Predictive maintenance based on event-log analysis: A case study

J. Wang
C. Li
S. Han
S. Sarkar
X. Zhou

Predictive maintenance techniques are designed to help anticipate equipment failures to allow for advance scheduling of corrective maintenance, thereby preventing unexpected equipment downtime, improving service quality for customers, and also reducing the additional cost caused by over-maintenance in preventative maintenance policies. Many types of equipment—e.g., automated teller machines (ATMs), information technology equipment, medical devices, etc.—track run-time status by generating system messages, error events, and log files, which can be used to predict impending failures. Aiming at these types of equipment, we present a general classification-based failure prediction method. In our parameterized model, we systematically defined four categories of features to try to cover all possibly useful features, and then used feature selection to identify the most important features for model construction. The general solution is sufficiently flexible and complex to address failure prediction for target equipment types. We chose ATMs as the example equipment and used real ATM run-time event logs and maintenance records as experimental data to evaluate our method on the feasibility and effectiveness. In this paper, we also share insights on how to optimize the model parameters, select the most effective features, and tune classifiers to build a high-performance prediction model.

1. Introduction

Product reliability is a key contributor to the success of companies that manufacture equipment. In order to avoid the impact of unexpected breakdowns, scheduled maintenance of the equipment is necessary. One simple process for performing scheduled maintenance is to assign technicians to check the equipment on a regular basis. This is referred to as preventative maintenance (PvM) [1]. However, this approach is not cost-effective, since too frequent visits result in wasted labor and travel costs, while too large a gap between visits can result in problems occurring without warning. Predictive maintenance (PdM) [2] is a promising alternative due to its ability to automatically make predictions about when and what repairs should be performed. The main goal of PdM is to schedule maintenance at a point in time when the maintenance activity is most cost-effective, and before the equipment loses performance and function.

In order to construct a PdM model, it is important to have access to data that can provide insights on the running condition of a piece of equipment. Most types of critical enterprise equipment record their internal operating state in the form of logs or system messages. For example, in the case of bank ATMs (automatic teller machines), all device operations—e.g., inserting a card, querying one's balance, withdrawing cash, and generating a receipt—are stored in its system logs. In addition, all errors and warnings occurring in an ATM are also recorded in logs. This is very important, since the errors contain informative messages, and incorporating them into a PdM model is essential for monitoring the condition of an ATM and gaining insights to be able to detect potential problems in advance of their actual occurrence. In the following sections, we will focus on ATM PdM for our case study.

The task of predicting equipment failure based on a continuous feed of the content of its log poses many challenges. First, transforming a PdM problem into a machine learning task is a challenge. Once we formulate

Digital Object Identifier: 10.1147/JRD.2017.2648298

0018-8646/17 © 2017 IBM

it as a standard machine learning problem, we can adopt many mature machine learning techniques to solve the problem. A second challenges is the extraction of features from log entries in order to represent "event sequences" that are indicative of an impending fault. For any given machine learning technique, the quality of the features used as model predictors has a direct impact on model quality, since discriminative features can improve the generalization ability of the prediction model. Third, since feature extraction is usually based on external prior knowledge, it is inevitable that noisy or redundant features will be identified as predictive features. Noisy features or redundant features may result in the over-fitting of the model in training, especially when the volume of the training data set is smaller; thus, removal of these features is essential for improving the accuracy of the model.

From a business perspective, PdM represents a new opportunity for both equipment owners as well as maintenance service providers (MSP). The IBM Technology Support Services (TSS) organization provides maintenance services to customers who own IBM and non-IBM devices such as server, storage, network equipment, and ATMs. Enterprise-class systems already include predictive failure models for key components (e.g., disk drives) that can "call home" with a prediction (including timeline) of a failure, allowing TSS to schedule preventive maintenance and prevent a disruption at the customer location. Extending that capability to ATM-class equipment would be a key benefit for banks, whose equipment does not have such features. As the consumer of PdM technology, MSPs have two key requirements, the first of which is that PdM models should be able to accurately predict potential equipment failures and with sufficient lead time in order for the MSP to respond with a corrective action before the failure occurs. The second requirement is that PdM models should be built by using frameworks that are generalized for a certain class of equipment so that predictive failure models can be easily customized and optimized for different machine models, such as ATMs manufactured by Wincor, NCR, or Diebold. The ability to use a common model-building technology—even for different classes of equipment such as ATMs, servers, and storage—is an even more challenging goal.

