Journal of Computing and Information Technology - CIT 19, 2011, 3, 193–206
doi:10.2498/cit.1001113

Semi-automatic Maintenance of Regression Models: an Application in the Steel Industry

Ilmari Juutilainen, Lauri Tuovinen, Perttu Laurinen, Heli Koskimäki, and Juha Röning

Data Mining Group, Department of Electrical and Information Engineering, University of Oulu, Finland

Software applications used in the controlling and planning of production processes commonly make use of predictive statistical models. Changes in the process involve a more or less regular need for updating the prediction models on which the operational software applications are based. The objective of this article is

- to provide information which helps to design semi-automatic systems for the maintenance of statistical prediction models and
- to describe a proof-of-concept implementation in an industrial application.

The system developed processes the production data and provides an easy-to-use interface to construct updated models and introduce them into a software application. The article presents the architecture of the maintenance system, with a description of the algorithms that cause the system's functionality. The system developed was implemented for keeping up-to-date prediction models which are in everyday use in a steel plate mill in the planning of the mechanical properties of steel products. The conclusion of the results is that the semi-automatic approach proposed is competitive with fully automatic and manual approaches. The benefits include good prediction accuracy and decreased workload of the deployment of updated model versions.

Keywords: data analysis, software architecture, maintenance system, predictive modelling, model updating

1. Introduction

Large amounts of data are measured from production processes in different industries. The production data is often utilized by applications that employ statistical predictive models to control, optimize and plan the production. The utilized statistical models are fitted using measurement data collected from the process. As new products and production methods are continuously developed and novel process settings are taken into use, data from previously unseen conditions occur – thus there is a constant need to update the prediction models. Another major reason that causes a need for model updates is process drift.

A commonly used practice to keep the predictions up-to-date is to employ adaptive methods that automatically update the prediction formula based on the newest data, and thus adapt themselves to the latest process conditions. There are two commonly used approaches to adaptive modelling: the moving window method, with its weighted versions, and recursive updating of parameters. The term 'on-line learning' [2] refers to the latter approach. Also, some other approaches, for example change point models [8], have been studied. Adaptive models are especially useful in those cases where process drifts cause rapid changes in the modelled relationship and only the most recent data is valuable for the modelling.

Less methodological attention has been paid to the situation where the distribution of explanatory process variables is changing, and previously unobserved data regions are regularly taken into use. Because there are no earlier data on these regions, the model cannot work reliably, although the relationship modelled remains unchanged. However, similar adaptive approaches have been commonly employed.

An adaptive learning approach is not applicable if the generation of up-to-date prediction models

with satisfactory quality requires human guidance. This is the case when the updating of the model requires the redefinition of the model inputs, structural changes, careful selection of fitting data or the utilization of domain knowledge.

In non-automated model maintenance, a new prediction model is developed from scratch whenever it is decided that the current prediction model does not work satisfactorily. A semi-automatic maintenance approach automatizes the data processing and the generation of new model candidates, but allows human control of the model employed to produce the predictions. The approach can improve prediction accuracy and give resource savings compared to the non-automated maintenance approach. The semi-adaptive approach for updating of prediction models may be the most reasonable approach for many applications. Although a semi-automatic maintenance approach has not received much attention in research papers, simplified variations of it may be utilized in industry relatively commonly.

Information that explains the aspects related to the architecture, functionality and implementation of semi-automatic model maintenance systems can be a substantial aid in solving a particular model maintenance task. Although maintenance of predictive models seems to be a common issue in industry [12], the issue is rarely discussed in the literature: In our literature review, we succeeded to find one throughout description on the architecture of a prediction model maintenance system [7]. We did not find any research articles that issue semi-automatic model maintenance systems. However, the architecture of the fully automatic prediction model evolving system by [7] could be applicable also for semi-automatic model maintenance. Their architecture maintains separate pools for data preprocessing algorithms and prediction models. Process data is employed to validate the predictive performance of the different combinations of preprocessing and learning methods. The validation results are employed to adaptively select the optimal combination of prediction functions which is utilized to produce the final prediction using model ensemble methods.

This article presents a case study where a semi-automatic model maintenance system was implemented to maintain up-to-date product property models in a steel plate mill. In contrast to [7], the system presented treats the prediction model as a separate software module that may be distributed for several model applications. The approach enables the full distinction between prediction models and the applications that employ the models. Thus, the maintenance of models is separated from that of application software. This is an advantage, especially if there are multiple software applications that utilize the model. What is more, the implementation of models as independent modules allows the exchange of models between different systems and the integration of prediction models into existing software [13]. The goal of this article is to provide guidelines which help in the design of semi-automatic model maintenance systems for different application domains and industries.

2. Data and the Regression Models

This section gives an overview of the application domain for which the studied model maintenance system was developed. The problem at hand was to keep up-to-date three non-linear regression models that predict the distribution of the mechanical properties of steel plates. These models are utilized by simulation tools which are used daily for planning the composition and thermomechanical treatments of the production process for plate products in Ruukki production, Raahe works. As a project requirement, it was decided that the updating of these models must be kept beyond human control, and that a semi-automatic system is needed to make the updates easier.

The prediction models studied belong to the family of double-generalized linear models [14]. It is assumed that some monotone transformation of the response variable observations y_i are normally distributed with both mean and variance depending on input variables through a link-linear relationship

$$y_i^* = h_\lambda(y_i) \quad (1)$$

$$y_i^* \sim N(\mu_i, \sigma_i^2) \quad (2)$$

$$\mu_i = f(x_i^{\mathrm{T}}\beta) \quad (3)$$

$$\sigma_i^2 = g(z_i^{\mathrm{T}}\tau), \quad (4)$$

where $h_\lambda(s) = (s^\lambda - 1)/\lambda$ is a Box–Cox transformation function. Here, $f(\cdot)$ and $g(\cdot)$ are monotone functions and $x_i \in \mathbf{R}^p$ is the input variable vector for the mean model and $z_i \in \mathbf{R}^q$ is the input variable vector for the variance model. Both x_i and z_i may include any transformations of the original explanatory variables.

