# AMUSE (Advanced MUSic Explorer) User Manual

Igor Vatolkin

TU Dortmund,
Chair of Algorithm Engineering (Ls11),
Otto-Hahn-Str. 14,
44227 Dortmund, Germany
`igor.vatolkin@udo.edu`
`http://ls11-www.cs.tu-dortmund.de/`

## 1 Introduction

### 1.1 License Notes

AMUSE is an open-source Java framework for Music Information Retrieval tasks. It is developed within research group Computational Intelligence (Prof. Dr. Günter Rudolph) on the Chair of Algorithm Engineering at the Technical University of Dortmund.

### 1.2 Remarks about the Current Version (Beta 0.1)

Until now AMUSE was mainly developed for several cooperation projects and empirical studies published in [1–6]. Therefore we cannot guarantee that AMUSE will run without any problems on your machine. From August 2010 AMUSE is available on SourceForge and the current state is Beta-version 0.1. You are welcome to try it for your research and provide us the information about your experience with AMUSE.

This manual is CURRENTLY UNDER DEVELOPMENT and does not contain the complete information how to use AMUSE.

Please contact us, if you encounter any problems or bugs. We plan to release the version 0.1 during this year (2010). Developer manual will be also available during the fall of 2010. Please subscribe to the email-list (`https://lists.`

`sourceforge.net/lists/listinfo/amuse-framework-news`) if you wish to be informed about new development issues.

If you use AMUSE for your research, please cite the publication [7]

### 1.3 Requirements

You should have a 1.6 Java Development Kit or Java Runtime Environment installed on your system.

Some of the AMUSE components use Matlab, however it is not required to run AMUSE in general.

We have tested the current version on Ubuntu Unix, Windows and MacOS systems.

## 2 Installation Steps

You can get the current version from SourceForge SVN repository either using the command

```
svn co https://amuse-framework.svn.sourceforge.net/svnroot/
amuse-framework amuse-framework
```

or checking it out as project in your development environment (e.g. Eclipse) from

```
https://amuse-framework.svn.sourceforge.net/svnroot/
amuse-framework.
```

Before starting of AMUSE for the first time, set the environment variable AMUSEHOME by editing the file amuseGUI.bat (for Windows) or amuseGUI.sh (for Unix). Check if the sh-script is executable. If you cannot start java just using a “java” command from your command line, please provide the path to it in these scripts.

Now run amuseGUI.bat or amuseGUI.sh. A start screen will appear, cf. Fig. 1.

Click on the button “Edit AMUSE Settings” for the further setup.

- **Path Settings:** Here the path to AMUSE folder must be set as well as several folders for AMUSE input/output data (called AMUSE databases). Music Database is a folder with music files (mp3s or waves) - however it is also possible to work with the music files from other directories on your machine. Features are stored in Feature Database, processed features in Processed Feature Database. Category Database is a folder for ground truth information, however here a path to an ARFF with the list of defined categories must be set. In Model Database the classification models are saved. The metrics estimated during the validation routine are saved in Metric Database, and the optimization data is logged to Optimization Database.