To address the challenges posed by the two MSP requirements outlined above, and targeting predictive models that primarily use system logs, we propose a flexible model-building solution that can support PdM for a broad class of machines, and incorporate a set of methods and toolkits to support the end-to-end customization and optimization of failure prediction models. The solution can be easily extended to include real-time performance monitoring data as model features, but since such data is challenging to collect in deployments where IBM as an MSP does not have direct access to the customer environment, the incorporation of such data for modeling is not highlighted in this paper.

We treat PdM as a binary classification problem, where the classes represent the outcomes that the equipment is either likely to fail, or to not fail, in a given time window. By analyzing the event or log streams, we identified five types of features for representing the training/testing instances used for predictive modeling—basic features, advanced statistics-based features, pattern-based features, failure similarity features, and device profile features. In order to reduce the adverse effects of including redundant (correlated) features on modeling, we applied feature-selection algorithms [3] to remove them before training a model. After feature selection, we built failure prediction models using multiple machine learning techniques. Experimental results using real-world ATM failure data demonstrate that our proposed general solution is effective for building event/log-based PdM models, and the method and toolkit can effectively support the end-to-end process of PdM model customization and optimization.

The remainder of this paper is organized as follows. In Section 2, we provide a brief description of related work. In Section 3, we introduce our problem formulation for event/log based equipment. We present the proposed failure prediction method, technology, and toolkits in Section 4. The extensive experimental results using real-life ATM data sets are presented in Section 5 to prove the effectiveness of the prediction model and the general solution. We end with some concluding remarks in Section 6.

2. Related work

The work most closely related to our approach is log-based machine learning model design. In [4], the authors attempt to find and understand the common patterns that result in a failure, but they do not consider proactive prevention of failures. In addition, the authors of [5–7] take advantage of principal component analysis (PCA) to detect anomalies, based on the parsed textual log messages, and then use PCA to detect anomalies. Recently, [8] presented an approach for log-based PdM. This work utilizes a Multi-Instance Learning (MIL) [9] technique to build predictive models from log data. The authors' approach was developed with active involvement from domain experts and was evaluated and shown to be effective by both machine learning and domain standards, but it has not presented guidance on feature extraction from equipment logs.

Another related work involves sequential pattern mining [10, 11] the goal of which is to find statistically relevant patterns across data points, where the values appear in a sequence. The authors of [12] discuss a typical case where sequential pattern mining is used for failure prediction. Patterns of log events have a strong association with failures; thus, it is an important predictor for failure prediction. However, sequential patterns do not include the

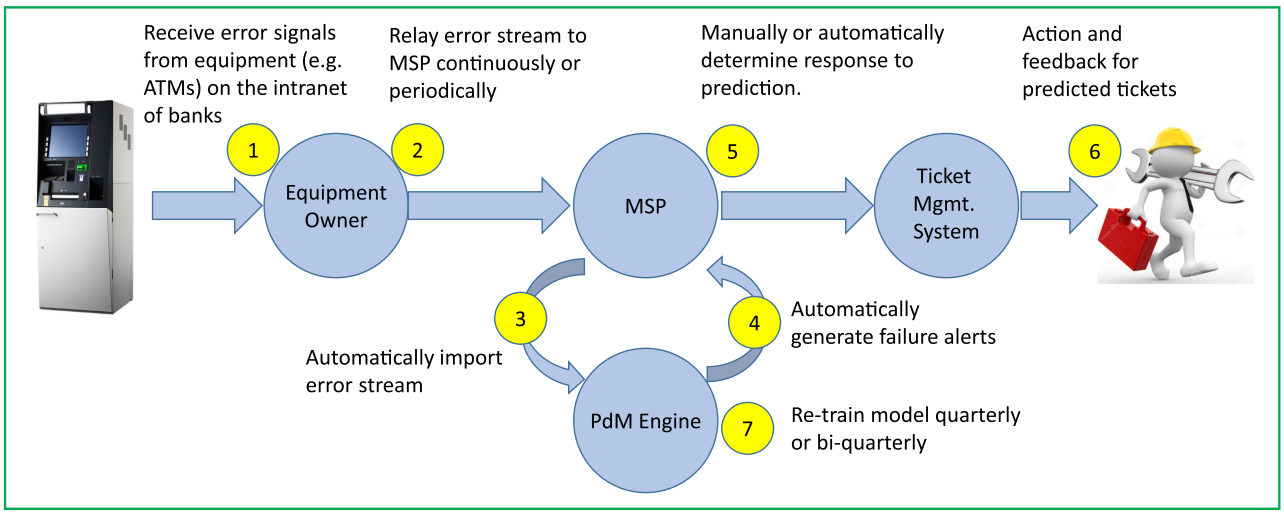

Figure 1

The procedure of failure prediction.

