Extracting Knowledge and Computable Models from Data - Needs, Expectations, and Experience

Thomas Natschläger, Felix Kossak, and Mario Drobics
Software Competence Center Hagenberg, A-4232 Hagenberg, Austria

Email: {Thomas.Natschlaeger, Felix.Kossak, Mario.Drobics}@scch.at

Abstract In modern industrial manufacturing, a great amount of data is gathered to monitor and analyze a given production process. Intelligent analysis of such data helps to reveal as much information about the production process as possible. This information is most useful if it is available in the form of interpretable and predictive models. Such models can be generated from data by means of (fuzzy logic based) machine learning methods. In this contribution we will describe industrial applications in the areas of process optimization and quality control where we have successfully established machine-learning methods as intelligent data analysis tools.

1 Introduction

In order to be competitive on the market, the major goal of any industrial production process is to produce – as fast as possible – a large quantity of high-quality products in a (cost-) efficient way. To tackle this challenge, major requirements are an optimized production process and constant quality control. Obviously it is desirable that each of these issues is supported by suitable *i*ntelligent *d*ata *a*nalysis (IDA) tools which operate on the process data, i.e. signals which are measured by appropriate sensors connected to the production process.

Fig. 1 shows a possible scenario where IDA tools are used in an offline manner: the knowledge gained by IDA operating on large amounts of stored data influences “online” analyses like quality control, trend analysis, and fault detection, as well as the production process itself, i.e. process optimization. It turns out that the knowledge extracted by the IDA is particularly useful if it comes in the form of mathematically well defined models of certain dependencies within the data. A potentially success-

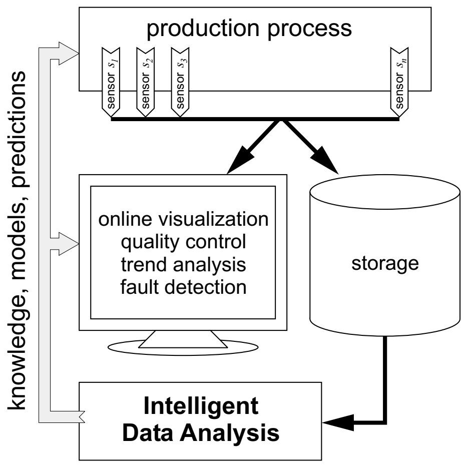


Figure 1: Possible scenario of *i*telligent *d*ata *a*nalysis (IDA)

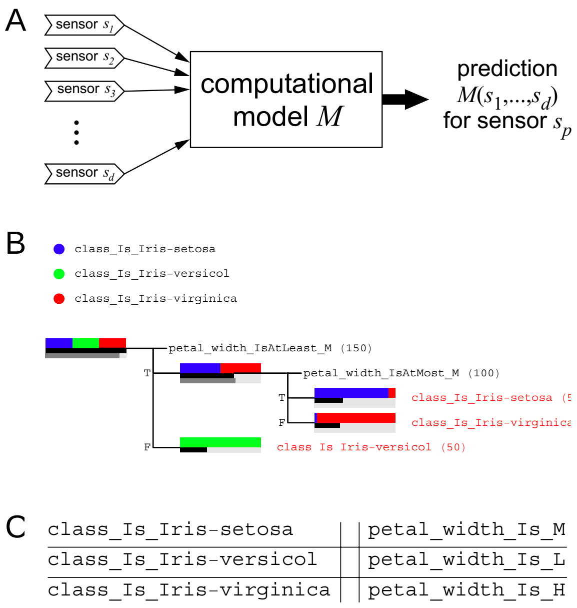

A
sensor s_1
sensor s_2
sensor s_3
sensor s_d
computational model M
prediction $M(s_1,...,s_d)$ for sensor s_p
B
class_Is_Iris-setosa
class_Is_Iris-versicol
class_Is_Iris-virginica
petal_width_IsAtLeast_M (150)
petal_width_IsAtMost_M (100)
class_Is_Iris-setosa (5
class_Is_Iris-virginica
class Is Iris-versicol (50)
T
F
T
F
C
class_Is_Iris-setosa | petal_width_Is_M
class_Is_Iris-versicol | petal_width_Is_L
class_Is_Iris-virginica | petal_width_Is_H