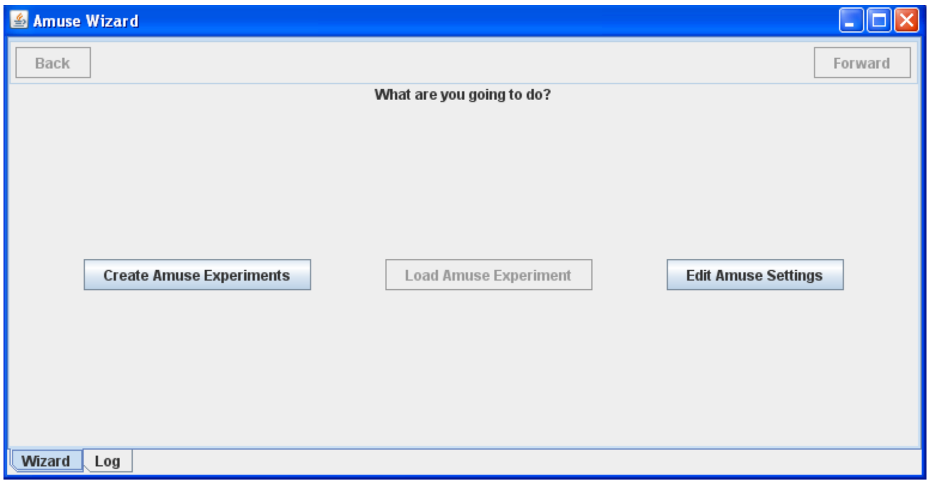


**Fig. 1.** AMUSE Start Screen.

- **External Tools:** Here you can type the paths to your Java and Matlab executables.
- **Wave-Sampling Settings:** The default option is to downsample the given mp3s or waves to 22050 Hz and run stereo to mono conversion.
- **Miscellaneous Settings:** Log level defines the minimal level of the messages which will be displayed in the log window. Setting this level to debug can help to see more information if any problems occur. Maximal number of task threads can be changed if you want to take usage of several processing units in your machine.

You can run AMUSE from amuseGUI.bat or amuseGUI.sh or from your development environment - in the last case you should use AMUSE.SCHEDULER.GUI.CONTROLLER.WIZARDCONTROLLER as a main class and the environment variable AMUSEHOME must be set to the path of your AMUSE folder.

## 3 Design of Experiments with AMUSE

After the successful initial setup of AMUSE you can start to design your experiments. Please click on the button "Create Amuse Experiments" from the start screen. Now you can add experiments to the experiment table, cf. Fig. 2.

The AMUSE components and tasks are illustrated in Fig. 3.

### 3.1 Feature Extraction

Feature extraction screenshot is given in Fig. 4.

On the left panel you can select the music files. On the right panel, you can enable/disable the audio signal features which should be extracted. The file and

