Automated Model Selection with AMSF in a production process of the automotive industry

Florian Grewe and **Peter Owotoki**
Hamburg University of Technology (TUHH)
21077 Hamburg, Germany
florian.grewe|owotoki@tu-harburg.de

Abstract

Machine learning, statistics and knowledge engineering provide a broad variety of supervised learning algorithms for classification. In this paper we introduce the *Automated Model Selection Framework* (AMSF) which presents automatic and semi-automatic methods to select classifiers. To achieve this we split up the selection process into three distinct phases. Two of those select algorithms by static rules which are derived from a manually created knowledgebase. At this stage of AMSF the user can choose between different rankers in the third phase. Currently, we use instance based learning and a scoring scheme for ranking the classifiers. After evaluation of different rankers we will recommend the most successful to the user by default. Besides describing the architecture and design issues, we additionally point out the versatile ways AMSF is applied in a production process of the automotive industry.

1 Introduction

Following CRISP, the process of Data-Mining can be broken down into the phases of *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation* and *deployment* [CRI, 2000]. In the modeling phase algorithms are selected and applied to generate models. The issue of selecting appropriate methods in this phase we refer to as *Model Selection*. CRISP describes algorithm selection as an exploratory process, highly dependant on the analyst's knowledge and on the problem domain.

In the field of *Knowledge Discovery in Databases* different projects with different approaches exist dealing with the issue of Model Selection. The most recent project is MetaL, combining the results of previous research projects in an online service [Met, 2004]. Users can upload their datasets and receive a ranking of algorithms.

The *Waikato Environment for Knowledge Analysis* (WEKA) is a tool for data analysis and includes implementations of different classification algorithms. A book describing the software was published in 2005 by Ian H. Witten and Eibe Frank [Witten and Frank, 2005]. WEKA's binaries and sources are freely available. Implemented methods include instance-based learning algorithms, statistical learning like Bayes methods and tree-like algorithms like ID3 and J4.8 (slightly modified C4.5). Including combinations of classifiers, e.g. bagging and boosting schemes, there are over sixty methods available in WEKA. Table 1 shows a list of all non-combined methods with a short description. All algorithms can be utilized for supervised learning with their standard parameters.

This paper introduces AMSF which's intention is to provide help to human users by semiautomatically selecting appropriate methods. Furthermore its selection schemes can be utilized to automatically analyze problem domains by analyzing datasets and ranking methods. For the semi-automatic use, support is provided by a user interface in form of a wizard[1].

2 AMSF Components

This Section introduces the different components AMSF consists of. From an architectural point of view the components and classes are separated according to a three layer architecture. In the data-layer we put two database files

- the *Knowledgebase of Classifiers* and
- the *Performance Database*.

In the logic layer there are the classes to handle the selection and ranking of the algorithms. Important components here are

- the *Preselection Component* and
- the *Ranking Component*.

In the presentation layer we find the wizard and other GUI-components.

2.1 Knowledgebase of Classifiers

The Knowledgebase of Classifiers is a database and contains an entry for every algorithm WEKA 4.5 provides.

The entries contain information about the applicability of an algorithm on a dataset. Here, we mean wether an algorithm can handle missing values or unlabeled instances. Furthermore the algorithms vary in their ability of dealing with different types of attributes and classes. Following the conventions of the Attribute-Relation File Format (ARFF) each attribute may be either numeric or nominal.

In addition to applicability characteristics the algorithms in the Knowledgebase of Classifiers are categorized according to the type of knowledge representation they generate. J4.8 for example generates a decision tree. Figure 1 exemplarily shows the Knowledgebase of Classifiers' entry for J4.8.

J. Gama and P. Brazdil provide additional criteria in [Gama and Brazdil, 1995] to characterize classifiers. In contrast to the Performance Database described in Section 2.2 the Knowledgebase of Classifiers was created manually by considering relevant literature and the sources of WEKA.