The modelled mechanical properties are measured in tensile tests. A test bar is cut out from the edge scrap of a steel plate. Tensile testing belongs to regular quality assurance that has to be performed before the products can be delivered to the customer. On average, over a hundred tensile tests are made daily. The modelling data includes the results of the tensile tests, the composition of steel and the thermomechanical treatments made during the production process. There are totally about 60 variables in the data. About 30 variables are used in the model fitting, and the rest of the variables are used only in data preprocessing and identification.

The algorithms for the model maintenance system were developed using a large data set that included all tensile tests that were performed during the period 2001–2006. The data set included over 250 000 tensile test results, and provided an excellent test bed to analyse and characterize the need for model updates and develop optimal algorithms for the semi-automatic maintenance. [5] explains the data and the application domain in more detail. The dependence of the modelled mechanical properties from the input variables can be assumed to remain relatively unchanged, but the introduction of new steel compositions and processing methods cause a need to update the models so that they work also for the new products. The data analysis indicated that the optimal interval for model updates varies, depending on the introduction of new products and compositions – typically a model update was needed once or twice in a year.

A special feature of our data is that there is a remarkable number of rare steel grades that are manufactured irregularly and occur after intervals of several years. It is important that the simulation tool can predict reliably the properties of these rare steel grades. Thus, all the data obtained on the rare steel compositions and production conditions should be archived for the modelling.

3. Architecture

The model maintenance system developed is semi-automatic in the sense that the final decision to update the operational prediction model is made by a human, but all the hard work needed to develop, validate and implement an updated model is automatized. A single person is responsible for model maintenance using the system, he or she is later called a *model maintainer*.

3.1. The operational environment

The operational environment of the model maintenance system developed is illustrated in Figure 1.

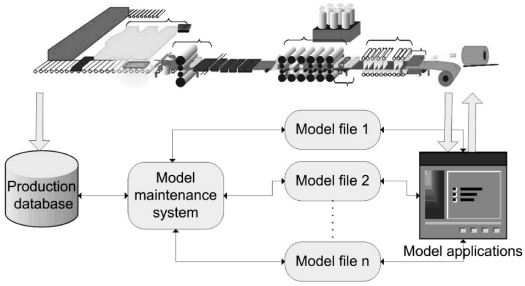

Figure 1. The model maintenance system processes production data to keep up-to-date the prediction models used by software applications whose purpose is to improve production efficiency.

We shall now clarify some essential concepts:

- **The production process** consists of the manufacturing, planning and scheduling operations.
- **The production database** stores the measurements made on the production process.
- **The model application** is a software application that is utilized by the production process to optimize its quality and efficiency. The model applications utilize one or more model files to produce its output.
- **The model file** is a file with which the model application produces predictions. The model file can be a library file whose functions can be called by the model application. Alternatively, the model file can be, for example, an XML file that defines the model and on the basis of which the model application can produce predictions.

- **The model maintenance system** is a software application that communicates with the production database and whose output is utilized to update the model files.

From the model maintenance point of view, it is important that the prediction model is separated from the model application. The communication between the model and the application should be based on a general and well-formatted interface such as predictive model markup language (PMML) [13]. This approach provides substantial advantages:

1. The maintenance of the prediction model is much simpler than that of the model application. Now the prediction model can be maintained by updating only the model file without taking care of the model application. The model can be seen as a 'plug-and-play' component.
2. The same model file can be used by several applications. Instead of updating all the applications, it is enough to update the model file.
3. A general interface between the application and model file allows drastic changes in the model structure and the model family. For example, it is possible to replace double-generalized linear models by regression trees, neural networks or any other model type.

Dynamic link library (dll) is an established format to implement functions that can be loaded run-time from software applications independently of the programming language. Thus, dll is an appealing way to implement 'plug-and-play' prediction models. Based on these arguments, it was decided that the maintenance system uses a specialized text file format designed to describe double-generalized linear models (Eq. 4). The model file (dll) reads the model details from the text file and the simulation tool calls the functions of the dll. The model is updated by rewriting the estimated parameters and optionally also terms and link functions in the text file.

3.2. System architecture

The reuse of code and designs is considered desirable in software engineering. Efficient reuse of code and expandability of the system by application-tailored algorithms are the leading principles in the architectural design of the software developed. These aims are achieved by implementing the system using an appropriate application framework. An application framework is a software skeleton that can be specialized into different applications by plugging in a comparably small quantity of new code. Application framework provides both a reusable design and a body of code that implements the design in a reusable form. Volatile functionality and implementation details of the data processing algorithms are encapsulated behind stable interfaces. Component-based application frameworks have been proved to suit well for data mining software [1].

An intuitive schema for solving a data mining problem is to link data mining algorithms to one another (if more than two) and to data sets. This schema can be schematically supported by a pipes-and-filters architecture. Thus, the pipes-and-filters concept has been often employed as the architectural design of data mining software: For example, the popular component-based data mining libraries KNIME, Weka and RapidMiner are based on pipes-and-filters architectures.

Based on the above arguments, our model maintenance system was designed using a component-based pipes-and-filters architecture. Each of the algorithms needed in model maintenance was implemented as a reusable component so that any data processing operation can be implemented by forming a component graph in which data is transferred from one component to another. The advantage of this approach is that it allows flexible re-use and combination of different data sources and algorithms used in different stages of data processing, which improves the expandability and modifiability of the system.