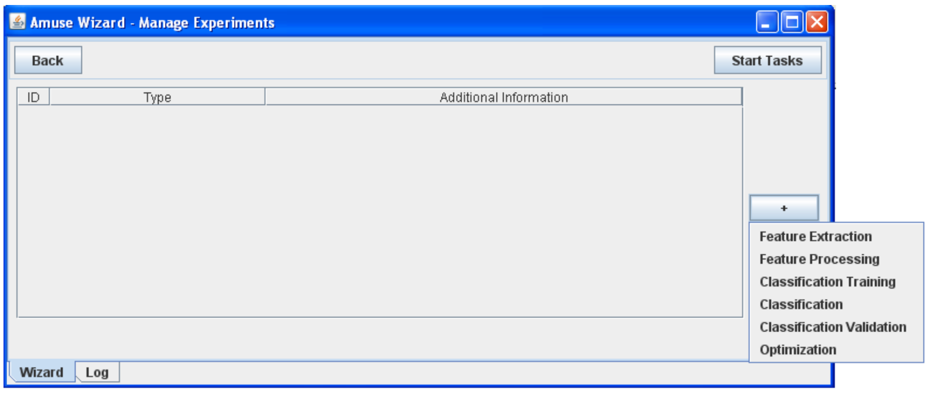


**Fig. 2.** Experiment Table.

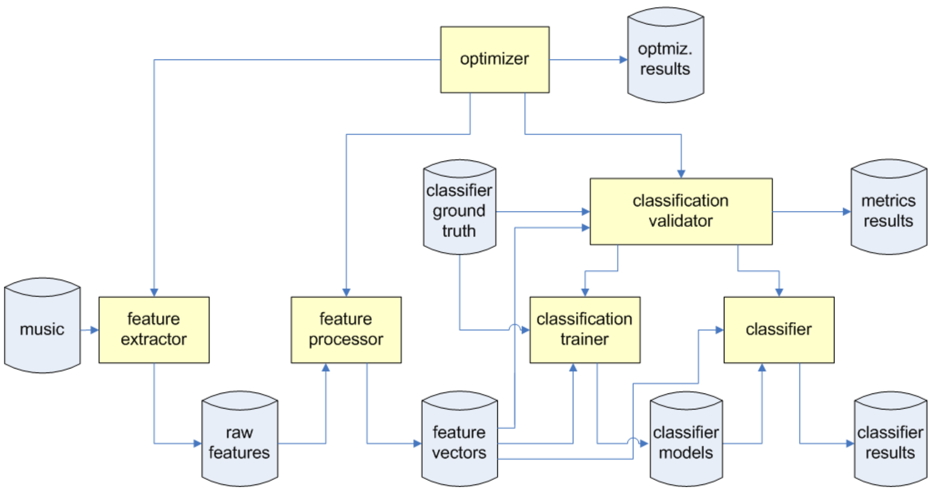
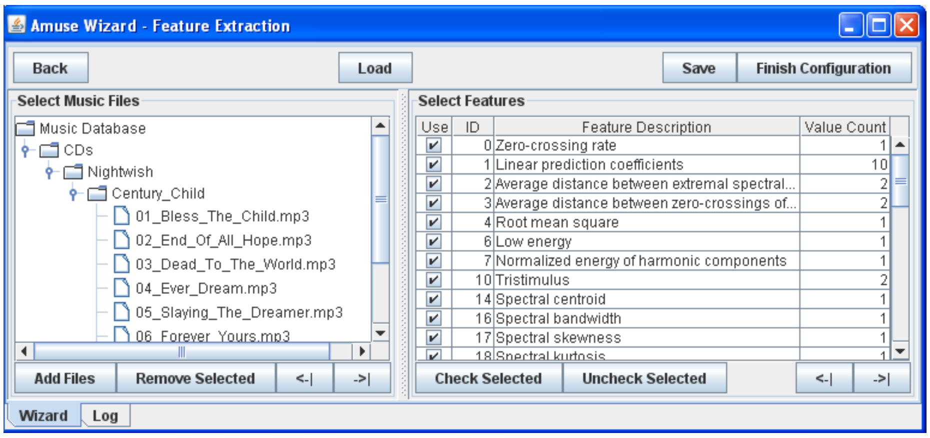


**Fig. 3.** AMUSE Tasks.



- Music Database
  - CDs
    - Nightwish
      - Century_Child
        - 01_Bless_The_Child.mp3
        - 02_End_Of_All_Hope.mp3
        - 03_Dead_To_The_World.mp3
        - 04_Ever_Dream.mp3
        - 05_Slaying_The_Dreamer.mp3
        - 06_Forever_Yours.mp3

| Use | ID | Feature Description | Value Count |
|---|---|---|---|
| ✓ | 0 | Zero-crossing rate | 1 |
| ✓ | 1 | Linear prediction coefficients | 10 |
| ✓ | 2 | Average distance between extremal spectral... | 2 |
| ✓ | 3 | Average distance between zero-crossings of... | 2 |
| ✓ | 4 | Root mean square | 1 |
| ✓ | 6 | Low energy | 1 |
| ✓ | 7 | Normalized energy of harmonic components | 1 |
| ✓ | 10 | Tristimulus | 2 |
| ✓ | 14 | Spectral centroid | 1 |
| ✓ | 16 | Spectral bandwidth | 1 |
| ✓ | 17 | Spectral skewness | 1 |
| ✓ | 18 | Spectral kurtosis | 1 |

**Fig. 4.** Feature Extraction Configuration.

feature lists can be saved and loaded with buttons "->|" and "<-|". You can sort the features due to their selection, name, id and dimension number by pressing of the corresponding column name.

You can save the complete experiment description or load the previously defined description using "Save" and "Load" buttons. After the design is ready, push the "Finish Configration" button. You will be taken back to the experiment table.

If you want to run the feature extraction, push "Start Tasks". Or you can design other experiments at first.