[1] A wizard is a user friendly stepwise dialog

```
<Method name="J48">
    <Characterization>
        <Group>
            <GroupEntry>trees</GroupEntry>
        </Group>
        <ModelInterpretability>interpretable</ModelInterpretability>
        <KnowledgeRepresentation>
            <KnowledgeRepresentationEntry>Decision Tree</KnowledgeRepresentationEntry>
        </KnowledgeRepresentation>
        <Input>
            <NumericalValues value="true"/>
            <NominalValues value="true"/>
            <MissingValues value="true"/>
        </Input>
        <Output>
            <NumericalValues value="false"/>
            <NominalValues value="true"/>
            <BinaryClass value="true"/>
            <MissingValues value="true"/>
        </Output>
    </Characterization>
    <Information>
        <Source>WekaAPI</Source>
        <Text>Class for generating an unpruned or a pruned C4.5 decision tree. For more
            information, see Ross Quinlan (1993). C4.5: Programs for Machine
            Learning, Morgan Kaufmann Publishers, San Mateo, CA. </Text>
    </Information>
</Method>
```

Figure 1: J4.8 entry in the Knowledgebase of Classifiers

2.2 Performance Database

In addition to the described Knowledgebase of Classifiers, we generated a Performance Database in an automated creation process. Details about the creation process are outlined in Section 2.3.

The Performance Database comprises information about the method's error rate, training and testing time when applied to a particular dataset. It currently contains about two-hundred entries, one for each dataset. To summarize, each entry of the Performance Database contains

- dataset characterization measures described in Section 2.4 and a
- list of performance related data consisting of one entry per applied learning algorithm.

The Performance Database and the Knowledgebase of Classifiers described in Section 2.1 are stored in XML-format and can be validated against schema documents after new entries have been added.

2.3 Performance Database Creator

The Performance Database contains empirical data. It was generated by applying the algorithms provided by WEKA in their standard setup. Here, for every dataset we utilized the Knowledgebase of Classifiers to select a subset of methods applicable to the dataset. The methods were applied using stratified ten-fold cross-validation. The resulting Error rates, training and testing times were averaged according to the cross-validation settings.

We expect the performance of AMSF to increase when additional entries are added to the Performance Database. AMSF therefore includes a graphical component to add new entries comfortably.

2.4 Dataset Characteristics

Besides error rate and time dependent information, we store information about the datasets themselves in the Performance Database. The coverage of characterization information stored follows the proposals of R. Engels and C. Theusinger in [Engels and Theusinger, 1998]. Referring to their terminology, the characteristics include

- *simple characteristics* like number of cases, number of defective cases and number of binary attributes,
- *statistical measures* like median and median deviation of classes and
- *information theoretical measures* like class entropy.

2.5 User Interface

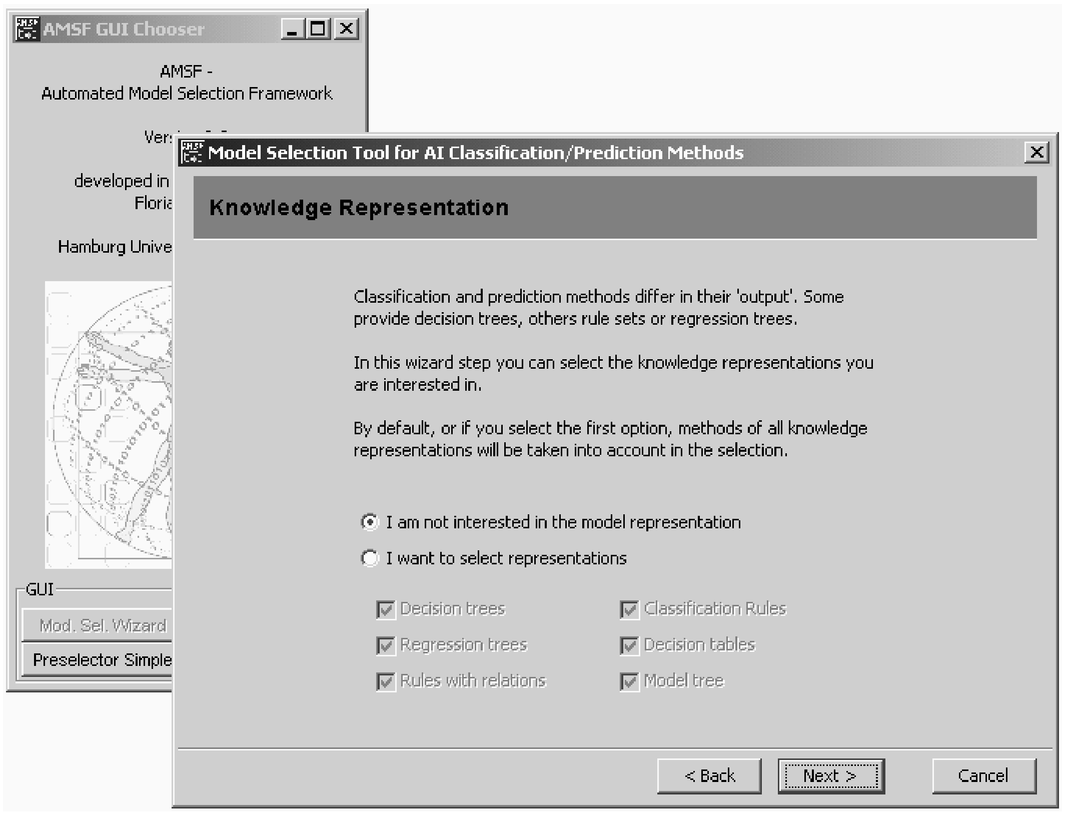
The main GUI component of the AMSF so far is the wizard already mentioned in Section 1. Figure 2 shows a screenshot of one step in the model selection process. Besides the wizard AMSF also provides an user interface for controlling the creation and extension of the performance database. We furthermore developed a GUI for the preselection process described in Section 3.1 to enable fast and direct access to the information of the Knowledgebase of Classifiers.