information like time intervals among events, the distribution information in time series. Using only sequential patterns to predict failures is not the best approach to our problem.

Rule-based expert systems [13, 14] have some similarities with sequential pattern mining in that rules are defined based on preconditions, which can be considered to be patterns that can be matched to input data. Once equipment runtime conditions satisfy the precondition of one or more rules, one rule is triggered (the selection based on a conflict resolution algorithm), and its action part is executed to either generate a failure alert, or to change the internal state of the rule-based system, which may trigger other rules. The rule preconditions (patterns) are not mined by processing data, but are defined by domain experts. In practice, this approach is easy and effective for small-scale systems. However, the challenges of engaging SMEs (subject-matter experts) to manually keep the rules up-to-date for large scale systems have been the major impediment to using rule-based systems for real-world tasks.

The survival model [15, 16] and Cox model [17] are also related to our work. These are a branch of statistics for analyzing the expected duration of time until one or more events occur, like die or fail. In the survival model and Cox model, the failure records are required, and error logs are not necessary. This is an advantage of the statistics-based approach. However, without error logs, some important predictive information will be missed, and the prediction performance will be significantly decreased.

In summary, some of the above methods may be able to solve failure prediction problems for a specific class of equipment. However, any method in isolation is not designed to serve our purpose—to provide MSPs a general solution and method to predict failures for log-based equipment. However, some concepts and insights from these references are useful and valuable for building our general solution. For example, in [8], the concepts of prediction interval, infected interval, and responsive duration are helpful for us to formulate our problem. Sequential pattern mining was adopted in our model to build a group of pattern-based features. The concept of survival probability in the survival model is used to replace the device age as one of the profile-based features.

3. Problem description

In traditional maintenance support services, once a device fails, the equipment owner issues a maintenance service request (or a "ticket") for the MSP, who may send a technician to repair the failed device if it cannot be fixed remotely (e.g., by applying a software update). In the PdM scenario, devices (e.g., ATMs) generate log data, which is the property of the equipment owner (e.g., the bank), whereas the MSP owns the ticket data. To support PdM, we need to collect both log and ticket data to apply machine learning techniques to build a prediction model, and then deploy the prediction model (including a runtime engine for running the model) which receives real time log/system messages, and generates failure alerts and reports those alerts to the MSP. **Figure 1** shows the typical process of deploying and using PdM technology. The success of this new business model depends highly on the performance and effectiveness of the prediction model.

In our solution, we transform the prediction problem into a classification problem. We develop and optimize a prediction model capable of classifying devices into two categories: likely to fail or likely not to fail in the near future.

Table 1 Example of ticket data. (Raw comments are reproduced with undefined acronyms and typos.)

Ticket ID	*ATM ID*	*Ticket start*	*Ticket stop*	*Activity comment*
1	ATM01	2014/1/1 14:41:00	2014/1/1 19:13:00	REPLACED STACKER, INIT SENSORS:#1,#2 TEST OK.
2	ATM02	2014/1/1 11:55:00	2014/1/1 15:55:00	CLEARED JAMMED BILLS FROM SINGLE REJECT. CLEANED SENSORS AND BELTS. TESTED OK.
3	ATM03	2014/1/1 10:18:00	2014/1/3 11:22:00	PRINTER FAILING BECAUSE OF OUTSIDE TEMPERATURE. ABM HAS HISTORY OF THIS ISSUE. BANK HAS NEGATIVE AIR PRESSURE AND DUCTS COLD AIR INTO ABM. SPOKE WITH COUNTRY SUPPORT.