3.3. Internal data storage

A relational database is a standardized interface to distribute data. A database is an efficient method to exchange data between software components and applications. These arguments encouraged us to employ a relational database to store model maintenance data.

Table name	Stored information
primary_data	data gathered from the production database and preprocessed
predictions	the response values predicted by different models
proximities	the proximity measures for the observations
clusters	clustering structure that maps each observation into one of the clusters
update_need	the results of the model update need measure calculation
model_performance	the calculated model goodness measures and related statistical significance
preprocess	the preprocess rules and the number of excluded observations for each rule
sent_mails	the notices sent to the model maintainer with time stamps
variable_definitions	names and expressions that define the variables available for modelling
settings	the settings that control the algorithms used in the model maintenance
model_descriptions	descriptive listing of the archived models
fit_obs	mapping that indicates the observations used in the fitting of each model
models	archived models in the standard file format

Table 1. The database tables that the model maintenance system employs.

The system employs an internal relational database to store the data and information needed in the model maintenance. The stored information includes a model archive, i.e. all the models that have been in operational use or in test use earlier. The contents of the model maintenance database are listed in Table 1.

The case implementation uses MySQL database. The table primary_data contains the measurements that are read from production database, includes a database key for identifying observations and flags the observations that are valid for inclusion in the analysis. The table model_descriptions include a database key for identifying models and flags that indicate the current operational model and current test model. Other tables make use of these keys to identify observations and models.

3.4. External interfaces

The model maintenance system developed communicates with humans and the information system of the production plant with four interfaces:

- **Data in** The system reads data from formatted text files and SQL-databases.
- **Model file out** The system generates a text file and sends it to overwrite the current operational model file.
- **Notice messages** The model maintainer is informed of the update needs by e-mail.
- **Graphical user interface** Enables the model maintainer to perform model updating actions, obtain detailed data about the models and their performances, and control the functionality of the model update algorithms.

3.5. Technical implementation

The implementation was carried out using a software framework called Smart Archive [11] in order to benefit from ready interfaces and advance the reuse of code and designs. The purpose of Smart Archive is to provide general functionality for implementing real-time data mining applications and for utilizing efficiently large amounts of historical measurement data, and thus shorten the time needed to develop tailored data mining applications. The design of Smart Archive is database-oriented: the system database is utilized also to exchange data between the components.

The model maintenance system was implemented using java, mainly because Smart Archive has been implemented in java. The arrangement of components employed in Smart Archive determined also the arrangement of classes of the maintenance system:

- **Data in** The interface to import data to the model maintenance system from the production database and the preprocessing of the raw data.

- **Components** Individual data analysis algorithms which can be used in the execution graphs.
- **Input receivers** The data interface to get measurement data from the system's internal database.
- **Execution graphs** The component graphs used for performing data analysis operations. An execution graph consists of at least one input receiver and at least one component.
- **User interface** The graphical user interface of the maintenance system.
- **Utilities** Utility functions common to many classes, such as basic database and model file operations.
- **Model file out** The interface to generate and deliver model files according to the requirements of the model applications.

4. Functionality

The semi-automatic model maintenance system developed consists of two separate software applications. The *data processing application* autonomously reads and preprocesses new data once a day, and determines if the new data gives a reason for maintenance action. The *model updating application* is a tool for easy, human-controlled fitting, validation and distribution of new prediction models.

A baseline principle is that a new model is never taken into use before its performance is validated. Let the *operational model* be the model that is used by a model application and let the *test model* denote a newly generated model being validated. The system takes care that the test model is not promoted to the new operational model until the newest data has confirmed that the test model performs better than the current operational model.

4.1. Data processing application

The functionality of the data processing application is presented in Figure 2. When the data processing application is launched, it performs the following steps:

1. Read all unprocessed data from production database.
2. Preprocess the data using a rule set that rules out erroneous measurements and observations gathered during abnormal process conditions. Store the preprocessed data into the system's internal database. If the data change so that the preprocessing rules do not anymore work correctly, a notice is sent to the model maintainer.
3. Calculate predictions for the new data using both the operational model and the test model. Store the predictions in the internal database.
4. Calculate proximity measures for the new data. Store the proximity measures in the internal database.
5. Calculate the model goodness criteria for both the operational model and the test model in the accumulated test data set. Test for the statistical significance between the model performances. If it is proved that the test model performs better, send a notice to the model maintainer.
6. Calculate the *model update need measure* using predictions and proximity measures using the data collected after the current operational model was fitted. Send a notice to the model maintainer if the calculated model update need measure indicates that the prediction accuracy could be significantly im-

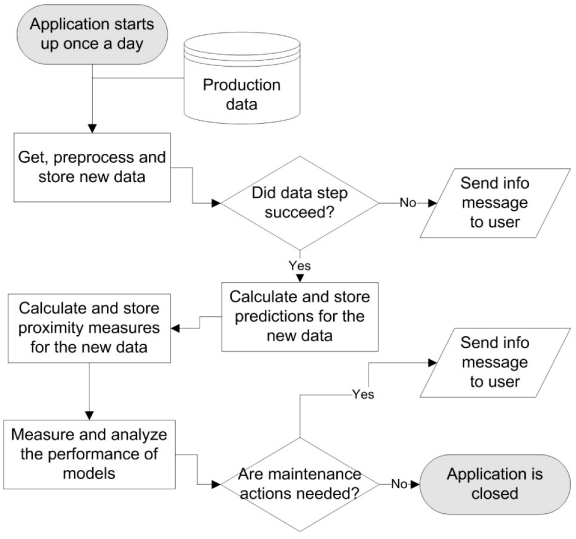

Figure 2. Operational flowchart for the data processing application.

proved by fitting a new model employing also the newest data.