3 Selection Process

In this Section we explain how the components described in Section 2 are utilized to perform the selection and ranking of the classifiers.

3.1 Preselection Component

Fed by the Knowledgebase of Classifiers the Preselection Component selects the methods applicable to a given domain description. Here, the domain description includes the user preferences regarding the method's knowledge representation and furthermore the characterization of the input attributes and the class attribute as described in Section 2.1. The preselection is carried out by simply suppressing the algorithms which cannot handle the properties



Knowledge Representation

Classification and prediction methods differ in their 'output'. Some provide decision trees, others rule sets or regression trees.

In this wizard step you can select the knowledge representations you are interested in.

By default, or if you select the first option, methods of all knowledge representations will be taken into account in the selection.

[✓ I am not interested in the model representation]
I want to select representations

✓ Decision trees ✓ Classification Rules
✓ Regression trees ✓ Decision tables
✓ Rules with relations ✓ Model tree

Figure 2: The third wizard step of the algorithm selection

of the dataset. Their properties can be comfortably calculated directly whereas those have to be provided in ARFF-format. Some of the input and class attribute information is included in the ARFF header and does not have to be derived from the data itself. It is therefore important that the ARFF-file was created according to the specification.

3.2 Ranking Component

Currently, AMSF includes two rankers. The first ranker utilizes an instance-based learner to find the k nearest neighbors in the dataset characterization space introduced in Section 2.4.

To find the nearest neighbors, the user has to provide a dataset that the characteristics can be calculated of. In the current version we have not yet implemented any special weighting of the dimensions. After the k nearest neighbors have been identified, only the error rate information is considered to perform the ranking by now.

For this purpose we created a scoring scheme: Within every of the k nearest neighbors the classifiers are sorted according to error rate. The best algorithm obtains three points, the second best two and the third best one point. The points are then accumulated over the k datasets for each method and then ranked according to their score. This is particulary useful in this stage of the software, as we presume the methods to become more obvious winners when we adjust the weighting of the dataset characteristics dimensions.

The second ranker is applicable without providing a concrete dataset and only ranks the methods by the standardized error rate sum

$$err_s = \frac{1}{N}\sum_{i=1}^{N} err_i \tag{1}$$

where N denotes the number of datasets the algorithm was applied to and err_i is the error rate of the method applied to the i-th dataset. The only information the user has to provide besides the preselection criteria is the nature of the class attribute. The methods are then either ranked according to Equation 1 or according to the corresponding equation of the sum of squares.

The whole selection procedure is illustrated in Figure 3 and summarized in the caption.