Figure 2: Computational models. In contrast to “black box models” (A) such as neural networks, descriptive models such as decision trees (B) or rule bases (C) are easy to interpret. The drawings in panels B and C are obtained with *mlf* (see Sec. II) applied to the well-known iris data set.

ful approach is to generate such models by means of machine learning methods; see e.g. [1]. Typically a model “learned” from the data predicts a sensor signal s_p based on a set of different sensor signals $s_1, s_2, ..., s_d$, i.e. $s_p = M(s_1, s_2, ..., s_d)$ (see also Fig. 2A).

The use of such predictive models can be twofold: one can exploit the *predictive capabilities* of a model on the one hand and the *structure* of the model on the other hand. The prediction obtained from a given (and previously learned) model can for example be used to detect anomalous working conditions of the production process [2] or to get an idea about the quality of the product if certain process parameters are varied. In such applications, the accuracy of the prediction is of primary interest while the structure of the model plays a minor role. However, if the structure of the model is easily interpretable, like it is the case for decision trees and rule bases (see Fig. 2B), this structure can help to understand the process in more depth. In consequence, such enhanced knowledge can often be used to optimize the production process.

Even from this short introduction, one can already specify two general requirements of machine learning methods when applied as intelligent data analysis tools: the resulting models should have good predictive capabilities while the structure of a model should be easily interpretable. In Sec. 2, we will describe a framework for machine learning, called *mlf*, which tries to meet this requirements. *mlf* has been successfully used as an intelligent data analysis tool in several applications. Two particular applications are described in Sec. 3 and Sec. 4. Throughout the sections 2, 3, and 4, we will draw our attention to the features of the applied machine learning tools which turned out to be important to successfully establish them in an intelligent data analysis tool.

2 *mlf*: A Machine Learning Framework for Mathematica

The machine learning framework for Mathematica[1] (*mlf*) is a collection of powerful machine learning algorithms integrated into a framework for the main purpose of intelligent data analysis [3]. *mlf*[2] combines an optimized computational kernel - the core engine - realized in C++ with the manipulation, descriptive programming, and graphical capabilities of Mathematica (see Fig. 3).

[1]Mathematica is a registered trademark of Wolfram Research Inc. (www.wolfram.com).

[2]*mlf* is developed and supported by the Knowledge Based Technology area of the Software Competence Center Hagenberg GmbH (A-4232 Hagenberg, Austria, http://www.scch.at). *mlf* is owned and distributed by uni software plus GmbH (Kreuzstrasse 15a, A-4040 Linz, Austria, http://www.unisoftwareplus.com/) which has ten years of experience in distributing Mathematica and Mathematica-based solutions and collaborates with Wolfram Research, Inc., for worldwide distribution.

Descriptive models via fuzzy logic

The design of *mlf* was strongly influenced by the requirements of being able to generate descriptive and highly predictive computational models. To a large part, this was achieved by integrating fuzzy logic wherever possible. As a result, the models generated by *mlf* contain easily interpretable phrases like “*the value of sensor* s_1 *is large*” while still maintaining numeric accuracy and predictive capabilities. In addition, smooth results very often model the underlying dependencies within the data more realistic than crisp ones.

Wide range of algorithms

As it is a matter of fact that for a given problem, it is not clear in advance which learning algorithm will yield the best results, it is always a good advice to try different machine learning algorithms or to combine them to solve one single problem. Such combinations of distinct algorithms may give the user unforeseen insights into their data. Therefore *mlf* covers a wide range of machine learning algorithms, which are listed in Tab. 1 together with the corresponding references.