4.2. Model updating application

The functionality of the model updating application is presented in Figure 3. The model maintainer launches the application by opening a graphical user interface window. The purpose is to provide the model maintainer with a user-friendly interface to perform the actions related to model maintenance: most importantly, to generate new models, validate the performance of the models and determine the model that is used as the operational model. The main window gives a numerical and verbal presentation on the goodness criteria of the test model and operational model, and on the statistical significance between the models. Also, the model update need measure is illustrated. The menus and buttons of the interface enable the following user actions:

- Accept the current test model as the new operational model. The action writes a model file in a format compatible with the model applications and delivers it outwards to replace the model file being currently used.
- Specify and fit a new test model. The action opens a guided window to determine the structure of the model to be fitted and the observation count and date limits for the data which is used in the model fitting. Also, any of the archived models can be set as the test model.
- Define a restricted data subset using an SQL-query. The action calculates predictions and compares model performances in a user-defined data subset.
- Show archived model details and browse the model archive.
- Examine the results of data preprocessing and modify the preprocessing rule set.
- Exclude or restore observations used in the modelling and model validation using SQL-commands.
- Show and modify variables which are available in new model specification and validation. The variable definitions are written using SQL-syntax.
- Modify the settings of the model maintenance system; for example, e-mail addresses or the parameters of different algorithms.
- Export data and predictions to a text file.

5. Algorithms

The aim of this section is to give an overview of the algorithm categories and examples of algorithms that are needed to implement a maintenance system in other applications. In this

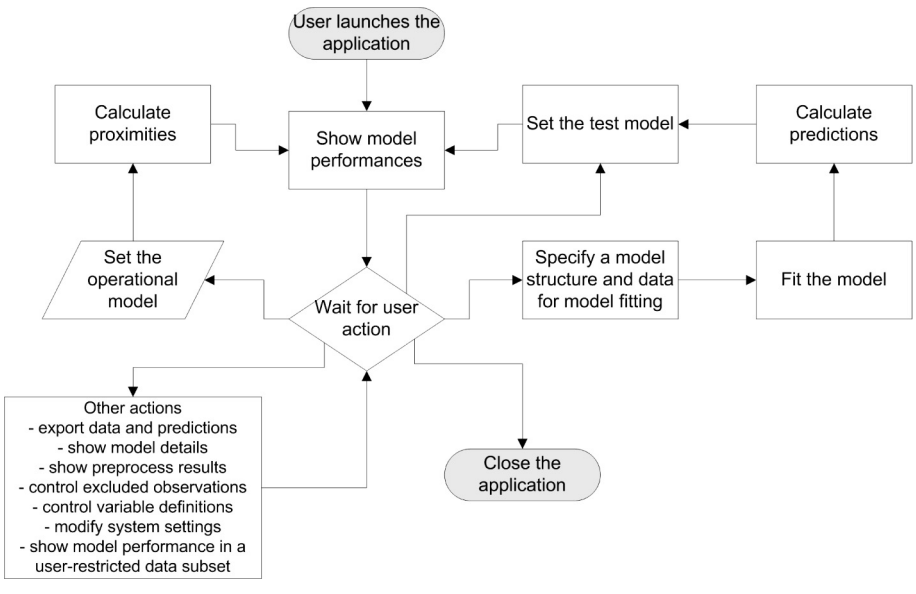

Figure 3. Operational flowchart for the model updating application.

study, five statistical algorithms were implemented to form the functionality of the model maintenance system: data preprocessing, selective data restriction, model fitting, predictive model validation, and detection of the optimal model update moment. These five elements of maintenance system functionality are justified as established stages of a data mining process. CRISP-DM is an industry-oriented process model for data mining and predictive modelling. According to CRISP-DM, a data mining process consists of six stages: business understanding, data understanding, data preparation, modelling, model evaluation and model deployment. [3]. The prediction model evolving system [7], too, makes use of these elements.

The algorithms used in this study were developed for the steel mill case application. This section illustrates the algorithm categories needed in model maintenance by describing the algorithms implemented in more detail. Because the algorithms implemented are not expected to be suitable in all maintenance applications, the article does not discuss the data analysis results which show that the developed algorithms work well in steel data. However, [6], [9] and [10] explain the application specific algorithmic decisions.

In further applications of the maintenance system, the algorithms should be designed and optimized using the available process data to fulfil the needs of the particular maintenance problem. Each of the implemented algorithms is an individual re-usable component or a graph of several components. Further applications of the maintenance system may re-implement some of the components, but maintain their elementary organization. The re-usability of the components lowers the threshold to use the proposed architecture in different applications. The implemented components and their relation to the system's functionality are illustrated in Figure 4.

