A Generic Data Analytics System for Manufacturing Production

Hao Zhang, Hongzhi Wang*, Jianzhong Li, and Hong Gao

Abstract: The increase in the amount of manufacturing information available means that big data can be collected and, with appropriate deep analysis, could be of great value to manufacturers. However, most small manufacturers cannot afford the overhead of a professional data analytics team. To address this problem, in this paper a generic data analytics system, Generic Manufacturing Data Analytics system (GMDA), is proposed. This system can perform most manufacturing data analytics tasks and users can easily carry out data analysis even if they have no prior knowledge or experience of data analytics. To establish such a system, we designed an abstract language, GMDL, to describe the manufacturing data analytics tasks. Aimed at factory data analytics, several algorithms were selected, tuned, optimized, and finally integrated into the system. Some noteworthy techniques were developed in GMDA such as proper algorithm selection strategy and an optimal parameter determination algorithm. Case studies show the practicability and reliability of the system.

Key words: manufactory; data analytics; data mining; optimization

1 Introduction

Knowledge is always the most valuable asset in a manufacturing enterprise[1]. However, a new Industrial Revolution, named Industry 4.0, promoting smart manufacturing is emerging in an increasing number of manufacturing enterprises[2]. In these enterprises, large amounts of data are generated and collected. Analysis of such big data could bring great opportunities for innovation, lower costs, better response to customer needs, optimal solutions, intelligent systems, etc.[2] Most existing data analytics techniques used in manufacturing are aimed at specific scenarios[3–10]. This type of data analytics can achieve a satisfactory result, but is not practical because it is not universal and requires a team of data analysis experts. Problems also exist in commercial applications. Some manufacturing enterprises have used big data analysis platforms, e.g., Mercedes-AMG combines SAP Business Suite and sensor technology to early detect engine failures[11] and the Daimler Group implemented a comprehensive IBM SPSS Modeler that increased productivity by about 25%[12]. Therefore, large-scale manufacturing enterprises are already benefiting from data analytics. However, it is impossible for small and medium manufacturing enterprises to afford the time and money required for big data analytic services, and neither do they need the complex data analytic services provided by other companies. A complete data mining process can be divided into six phases according to CRISP-DM methodology[13], i.e., business understanding, data understanding, data preparation, modeling, evaluation, and deployment. However, the complexity and diversity of manufacturing processes means that it is extremely difficult to make the whole data analytics process generic and suitable for a variety of manufacturing processes and problems[14]. At this point in the study we focus on the modeling phase and propose a data analytics system named GMDA (Generic Manufacturing Data Analysis system), which has the following three major advantages:
Highly automated: With the data analytics requirements and data as the inputs, it can automatically accomplish end-to-end data analytics tasks with minimum human effort. It solves the conflict between the need for urgent data analysis in small and medium manufacturing enterprises and existing data analysis platforms being time and money consuming.

Friendly to non-expert users: It provides a data analytics task description language, GMDL (Generic Manufacturing Data analysis task-describing Language), to describe the data analytics tasks that may appear during the manufacturing process. Non-expert users can easily describe data analytics tasks using GMDL without any domain knowledge. Thus, GMDA is friendly to non-expert users and is therefore appropriate for small and medium manufacturing enterprises, because they do not generally employ data analytics experts.

Generic: GMDA and GMDL cover most of the data analysis tasks in manufacturing processes, e.g., socket pogo pins’ status monitoring, optimization of setting parameters, and identification of defects and failures and their causes. Common manufacturing data analytics tasks, like those mentioned in Section 2.1, can be described by GMDL and processed by GMDA.

The rest of this paper is organized as follows: Section 2 discusses related works. Section 3 details the design and definition of the abstract language GMDL. Section 4 introduces the GMDA framework and presents the whole workflow of a data analysis task. A parameter optimization approach is proposed in Section 5. Optimal algorithm selection strategies are discussed in Section 6. Section 7 presents several case studies to show the practicality and reliability of this system. Finally, the whole paper is summarized in Section 8.

2 Background

In this section, we first discuss related research on manufacturing data analysis, then introduce three use cases to fully describe GMDA.