4 AMSF in the real world

Data Mining methods can be utilized to uncover new valuable insights. Still, one major problem when those methods are to be applied is insufficient usability. Although tools like WEKA offer a comprehensive spectrum of analysis alternatives, the appropriate selection of applicable methods is still an issue of the user.

In modern production lines hundreds to thousands of parameters can be set and adjusted not to mention the high number of measurement values that are recorded during production. Analyzing and interpreting these data can be of high benefit with respect to quality and cost optimization.

In a production process of the DaimlerChrysler plant in Hamburg we therefore set up an *Integrated Database* which collects data of the complete production process including quality data about input factors and former auxiliary conditions like data about hall temperature and humidity. In doing so, it is possible to identify the produced parts throughout the production process and mine valuable knowledge.

In order to increase quality characteristics or to determine the process robustness the parameters are often varied systematically. The methodology of *Design of Experiments*

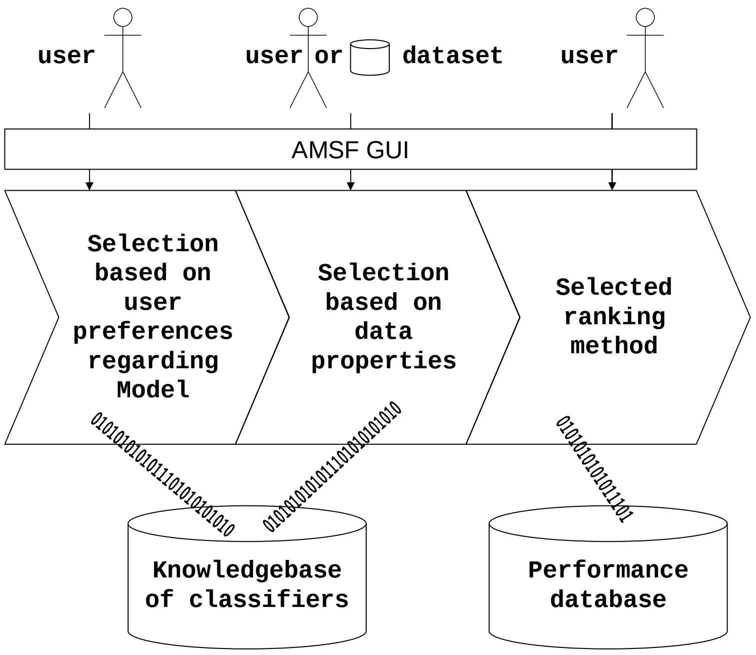

user
user or
dataset
user
AMSF GUI
Selection based on user preferences regarding Model
Selection based on data properties
Selected ranking method
Knowledgebase of classifiers
Performance database

Figure 3: The selection process split up in the three distinct phases *user preferences*, *data properties* and *ranking*. The user employs the wizard to control the process and analyze the optionally provided dataset. The third phase utilizes the performance database. The previous phases generate rules from the Knowledgebase of Classifiers.

(DOE) provides procedures to maximize the knowledge that can be gained with a fixed preferably small number of experiments. Classically, the input as well as the output parameters of such experiments are assumed to be numerical. After the experiments have been carried out linear models are adjusted by the method of linear regression. In doing so only a small number of parameters, namely the varied input parameters and the output parameter, are utilized to build the models. So, almost all the data collected during the experiments is not considered. This is especially unsatisfactory if the created linear models do not sufficiently fulfill the necessary requirements and are discarded after the procedure for that reason.

By mining the data and carefully looking at the created interpretable models (in this case the models have to be 'interpretable', as described above), we gained valuable new insights in the production process. The dependencies revealed are used for example to classify produced parts avoiding expensive experiments.

5 Evaluation of preselection and rankers

As mentioned in Section 2.2 we created performance data for about two-hundred datasets. In that process, the preselection component described in Section 3.1 was utilized and tested. It attracted attention that a relatively large ration of classifiers could be applied to the datasets (approximately 70% in average). We used UCI and our own datasets to calculate the performance data and are currently in the process of producing more data.

A comparison of rankers provided by different ranking methods of AMSF and ideal ranking (calculated later on) is not yet available. Anyhow, the described ranker already produces reasonable results although the possibilities are not yet exploited.