5.1. Model fitting

The ability to aid in the generation of new predictive models is a primary activity for the model maintenance system. The system estimates the model parameters after the specification of structure, parameters and variables are confirmed by the model maintainer. The implemented model fitting algorithm produces the parameter estimates $(\widehat{\beta}, \widehat{\tau})$ for the double-generalized linear model framework (Eq. 4) [5]. The implemented algorithm needs two obligatory inputs: the response variable vector

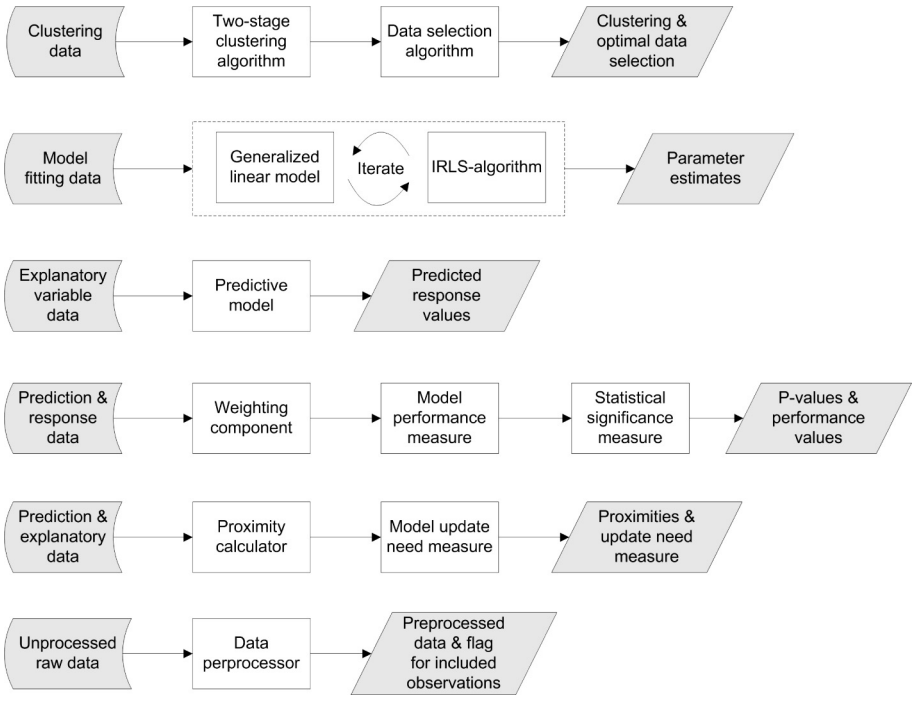

Figure 4. The algorithm components implemented to form the model maintenance functionality.

$y^* = (y_1^*, \ldots, y_n^*)^{\mathrm{T}}$, and the input data matrices $X = (x_1, \ldots, x_n)^{\mathrm{T}}$ and $Z = (z_1, \ldots, z_n)^{\mathrm{T}}$. It also has two optional inputs: the observation weight vector $w = (w_1, \ldots, w_n)^{\mathrm{T}}$ and the initial parameter estimates $(\widehat{\beta}_{\text{initial}}, \widehat{\tau}_{\text{initial}})$.

The algorithm maximizes the Gaussian log-likelihood function in (β, τ) using the iterative ML method [14]. QR-decomposition and the IRLS-algorithm [4, , pp. 28] are employed iteratively to solve the normal equations inside both the mean and variance model estimation.

5.2. Selective data restriction

A large number of observations prevents the usage of all available data for the model fitting. An algorithm for the smart selection of observations to be used in the training data is needed to utilize available historical data efficiently. All observations with rare and interesting explanatory variable values should be included in the training data, but the majority of observations with typical input values can be excluded. The selective data restriction algorithm developed places each observation into a cluster, and uses the clustering to construct a boolean vector $b = (b_1, \ldots, b_n), \quad b_i \in \{0, 1\} \ \forall i$ that indicates which observations will be included in the training data set. The output cluster structure is a vector $c = (c_1, \ldots, c_n)^{\mathrm{T}}$ that assigns each observation to one of the found C clusters $C_1, \ldots, C_C$ i.e. $c_i \in \{C_1, C_2, \ldots, C_C\} \ \forall i$. The inputs of the algorithm are the previous cluster structure c and a matrix of explanatory variable observations $\Xi = (\xi_1, \ldots \xi_n)^{\mathrm{T}}$ where the explanatory variable vector $\xi_i \in \mathbf{R}^r$ is a subset of possibly scaled or transformed original inputs.

The clustering algorithm [9] first updates current coarse clustering and then performs k-means clustering inside each coarse cluster. The *Model fitting information value* is calculated for each cluster using the proximities inside the cluster and between the clusters. The observations to be included in the training data are selected in proportion to the calculated model fitting information values.

5.3. Predictive model validation

A model performance measure is needed to evaluate and compare the suitability of models to the application. A statistical test is useful to ensure the statistical significance between the measured performances. The model performances are measured in a validation data set V, which is not used for the fitting of the models.

In the case application, the model performance is measured with weighted average robust negative log-likelihood

$$P_m = \sum_{i \in V} \omega_i \{ \log(\widehat{\sigma}_{i,m}^2) + \rho_{25} \left[(y_i^* - \widehat{\mu}_{i,m})^2 / \widehat{\sigma}_{i,m}^2 \right] - (\lambda - 1) \log(y_i) \} / \sum_{i \in V} \omega_i, \quad (5)$$

where $(\widehat{\mu}_{i,m}, \widehat{\sigma}_{i,m}^2)$ is the prediction of model m and $\rho_\alpha(t) = \min(t, \alpha)$ is a robustifying function. The Jacobian of Box–Cox transformation $(\lambda - 1) \log(y_i)$ transforms the log-likelihood back to the original scale. In the case application, it is most important to predict reliably for rarely manufactured steel products. Thus, the weights ω_i are defined to reduce the influence of typical observations: $\omega_i = \min(1, 50/T_i)$, where T_i denotes the number of observations in V belonging to the same product as the ith observation.

A test for statistical significance [6] was implemented by estimating the standard errors of model performance measures P_m, and assuming that the model performance measures are normally distributed.

5.4. Detection of optimal model update moment

New data are measured every day. Re-fitting the prediction model using also the newest data probably improves the prediction accuracy and utility value of the model application. On the other hand, updating of the prediction model includes its own costs and risks. The optimal time moment to re-fit the prediction model is recognized by comparing the expected improvement in model accuracy to the theoretical costs of the model update.

