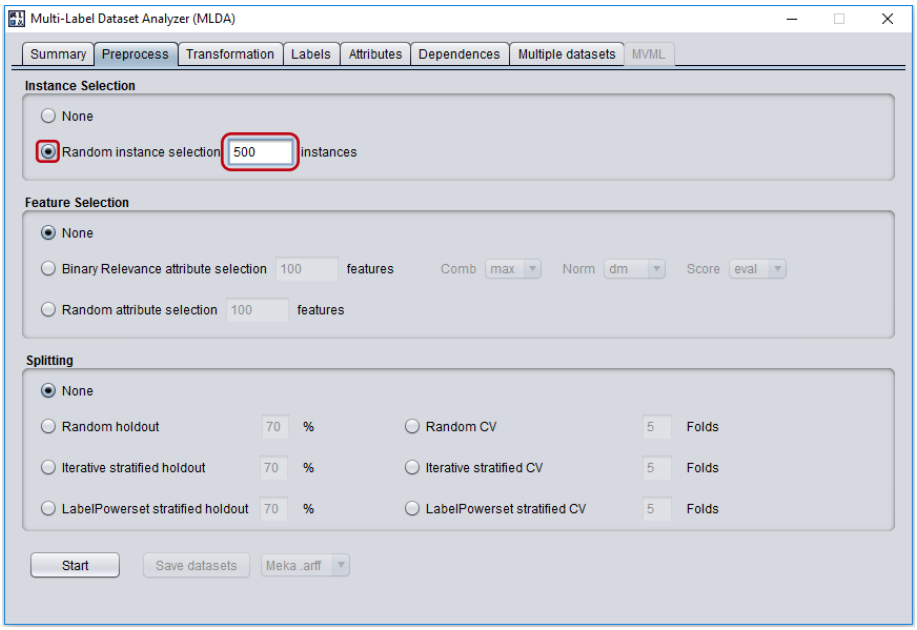

FIGURE 2.7: Random instance selection

2.4.2 Feature selection

For feature selection, the tool includes two methods: BR based [17] and random. For both types, the number of features must be indicated, always less than the number of features in the original dataset. As shown in Figure 2.8, the BR based method, which is specific for multi-label learning, needs three parameters: the combination approach method (which may be by maximum *max*, average *avg* or minimum *min*), normalization mode (which may be dividing by length *dl*, dividing by maximum *dm* or no normalization *none*), and score mode (which may be by evaluation score *eval* or by ranking score *rank*).

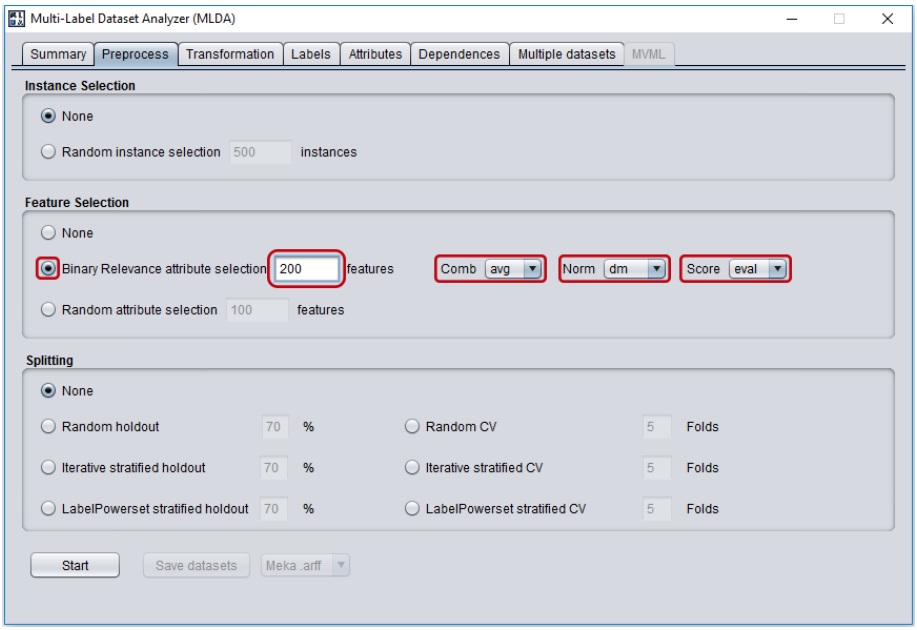

FIGURE 2.8: BR based feature selection

2.4.3 Data partitioning

Multi-label datasets allows splitting to be carried out in two different ways: holdout and k-folds cross-validation (cv) (Figure 2.9). For holdout partitioning, the percentage of instances for the train file must be specified, while for k-folds cv, the number of folds to split the original dataset has to be indicated. Holdout partitioning saves two new dataset files, a train file and a test file, while the k-folds cv saves k train files and k test files (Figure 2.10). For both types, there exists three methods for splitting: random, iterative stratified and LP stratified [13]. Iterative and LP stratification methods are specific for multi-label learning.

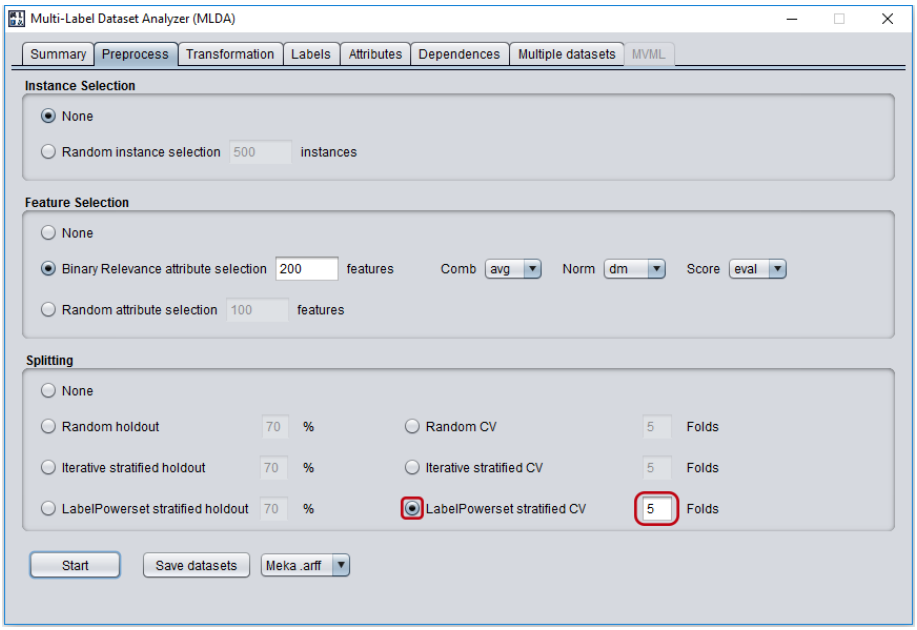

FIGURE 2.9: LP *k*-folds cv dataset splitting

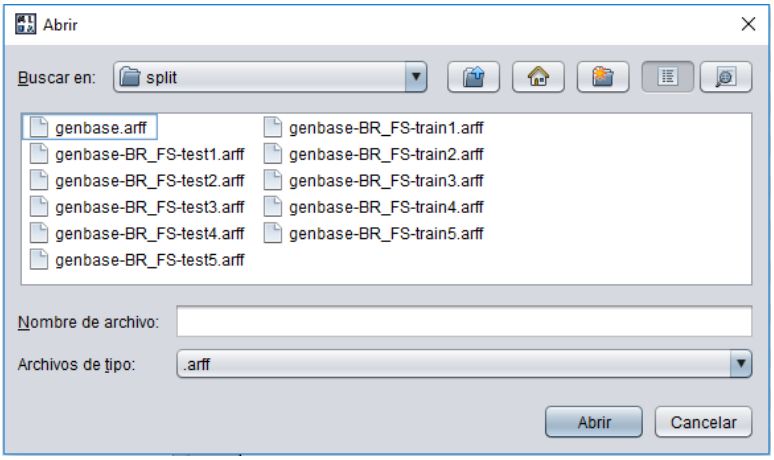

FIGURE 2.10: Result of a 5-folds cv splitting

2.4.4 Dataset format conversion

Since the tool is able to load both Mulan and Meka dataset formats and also allows to save the preprocessed datasets in both formats, if no preprocessing type is selected (Figure 2.11), the dataset format could be converted without modifying data. In order to convert the format of the dataset, *Start* button has to be clicked, and *Save datasets* to save the dataset with the selected format.

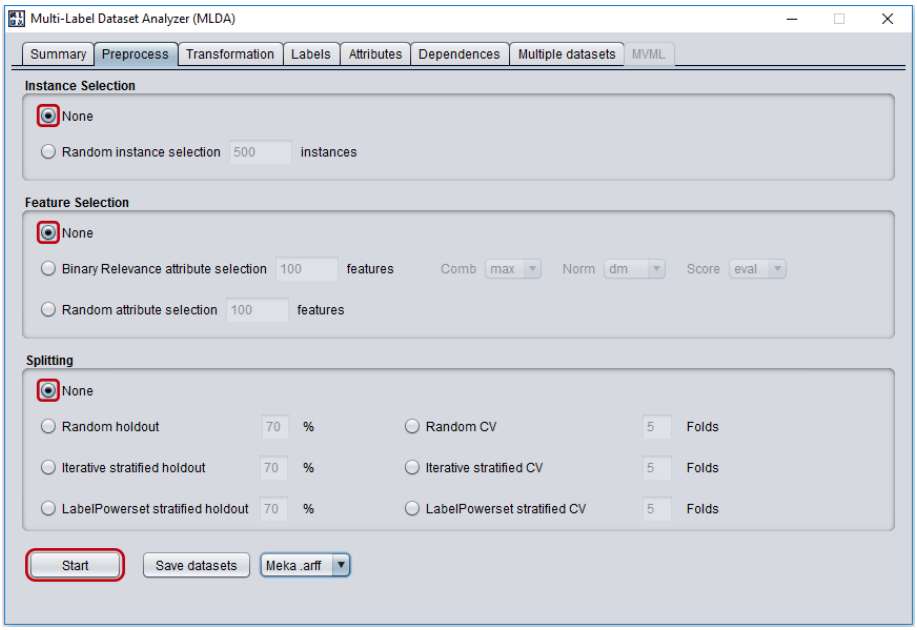

FIGURE 2.11: Dataset format conversion

2.5 Transformation

Transformation tab includes four transformation methods for multi-label datasets into single-label datasets. It includes: Binary Relevance transformation [17], Label Powerset transformation [1], include labels transformation and remove all labels transformation. Binary Relevance method generates q binary datasets, one for each existing label (Figure 2.12). Label Powerset generates one multi-class dataset, where the classes are the labelsets of the original dataset (Figure 2.13). Include labels transformation generates a binary dataset. For that, each instance is replicated q times, where each new instance is extended with the label name and the class is the label value of this instance (Figure 2.14). Finally, remove all labels transformation generates a new dataset removing all the label attributes (Figure 2.15).