## 3.2 Feature Processing

The first step of the feature processing is similar to the feature extraction: music files and features must be selected. Then you can setup different processing methods which should be applied to the raw audio signal features, cd. Fig. 5

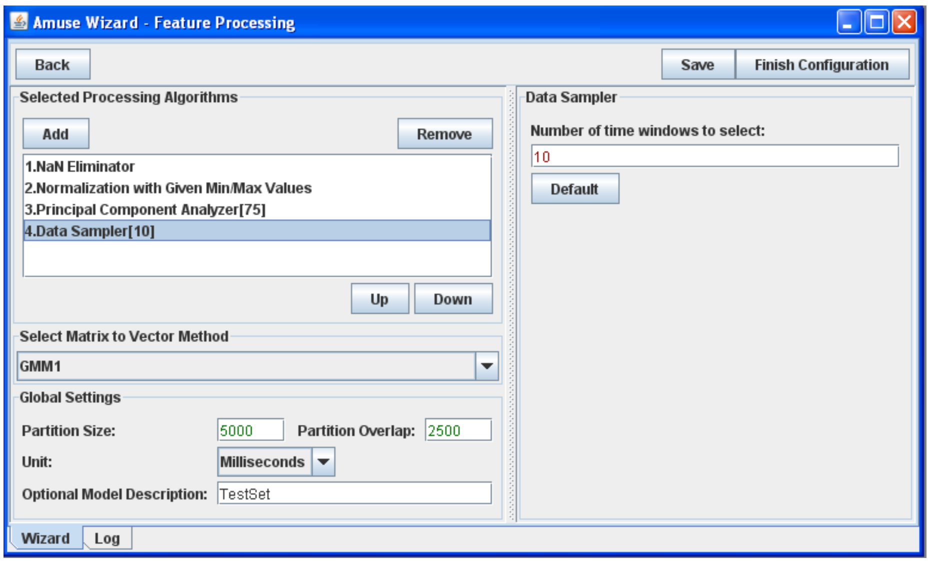
## 3.3 Classification Training

The setup of classification training task is depicted in Fig. 6.

The first step is to provide the ground truth (list with labeled music tracks) for the training of a model. You should create at first an ARFF file with the labeled list of music files as in the following example:

```
@RELATION musicfiles

@ATTRIBUTE Id NUMERIC
@ATTRIBUTE Path STRING
@ATTRIBUTE Unit {milliseconds,samples}
@ATTRIBUTE Start NUMERIC
```



Selected Processing Algorithms

1.NaN Eliminator
2.Normalization with Given Min/Max Values
3.Principal Component Analyzer[75]
4.Data Sampler[10]