The method developed to detect the optimal model update moment [10] makes use of the *average prediction error in neighbourhood*, APEN, to detect data regions of systematic error in the predicted mean and *average squared standardized residual in neighbourhood*, ASSQN, to detect data regions of systematic error in the predicted variance. These are calculated for each new observation:

$$\mathrm{APEN}_i = \frac{\sum_k (1 - d_{ik}/R)^+ \rho_5[(y_k^* - \widehat{\mu}_k)/\widehat{\sigma}_i]}{\sum_k (1 - d_{ik}/R)^+} \quad (6)$$

$$\mathrm{ASSQN}_i = \frac{\sum_k (1 - d_{ik}/R)^+ \rho_{25}[(y_k^* - \widehat{\mu}_k)^2/\widehat{\sigma}_k^2]}{\sum_k (1 - d_{ik}/R)^+}. \quad (7)$$

Here d_{ik} is the Euclidean distance between the explanatory variable measurements of the ith and kth observation. Observations within radius R from the ith input observation are included in the neighbourhood with a weight depending on the distance. The robustifying function $\rho_\alpha(\cdot)$ bounds the influence of single outliers. Let the data set V include the valid observations measured after the studied model was fitted. The model update need measure proposed, M, reflects the improvement in Gaussian log-likelihood that would be achieved if the data regions with the systematic prediction error were corrected:

$$M = \sum_{i \in V} \Big[\mathrm{APEN}_i^2 - \log(\mathrm{ASSQN}_i) + \mathrm{ASSQN}_i - 1 - \gamma_i \Big] - \mathrm{UCOST}. \quad (8)$$

Here $\gamma_i = E(\mathrm{APEN}_i^2 - \log(\mathrm{ASSQN}_i) + \mathrm{ASSQN}_i - 1 \mid \textit{model is correct})$ and UCOST is a user-defined penalty reflecting the costs of a model update. The updating of the model is recommended if $M > 0$. The studies performed showed that our model update need measure reacts rapidly if a new data region with a poor prediction accuracy comes into regular production.

5.5. Data preprocessing

Data for model maintenance is often combined from several production data sources. Data is violated by missing values and measurement errors that should be removed to avoid tangling the analyses. The data preprocessing component implemented performs a modifiable set of SQL-sentences that process the raw data measurements and flag the observations that will be excluded from all the analyses and modelling. The internal database stores also the reason why an observation is interpreted to be invalid for modelling. These reasons can be grouped into five categories:

- Mismatch in the observation identification and time stamps, or conflict in data.
- A missing value in an obligatory explanatory variable, or in the response variable.
- Gross measurement error: a measured value of some important variable is outside the possible data region.
- Obvious measurement error in response value. This is judged when the absolute standardized residual is high, i.e. $(|(y_i^* - \widehat{\mu}_i)/\widehat{\sigma}_i| > 6$, and in addition, the observation has several neighbouring observations in the training data of the current prediction model so that the prediction should be reliable.
- Unsuspected disturbance in the process. The aim of the model is to predict the product properties assuming that the production process succeeds according to plan. Thus, unsuccessful process runs are excluded.

6. Empirical Study

In this section, the benefits of the developed semi-automatic model maintenance system are validated by comparing the proposed approach with two alternative scenarios for model maintenance. The comparison is conducted using the data which is collected from the mechanical properties of steel plates during the years 2006–2010. The validation data has not been employed in the development of the maintenance algorithms and it consists of 200 000 tensile tests. During the examination period, the semi-automatic model maintenance system has been developed from a prototype to its final version and its experimental usage at Raahe Works began in 2008. During the examination period there were three major process improvements: introduction of new cooling line, introduction of new furnace and introduction of a new cooling program for a large product family. In addition to these major changes, also the introduction of new products caused update needs for models.

The compared scenarios are:

1. SEMI: New model versions are generated using the developed model maintenance system. The model versions are taken into production use at the time moments proposed by the algorithms. The maintenance system takes care of the creation and delivery of the new model files. At each model update operation, the model maintainer may make improvements to the term structure of the models.
2. AUTO: The parameters of the prediction model are re-estimated once a week. The estimation is done using a rolling window scheme with a two year long data window. It is assumed that the model application or the model file reads the re-estimated parameters from plain text files so that there is no need for the generation of more complex model files.
3. MANUAL: The model is developed and re-implemented from scratch once during the examination period. The model development efforts consist of several tasks: data gathering, data preprocessing, model fitting and selection and the implementation of model file. The time moment of the model update is 3.5 years after the beginning of the examination period. It is assumed that data for model re-estimation is available from the previous two years.

Scenario SEMI reflects the planned and actual usage of the system at Raahe Works. The development of prediction error in the different scenarios was evaluated in a simulation study in which the scenarios were applied to the validation data. The costs of different scenarios were coarsely estimated using domain knowledge. Table 2 summarizes the advantages and disadvantages of the different scenarios.

	SEMI (semi-automatic)	AUTO	MANUAL
Maintenance tasks	Monthly follow-up of the model maintenance system (1.5 hours/month) + the estimation and acceptance of new model versions and delivery of new model files (5 deliveries, 2 hour/delivery) + determination of structural changes to the models (5 changes, 6 hours/change)	None	The model files are re-implemented from a scratch once per three year. Each reimplementation include data gathering (1 day), data preprocessing (4 days), model estimation and selection (5 days) and implementation and testing of the model files (5 days)
Total workload	16 days	0 days	30 days
Version control	Manual: The users can know the versions and notice version differences	Automatic: The users cannot stay abreast of the high number of model versions	Manual: The users know the versions and recognize version updates
Error risk	Eliminated: model files are pretested	Small: Bad estimation results or bugs in the model file are not observed until the model application	Eliminated: model files are pretested
Flexibility	Moderate: the term structure between the model versions may differ	Small: the term structure is similar for each model version	Moderate: the term structure between the model versions may differ

Table 2. Comparison of the estimated workloads, advantages and disadvantages of model update operations in the different scenarios.