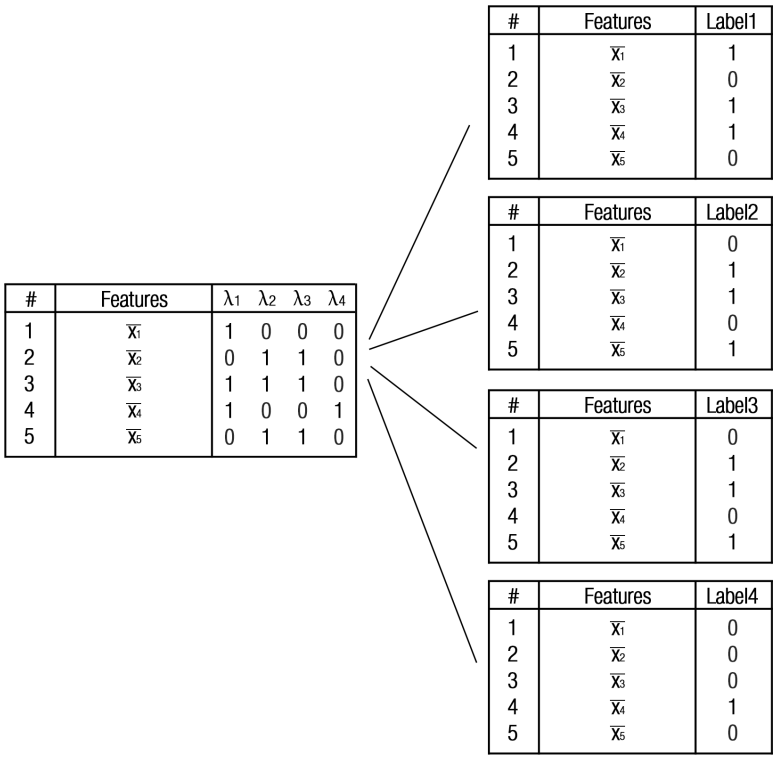

FIGURE 2.12: BR transformation

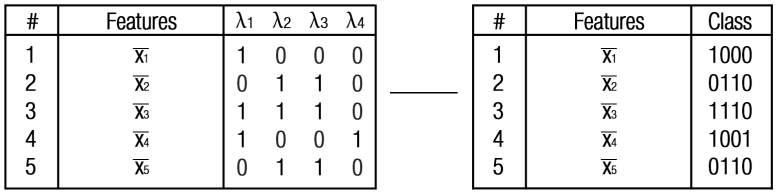

FIGURE 2.13: LP transformation

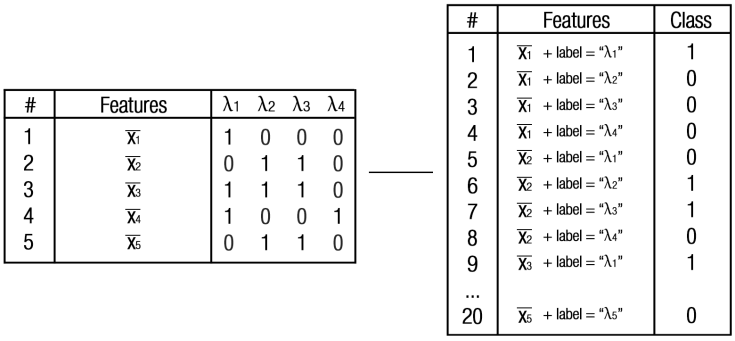

FIGURE 2.14: Include labels transformation

#	Features	λ_1	λ_2	λ_3	λ_4
1	$\overline{x_1}$	1	0	0	0
2	$\overline{x_2}$	0	1	1	0
3	$\overline{x_3}$	1	1	1	0
4	$\overline{x_4}$	1	0	0	1
5	$\overline{x_5}$	0	1	1	0

#	Features
1	$\overline{x_1}$
2	$\overline{x_2}$
3	$\overline{x_3}$
4	$\overline{x_4}$
5	$\overline{x_5}$

FIGURE 2.15: Remove all labels transformation

To transform the current dataset, we just have to select the desired method, click on the *Transform* button, and then save it with the *Save* button, as shown in Figure 2.16.

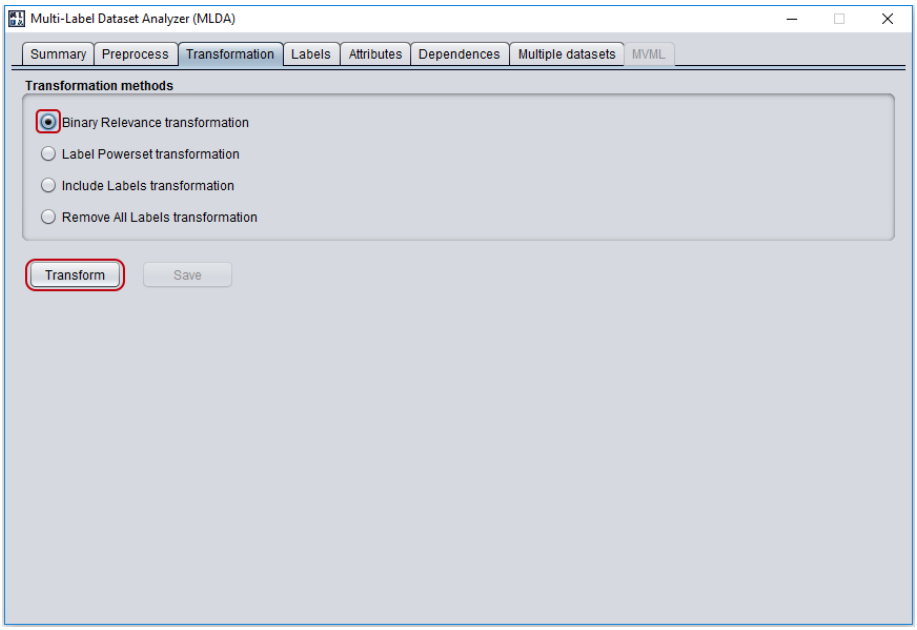

FIGURE 2.16: Transformation tab

2.6 Labels tab

Labels tab allows to obtain information and charts about the distribution and imbalance of labels. On the top appears a dropdown menu (Figure 2.17) where the graphic type can be selected. In all cases, on the left sidea table with the information appears, which also can be saved in *.csv* format by clicking on the *Save* button. The chart appears on the right side. By clicking in the chart with the right mouse button (Figure 2.18), some options are provided: zoom in, zoom out, auto range, print, properties, or saving the chart as a *.png* file in the *Save as* option (Figure 2.19) .

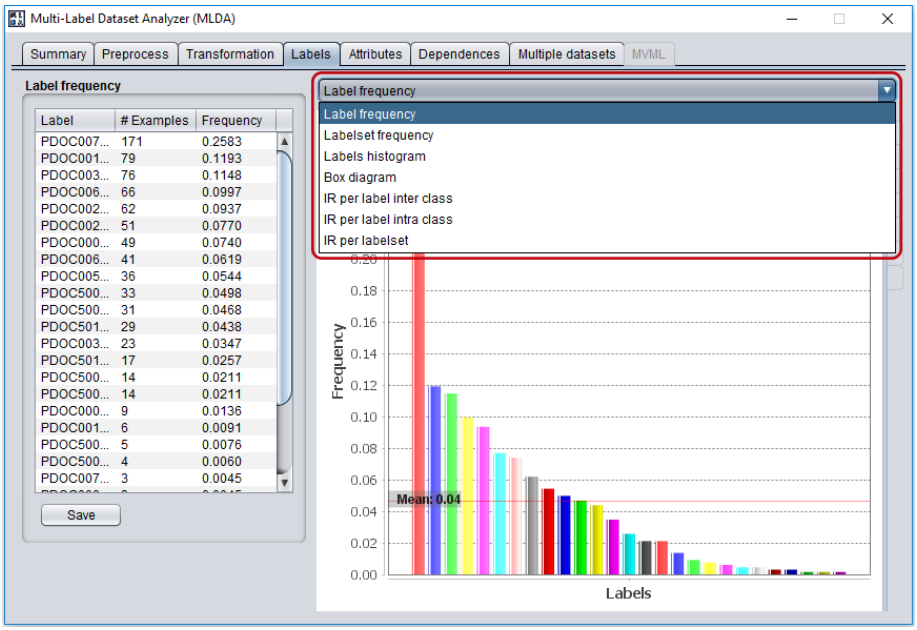

FIGURE 2.17: Dropdown menu on labels tab

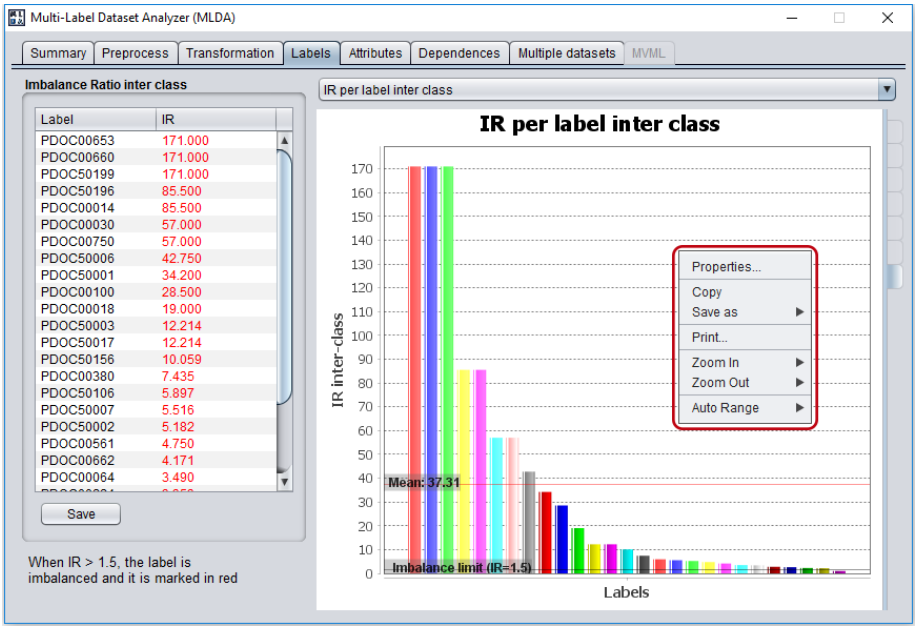

FIGURE 2.18: Chart options

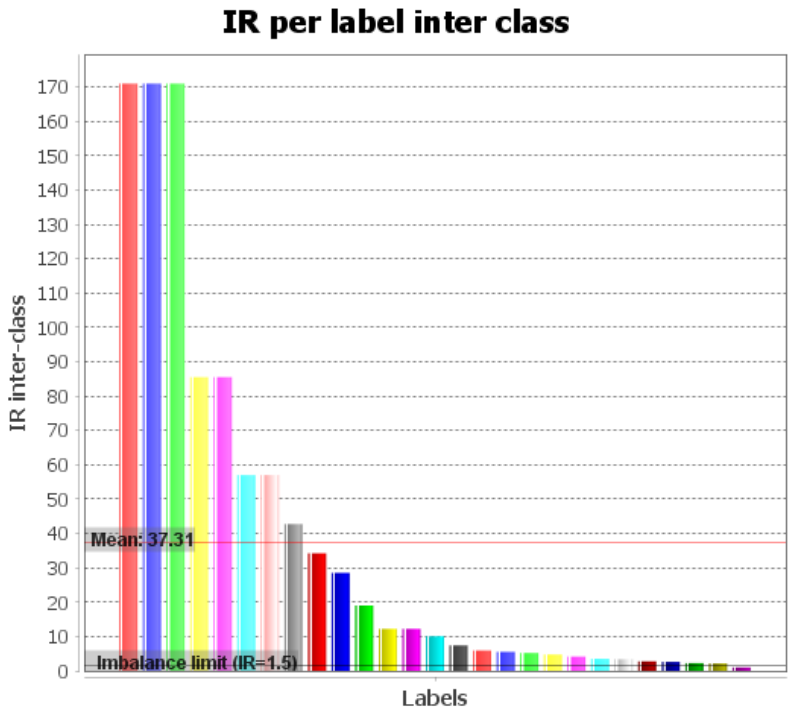

FIGURE 2.19: Example of chart saved in *.png* format

In the dropdown menu, the following options are provided:

- **Label frequency:** on the left side it shows a table with the labels, the number of examples to which it is associated and the frequency of the label. On the right side it shows a bar chart where each bar corresponds to a label and its height indicates the frequency. Besides, it shows the average frequency value as a horizontal line. If the mouse is over a bar, a tooltip message with the label name and frequency is showed (Figure 2.20).

- **Labelset frequency:** this tab shows the frequency of each labelset. So, the left table shows, ordered by frequency, the number of instances associated to each labelset and its frequency. When clicking on each labelset in the table, it appears information about the labelset, showing the number of labels that compose it, the binary coded labelset (1s and 0s shows if each label is associated or not with the labelset respectively), the labelset frequency and the name and frequency of each label included in the labelset (Figure 2.21). In the bar chart of the right side, each bar corresponds to a labelset and the height shows its frequency.

- **Labels histogram:** in this case, the table shows how many examples are associated with a certain number of labels and its frequency. That is to say, it shows how many examples have only one label associated, how many have two labels, how many have three, and so on. The bar chart indicates the frequency of the previous table. If the mouse is over a bar, a tooltip message with the number of associated labels and their frequency is showed (Figure 2.22).

- **Box diagram:** it shows two possible boxplot charts: distribution of the number of examples by label (Figure 2.23) or distribution of number of examples by labelset (Figure 2.24).

- **IR inter class:** the IR inter class shows the imbalance ratio between labels [2]. The table on the left side shows each label name and its IR inter class value, which are sorted by descending IR value. The bar chart shows the IR inter class value for each label. Also it shows a line for the mean IR inter class value and other line for the imbalance bound (Figure 2.25). This bound is fixed to 1.5, and each label whose IR is greater than this bound, is considered to be imbalanced. The IR values in the table are shown in red if they are greater than 1.5. As in previous charts, if the mouse is over a bar, it shows a tooltip message with the label name and its IR inter class value.

- **IR intra class:** IR intra class shows the imbalance ratio inside a label [15]. Left side table shows each label name and its IR intra class value. The labels in the table are sorted by descending IR value. The bar chart shows the IR intra class value for each label. Also it shows a line for the mean IR intra class value and other line for the imbalance limit (Figure 2.26). This limit is also fixed to 1.5. IR values in the table are shown in red if they are greater than 1.5. As in previous charts, if the mouse is over a bar, it shows a tooltip message with the label name and its IR intra class value.

- **IR per labelset:** IR per labelset is similar to IR inter class, but it shows the imbalance ratio between labelsets. Left side table shows each labelset and its IR value. The labelsets in the table are sorted by descending IR value. The bar chart shows the IR value for each labelset, and also it shows a line for the mean IR value (Figure 2.27). IR values in the table are shown in red if they are greater than the imbalance limit fixed to 1.5. As in previous charts, if the mouse is over a bar, it shows a tooltip message with its IR value.