2.1 Related work

Manufacturing data are typically noisy, highly correlated, and very often are randomly missing for various reasons such as faulty sensors and computer communication errors. The objects of analysis are often closely tied to real applications. Among many fields, the semiconductor industry is the one in which the applications are implemented most. The authors of Ref. [6] implemented a framework with a neural network to extract hidden knowledge in production data and speed up quality control without any further knowledge of the manufacturing environment. In Ref. [3], two different methods were used to detect unqualified pogo pins to guarantee the correctness of product quality tests. Furthermore, iron and steel manufacturing uses specialist techniques and is a capital-intensive process system composed of mining, coking, puddling, and preprocessing of melted iron and steel scrap, etc. In Ref. [7], an improved APRIORI algorithm based on rough set theory to extract the relationship among different production attributes was proposed. In Ref. [4], several regression models were established to model the annealing process of cold rolled steel sheets.

Of course, data analytics can also be applied to many other manufacturing fields. The authors of Ref. [5] built a decision tree with a C4.5 algorithm to extract rules from carpet manufacturing data. In Ref. [8], rough set theory was used to extract the rules for solder-ball defects. Multi-layer perception networks to estimate the quality of a template were established in Ref. [9], and in Ref. [10], neural networks and CART were used to predict the quality of a glass coating. To the best of our knowledge, current studies concerning manufacturing production are mainly aimed at specific data analysis scenarios, no one has yet proposed a generic data analytics system for non-expert users. Even the most popular statistical analysis software, IBM SPSS, which has visual interfaces, cannot perform data analytics tasks until a user manually chooses a model and sets some parameters. To make data analytics available to small and medium manufacturing enterprises and non-expert users, we aim to standardize the modeling phase method for manufacturing data analytics and establish the first highly automated and generic data analytics system, i.e., GMDA. Our goal is to develop a system that, instead of requiring users to select models and set parameters manually, only requires basic information about a task, e.g., task type, target attribute, and data address, to accomplish data analytics tasks.

2.2 Use cases for GMDA

We use the following three different use cases to describe GMDA more clearly. These three use cases are common in manufacturing, covering most of
the scenarios for data analytics in the industry. Each use case represents a type of data analytics task and shows some of the features of the system.

**Inventory Forecasting Use Case:** Inventory forecasting is one of the major tasks in manufacturing and includes raw material inventory forecasting. Accurate and reliable inventory prediction can guarantee a smooth production process\[^{16}\]. Here we assume that the forecasting target is the inventory of a PCB (Printed Circuit Board) and the address of the training data is `inventoryOfPCB.csv`.

**Car Evaluation Use Case:** When manufacturing enterprises are going to launch a new product, it needs to match the public’s aesthetic. We can use a rule extraction model to extract rules regarding the public’s aesthetic that were hidden in previous user feedback data on various car models. Here we take a Car Evaluation Data Set from the UCI machine learning repository as training data\[^{17}\]; the address is `carEvaluation.csv` and the target is `carAcceptability`.

**Tool Condition Monitoring Use Case:** Manufacturing systems are becoming more complex and are subject to failures that adversely impact their reliability, availability, safety, and maintainability\[^{18}\]. For example, in the high-speed milling process, a worn milling tool might irreversibly damage a workpiece\[^{19}\]. In such a case, real-time monitoring of the condition of the tools can help the operator avoid catastrophic events. Here we take a Steel Plates Faults Data Set as training data\[^{20}\]; the address is `steelPlatesFaults.csv` and the target attribute is `Faults`.

### 3 GMDL

Different from other data analytics platforms, GMDA only needs users to input a basic description of the task, e.g., task type, data address, etc., rather than the whole process configuration including parameter values. As the interface for the whole system, GMDL is one of its most essential parts. It directly determines what tasks GMDA can perform and thus is crucial to the whole system, therefore it requires perfect design. We attempt to make it concise and easy to understand for inexperienced users.

We first give some examples of GMDL in Section 3.1, then give the whole definition of GMDL and explain its essentiality in Section 3.2.

#### 3.1 GMDL for use cases

To fully describe a task, we must specify at least three key points:

- The goal of task;
- The dataset;
- Target attribute (if the task is prediction).