Select Matrix to Vector Method: GMM1

Global Settings

Partition Size: 5000 Partition Overlap: 2500

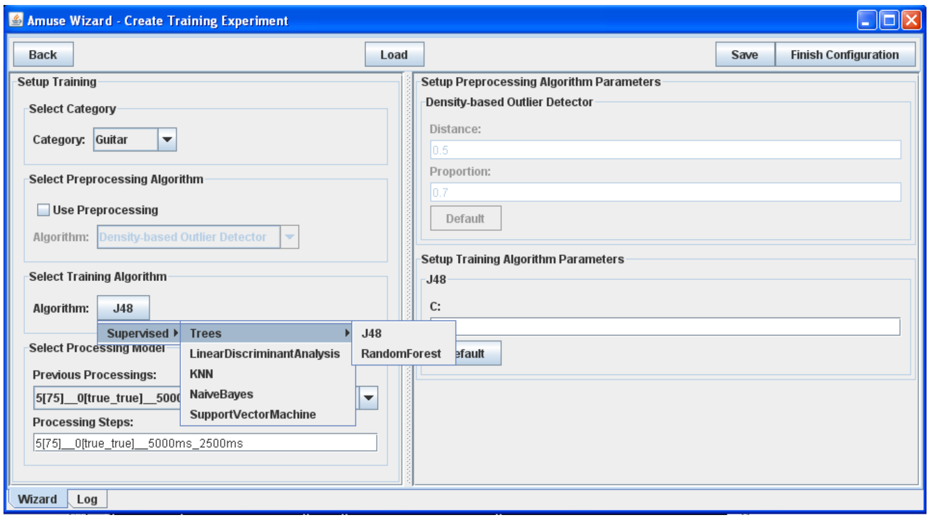
Unit: Milliseconds

Optional Model Description: TestSet

Data Sampler

Number of time windows to select: 10

**Fig. 5.** Feature Processing Configuration - Step 2.



Setup Training

Select Category

Category: Guitar

Select Preprocessing Algorithm

Use Preprocessing

Algorithm: Density-based Outlier Detector

Select Training Algorithm

Algorithm: J48

Select Processing Model

Previous Processings: 5[75]__0[true_true]__500(

Processing Steps: 5[75]__0[true_true]__5000ms_2500ms

Setup Preprocessing Algorithm Parameters

Density-based Outlier Detector

Distance: 0.5

Proportion: 0.7

Setup Training Algorithm Parameters

J48

C:

**Fig. 6.** Classification Training.

```
@ATTRIBUTE End NUMERIC
@ATTRIBUTE Category STRING
@ATTRIBUTE Relationship NUMERIC

@DATA
0,'C:\\Music\\Database\\Guitar\\tr112.wav',milliseconds,0,-1,Guitar,1.0
1,'C:\\Music\\Database\\Guitar\\tr114.wav',milliseconds,0,-1,Guitar,1.0
2,'C:\\Music\\Database\\Guitar\\tr115.wav',milliseconds,0,-1,Guitar,1.0
10,'C:\\Music\\Database\\Piano\\tr011.wav',milliseconds,0,-1,Guitar,0.0
11,'C:\\Music\\Database\\Piano\\tr013.wav',milliseconds,0,-1,Guitar,0.0
12,'C:\\Music\\Database\\Piano\\tr014.wav',milliseconds,0,-1,Guitar,0.0
```

The name of the category is here "Guitar", the first three waves belong to it (relationship 1.0), the last three does not belong to it (relationship 0.0). You can save this ground truth file as guitar.arff in the AMUSE category database. Now you should edit the list with all categories. If you do not have any list, create an ARFF with the following text:

```
@RELATION categories

@ATTRIBUTE Id NUMERIC
@ATTRIBUTE Path STRING
@ATTRIBUTE CategoryName STRING

@DATA
0,'C:\\Music\\Categories\\guitar.arff',Guitar
```

The path to this second file (default name is categoryTable.arff) must be properly set as AMUSE Category Database in the Settings panel!

Now you should decide if you want to use preprocessing outlier removal and configure it. This step is not necessary.

Then you should select the classification method ("Select Training Algorithm" frame). Some of the algorithms have parameters and the default values will appear on the right side of the training configuration window.

The last step is to type the description of the processing configuration which was used to save the unlabeled feature vectors. The processing history saves the last processing descriptions. They have the following format: at first the chain of the processing methods is given, separated by a minus sign ('-'). Method parameters are given in brackets and separated by underscores ('_'). Then two underscores are used. After them the feature matrix conversion description is given, following by two underscores. At the end come the partition sizes separated by underscores.

As an example, "5[75]__0[true_true]__5000ms_2500ms" means, that the processing was done with method 5 (PCA) using 75% of components, the conversion method was GMM1 (both mean value and standard deviation are saved), and the partition size is 5000 ms with 2500 ms overlap.

Now you can finish / save your configuration and train a model.