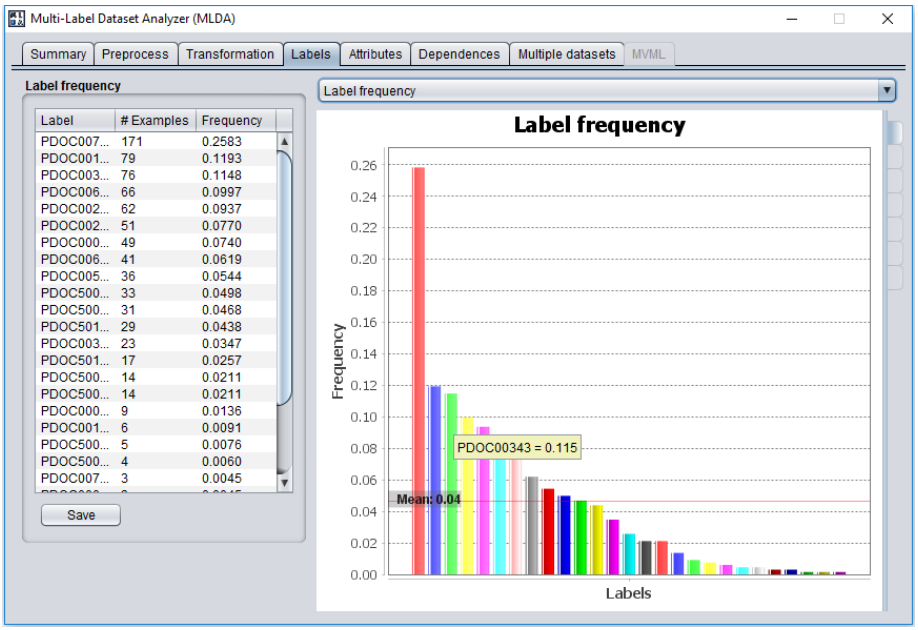

FIGURE 2.20: Example of label frequency chart

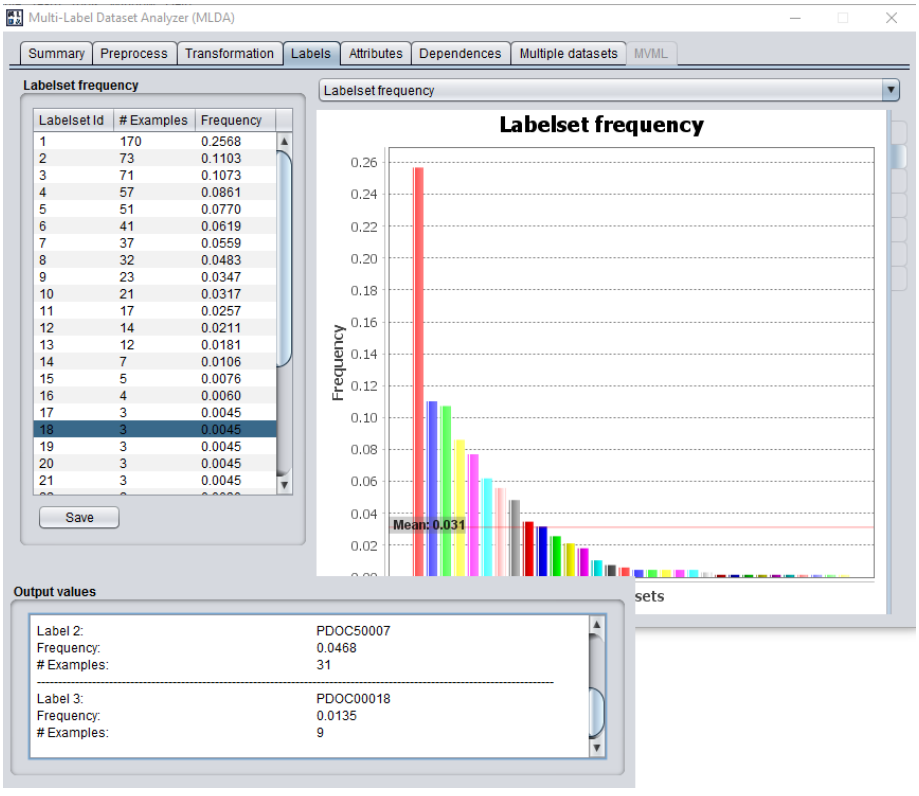

FIGURE 2.21: Example of labelset frequency chart

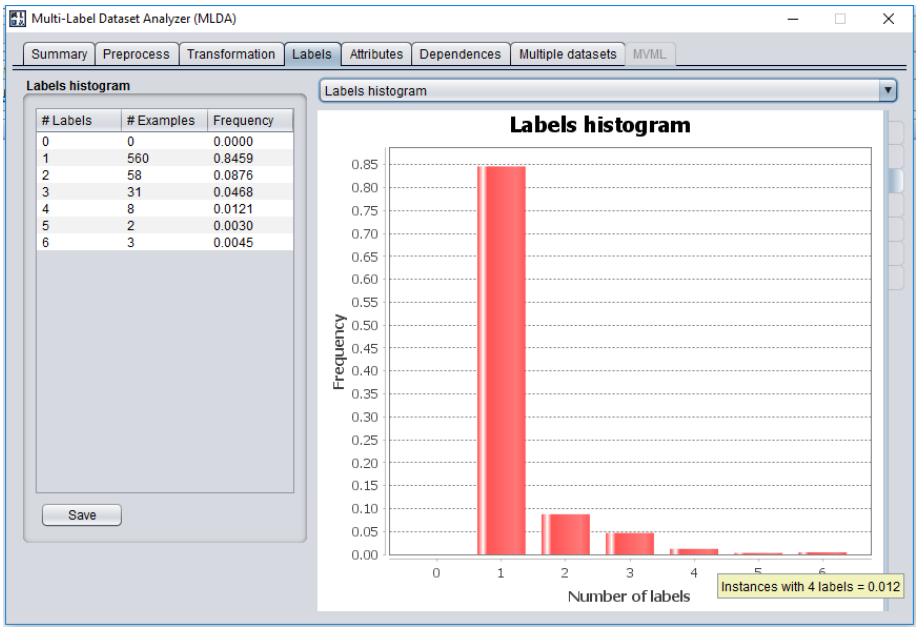

FIGURE 2.22: Example of labels histogram chart

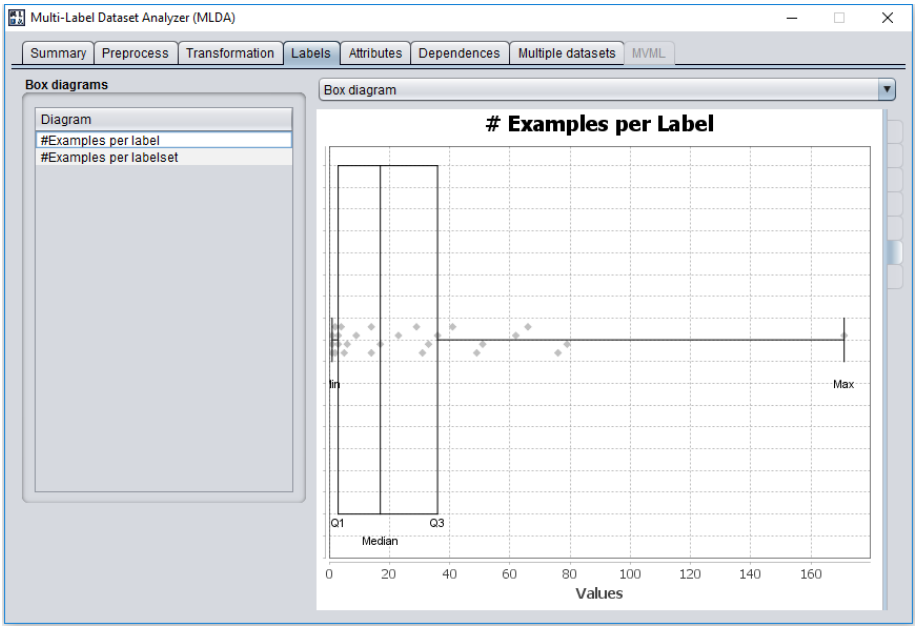

FIGURE 2.23: Example of boxplot for number of examples by label

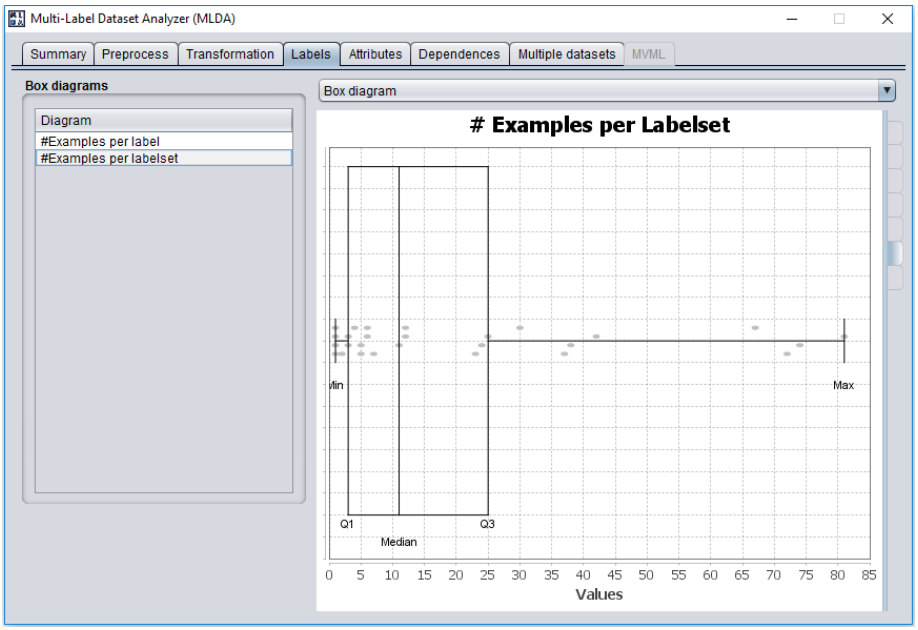

FIGURE 2.24: Example of boxplot for number of examples by labelset

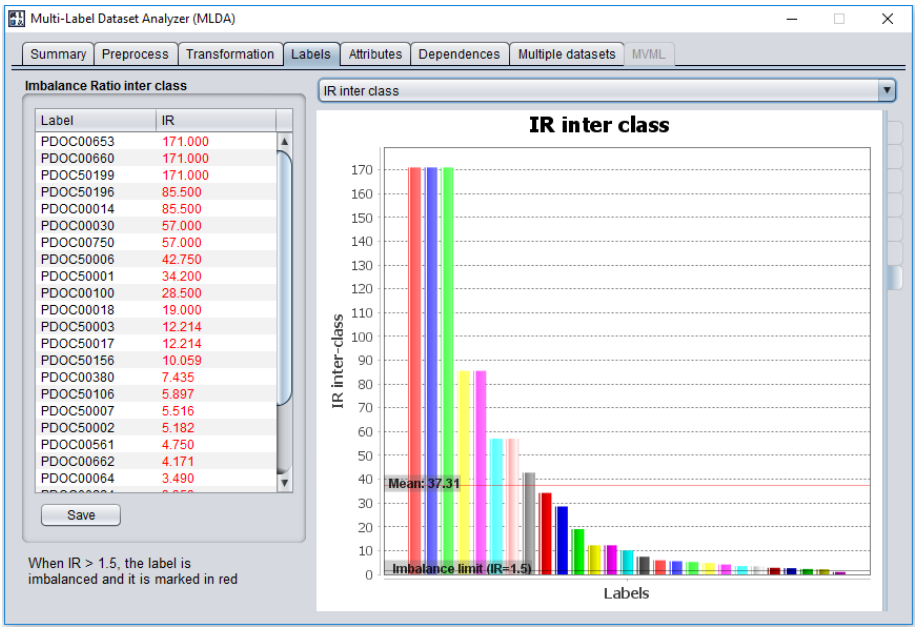

FIGURE 2.25: Example of IR inter class chart

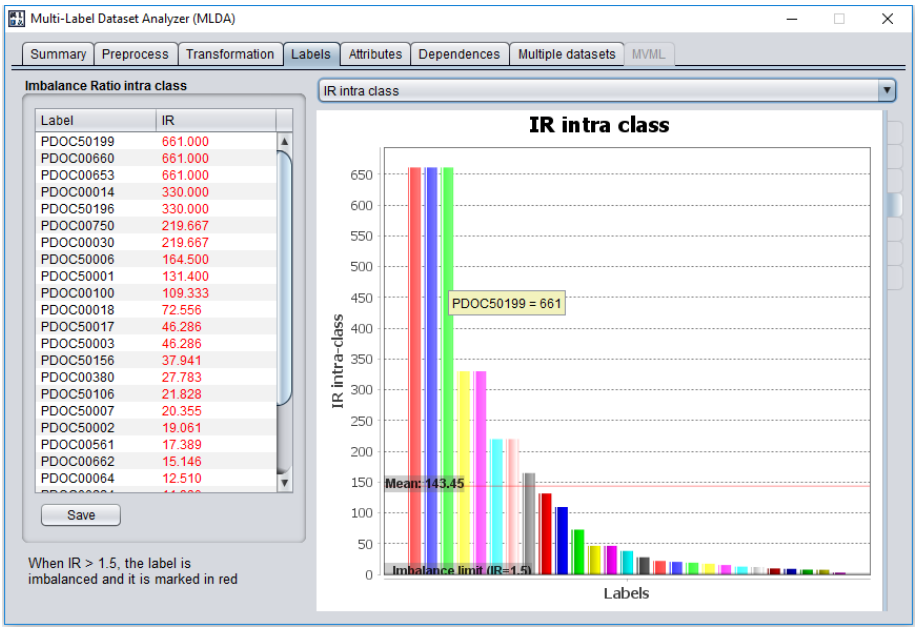

FIGURE 2.26: Example of IR intra class chart

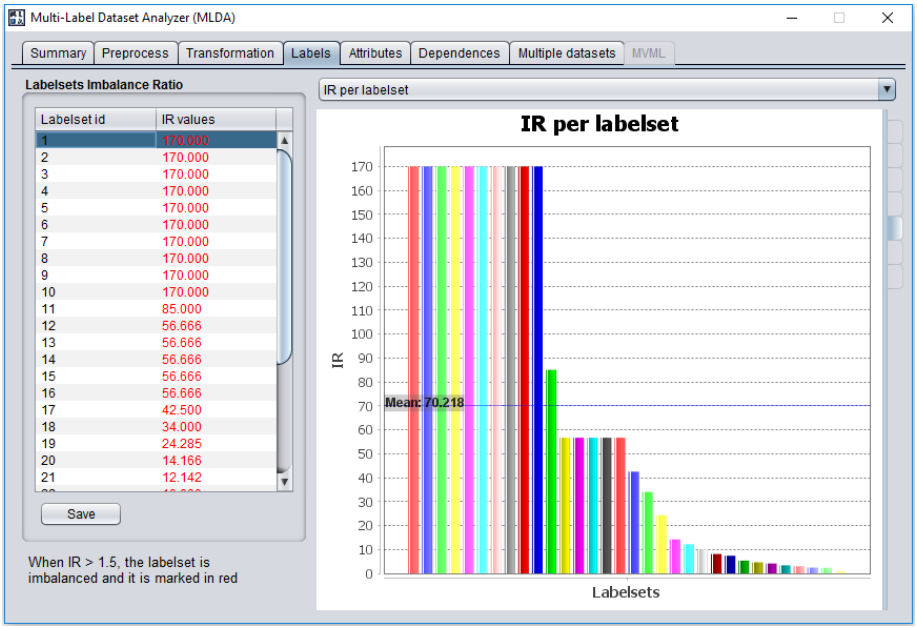

FIGURE 2.27: Example of IR per labelset chart

2.7 Attributes tab

The attributes tab shows boxblot charts for each numeric attribute. As shown in Figure 2.28, on the left side there is a table listing all the numeric attributes, and on the right, there is a boxplot chart showing the values of the selected numeric attribute. When an attribute is selected in the list, a boxplot with its values is showed.

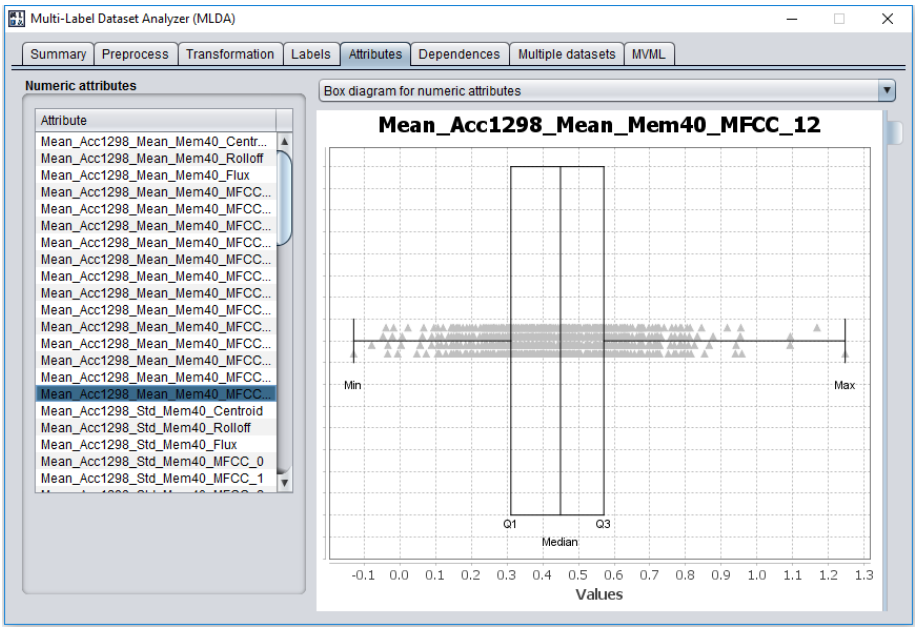

FIGURE 2.28: Example of numeric attributes boxplot

2.8 Dependences tab

The dependences tab shows information about the dependences or relationship between labels. This tab is divided in five sub-tabs, including three dependence types: chi and phi coefficients, co-occurrence and heatmap.