Visualization and structure of data

In addition to *supervised* learning algorithms which generate the kind of computational models depicted in Fig. 2A, the

Table 1: Algorithms implemented in *mlf*

Supervised Analysis
Fuzzy decision tree learning (FS-ID3) [4]
Fuzzy rule base learning (FS-FOIL, FS-MINER) [5]
Numerical optimization of arbitrary rule bases (RENO) [6]

Unsupervised Analysis
Clustering algorithms (WARD, FCM) [7, 1]
Self-organizing maps (SOM) [8]

machine learning framework also contains several *unsupervised* learning algorithms. Such algorithms can help to understand how data is structured (e.g. which groups of customers a company has) or to visualize the often very high dimensional data (e.g. by using SOMs [8]). Apart from machine learning algorithms, *mlf* includes a wide range of visualization methods using all the power from the graphics capabilities of Mathematica.

High speed

All algorithms are highly parameterizable to be able to adjust them to a particular problem. Given this parameterizeability and the range of learning algorithms combined with the efficient core engine (realized in C++) of *mlf*, the user is able to "look at their data from different points of view" in real time.

Powerful architecture and interface

The machine learning framework for Mathematica is used from the Mathematica front end. By using the Mathematica front end, one has access to all Mathematica functions, including the powerful graphical manipulation tools, and, with the Mathematica programming language, one has access to an elegant scripting language. However, computationally intensive algorithms are implemented in an optimized computational kernel (the core engine realized in C++) to yield fast response times. The C++ kernel is completely independent of the Mathematica front end and can be utilized from any environment capable of calling C++ functions (see Fig. 3).

3 Process optimization in paper production

In this section, we will describe an application of *mlf* for process optimization in paper production.

Paper production is a complex flow process with several process levels and hundreds of process parameters which potentially influence the quality of the produced paper, assessed by dozens of quality measurements. Hence a major goal is to adjust the process parameters for optimum paper quality under tough economic constraints. Knowledge about dependencies within the data, i.e. the process parameters and the quality measurements, is valuable information in assisting the optimum adjustment of parameters which needs enormous expert knowledge and experience. Unfortunately the process of paper production is so complex that the creation of explicit models for such dependencies is hardly possible. In such a case, intelligent data analysis by means of machine learning methods can help to discover such required knowledge from previously measured data.

This approach turned out to be very fruitful and culminated in a project together with VOITH Paper (www.voith.com) and SCA Graphic Laakirchen AG (www.sca.at) where a machine learning application called PaperMiner was (and is still being) developed. PaperMiner is heavily used in practice and has led to a number of surprising insights by the users.

A major requirement for the PaperMiner from the onset was a graphical user interface (GUI) for accessing some of the functionality of *mlf* (see Fig. 3 for a block diagram of PaperMiner). The requirement for a GUI reflects the more general requirement that basic machine learning functionality must be accessible for users without a special mathematical background. On the other hand, experts should be able to dig deeper using the same tools (e.g. with respect to co-operative work). This results in the following requirements:

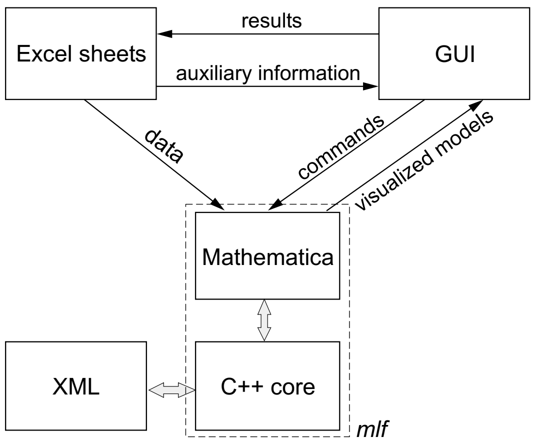


Figure 3: The architecture of PaperMiner which, is based on *mlf*.

GUI for standard problems

A user should be able to obtain useful results for standard problems by following a rather short series of simple steps. The PaperMiner GUI represents the required steps in a hierarchical way which makes the whole workflow visible at first sight; see Fig. 4.