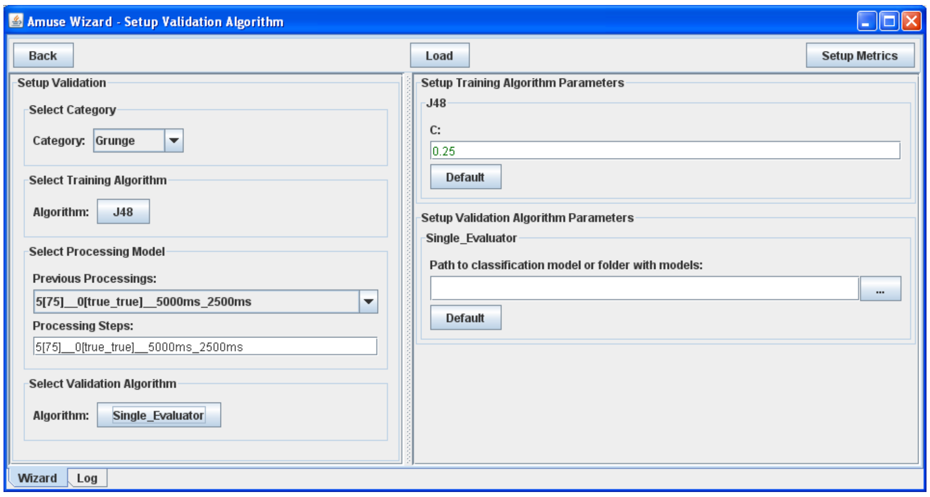
## 3.4 Classification

After one or more classification training models have been created, the unlabeled music file lists can be classified.

In the first step the file list should be created or loaded. Then the model characteristics must be set: the category used for the training of the model, the algorithm used and the configuration of feature processing. The path to classification results must be given. As default, the results applied on song partitions are averaged across the complete songs.

## 3.5 Validation

For the measurement of classification quality the validation task can be run.


Amuse Wizard - Setup Validation Algorithm

Setup Validation
Select Category
Category: Grunge
Select Training Algorithm
Algorithm: J48
Select Processing Model
Previous Processings: 5[75]__0[true_true]__5000ms_2500ms
Processing Steps: 5[75]__0[true_true]__5000ms_2500ms
Select Validation Algorithm
Algorithm: Single_Evaluator

Setup Training Algorithm Parameters
J48
C: 0.25

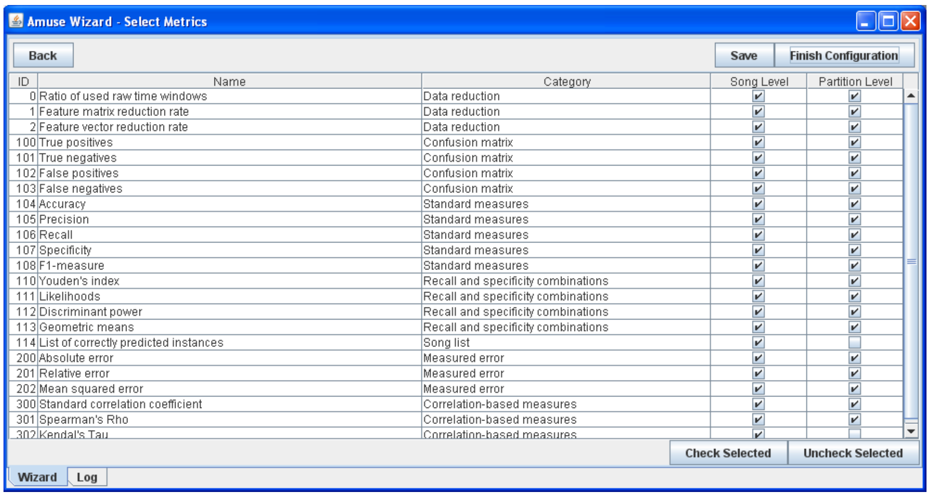
Setup Validation Algorithm Parameters
Single_Evaluator
Path to classification model or folder with models:

**Fig. 7.** Classification Validation - Step 1.

In the first step as shown in Fig. 7 the test category and the training category have to be configured. Training category describes the ground truth used for evaluation - the unlabeled processed features must exist in the Processed Features Database. The classification algorithm and processing description must be set.

Then you should decide if you wish to validate a single previously trained model ("Single_Evaluator" method) and give a path to this model in the right panel, or if you wish to run a n-fold cross-validation where n models are trained automaticly from the given category.