2.8.1 Chi and Phi coefficients

Chi coefficient (χ^2) identifies the unconditional relationship between label pairs, using Chi-square tests between each label pair. If Chi value is greater than 6.635, labels will be considered dependent at 99% confidence [8]. Phi (ϕ) coefficient [5] can be calculated from Chi as $\chi^2 = n \cdot \phi^2$, being n the number of instances. The *Chi & Phi coefficient* sub-tab shows a table with the Chi and Phi values among each pair of labels. As shown in Figure 2.29, Chi values are in white cells and Phi values in gray cells. Also, Chi values which are greater than 6.635 are marked in red, in order to show the dependence between these labels. If the mouse is over a cell, a tooltip message with the label names to which correspond the value is shown. Finally, the table can be saved in *.csv* format by clicking on the *Save* button. It will store a file with two tables, one for each coefficient.

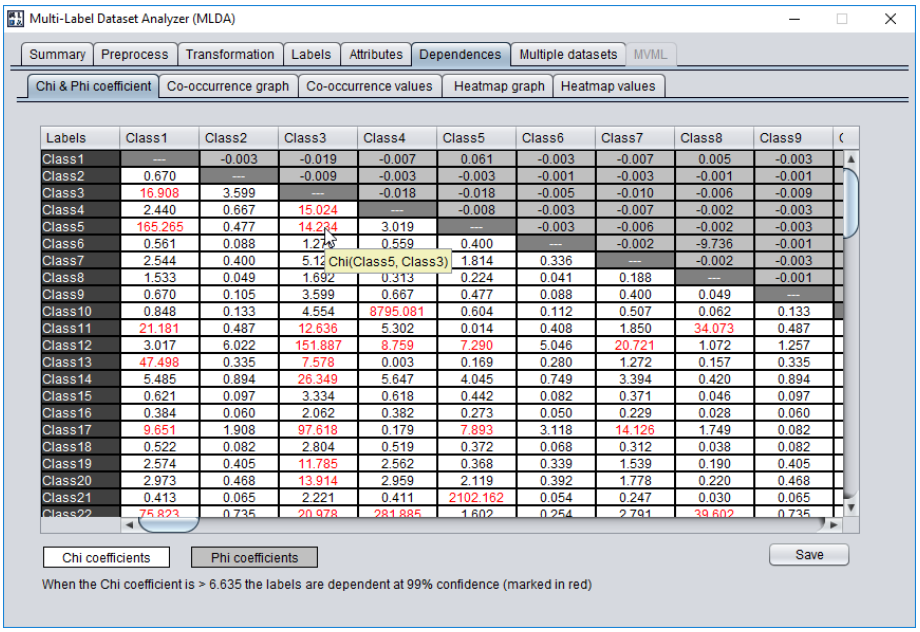

FIGURE 2.29: Chi and Phi coefficients table in Dependences tab

2.8.2 Co-occurrence values and co-ocurrence graphs

The co-occurrence measures how many times a label appears with another. The sub-tab *Co-occurrence values* show the values of co-occurrence between each pair of labels, as shown in Figure 2.30. This table is triangular, because co-occurrence between i and j is the same as between j and i. If the mouse is over a cell, a tooltip message with the labels that correspond to this value is shown. White cells with no value means no co-occurrence, the same as a 0 value. This table could be saved by the *Save* button.

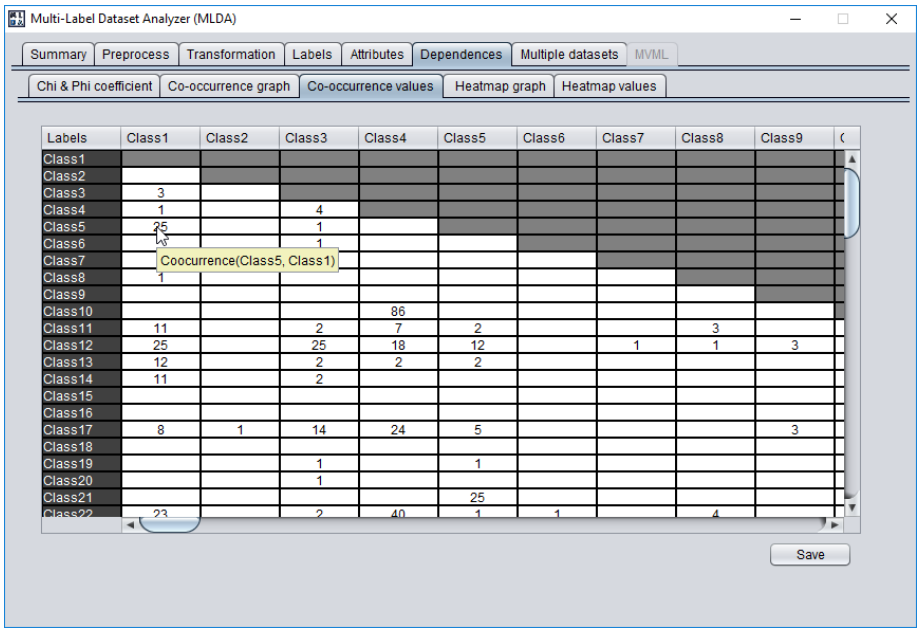

FIGURE 2.30: Co-occurrence values table in Dependences tab

The *Co-occurrence graph* sub-tab shows a graph representing the previous table. In this graph, the label thickness represents the *priori* probability of the current label $P(\lambda_i)$ and the link thickness represents the co-occurrence probability between two labels $P(\lambda_i \wedge \lambda_j)$. On the left side, it is shown a table with the label names and frequencies, which is ordered by descending frequency. Under the table, there are four different options:

- Show selected: creates a graph with the selected labels in the table.
- Show most frequent: creates a graph with the n most frequent labels. The value of n must be less than the number of labels.
- Show most related: creates a graph with the n most related labels. For that, it selects the pairs of labels with higher value of co-occurrence between them. The value of n must be less than the number of labels. This graph is shown by default with the 10 most related labels.
- Show most frequent $\cup$ most related: creates a graph with the union of the n most frequent labels and the n most related labels. The number of labels in the graph may vary between n and $2n$, because some labels could be in both most frequent and most related sets. The value of n must be less than the number of labels.

Figure 2.31 shows an example of co-occurrence graph, selecting the 10 most related labels. As it can be seen, after clicking on the *Show most related* button, the table automatically selects the rows of the labels in the graph. The graph may be modified, moving the nodes or resizing them. Then, the *Save* button allows us to save the graph in *.png* format.

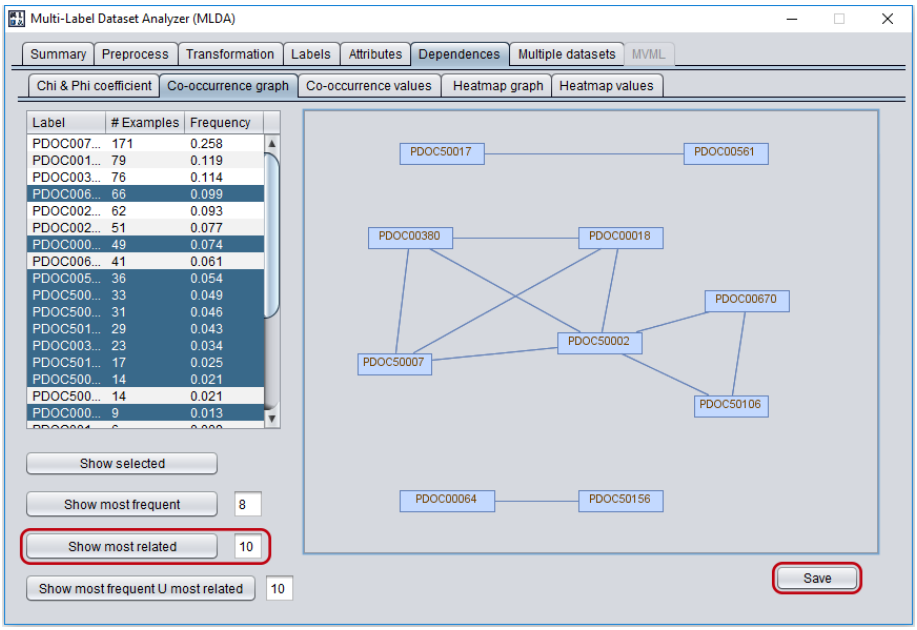

FIGURE 2.31: Co-occurrence graph for 10 most related labels in Dependences tab

2.8.3 Conditional probabilities and heatmap graphs

Heatmap table in *Heatmap values* sub-tab also represents the relationship among labels. Each cell in the table or matrix $\mathbf{M}$ represents the conditional probability, being $\mathbf{M}_{ij} = P(\lambda_i|\lambda_j) = P(\lambda_i \wedge \lambda_j)/P(\lambda_j)$, where i is a row and j a column. The diagonal (gray background) represents the *a priori* probabilities $\mathbf{M}_{jj} = P(\lambda_j)$. White cells with no value represents a probability of 0.0. And as in previous cases, if the mouse is over a cell, it shows a tooltip message with the labels that correspond to this value. This table could be saved by the *Save* button (Figure 2.32).