Each point is independent, and to make GMDA easier for inexperienced users, we initially propose a basic formal structure for the GMDL language, where COMMAND is the description of the task that GMDA can understand. That is:

**COMMAND \rightarrow SETTING COMMAND | SETTING:**

**SETTING \rightarrow PROPERTY = “VALUE”**.

Next, we give the GMDL for the Use Cases in Section 2.2.

#### 3.1.1 Car evaluation

**The goal of task:** The goal of the task is to extract the rules on the public’s aesthetic hidden in previous user feedback data and predict user reaction to new car models. Therefore, two SETTINGs are added to COMMAND.

- **object =“prediction”:** indicates that the type of this task is prediction;
- **rules =“T”:** indicates that this task needs to extract rules from the dataset.

**The dataset:** The data to be analyzed in this example is `carEvaluation.csv`. Therefore, SETTING `trainFile=“carEvaluation.csv”` is added.

**Target attribute:** The attribute to be predicted from the dataset is `carAcceptability`, which means the acceptability of this type of car by the masses. Hence SETTING `target = “carAcceptability”` is added.

Finally, the most concise COMMAND for the Car Evaluation Use Case is: `trainFile = “carEvaluation.csv” object =“prediction” rules =“T” target =“carAcceptability”`.

The COMMANDs for the other two Use Cases in Section 2.2 can also be obtained by following the steps mentioned above.

#### 3.1.2 Inventory forecasting

The most concise COMMAND in the Inventory Forecasting Use Case is: `trainFile = “inventoryOfPCB.csv” object = “prediction” sequential =“T” target = “currentNumOfPCB”`.

#### 3.1.3 Tool condition monitoring

The most concise COMMAND in the Tool Condition Monitoring Use Case is: `trainFile = “steelPlatesFaults.csv” object = “prediction” target=“Faults”`.

#### 3.1.4 Example of alternative SETTING

Apart from the key points mentioned above, some
alternative SETTINGS should be added when they are needed. For example, header indicates whether the first line of the data is the name of the variables or not. The value of the header is “T” by default. When the data does not contain the names of the variables, SETTING header = “F” is added.

3.2 Definition of language GMDL

COMMAND includes multiple independent SETTINGS, which are independent because they are assignments for different PROPERTYs. The meaning of each PROPERTY is shown in Table 1. Table 2 shows the configuration of each PROPERTY. Next, the meaning of each group is described and the task classification is detailed.

3.2.1 Group of PROPERTY

According to Refs.[3–10], PROPERTYs can be divided into three groups: Essential, Alternative, and Additional. Essential: PROPERTYs in this group must be specified in COMMAND. They are the core of COMMAND and cannot be set to default. In addition, when the value of object is prediction, target also belongs to this group. Alternative: PROPERTYs in this group have default values. They will be in COMMAND when their values need to be changed (as in Section 3.1.4). If these PROPERTYs in COMMAND are missing, the system can still work but it cannot achieve best performance. Additional: This group consists of the PROPERTYs that exist just for additional operations. PROPERTYs in this group can be specified by users if necessary.

3.2.2 Task type classification

Among all the configurations in Table 2, object is the most important PROPERTY to design. It is impossible for our system to make every single task a unique object value. Therefore, the tasks are classified into multiple types and each type is represented by a unique object value. The classification of the tasks is subject to the following criteria:

- Tasks can be categorized into at least one task type.
- Each task type has a practical application.

These criteria make sure that each task type is indispensable. Finally, tasks are classified into the following five categories:

Clustering groups a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). For example, in Ref. [21] cluster analysis (K-Means) is used to group similar fermentation behaviors. It was found that the final quality of a wine could be evaluated using the first three days’ fermentation data.

Association Analysis discovers interesting relations between variables in large databases. For example, the authors in Ref. [7] proposed an improved APRIORI algorithm based on rough set theory and extracted association rules from steel manufacturing data.

General Prediction: All the prediction tasks, which are not Prediction via Decision Tree or Time-Series Data Prediction, are general prediction tasks. The authors in Ref. [10] used neural network and CART approaches to predict product quality and find the best process variables.

Prediction via Decision Tree obtains a tree-like model, which represents several classification rules, from large datasets. The authors of Ref. [5] built a C4.5 decision tree model from carpet production data and improved the production system performance using
the rules obtained.