Table 2 Example of system error messages. We can extract meaningful fields from these raw system messages. Here, *Compo* stands for component, which indicates the component generating the message. *DEP* stands for depository, and *PTRP* stands for printer. *StClass* and *StCode* stand for the state class and the state code. In the Wincor user manual, the detail error description can be looked up by using the two fields.

ATM ID	*Date*	*Time*	*Raw error codes*
ATM01	2014/12/17	16:36:12	Compo = DEP,CN = {StClass = 0x00000017,StCode = 0x28008025, StWarn = 0x00000000,. . .}
ATM01	2014/12/27	10:59:04	Compo = PTRP,CN = {StClass = 0x00000037,StCode = 0x28008101, StWarn = 0x00000000,. . .}

3.1. Data description

Data is the key to building a prediction model. For log-based equipment failure prediction, we collect valuable data from the following categories as inputs to extract features: maintenance records, system log/messages, inventory data, utilization data, environmental data, and configuration information. As data sources, equipment owners must provide system log/messages and utilization data and the MSP must provide maintenance records. Both the equipment owner and MSP should be able to provide additional data. The maintenance records are usually recorded as service requests/tickets, which may include the date and time of creation, support teams involved, reported symptom, resolution, call-backs if the resolution was not successful, and parts used, if any. The real-time system log/messages from the equipment may be structured or unstructured, and include device identity, timestamp, message content, priority, code, and action. The inventory data includes equipment manufacturer information, machine type, serial number, location, and installation date. The utilization data, if available, can include, for example, transaction records for banking devices and operational logs for medical devices. The environmental data can include temperature, humidity, pollution levels, and location information. The configuration information for IT devices can include hardware configuration, software configuration, and scheduled and planned updates. Although all such data may be useful for failure prediction, the most useful are maintenance data, system log/messages, and inventory data. In addition, if both maintenance data and system log/messages include component information, failures can be predicted at a component level. Otherwise, it must be limited to a device level.

In the ATM case, system log includes the runtime error messages generated by ATMs. Some examples of tickets, error messages, and ATM inventory data are listed in **Tables 1**, **2**, and **3**, where we only show the key content for building the prediction model. In our proposed prediction method, such data is treated as model inputs. All ATM data used in our experiments is structured and only needs to be formatted to meet the requirements of various machine learning algorithms. For unstructured log data, one additional step is to extract key event data using a log formatter, which is customized for a specific type of log. The extracted event data can be utilized in our proposed failure prediction method.

3.2. Key time-related concepts

Based on our study of ATM logs, system messages, and service requests submitted to IBM, two major types of error events appear in logs. The first type consists of minor (non-fatal) errors, such as would be the case for a card reader slowing by 5% or connection time-outs, which indicate a problem of some sort in the equipment, and which may or may not be a good predictor of whether some

Table 3 Example of inventory information.

ATM ID	*Manufacturer*	*Model*	*Installation date*
ATM01	WINCOR/NIXDORF	ProCash 2100 RL	2004/8/18
ATM01	WINCOR/NIXDORF	ProCash 1500 RL	2006/7/4

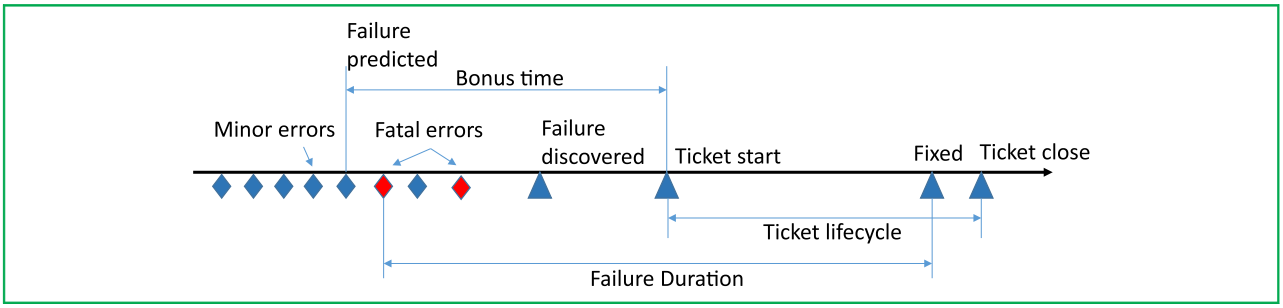

Figure 2

The key concepts in a time series of events.