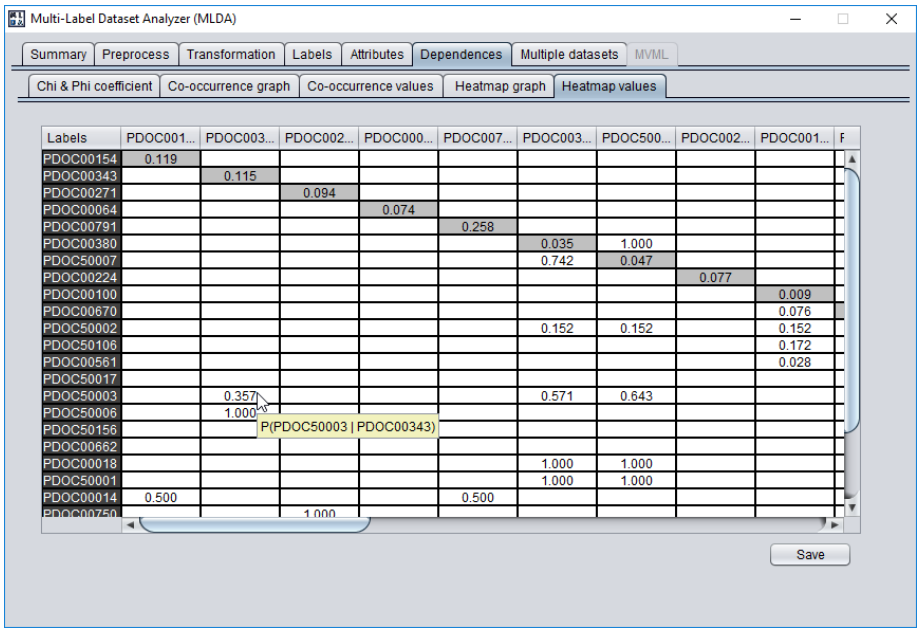

FIGURE 2.32: Heatmap values table in Dependences tab

The *Heatmap graph* sub-tab shows a graph representing the previous heatmap matrix. In this graph, probabilities are represented in grayscale, where black is a conditional probability of 0.0 and white is a conditional probability of 1.0, being the gray colors values between these. The diagonal running from the bottom left to the upper right corner represents the *prior* probabilities. On the left side, it is shown a table with the label names and frequencies, which is ordered by descending frequency. Under the table, there are four different options:

- Show selected: creates a graph with the selected labels in the table.
- Show most frequent: creates a graph with the n most frequent labels. The value of n must be less than the number of labels.
- Show most related: creates a graph with the n most related labels. For that, it selects the pairs of labels with higher value of conditional probability. The value of n must be less than the number of labels.
- Show most frequent $\cup$ most related: creates a graph with the union of the n most frequent labels and the n most related labels. The number of labels in the graph may vary between n and $2n$, because some labels could be in both most frequent and most related sets. The value of n must be less than the number of labels.

Figure 2.33 shows an example of heatmap. It has been created with all the labels (default), in order to prove that this type of graph is better when the number of labels is high. Finally, the *Save* button allows to save the graph in *.png* format.

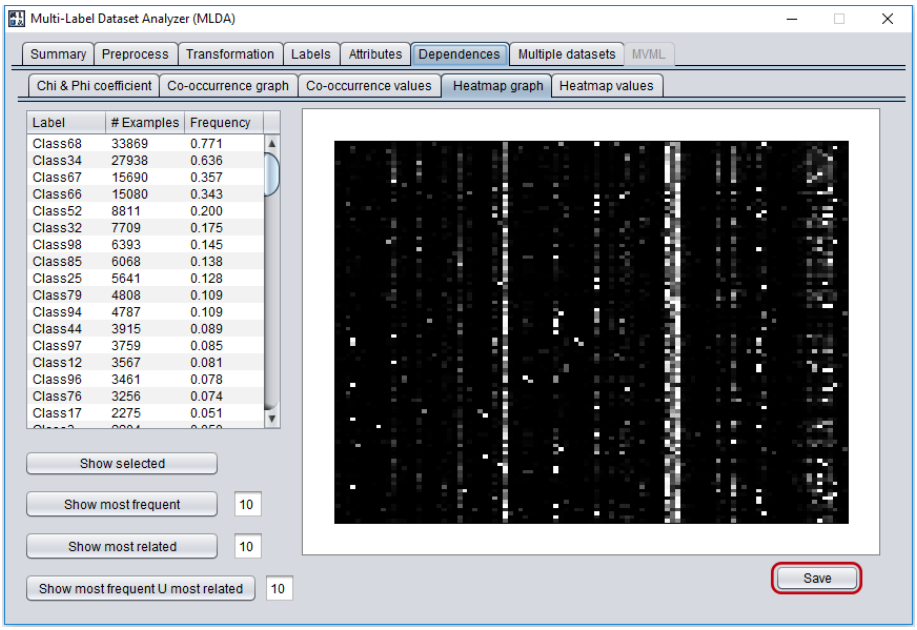

FIGURE 2.33: Heatmap graph in Dependences tab

2.9 Multiple datasets

The multiple datasets tab allows to load several datasets simultaneously, in order to calculate a set of metrics for all of them. As shown in Figure 2.34, on the left side there is an empty list and two buttons: *Add* and *Remove*. To add datasets to the list, we have to click on the *Add* button and select one or more datasets (Figure 2.35). We can add as many datasets as we want. Also, with the *Remove* button, the current selected dataset could be removed from the list.

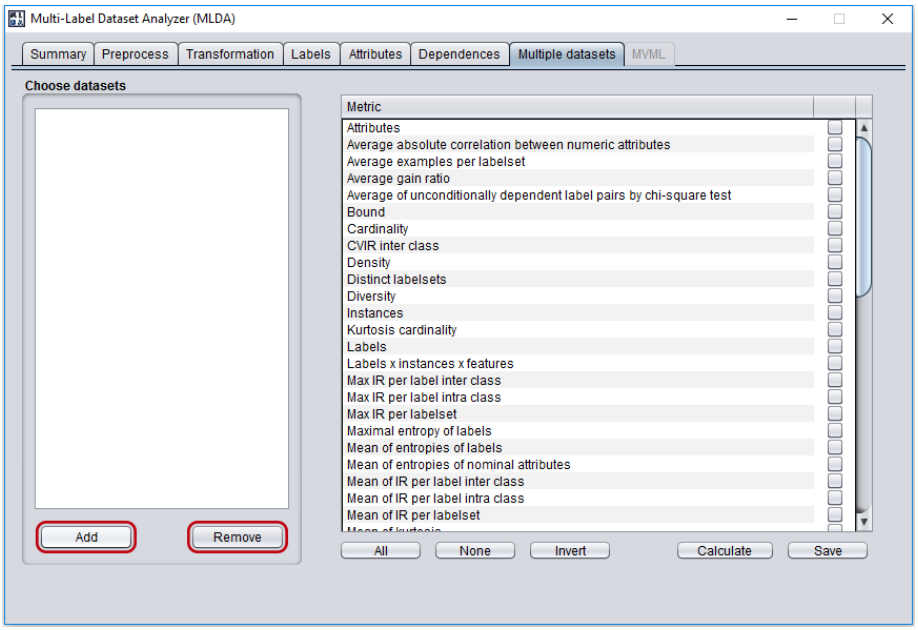

FIGURE 2.34: Add/remove datasets in multiple datasets tab

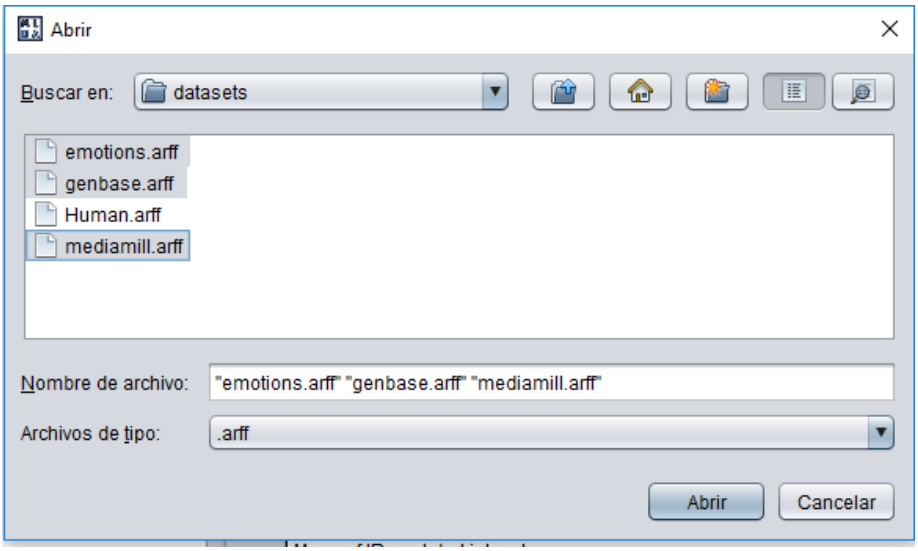

FIGURE 2.35: Adding datasets in multiple datasets tab

Once the datasets have been added, on the right side there is a table with metrics as in the summary tab. A set of metrics can be selected and calculated clicking on the *Calculate* button. Then, the metric values can be saved with the *Save* button in *.csv*, *.arff*, *.txt* and *.tex* formats (Figure 2.36).

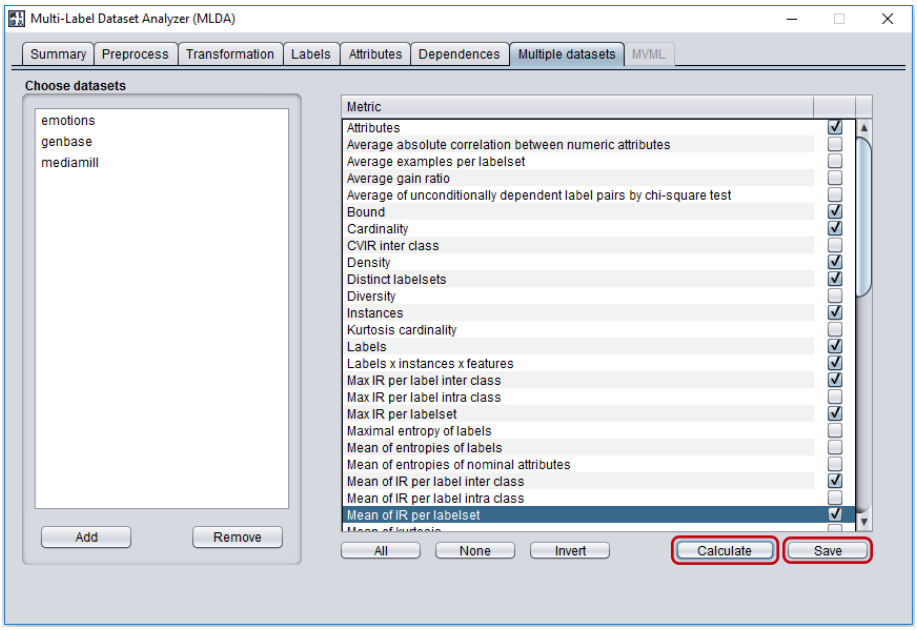

FIGURE 2.36: Calculating metrics for multiple datasets

2.10 MVML

The MVML tab gives an extra utility to datasets which are multi-view multi-label (MVML). Multi-view datasets are those in which the input space is partitioned in some views. This tab is disabled if the dataset is not MVML.

The difference in format between a multi-label dataset and a MVML dataset is in its header. The *@relation* clause in the *.arff* file includes a `-V` parameter indicating the views. For example, this could be the header of a MVML *.arff* file: `@relation 'relationName -V:0-31!32-63!64-71'`. The format for MVML dataset was described in Section 2.2.2.