In the second step, cf. Fig. 8, you can enable/disable the metrics which should be calculated. They will be stored in the AMUSE Metric Database folder.



| ID | Name | Category | Song Level | Partition Level |
|---|---|---|---|---|
| 0 | Ratio of used raw time windows | Data reduction | ✓ | ✓ |
| 1 | Feature matrix reduction rate | Data reduction | ✓ | ✓ |
| 2 | Feature vector reduction rate | Data reduction | ✓ | ✓ |
| 100 | True positives | Confusion matrix | ✓ | ✓ |
| 101 | True negatives | Confusion matrix | ✓ | ✓ |
| 102 | False positives | Confusion matrix | ✓ | ✓ |
| 103 | False negatives | Confusion matrix | ✓ | ✓ |
| 104 | Accuracy | Standard measures | ✓ | ✓ |
| 105 | Precision | Standard measures | ✓ | ✓ |
| 106 | Recall | Standard measures | ✓ | ✓ |
| 107 | Specificity | Standard measures | ✓ | ✓ |
| 108 | F1-measure | Standard measures | ✓ | ✓ |
| 110 | Youden's index | Recall and specificity combinations | ✓ | ✓ |
| 111 | Likelihoods | Recall and specificity combinations | ✓ | ✓ |
| 112 | Discriminant power | Recall and specificity combinations | ✓ | ✓ |
| 113 | Geometric means | Recall and specificity combinations | ✓ | ✓ |
| 114 | List of correctly predicted instances | Song list | ✓ | ☐ |
| 200 | Absolute error | Measured error | ✓ | ✓ |
| 201 | Relative error | Measured error | ✓ | ✓ |
| 202 | Mean squared error | Measured error | ✓ | ✓ |
| 300 | Standard correlation coefficient | Correlation-based measures | ✓ | ✓ |
| 301 | Spearman's Rho | Correlation-based measures | ✓ | ✓ |
| 302 | Kendal's Tau | Correlation-based measures | ✓ | ☐ |

**Fig. 8.** Classification Validation - Step 2.

### 3.6 Optimization

Currently the feature selection by evolutionary strategies is supported. During the optimization setup you should select the category for training of the models using the fatures selected by ES. The second category is the optimization category for the model evaluation. The last is the independent test category (is not used per default).

The ES configuration is saved in an XML file, see amuse\docs\optimization.xml for an example. In this XML the initial feature table, processing steps and the metric table must be set. The first metric in the given metric list is used for optimization.

## Acknowledgements

The AMUSE development was partly supported within a project funded by the Klaus Tschira Foundation.

Many contributions to the source development were done by Clemens Wältken, student at the TU Dortmund.

We thank the co-authors of the publications [1–6] for the extensive discussions and many ideas which helped to develop AMUSE.

## References

1. Theimer, W., Vatolkin, I., Botteck, M. and Buchmann, M.: Content-Based Similarity Search and Visualization for Personal Music Categories. In: Proc. of the Sixth International Workshop on Content-Based Multimedia Indexing, London (2008)

2. Theimer, W., Vatolkin, I., and Eronen, A.: Definitions of Audio Features for Music Content Description, Algorithm Engineering Report TR08-2-001, Technische Universität Dortmund (2008)
3. Vatolkin, I. and Theimer, W.: Optimization of Feature Processing Chain in Music Classification by Evolution Strategies, in Proc. of the 10th International Conference on Parallel Problem Solving from Nature (PPSN), Dortmund, pp. 1150-1159 (2008)
4. Vatolkin, I., Theimer, W., Rudolph, G.: Design and Comparison of Different Evolution Strategies for Feature Selection and Consolidation in Music Classification. In: Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009), IEEE Press, Piscataway (NJ) (2009)
5. Bischl, B., Vatolkin, I. and Preuss, M.: Selecting Small Audio Feature Sets in Music Classification by Means of Asymmetric Mutation, accepted for Proc. of the 11th International Conference on Parallel Problem Solving from Nature (PPSN), Krakow (2010)
6. Nagathil, A., Vatolkin, I., Theimer, W.: Comparison of Partition-Based Audio Features for Music Classification, accepted for the 9th ITG Fachtagung Sprachkommunikation, Bochum (2010)
7. Vatolkin, I., Theimer, W. and Botteck, M.: AMUSE (Advanced MUSic Explorer) - A Multitool Framework for Music Data Analysis, Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR), Utrecht (2010)