component of the ATM will break down in the near future. In spite of these minor errors being reported, the equipment can continue to function well, and end users may not be aware of any impending problem. The second category of error events consists of fatal errors, such as would be the case of a motor problem related to the conveyor transport, which indicates that the device, or some components within it, has failed, and the equipment cannot function normally or may have stopped working altogether. **Figure 2** illustrates such log messages as a time series of error events and ticket creation events concerning a failure of an ATM. After the breakdown is discovered by the equipment owner, a service request is submitted to the MSP, a ticket is created to track the problem, and repair service is scheduled. For ATMs, where remote problem determination and application of a software fix is not the norm, typically a technician arrives on-site to resolve the problem. After the problem is fixed, the ticket is closed and updated with details about the service provided, such as the parts replaced for the failed component(s), the hours spent on the service call, etc. Time-related concepts are associated with the entire process, starting with the occurrence of the fault until the problem is resolved. Several concepts include:

- *Failure duration*: the downtime of the equipment, which is the time duration between the initial occurrence of the fatal error and the final resolution of the problem when the device is operational again.
- *Ticket lifecycle*: the duration between the time when a ticket is created and the time when the ticket is closed with a resolution.
- *Bonus time*: the duration between the time when equipment failure is reported by a predictive failure model, and the time when a ticket is created because a fatal error was reported. Bonus time represents the response time available to the MSP, brought by the PdM, to prepare for a maintenance action. It enables the transformation of the MSP's business from being reactive to proactive, replacing unscheduled maintenance actions with scheduled actions.

Most classes of equipment that log error events have the same failure behavior and problem resolution process as described above. This commonality enables the development of a methodology and a general solution to identify and analyze predictive error events and predict failures before they occur. An additional point to be mentioned is that the outputs of prediction models are the creation of tickets instead of the occurrences of the triggering failures. This is caused by the fact that ticket creation, instead of failure occurrence, is recorded as data and used the dependent variable of the model.

4. Proposed failure prediction method

For many types of equipment that produce system logs, we can build prediction models based on a common failure prediction method. Since the consumers of the prediction model are the MSP and equipment owners, we should consider their concerns as the key performance evaluation metrics of the prediction model. For equipment owners, the key requirement is that virtually all failures should be predicted and corrected (prevented) before they occur; thus, *recall* is the most important model accuracy metric. (Generally speaking with respect to information retrieval, recall, also known as sensitivity, is the fraction of relevant instances that are retrieved.) For the MSP, the main concern is *precision*, since false alarms raised by the model will lead to wasted human labor and unnecessary parts replacement cost. Depending on contracts and the collaboration models, both metrics may be interesting for both the equipment owner and MSP. The following metrics are therefore suitable for evaluating the failure prediction models that we build: precision and recall, AUC (area under ROC curve, where ROC refers to receiver operating characteristic), AUPRC (area under the precision recall curve), and F1-Measure.

4.1. Construction of the prediction model

As described in Section 3.1, error event logs and ticket creation data are the major inputs available to the failure