Expert interface

In many cases, however, standard procedures will not yield the desired results. In these cases, the user should be able,

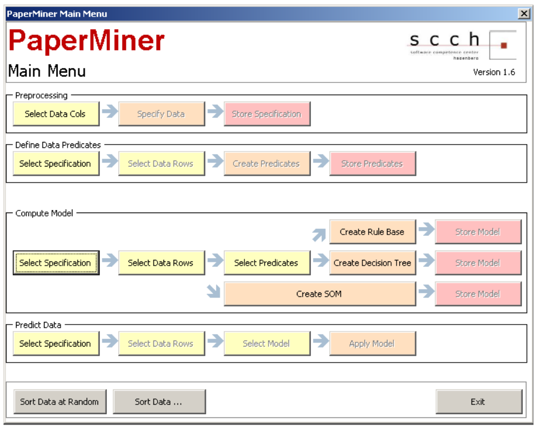
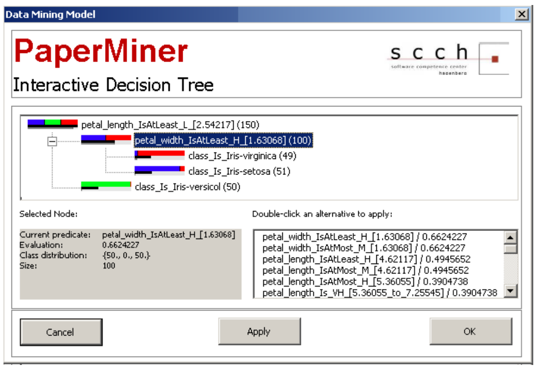


Figure 4: Graphical user interface (GUI) of PaperMiner (screen shot)


Data Mining Model
PaperMiner
Interactive Decision Tree
s c c h
software competence center hagenberg
Selected Node:
Double-click an alternative to apply:
Cancel
Apply
OK

```
petal_length_IsAtLeast_L_[2.54217] (150)
    petal_width_IsAtLeast_H_[1.63068] (100)
        class_Is_Iris-virginica (49)
        class_Is_Iris-setosa (51)
    class_Is_Iris-versicol (50)
```

```
Current predicate:    petal_width_IsAtLeast_H_[1.63068]
Evaluation:           0.6624227
Class distribution:   {50., 0., 50.}
Size:                 100
```

```
petal_width_IsAtLeast_H_[1.63068] / 0.6624227
petal_width_IsAtMost_M_[1.63068] / 0.6624227
petal_length_IsAtLeast_H_[4.62117] / 0.4945652
petal_length_IsAtMost_M_[4.62117] / 0.4945652
petal_length_IsAtMost_H_[5.36055] / 0.3904738
petal_length_Is_VH_[5.36055_to_7.25545] / 0.3904738
```

Figure 5: PaperMiner's interactive decision tree. The screen shot shows a decision tree generated for the well known iris data set.

for instance, to optimize parameters of the model-generating algorithms. With the current architecture of the application, this is possible through the Mathematica interface of *mlf*.

Another primary requirement for the PaperMiner was that it can be used in connection with Microsoft Excel, for the following reasons: a) Excel is a program which potential users are typically already familiar with; b) Excel is capable of displaying and transforming data in a convenient way; and a) It is relatively easy to extract data from a database into an Excel file (and respective scripts were already available for the pilot application). From the requirement to use Excel as the data source, one can derive the more general requirement that applying the machine learning algorithms is only half of the story about intelligent data analysis. One also needs flexible, powerful and easy-to-use tools to access and preprocess the data.

During the application of PaperMiner to practical problems in paper production, it turned out that whereas the resulting models often describe highly interesting dependencies, sometimes the chosen attributes/measurements are judged by the domain experts not to be the "best" ones. To examine whether the original measurement suggested by the learning algorithm can be replaced by a "better" measurement, PaperMiner features an interactive tool for the construction of a decision tree. This tool allows a user to modify individual nodes of a given (most commonly a previously learned) decision tree; see Fig. 5.

A further requirement is that the standard presentation of the output should be as appealing and intuitive as possible. This is most easily achieved by graphical representations. We experienced that people could easily interpret decision trees with color encoding of key information at the nodes without knowing anything about the theoretical background. Simple diagrams and similar graphical depictions can be interpreted by humans intuitively, at first sight, while the information conveyed by such pictures is still sufficient for most applications. Still, for fine-grained process optimization, more detailed output must be available as well.