**Time-Series Data Prediction:** A significant volume of manufacturing data is time-series in nature⁹, e.g., inventory of items and power consumption. In Ref. [23], a time-series analysis technique was used to correctly predict the required amount of inventory items in Iran’s ZoabAhan steel company.

To sum up, Fig. 1 shows frequently occurring data analysis tasks in manufacturing and their corresponding task types. These five task types are easy to distinguish and are thus friendly to non-experts.

### 4 Framework

In this section, we present the framework design of GMDA and demonstrate its workflow. Next, we describe the challenges in the design and implementation of the system that are resolved in the latter sections.

#### 4.1 System framework

GMDA is based on R language. All the data analysis operations in GMDA are translated into R statements. GMDA is like an interlayer between data analysis requirements and data analysis tasks described by R language, which can be processed on existing engine such as Rgui or Rstudio.¹⁰ The overall framework design of GMDA is shown in Fig. 2. To be highly automated and generic, GMDA mainly relies on three major system components, a **data processor**, an **algorithm selector**, and a **parameter optimizer**. In the next section we introduce these three components by demonstrating the GMDA workflow.

#### 4.2 System workflow

A complete data analytics process begins with a user’s command. GMDA receives the task description and data processor automatically decides the attributes’ type and displays it to the user. The user can adjust the attributes’ type using commands if they wish. Then, **algorithm selector** selects the most appropriate algorithm according to the task type and data characteristics detailed in Section 6. If necessary, **data processor** will process the data to make sure that it is suitable for the selected algorithm. For example, multi-layer perceptron does not tolerate missing or discrete values and these must be removed. Next, if the algorithm has parameters to be determined, **parameter optimizer** will determine the optimum parameter values, as detailed in Section 5. Following this, GMDA uses the selected algorithm and determined parameter settings to analyze the processed data and establish an analysis model. Finally, the analysis results are obtained and users can confirm if they are satisfied with it, if not, they can adjust the parameters or even change the algorithm until they are.

#### 4.3 Challenges

The framework of our system brings the following two challenges.

##### 4.3.1 Challenges in parameter optimizer

Parameters greatly affect the performance of models. For example, Ref. [25] examines parameters such as splitting criterion, minimum parent size, minimum leaf size, and maximum number of splits for the construction of an optimal decision tree. Table 3 shows the parameters that are to be determined in GMDA. It is
very difficult for parameter optimizer to automatically determine their appropriate values. Section 5 presents different optimization methods for various algorithms.

4.3.2 Challenges in algorithm selector

As shown in Fig. 1, common data analysis can be classified into five types. Some types of task have a specific model. For example, GMDA will choose decision tree model if a task needs to extract rules about a certain attribute from the dataset. As a result, some models are integrated into GMDA because they are the most appropriate for this type of task. However, when it comes to general prediction, a single algorithm is not enough because of the variety in the dataset. Therefore, we chose three candidate algorithms, see Table 4, to increase the performance of GMDA. These were selected because each covers a different type of dataset. For example, random forest is generally suitable for a dataset in which the target attributes are discrete, while LASSO is suitable for datasets with a linear relationship between attributes. Other algorithms can also be integrated into GMDA provided they are suitable for datasets that have not been covered before and the optimum values of their parameters can be determined automatically. It is obvious that when GMDA performs a general prediction task, algorithm selector must choose the most suitable algorithm for the data. This feature of algorithm selector is undoubtedly challengeable. Section 6 describes the technical details for the implementation of algorithm selector.

5 Parameter Optimization

Table 3 shows the parameters to determine. These parameters must be carefully determined as they can greatly affect the performance of these models. One of the reasons for integrating the models in Table 4 into GMDA is to find ways to automatically determine the optimum values of their parameters. Generally, two types of parameter are optimized:

- **R-parameter**: parameters whose optimum values can be determined by rules.
- **C-parameter**: parameters that are not R-parameters.

We developed different optimum parameter determination methods for these two types of parameter. In this section, first we propose a parameter optimization algorithm to determine the C-parameter, then we present the rules of other parameters to respectively determine the R-parameter.