6 Conclusion and Future Work

AMSF provides user support in selecting appropriate methods. To reach this goal it compares the domain-description provided by the user and the dataset on one hand with certain databases on the other hand.

AMSF provides assistance to the user in different ways:

- It automatically analyzes datasets.
- In a preselection step it identifies applicable methods.
- It provides a ranking of the applicable by different ranking methods.

Considering the ranking component described in Section 3.2 other approaches, e.g. the *Zoomed Ranking* method of P. Brazdil and C. Soares described in [Soares and Brazdil, 2000] also take time issues into account. This is particularly valuable if the user has fixed notions about the runtime, for example if a method for real-time systems has to be chosen.

Acknowledgments

We want to thank Dr. Oliver Kropla, Eckhard Reese, Michael Picker, Matthias Plagmann, Jan Zimmermann, Klaus Vukelic, Jürgen Westphal, Henning Bendfeldt and Heiko Inert of the DaimlerChrysler AG for the valuable discussions and for providing the data.

References

[CRI, 2000] Cross industry standard process for data mining (crisp-dm 1.0), www.crisp-dm.org, 2000.

[Engels and Theusinger, 1998] Robert Engels and C. Theusinger. Using a data metric for preprocessing advice for data mining applications. In *European Conference on Artificial Intelligence*, pages 430–434, 1998.

Name	Description
AODE	Averaged, one-dependence estimators
BayesNet	Learn Bayesian nets
Complement NaiveBayes	Build a Complement Nave Bayes classifier
NaiveBayes	Standard probabilistic Nave Bayes classifier
NaiveBayes Multinomial	Multinomial version of Nave Bayes
NaiveBayes Simple	Simple implementation of Nave Bayes
NaiveBayes Updateable	Incremental Nave Bayes classifier that learns one instance at a time
ADTree	Build alternating decision trees
DecisionStump	Build one-level decision trees Id3 Basic divide-and-conquer decision tree algorithm
J48	C4.5 decision tree learner (implements C4.5 revision 8)
LMT	Build logistic model trees M5P M5 model tree learner
NBTree	Build a decision tree with Nave Bayes classifiers at the leaves
RandomForest	Construct random forests
RandomTree	Construct a tree that considers a given number of random features at each node
REPTree	Fast tree learner that uses reduced-error pruning
UserClassifier	Allow users to build their own decision tree
ConjunctiveRule	Simple conjunctive rule learner
DecisionTable	Build a simple decision table majority classifier
JRip	RIPPER algorithm for fast, effective rule induction
M5Rules	Obtain rules from model trees built using M5
Nnge	Nearest-neighbor method of generating rules using nonnested generalized exemplars
OneR	1R classifier Part Obtain rules from partial decision trees built using J4.8
Prism	Simple covering algorithm for rules
Ridor	Ripple-down rule learner
ZeroR	Predict the majority class (if nominal) or the average value (if numeric)
LeastMedSq	Robust regression using the median rather than the mean
LinearRegress.	Standard linear regression
Logistic	Build linear logistic regression models
Multilayer Perceptron	Backpropagation neural network
PaceRegression	Build linear regression models using Pace regression
RBFNetwork	Implements a radial basis function network
SimpleLinear Regression	Learn a linear regression model based on a single attribute
SimpleLogistic	Build linear logistic regression models with built-in attribute selection
SMO	Sequential minimal optimization algorithm for support vector classification

Table 1: Non-combined methods in the knowledgebase [Witten and Frank, 2005]. The first block shows the group of Bayes methods, the second tree algorithms, the third instance-based learners and the last the functions group.

[Gama and Brazdil, 1995] J. Gama and P. Brazdil. Characterization of classification algorithms. *Progress in Artificial Intelligence, 7th Portuguese Conference on Artificial Intelligence, EPIA-95*, pages 189–200, 1995.

[Met, 2004] Metal, www.metal-kdd.org, 2004.

[Soares and Brazdil, 2000] Carlos Soares and Pavel Brazdil. Zoomed ranking: Selection of classification algorithms based on relevant performance information. pages 126–135, 2000.

[Witten and Frank, 2005] Ian H. Witten and Eibe Frank. *Data Mining: practical machine learning tools and techniques*. Morgan Kaufmann, 2005.