4 Machine Learning in Discrete Manufacturing

In this section, we will describe applications of machine learning in the are of discrete manufacturing. The process of assembling discrete parts from its discrete components (produced elsewhere) has different characteristics from a flow process like paper production (see previous section). In a typical assembly line, the components to be assembled (provided by certain feeder modules) are fixed at a workpiece holder which goes through several processing modules until the part is finished either as “o.k.” or “not o.k.” (cf. Fig. 6). The whole assembly process is monitored by a shop floor management system which collects the process data and provides statistical information to operators and the management.

In a project together with AMS Engineering Sticht GmbH (www.ams-engineering.com), we set out to apply machine learning tools like *mlf* to gain more knowledge out of the information provided by the shop floor management system. The high level goal is to achieve

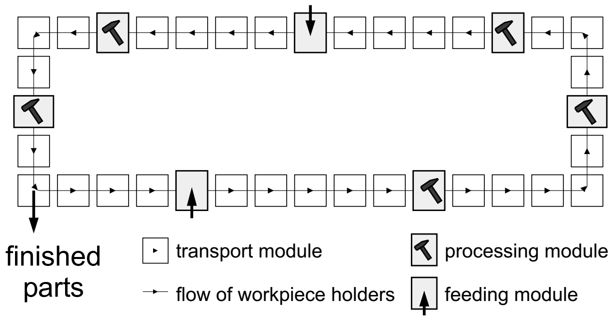


Figure 6: Schematic of a loosely coupled assembly line.

an overall equipment efficiency (OEE) as high as possible and to analyze the causes if the OEE is not satisfactory. The OEE is a product of equipment availability (EA), production efficiency (PE), and quality rate (QR). Therefore each of the three factors must be as high as possible.

If one of those factors is strongly varying over time, it is of interest to figure out the factors which cause low values. For example, one may investigate whether the people operating the assembly line and the type of the produced parts influence the average time needed to produce one part. A result of such analysis of a hypothetical assembly line which produces different types of footwear is shown in Fig. 7. From such a decision tree, one can not only conclude that the average time indeed depends on the type and operator but one can also see groupings of operators and types. Similar results were obtained for real data (from a given assembly line with more then 30 processing modules made available by AMS engineering Sticht GmbH) when analyzing

```
if operator $\in \{$Sue, Pam, Tim, Sam, Bob, Kim$\}$ then
    if type $\in \{$casuals, sandals$\}$ then $T = 13.7$
                                   else $T = 12.7$
else
    if type $\in \{$sandals, galoshes, slippers$\}$ then $T = 13.05$
                                              else $T = 11.34$
```

Figure 7: A binary decision tree (written as if-then-else statements) which shows how the average production time T depends on the produced type and the operator of a hypothetical footwear assembly line.

EA, PE, and QR in this manner.

Obviously there are much more problems regarding the analysis of EA, PE, and QR which can potentially be addressed by means of machine learning tools. Below, we describe for each of the three factors a typical issue which was not yet solved in a satisfactory way in the analysis tools which are currently integrated in the shop floor management system.

Equipment availability

Equipment availability (EA) measures how much time is lost by (technical or organizational) standstills of the assembly line. Minimizing the number of standstills maximizes EA. Since a standstill of the assembly line is caused by the standstill of one or more processing modules, one must detect the "guilty" modules and remedy deficiencies in those modules. In a rigidly coupled assembly line, all the processing modules work synchronously,[3] and it is rather straightforward to detect which module caused the standstill. In contrast, a loosely coupled system consists of asynchronously working processing modules, and the individual workpiece holders are moved forward by separate transport modules which also act as buffers (cf. Fig. 6). In this case, it is not trivial to always detect the correct "guilty" modules. Having a model which relates standstills of the assembly line and standstills of modules to each other will help to find the reasons for low EA. In the current project, first promising results have been obtained by a combination of intelligent data visualization and machine learning tools.

Production efficiency

Production efficiency (PE) describes how efficient/fast the individual parts are produced. On the level of individual modules,

[3]To ensure that all the workpiece holders are at the right place at the right time they are usually attached to some kind of transportation belt.