5.1 Parameter optimization algorithm

Here, we propose a parameter optimization algorithm, but first a few terms are explained:

- **F(θ)**: the evaluation of function F when the parameter value is θ. The specific index of algorithm performance evaluation is detailed in Section 6.2.1.
- **R{m, n};**: means that the parameter can take values greater than m and less than n.
- **P**: means that the parameter keeps P decimals places.
- **C**: indicates that during each iteration, R{m, n} will equally divide into C parts.

5.1.1 Algorithm description

Algorithm 1 shows a method to determine the desired value for the C-parameter. It starts with an initial R{m, n}. Initial R{m, n} is equally divided into C parts to create several new smaller R’{m, n}. These R’{m, n} are stored in rangeList. Then, the algorithm enters the appropriate value by an iterative search process. In each iteration, tempList is cleared, and minimum is assigned to e if parameterList is empty. e records current highest F(v). The algorithm randomly selects a value v for each R{m, n} in rangeList (Line 7). For each v, the algorithm gets an evaluation of F, e.g., F(v). If F(v) is higher than e, the corresponding R{m, n} will reserve, F(v) will be assigned to e, and parameterList and tempList are cleared at the same time (Lines 9–11). Otherwise, corresponding R{m, n} are removed. For each reserved R{m, n}, the algorithm calculates the number of available values in it (depend on P) (Line 15). If only one value is available, this will be added into parameterList (Line 17). If more than C available values are available, R{m, n} will be evenly divided into C parts and added into tempList (Line 22). If there are no more than C available values in R{m, n},
Algorithm 1 Parameter Optimization Algorithm

Input: \( F(\theta), R[m, n], P, C \).

Initialize: \( e \leftarrow \min \), \( parameterList \leftarrow \text{null} \), \( rangeList \leftarrow R[m, n] \), \( flag \leftarrow \text{null} \).

Output: parameterList

1: \( \text{while rangeList is not null do} \)
2: \( \text{tempList} \leftarrow \text{null} \)
3: \( \text{if parameterList is null then} \)
4: \( e \leftarrow \min \)
5: \( \text{for all } R[m_i, n_i] \in rangeList \text{ do} \)
6: \( \text{flag} \leftarrow \text{false} \)
7: \( \text{randomly select a value } v \text{ between } [Rm_i, n_i] \text{ (precision depends on } P) \)
8: \( \text{if } F(v) > e \text{ then} \)
9: \( e \leftarrow F(v) \)
10: \( \text{flag} \leftarrow \text{true} \)
11: \( \text{clear parameterList and tempList} \)
12: \( \text{else if } e = F(v) \text{ then} \)
13: \( \text{flag} \leftarrow \text{true} \)
14: \( \text{if } \text{flag} = \text{true then} \)
15: \( \text{num} \leftarrow \text{number of the available value in } R[m_i, n_i] \text{ (influenced by } P) \)
16: \( \text{if num} = 1 \text{ then} \)
17: \( \text{add num into parameterList} \)
18: \( \text{else if } 1 < \text{num} < C \text{ then} \)
19: \( \text{for } k \in \text{all available value in } R[m_i, n_i] \text{ do} \)
20: \( \text{tempList} \leftarrow \text{tempList} \cup R[k, k] \)
21: \( \text{else} \)
22: \( \text{evenly divide } R[m, n] \text{ into } C \text{ subsets, add them all in tempList} \)
23: \( \text{return} \) parameterList

all the available values will be tried and added into tempList (Lines 19 and 20). The tempList will be used as rangeList in the next round (Line 23). Iteration continues until rangeList is empty. All the appropriate values of the parameters are stored in parameterList.

5.1.2 Algorithm analysis
The complexity of the algorithm is \( O(N \log N(M)) \cdot I_{\text{single decision tree}} \), where \( N \) is split size, \( M \) is the number of available values in the parameter value range, which is partly influenced by the precision index. A parameter optimization algorithm can be also applied to the real type parameters as long as the precision index is given.