As shown in Figure 2.37, this tab at the top shows some basic metrics for multi-view datasets, for instance the number of views and the minimum, maximum and average number of attributes per view. On the bottom, the tool shows a table with the different views, giving for each one the number of attributes and some metrics like LxIxF, ratio of number of instances to the number of attributes or average gain ratio. By selecting views on the table, it can be seen a list with the attribute names of the selected views. This tab allows to save both the table with the metrics for each view and the list of attributes for each selected view by clicking on the *Save table* button. Finally, the tool also allows to save a new multi-view multi-label dataset only with the selected views in the table. For that, we only have to select the format to save (Mulan or Meka) and click on *Save views* (Figure 2.37).

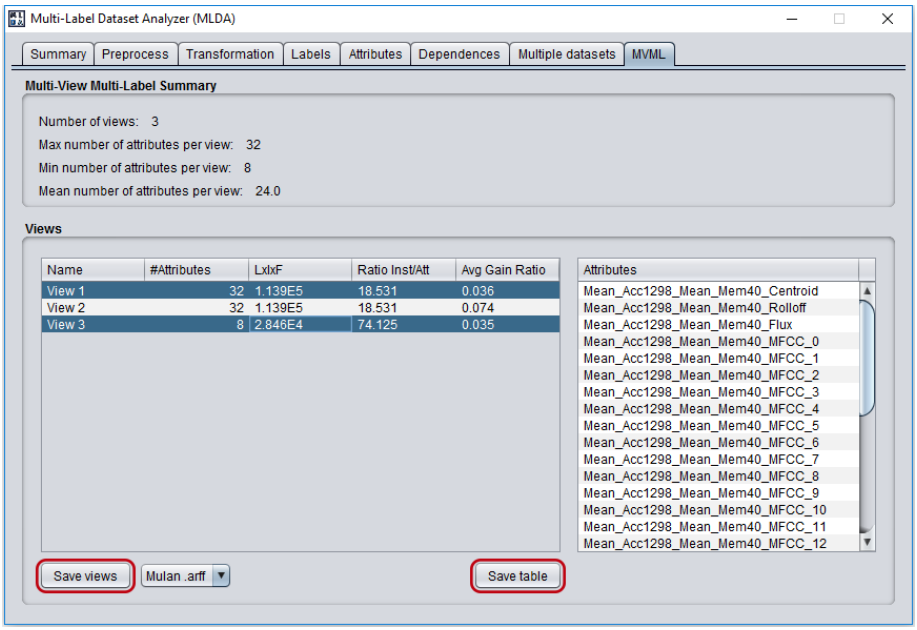

FIGURE 2.37: Saving views and table in MVML tab

Chapter 3

API

3.1 Metrics

As indicated previously, the set of metrics included in the API has been selected starting with the taxonomy proposed in [3], being extended with the metrics of both Mulan and Meka, and also the proposed metrics in [4]. Thus, the taxonomy is composed of 5 groups of metrics: dimensionality, labels distribution, labels imbalance, relationship among labels and attributes. Figure 3.1 shows the taxonomy and metrics composing each group.

3.2 Structure

The API have been implemented in Java, built over Mulan and Weka libraries. It is divided into the following packages:

- **base:** it includes *MLDataMetric* and *MLDataCharacterization*, the main classes for dataset characterization. The former is the base implementation for any implemented metric, including the *calculate()* method, which calculates the metric value. The latter is useful to calculate a set of metrics instead only one.
- **attributes:** it includes the implementations of attribute metrics.
- **dimensionality:** it includes the implementations of dimensionality metrics.
- **imbalance:** it includes the implementations of imbalance metrics.
- **labelsDistribution:** it includes the implementations of labels distribution metrics.
- **labelsRelation:** it includes the implementations of labels relationship metrics.
- **metricsName:** it includes the names of all implemented metrics.
- **util:** it includes some necessary methods for metrics calculation.

Figure 3.2 shows the API class diagram. The classes that implements the metrics inherits from *MLDataMetric* class, from *base* package. For simplicity, all the metrics are not included in the diagram, but they are included the necessary classes to understand the structure. The *labelsRelation* package

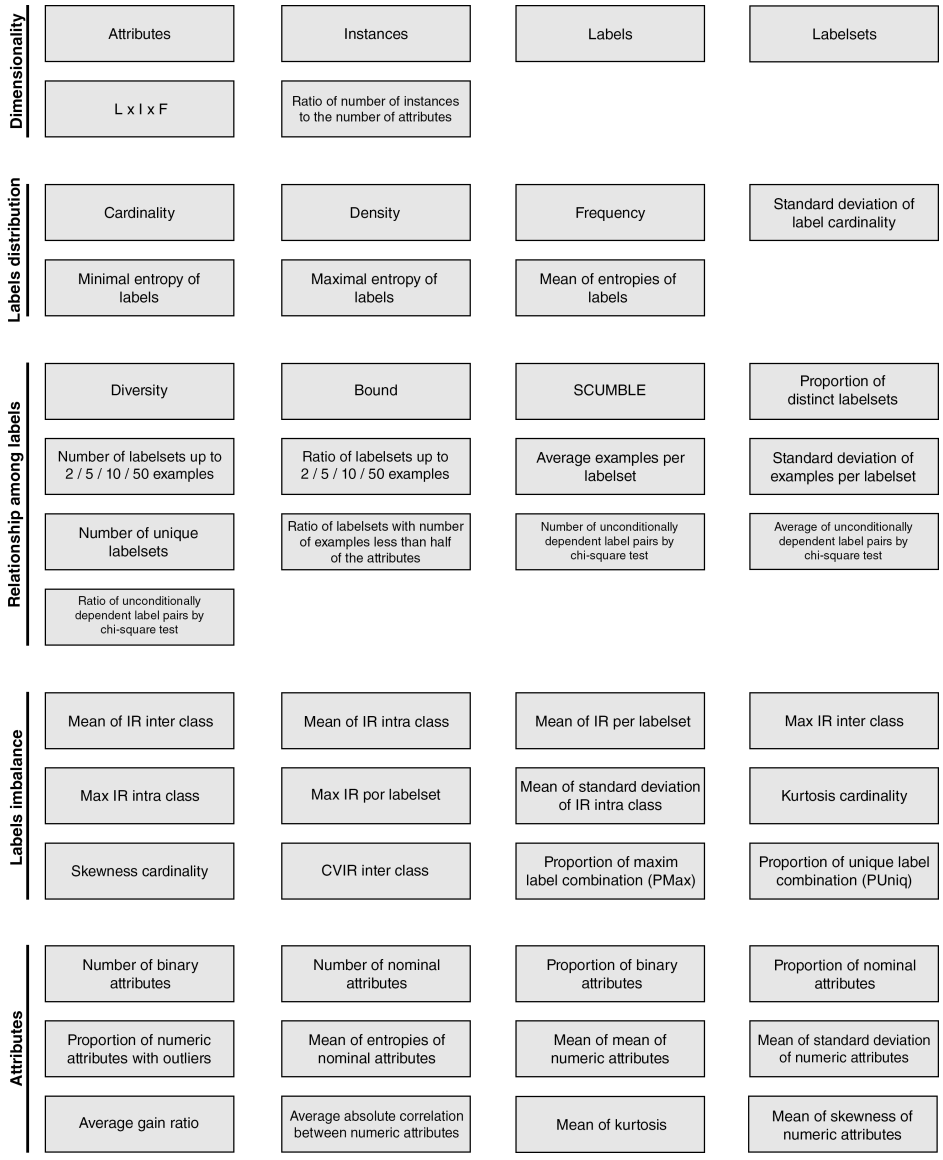

FIGURE 3.1: Taxonomy of metrics for MLL datasets characterization

has a generic class *LabelsetsUpToNExamples*, from which inherits 4 predefined classes for $n = 2$, 5, 10 and 50. On the other hand, *imbalance* package includes the *ImbalanceDataMetric* class, inheriting from *MLDataMetric* and from which inherits all the imbalance metrics. *ImbalanceDataMetric* includes a new field of type *ImbalancedFeature* in order to store the characteristics of imbalanced features. Table 3.1 shows the correspondence between each metric and the class that implements it.

3.3 Usage

The API can be downloaded from `https://github.com/i02momuj/MLDA/tree/master/bin`. Once downloaded, it has to be included in the Java project. This API has two main goals: calculate one characterization metric for a multi-label dataset or calculate a set of metrics for the same dataset.

To calculate one metric, an object of the desired metric have to be created. After creating the metric object, the *calculate()* method have to be called with the multi-label dataset as parameter. The metric value can be obtained in two ways: getting the returned value of the *calculate()* method, or accessing to the metric value with the *getValue()* method. Figure 3.3 shows an example for calculating one metric.

On the other hand, to calculate a set of metrics instead of only one, the API includes the *MLDataCharacterization* class. To create a *MLDataCharacterization* object it just need the multi-label dataset as parameter. Then, the metrics are added with *addMetric()* or *addMetrics()* methods, passing as parameter a metric or a list of metrics respectively. Once calculated with *calculateMetrics()* method, the *getMetric()* method returns a metric of the list identified by its name. The *getAvailableMetrics()* method returns a set with the names of all available metrics. Figure 3.4 shows an example of how to calculate some metrics for a dataset.