PE measures how efficient/fast the processing step of a given module is. To optimize the efficiency, it is, of course, interesting to know what parameters influence the processing speed (and thus PE) of a given module in which way. In the context of optimizing the PE of the whole assembly line, it is of special interest to analyze whether parameters which directly control the behavior of a given module influence the speed of any successor module. Making extensive use of machine learning tools, one can, for example, construct a diagram where it is shown how strong the processing speed of a given module depends on parameters associated with predecessor modules. With such an overview diagram, one can dig deeper to look at the individual models relating the parameters to the processing speed. This approach was successfully applied in analyzing real data (made available by AMS engineering Sticht GmbH).

Quality rate

The quality rate (QR) is the ratio between the number of “o.k.”-parts and the total number of produced parts. The quality (“o.k.” or “not o.k.”) of a part is determined by means of a great number of measurements (taken at several modules) such as force, length, and torque referred to as quality criteria. If any of these quality criteria is outside its allowed range, the part is marked as “not o.k.”. A central issue is to analyze relationships between such measurements. Such knowledge will not only help to increase the quality rate of the machine but also help to determine whether a measured quantity is “important” or “useless” (and may be skipped at all), i.e. attribute selection. Interesting results of this kind for real data were obtained by using so called model or regression trees for more details).

5 Conclusions

In this paper we have described two areas of industrial applications (paper production and discrete manufacturing) where we have successfully applied machine learning tools (in particular *mlf*) in an offline manner for process optimization and quality control. In fact, the knowledge gained by applying this type of intelligent data analysis helped to optimize the production processes.

The currently running projects are by no means unidirectional. We are constantly receiving feedback from our industry partners which often results in improvements of the used methods. Each particular improvement effects one ore more aspects of the used methods. According to our experience gained in the running projects, several aspects turned out to be important for machine learning methods to be a useful tool to analyze a company’s crucial data:

- *Interpretability* and *prediction accuracy* of computational models generated from data (e.g. fuzzy decision trees combine these two requirements)
- *Integration* of methods in existing *analysis and preprocessing tools* (e.g. Pa-

perMiner is built as an add-on to MS-Excel)

- Possibility to compare the results of *several methods* (with different parameters) in *reasonable time*.
- Appealing and easy-to-comprehend *graphical representation*
- *Different levels of analysis*; i.e. GUI (e.g. PaperMiner) versus programming language level (e.g. Mathematica).

If all these requirements are met, it is very likely that people start to "trust" in those methods of data analysis.

Acknowledgements

This work has been done in the framework of the K*plus* Competence Center Program which is funded by the Austrian Government, the Province of Upper Austria, and the Chamber of Commerce of Upper Austria.

References

[1] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[2] E. Lughofer and E. P. Klement, "Model-based fault detection in multi-sensor measurement systems," Fuzzy Logic Laboratorium Linz-Hagenberg, Tech. Rep. FLLL-TR-0303, 2003.

[3] M. Drobics, "machine learning framework for *Mathematica* — creating understandable computational models from data," in *Proc. of the 2003 Mathematica Developer Conf.* Champaign, IL: Wolfram Research Inc., 2003.

[4] M. Drobics and U. Bodenhofer, "Fuzzy modeling with decision trees," in *Proc. 2002 IEEE Int. Conf. on Systems, Man and Cybernetics*, Hammamet, Tunisia, October 2002, pp. 90–95.

[5] M. Drobics, U. Bodenhofer, and E. P. Klement, "FS-FOIL: An inductive learning method for extracting interpretable fuzzy descriptions," *Internat. J. Approx. Reason.*, vol. 32, no. 2–3, pp. 131–152, 2003.

[6] J. Haslinger, U. Bodenhofer, and M. Burger, "Data-driven construction of Sugeno controllers: Analytical aspects and new numerical methods," in *Proc. Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, Vancouver, July 2001, pp. 239–244.

[7] J. H. Ward Jr., "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, pp. 236–244, 1963.

[8] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cyb.*, vol. 43, pp. 59–69, 1982.