5.1.3 Experiments
We use the prediction accuracy to measure the model’s performance. Improvement in model performance refers to the ratio of accuracy improvement from the parameter optimization algorithm to the original accuracy, i.e., \( 100 \times \left( \frac{\text{accuracy}_{\text{optimized}} - \text{accuracy}_{\text{default}}}{\text{accuracy}_{\text{default}}} \right) \% \). The breast tissue data set was obtained from the UCI machine learning repository[17] to examine the performance of the parameter optimization algorithm. We used one of the decision tree’s parameters, i.e., minsplit, as an example. Figure 3 shows that decision tree will over-fit the training set if minsplit is too small. In this case, when the minsplit value keeps increasing, the accuracy first increases then decreases. We ran the parameter optimization algorithm to find the optimum value for minsplit, Fig. 4 shows that the parameter optimization algorithm can improve the model performance by an average of 10.1% and at most 50%.

5.2 Parameter optimization rules
Some models’ parameters’ optimum values can be derived from rules. This type of model can be integrated into GMDA without degrading the automation of the system. Following are two examples:

- ARIMA (Auto Regressive Integrated Moving Average) model is a generalization of the standard ARMA model[26]. It is widely used to deal with time-series data in manufacturing processes. It consists of three components: \( AR \) is determined by parameter \( p \), \( I \) is determined by parameter \( d \), and \( MA \) is determined by parameter \( q \)[27]. These three parameters’ optimum values can be determined by the method described in Ref. [28].

- LASSO algorithm can solve the multiple collinearity problems and tackles the redundant attribute problem in regression analysis. Its core parameter \( \lambda \) can be determined by invoking the cv.glmnet function in the GLMNET package, which can automatically find the \( \lambda \) that obtains the best result and records it as lambda.min.

6 Algorithm Selection
The features of factory data for analysis are different.
Hence, it is common that a prediction algorithm achieves very high accuracy for certain data, but its performance on uncertain data is poor. To overcome this deficiency, three alternative algorithms are implemented in GMDA, as shown in Table 4. Thus, algorithm selector is implemented to handle the problem of selection of the most appropriate algorithm. This section first specifies the problems encountered in the implementation of algorithm selector in GMDA. Each problem is then discussed and carefully dealt with. Finally, examples are given and the reliability of algorithm selector is tested experimentally.

6.1 Problems in realizing algorithm selector

Several problems arise in the implementation of algorithm selector. The first is that we need to determine the metrics to evaluate an algorithm’s performance. The second is that, as we want to predict a data’s most appropriate algorithm based on previous records, we must establish a knowledge base. To establish a knowledge base, we must choose the proper features to accurately describe the dataset and record the most appropriate algorithm. The last is that, once we have the record, we need to choose a proper method to predict the most appropriate algorithm.

6.2 Solution

6.2.1 Algorithm evaluation

When we chose the evaluation criteria, we were aware that the evaluation results were for non-expert users, but it was still necessary for them to have a general idea of the model’s performance and be able to choose the most appropriate model for a certain dataset when they attempted to update knowledge base.

When the target attribute is discrete, the accuracy is defined as the proportion of the number of correct predictions to the total number of predictions. We did not choose recall, precision, F1 Score, etc., as evaluation criteria, as such measures could have made the evaluation result too complex when the task was multi-classification. When the target attribute is continuous, the opposite number of the root means square error is used for accuracy because it is easy to understand and is also a common evaluation approach for regression. Thus, the accuracy decreases when the algorithm achieves better performance if the attribute is continuous or discrete.

In addition, execution time is another factor that needs to be taken into consideration.

6.2.2 Data features selection

There is no indication so far as to which combination of data features can accurately, completely, and concisely describe the data. After a large number of tests, the data feature set used in GMDA was finally obtained. It contains a total of nine data characteristics as follows: proportion of discrete attributes (shown as SymPr in Table 5), total number of attributes (#Attr), type of target attribute (TarType), number of different values in the target attribute (#TarVal), size of data set (#Obs), proportion of missing values (MisValPr), information entropy (Entropy), joint information entropy (MultiInf), and mutual information (MutInf).

6.2.3 Algorithm performance record

We recorded the most appropriate algorithm for the data instead of each performance on it. The reason is that, when evaluating algorithms, GMDA randomly divides the data into training set and test set, and the results of the parameter optimization varied each time. Therefore, it is unreliable to record the accuracy of each candidate algorithm in each data set. Examples of the knowledge base are shown in rows 2–4 of Table 5.