TABLE 3.1: Correspondencia entre métricas y clase que la implementa

Metric	Class
Dimensionality	
Attributes	`Attributes`
Labels	`Labels`
Instances	`Instances`
Labelsets	`DistinctLabelsets`
LxIxF	`LxIxF`
Ratio of number of instances to the number of attributes	`RatioInstancesToAttributes`
Labels distribution	
Cardinality	`Cardinality`
Density	`Density`
Maximal entropy of labels	`MaxEntropy`
Mean of entropies of labels	`MeanEntropy`
Minimal entropy of labels	`MinEntropy`
Standard deviaton of label cardinality	`StdvCardinality`
Relationship among labels	
Average examples per labelset	`AvgExamplesPerLabelset`
Average of unconditionally dependent label pairs by chi-square test	`AvgUnconditionalDependentLabelPairsByChiSquare`
Bound	`Bound`
Diversity	`Diversity`
Number of labelsets up to 2 examples	`LabelsetsUpTo2Examples`
Number of labelsets up to 5 examples	`LabelsetsUpTo5Examples`
Number of labelsets up to 10 examples	`LabelsetsUpTo10Examples`
Number of labelsets up to 50 examples	`LabelsetsUpTo50Examples`
Number of labelsets up to N examples	`LabelsetsUpToNExamples`
Number of unconditionally dependent label pairs by chi-square test	`NumUnconditionalDependentLabelPairsByChiSquare`
Proportion of distinct labelsets	`ProportionDistinctLabelsets`
Ratio of labelsets up to 2 examples	`RatioLabelsetsUpTo2Examples`
Ratio of labelsets up to 5 examples	`RatioLabelsetsUpTo5Examples`
Ratio of labelsets up to 10 examples	`RatioLabelsetsUpTo10Examples`
Ratio of labelsets up to 50 examples	`RatioLabelsetsUpTo50Examples`
Ratio of labelsets up to N examples	`RatioLabelsetsUpToNExamples`
Ratio of labelsets with number of examples less than half of the attributes	`RatioLabelsetsWithExamplesLessThanHalfAttributes`
Ratio of unconditionally dependent label pairs by chi-square test	`RatioUnconditionalDependentLabelPairsByChiSquare`
SCUMBLE	`SCUMBLE`
Standard deviation of examples per labelset	`StdvExamplesPerLabelset`
Number of unique labelsets	`UniqueLabelsets`
Labels imbalance	
CVIR inter class	`CVIRInterClass`
Kurtosis Cardinality	`KurtosisCardinality`
Max IR inter class	`MaxIRInterClass`
Max IR intra class	`MaxIRIntraClass`
Max IR per labelset	`MaxIRLabelset`
Mean of IR inter class	`MeanIRInterClass`
Mean of IR intra class	`MeanIRIntraClass`
Mean of IR per labelset	`MeanIRLabelset`
Mean of standard deviation of IR intra class	`MeanStdvIRIntraClass`
Proportion of maxim label combination (PMax)	`PMax`
Proportion of unique label combination (Puniq)	`Puniq`
Skewness cardinality	`SkewnessCardinality`
Attributes	
Average absolute correlation between numeric attributes	`AvgAbsoluteCorrelationBetweenNumericAttributes`
Average gain ratio	`AvgGainRatio`
Number of binary attributes	`BinaryAttributes`
Number of nominal attributes	`NominalAttributes`
Number of numeric attributes	`NumericAttributes`
Mean of entropies of nominal attributes	`MeanEntropiesNominalAttributes`
Mean of mean of numeric attributes	`MeanOfMeanOfNumericAttributes`
Mean of standard deviation of numeric attributes	`MeanStdvNumericAttributes`
Proportion of binary attributes	`ProportionBinaryAttributes`
Proportion of nominal attributes	`ProportionNominalAttributes`
Proportion of numeric attributes	`ProportionNumericAttributes`
Proportion of numeric attributes with outliers	`ProportionNumericAttributesWithOutliers`
Mean of kurtosis	`MeanKurtosis`
Mean of skewness of numeric attributes	`MeanSkewnessNumericAttributes`

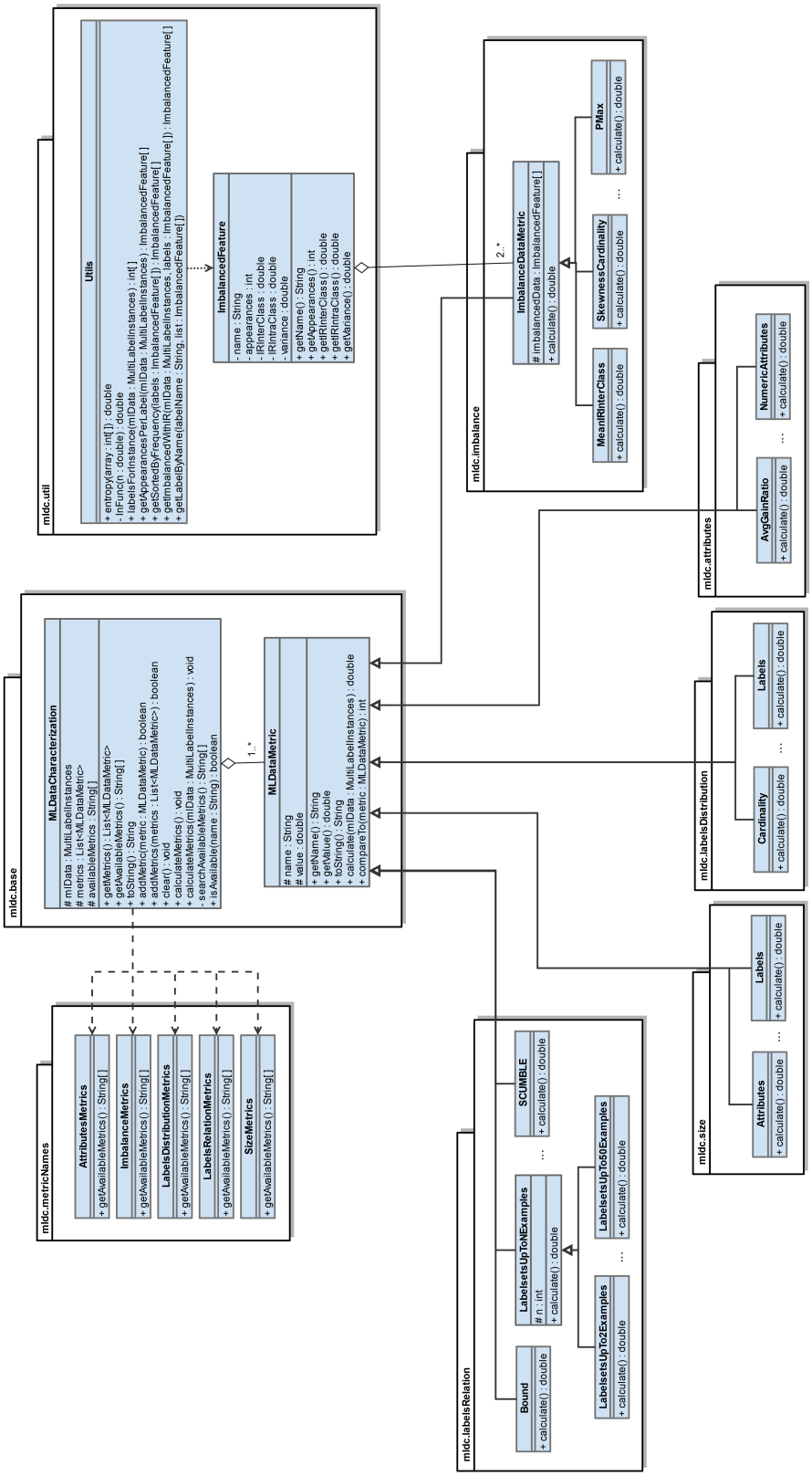

FIGURE 3.2: API class diagram

```java
//Creating the object corresponding to the metric
Density density = new Density();

//Calculating metric value
double value = density.calculate(mlData);

//Other way to get the metric value
 //After calling calculate() method
double value2 = density.getValue();
```

FIGURE 3.3: Calculating one metric

```java
//Creating object MLDataCharacterization
MLDataCharacterization mldc = new MLDataCharacterization(mlData);

//Including metrics with addMetrics method
ArrayList<MLDataMetric> m = new ArrayList<>();
m.add(new Attributes());
m.add(new Labels());
m.add(new Instances());
mldc.addMetrics(m);

//Including metrics with addMetric method
mldc.addMetric(new Cardinality());
mldc.addMetric(new Density());

//Calculating
mldc.calculateMetrics();

//Getting values
double attributes = mldc.getMetrc("Attributes").getValue();
double labels = mldc.getMetric("Labels").getValue();
double instances = mldc.getMetric("Instances").getValue();
double cardinality = mldc.getMetric("Cardinality").getValue();
double density = mldc.getMetric("Density").getValue();
```

FIGURE 3.4: Calculating some metrics

References

[1] M Boutell et al. "Learning multi-label scene classification". In: *Pattern recognition* 37 (2004), pp. 1757–1771.

[2] F. Charte et al. "A first approach to deal with imbalance in multi-label datasets". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8073 LNAI (2013), pp. 150–160. DOI: 10.1007/978-3-642-40846-5_16.

[3] Francisco Charte and David Charte. "Working with Multilabel Datasets in R: The mldr Package". In: *The R Journal* 7.2 (2015), pp. 149–162.

[4] L Chekina, L Rokach, and B Shapira. "Meta-learning for selecting a multi-label classification algorithm". In: 2011, pp. 220–227. DOI: 10.1109/ICDMW.2011.118.

[5] Jacob Cohen et al. *Applied Multiple Regression / Correlation Analysis for the Behavioral Sciences*. 2002.

[6] R. A. Fisher. "The use of multiple measurements in taxonomic problems". In: *Annals of Eugenics* 7 (1936), pp. 179–188.

[7] Eva Gibaja and Sebastian Ventura. "A tutorial on multilabel learning". In: *ACM Computing Surveys* 47.3 (2015). DOI: 10.1145/2716262.

[8] Priscilia E Greenwood and Michael S Nikulin. "A guide to chi-squared testing". In: *Wiley-Interscience* 280 (1996).

[9] Mark Hall et al. "The WEKA Data Mining Software: An Update". In: *SIGKDD Explor. Newsl.* 11.1 (2009), pp. 10–18. ISSN: 1931-0145. DOI: 10.1145/1656274.1656278.

[10] E Loza and J Fürnkranz. "Efficient multilabel classification algorithms for large-scale problems in the legal domain". In: *Semantic Processing of Legal Texts*. Vol. 6036. 2010, pp. 192–215.

[11] Y Luo et al. "Multiview vector-valued manifold regularization for multilabel image classification". In: *Neural Networks and Learning Systems, IEEE Transactions on*. Vol. 24(5). 2013, pp. 709–722.

[12] *MEKA: A Multi-label Extension to WEKA*. http://meka.sourceforge.net/. Último acceso: 21-04-2016.

[13] K. Sechidis, G. Tsoumakas, and I. Vlahavas. "On the stratification of multi-label data". In: *Lecture Notes in Computer Science* 6913 LNAI.PART 3 (2011), pp. 145–158. DOI: 10.1007/978-3-642-23808-6_10.

[14] H Shao et al. "Symptom selection for multi-label data of inquiry diagnosis in traditional Chinese medicine". In: *Sci China Ser F-Info Sci* 1 (2010), pp. 1–13.

[15] E Stamatatos. "Author identification: Using text sampling to handle the class imbalance problem". In: *Information Processing and Management* 44.2 (2008), pp. 790–799. DOI: 10.1016/j.ipm.2007.05.012.

[16] L Tang and H Liu. “Scalable learning of collective behavior based on sparse social dimensions”. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. New York, NY, USA, 2009, pp. 1107–1116.

[17] G Tsoumakas, I Katakis, and I Vlahavas. “Data Mining and Knowledge Discovery Handbook, Part 6”. In: Springer, 2010. Chap. Mining Multi-label Data, pp. 667–685.

[18] G. Tsoumakas et al. “Mulan: A Java Library for Multi-Label Learning”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2411–2414.

[19] Chang Xu, Dacheng Tao, and Chao Xu. “A Survey on Multi-view Learning”. In: *CoRR* abs/1304.5634 (2013). URL: http://arxiv.org/abs/1304.5634.

[20] Y Zhang et al. “Ensemble pruning via semi-definite programming”. In: *Journal of Machine Learning Research* 7 (2006), pp. 1315–1338.