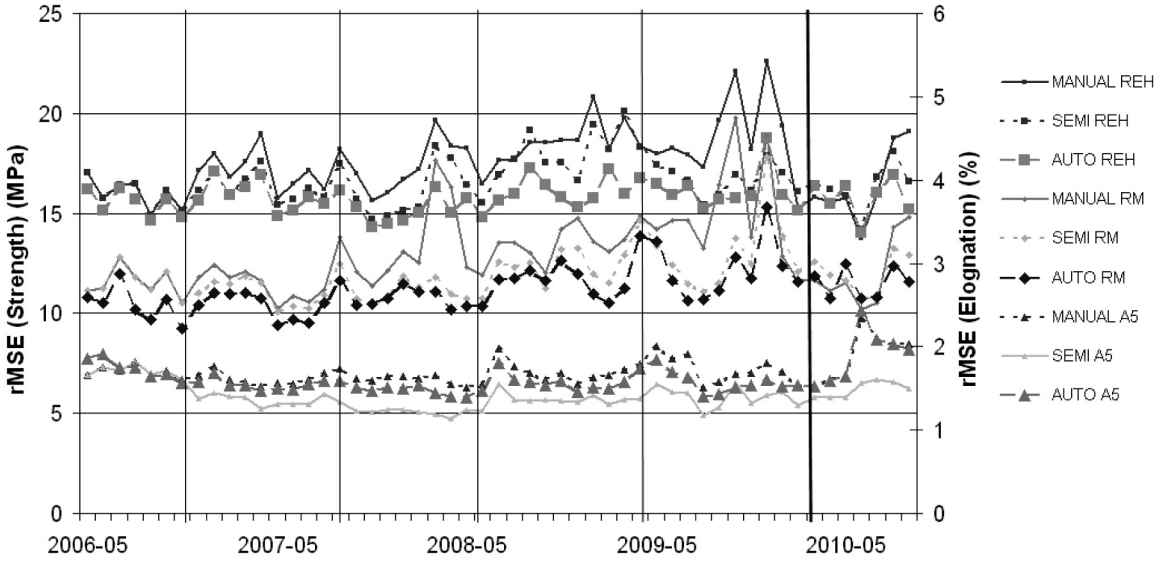

Figure 5. The observed monthly average of prediction error (root mean squared error) for the three model update scenarios. The model updates in the scenario SEMI are shown by vertical lines. The thick vertical line shows the model update moment in the scenario MANUAL.

The differences in the prediction accuracy between the scenarios were relatively small and similar for all of the modelled response variables. In the scenario MANUAL, the number of bad predictions increased until the model was updated and therefore the predictive performance was worst in this scenario. The predictive performance was best in the scenario AUTO (Figure 5).

The amount of work required by the model maintenance operations was highest in scenario MANUAL and lowest in the scenario AUTO. However, the difference between the scenarios MANUAL and SEMI is not very high compared to the complexity of the semi-automatic maintenance system. Based on the predictive performance and estimated workload and costs of the maintenance operations, the scenario AUTO seems most appealing. However, in this case, the semi-automatic approach was preferred because of its flexibility to support structural changes in the prediction formula. In addition, the semi-automatic approach was experienced more reliable and transparent for the users of the model application.

7. Discussion and Conclusion

The semi-automatic model maintenance system developed during the research has been successfully installed into the information system of a steel mill, and the testing stage is currently ongoing. However, it is still too early to evaluate the long-term success of the application. Thus far, the role of the model maintainer has been mainly to confirm the proposed model fitting and acceptance actions that the maintenance system has proposed in average two per year per model. Based on first experiences, it is expected that the system will give economic benefits because improved prediction accuracy in the product planning decreases production costs and the costs needed to maintain the good prediction accuracy will now be small.

The distinction of the application and the model file to separate software modules improves the re-usability and expandability of software and provides advantages in the maintenance of models and model applications in long-term. The drawback is that the distinction implies that each model update requires the generation of a new model file. Thus, the model maintenance system has to possess an ability to generate new model files. In the Ruukki case application, the distinction between the model files, model applications and the model maintenance was seen important, because it minimizes dependencies between the models, the model maintenance and the utilization of the models. An important motivation for the separation is that the model maintenance system developed

can be later expanded to support models other than double-generalized linear models and then utilized in the maintenance of other prediction models made use of in the production process.

The model maintenance may require the introduction of new explanatory variables, major changes in the model structure, new split to sub-models or careful, knowledge-based selection of data and model structure. Implementing a fully automatic system with these kinds of complex procedures into software requires hard work. A semi-automatic approach may leave some of the work to humans, which decreases the complexity and development costs of the model maintenance system.

Each application domain has its own premises which must be taken cognizance in the maintenance of related statistical models: the reasons that cause the update need, the distribution of process variables and its time-dependent changes, the number and characteristics of input variables, and the characteristics and stability of modelled relationship. Thus, the model update needs in different processes are different. In some cases, a semi-automatic model maintenance system may be a cost-efficient method to maintain good prediction accuracy. The semi-automatic approach seems most suitable to a situation where the prediction model is implemented as a separate software module independently of the model application and model updates are needed relatively often. If model update is needed rarely, the setup cost of the maintenance system overwhelms the benefit.