6.2.4 Algorithm selection methodology

GMDA adopts the idea of KNN to select the most appropriate algorithm to analyze the data. When new data arrives, GMDA extracts the features from it, then selects the three most similar records in knowledge base based on the KNN algorithm. These three records vote for the most appropriate algorithm for the data.

Not only that, if algorithm selector chooses an inappropriate algorithm, a user can manually run other candidate algorithms, choose the most appropriate algorithm, and record it into the knowledge base, thus updating it. The whole process requires no experience

<table>
<thead>
<tr>
<th>SymPr</th>
<th>#Attr</th>
<th>TarType</th>
<th>TarVal</th>
<th>#Obs</th>
<th>MisValPr</th>
<th>Entropy</th>
<th>MultiInf</th>
<th>MutInf</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.036</td>
<td>28</td>
<td>Factor</td>
<td>7</td>
<td>1941</td>
<td>0</td>
<td>61.08</td>
<td>53.51</td>
<td>267.473</td>
<td>Random forest</td>
</tr>
<tr>
<td>0.029</td>
<td>35</td>
<td>int</td>
<td>6</td>
<td>366</td>
<td>0</td>
<td>29.538</td>
<td>23.635</td>
<td>119.485</td>
<td>Random forest</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>int</td>
<td>7</td>
<td>4898</td>
<td>0</td>
<td>31.451</td>
<td>23.256</td>
<td>44.209</td>
<td>MLP</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>int</td>
<td>3</td>
<td>178</td>
<td>0</td>
<td>21.957</td>
<td>16.775</td>
<td>55.862</td>
<td>MLP</td>
</tr>
</tbody>
</table>
or knowledge of data analytics.

6.3 Algorithm selection of use case

Here we take the Tool Condition Monitoring Use Case in Section 2.2 as an example. This task is classified as General Prediction. As soon as GMDA receives the data, its features are extracted as shown in the first row of Table 5. Then, the three most similar records (shown in rows 2–4 of Table 5) in the knowledge base are selected. From rows 2–4 of Table 5, we know that the most appropriate algorithm for the Steel plates faults data set is random forest.

We manually ran the three alternative algorithms on the Steel plates faults data set to observe whether algorithm selector chose the most appropriate algorithm. From the first row of Table 6 we can see that the chosen algorithm outperforms other two.

6.4 Reliability of algorithm selector

Several different datasets were obtained from the UCI machine learning repository. We ran each alternative algorithm through GMDA on each data set, and record the performance, i.e., accuracy and run time. Then, we performed a General Prediction analysis task on each data set to check whether the algorithm selected by algorithm selector was the most appropriate. Rows 2–5. Table 6 shows the algorithm selection results from four different datasets. We can easily see that for each dataset, the performance of chosen algorithm is obviously better than the other two, which means that algorithm selector is reliable.

7 Case Studies

This section takes three common scenarios, belonging to different task types in the manufacturing industry, as examples to illustrate the feasibility, generality, and reliability of GMDA.

7.1 Time-series prediction

In manufacturing, some attributes may develop a deeper relationship over time and may change with a certain degree of regularity. Some of these attributes can reflect component status, such as the status of a socket pogo pin[3], and the PCB inventory in the Inventory Forecasting Use Case. Here, we use this case, see Section 2.2, as an example. After data processor preprocesses the time-series data, algorithm selector chooses ARIMA to perform the task, and the parameter optimizer automatically determines the parameters $p$, $d$, and $q$ of the ARIMA model.

The performance of the ARIMA model established by the auto.arima function is shown in Fig. 5, and

![Fig. 5 Prediction of auto.arima.](image)
the performance of the ARIMA model established by GMDA is shown in Fig. 6. We observe that the prediction curve in Fig. 6 is closer to the historical data compared with that in Fig. 5. This means the ARIMA parameters determined by GMDA are much more appropriate than those determined by the other method.

7.2 Rules extraction

In the manufacturing process, understanding the hidden rules between the attributes can be of great help. Manufacturers can work in accordance with rules to optimize product design, improve the production process, and determine better process parameters to meet the requirements of the manufacturing enterprise. The Car Evaluation Use Case in Section 2.2 is a typical analysis task in manufacturing. Upon receiving the command, algorithm selector chooses the decision tree algorithm for this use case. parameter optimizer then determines the optimum values of the three parameters (shown in Table 3), and GMDA establishes a decision tree, shown in Fig. 7. The accuracy of model reached 88.0%.

Figure 7 clearly shows that the most important features for car acceptability are safety and capacity, however, purchase price and price of maintenance can also influence this. Car manufacturers can refer to these rules during their design process so that they can produce cars that most people like.

7.3 General prediction

General prediction can be used almost everywhere. For example, manufacturers can predict the quality of product according to its manufacturing configuration so that the best configuration can be determined, or they can predict the quality of product according to its intermediate product status so that a defective product can be found and recycled immediately.

Taking the Tool Condition Monitoring Use Case in Section 2.2 as an example: After GMDA receives the command, algorithm selector first extracts the features of the dataset (shown in the first row of Table 5), then selects the three most similar records from the knowledge base (shown in rows 2–4 of Table 5). These three records determine the most appropriate algorithm for dataset, which is random forest in this use case. To ensure that the random forest algorithm was truly the most appropriate algorithm for this use case, all the alternative algorithms were applied to the dataset (shown in the first row of Table 6). The results show that random forest outperforms the other algorithms on this data set.

8 Conclusion and Future Work

A generic and low-user-threshold manufacturing data analytics system, GMDA, is proposed in this paper. This will enable small and medium manufacturers to conduct data analysis tasks using their own data and to benefit from it, even if they have no knowledge or experience of data analytics.

To establish such a system, a descriptive language was designed, through which the user can describe analysis task easily. A knowledge base was established so that our system could select, based on the KNN algorithm, the most appropriate algorithm for the data. Several algorithms, including APRIORI, decision tree, CMEANS, ARIMA, MLP, and GLMNET, were integrated into the system, according to data analytics tasks common in the manufacturing industry, to ensure that the system covered most tasks. A number of novel methods were implemented to enable the automatic accomplishment of the whole data analysis process even when the user was inexperienced (low-user-threshold). Experiments show that the system is practical and reliable enough to accomplish common data analytics tasks in manufacturing and can easily be used by non-experts. There is no doubt that such a system is a powerful tool for small and medium-sized manufacturers and could benefit thousands of
manufacturing enterprises.

However, with the constraints of the R language, GMDA cannot handle big datasets at present. In the future, we plan to replace the R part in GMDA with RHadoop or SparkR to make it available for use with big data.

Acknowledgment

This paper was partially supported by the National Natural Science Foundation of China (Nos. U1509216, 61472099, and 61602129), the National Key Research and Natural Science Foundation of China (Nos. U1509216, 1610YFB1000703), National Sci-Tech Support Plan (No. 2015BAH10F01), and the Scientific Research Foundation for the Returned Overseas Chinese Scholars of Heilongjiang Province (No. LC2016026).

References

[23] A. Fazel, M. Saraei, and P. Shamshinejad, Mining time series data: Case of predicting consumption patterns
in steel industry, in Proc. 2nd Int. Conf. Software Engineering and Data Mining (SEDM), Chengdu, China, 2010, pp. 501–505.


Hao Zhang is a postgraduate student in computer science at Harbin Institute of Technology. His research area is big data analytics. He received the BEng and MEng degrees from Harbin Institute of Technology in 2016 and 2018, respectively.

Hong Gao is a professor and doctoral supervisor at Harbin Institute of Technology. She is a senior member of CCF. Her research interests include data management, wireless sensor networks, and graph database. She received the PhD degree from Harbin Institute of Technology.

Hongzhi Wang is a professor and doctoral supervisor at Harbin Institute of Technology. His research area is data management, including data quality, XML data management, and graph management. He has published more than 100 papers in refereed journals and conferences. He is a recipient of the outstanding dissertation award from CCF and a Microsoft Fellow, and has an IBM PhD Fellowship. He received the BEng, MEng, and PhD degrees from Harbin Institute of Technology in 2001, 2003, and 2008, respectively.

Jianzhong Li is a professor and doctoral supervisor at Harbin Institute of Technology. He is a senior member of CCF. His research interests include database, parallel computing, and wireless sensor networks.

Jianzhong Li is a professor and doctoral supervisor at Harbin Institute of Technology. He is a senior member of CCF. His research interests include database, parallel computing, and wireless sensor networks.