The case study presented here gives guidelines for the implementation of a semi-automatic model maintenance system. The results achieved form a remarkable step forward, although this article does not formulate a general framework for semi-automatic model maintenance system. The development of a more general customable technical framework and algorithm library to aid the design of semi-automatic model maintenance systems is an important topic for further research.

Acknowledgment

The research was supported by Ruukki and the Finnish Funding Agency of Technology and Innovation.

References

[1] F. BERZAL, I. BLANCO, J-C. CUBERO, N. MARIN Component-based data mining frameworks. *Communications of the ACM*, (45) (2002), 97–100. elvex.ugr.es/idbis/dm/docs/2002-TMiner.pdf

[2] L. BOTTOU Online learning and stochastic approximations. In *Online Learning in Neural Networks* (D. SAAD, Ed.), (1998) pp. 9–42. Cambridge University Press, Campridge. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.4918&rep=rep1&type=pdf

[3] P. CHAPMAN, J. CLINTON, R. KERBER, T. KHABAZA, T. REINARTZ, C. SHEARER, R. WIRTH *CRISP-DM 1.0 Step-by-step data mining guide*. The CRISP-DM consortium, 2000. http://www.crisp-dm.org/CRISPWP-0800.pdf

[4] J. W. HARDIN, J. M. HILBE *Generalized linear models and extensions* (2nd edn.). StataCorp, Texas, 2007.

[5] I. JUUTILAINEN, J. RÖNING Planning of strength margins using joint modelling of mean and dispersion. *Materials and Manufacturing Processes*, (21), (2006), 367–373.

[6] I. JUUTILAINEN, J. RÖNING How to compare interpretatively different models for the conditional variance. *Journal of Applied Statistics* (37) (2010) 983–998.

[7] P. KADLEC, B. GABRYS Evolving on-line prediction model dealing with industrial data sets. Presented at 2009 *IEEE Workshop on Evolving and Self-Developing Intelligent Systems Proceedings*, (2009) pp. 24–31, Nashville. http://eprints.bournemouth.ac.uk/9530/

[8] G. KOOP, S. POTTER Estimation and forecasting in models with multiple breaks. *Review of Economic Studies*, 74 (2007), 763–789. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=994037

[9] H. KOSKIMÄKI, I. JUUTILAINEN, P. LAURINEN, J. RÖNING Two-level clustering approach to training data instance selection: a case study for the steel industry. Presented at *Proceedings of the IEEE International Joint Conference on Neural Networks*, (2008) pp. 3043–3048, Hongkong, China.

[10] H. KOSKIMÄKI H, I. JUUTILAINEN, P. LAURINEN, J. RÖNING (2008) Detection of Correct Moment to Model Update. *Lecture Notes in Electrical Engineering: Informatics in Control, Automation and Robotics*, (24) (2008), 87–94.

[11] P. LAURINEN P, L. TUOVINEN, J. RÖNING Smart Archive: A component-based data mining application framework. Presented at *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, (2005) pp. 20–26, Wroclaw, Poland.

[12] I. MILETIC, S. QUINN, M. DUDZIC, V. VACULIK, M. CHAMPAGNE An industrial perspective on implementing on-line applications of multivariate statistics. *Journal of Process Control*, 14 (2004), 821–836. http://www.sciencedirect.com/science?_ob=GatewayURL&_method=citationSearch&_uoikey=B6V4N-4BYNPFR-1&_origin=SDEMFRHTML&_version=1&md5=77fae9ae62d0f8aa7ed767a9272633fe

[13] R. PECHTER Conformance standard for the predictive model markup language. Presented at *Proceedings of the 4th international workshop on data mining standards, services and platforms*, (2006) pp. 6–13, Philadelphia, Pennsylvania. http://www.mpi-inf.mpg.de/~ifrim/KDD06/docs/w_5.pdf#page=8

[14] G. K. SMYTH Generalized linear models with varying dispersion. *Journal of the Royal Statistical Society, series B*, 51 (1989), 47–60. http://www.jstor.org/stable/2345840?cookieSet=1

Received: May, 2009
Revised: September, 2011
Accepted: October, 2011

Contact address:
Ilmari Juutilainen
Data Mining Group, Computer Engineering Laboratory
Department of Electrical and Information Engineering
PO BOX 4500, 90014, University of Oulu, Finland
e-mail: ilmari.juutilainen@ee.oulu.fi

ILMARI JUUTILAINEN completed an M.Sc. in statistics at the University of Oulu in 2002 and a Dr.Tech. in embedded systems at the same university in 2006. He works as a research scientist in the Data Mining Group at the University of Oulu. His research is focused on applied industrial statistics.

LAURI TUOVINEN completed an M.Sc. in information processing science at the University of Oulu in 2005. He works as a research scientist in the Data Mining Group at the University of Oulu. His research is focused on software engineering of data mining systems.

PERTTU LAURINEN completed an M.Sc. in statistics at the University of Oulu in 2000 and a Dr.Tech. in embedded systems at the same university in 2006. His research is focused on data mining systems. Currently, he works at Indalgo Inc.

HELI KOSKIMÄKI completed an M.Sc. in mathematics at the University of Oulu in 2003. She completed a Dr.Tech. in embedded systems at the same university in 2009. She works as a research scientist in the Data Mining Group at the University of Oulu. Her research is focused on industrial applications of machine learning.

JUHA RÖNING completed his M.Sc. in engineering at the University of Oulu in 1983 and a Dr.Tech. at the same university in 1992. He is a professor of embedded systems and he works as the head of the Department of Information Engineering at the University of Oulu. His main research interests are intelligent systems and mobile robots.