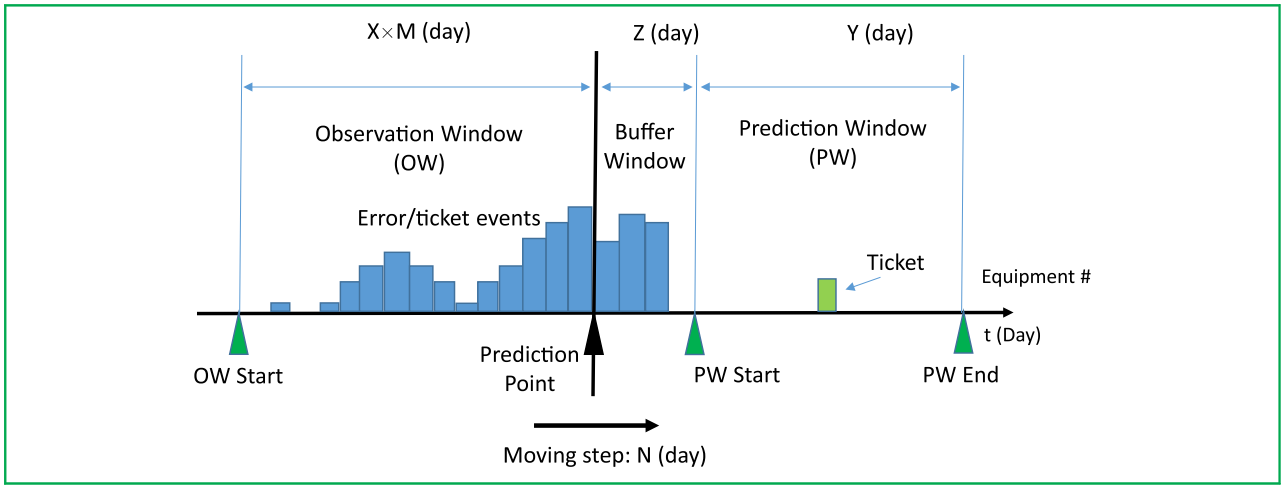

Figure 3

The moving-window-based failure prediction method for event-based equipment.

prediction model. They are collected from different data sources—the equipment owner and the MSP. Before building a prediction model, we do data alignment first—merging the two data sets together and sorting them by occurrence time stamp to generate a consolidated event stream for each equipment instance. The consolidated event stream is the input to the instance generation and feature extraction steps for building the prediction model.

Figure 3 shows how to generate training and testing instances. We consider the time point when the model makes a prediction as the *prediction point*. For each prediction point, we generate a learning instance **S**, which consists of a vector of features and a label. Before a prediction point, we set an *observation window* (OW), which includes a list of events involving errors and tickets. The observation window is split into X sub-windows. Each sub-window is a *measurement unit* where features are extracted. The size of the sub-window is the parameter M in days. Therefore, the size of the observation window is $X \times M$. We set a *prediction window* (PW) after the prediction point with the size parameter Y. If there is any ticket event in the prediction window, then the label of the instance is set as positive; otherwise, the instance label is negative. By moving the prediction point and corresponding windows, we can create different instances for training and testing the model. The *moving step* parameter N reflects the prediction frequency, which depends on the business requirement, and is also the sampling density, which has an impact on the total amount of learning instances generated from the log data. The distance from the prediction point to the nearest ticket is the *bonus time* available to the MSP to respond to the failure prediction. The MSP will prefer to have sufficient time to respond to the prediction; thus, we add an optional buffer window between the prediction point and prediction window to control the minimal bonus time. The size of the buffer window is another parameter Z. Although a larger buffer window is more beneficial to the MSP, setting Z to a larger value will decrease the performance of failure prediction. For one instance, if there is one or more tickets in a buffer window, we consider it as not a qualified training instance. We exclude such instances from the training set. For testing instances, if there are one or more tickets in a buffer window or prediction window, we regard it as a positive instance, otherwise, as a negative instance. For one device, we can obtain a set of instances by moving the prediction point along the time axis. By repeating this process for all devices, we can obtain all the training and testing instances to construct the prediction model.

During instance generation, we extract features from the observation window. We define four types of features: statistics-based, pattern-based, failure similarity based, and profile-based features.

4.1.1. *Statistics-based features*

Statistics-based features represent the distribution of each individual error type in the observation window. Using statistics-based features, we expect the classification algorithm to learn the relationships between the distribution of equipment errors and failures. Here, we note that if there are tickets in the observation window, we deal with tickets as a kind of special error type.

The basic statistical measure used to describe error distribution is the count of the number of error instances in each measurement unit, for each error type. We denote these basic statistics-based features as $\mathbf{B} = \{c_{ij}, i \in [1, T], j \in [1, X]\}$, where T is the number of error types, X is the

number of sub-windows, and c_{ij} is the error instance number of the *i*-th error type in the *j*-th sub-window. Thus, the length of **B** is $X \times T$.

Aside from basic features, we also define advanced statistics-based features to measure error distribution in details. For a given error instance of a given error type, we define its *distance to the prediction point* as $d = t_p - t_e$, where t_p is the timestamp of the prediction point and t_e is the timestamp of the error instance. Meanwhile, we define the *error interval* between two continuous error instances with the same error type as *v*. Then, we can define advanced features using a vector **A**, where $\mathbf{A} = \{\min(\mathbf{D}_i), \max(\mathbf{D}_i), \text{mean}(\mathbf{D}_i), \text{mean}(\mathbf{V}_i), stdDev(\mathbf{V}_i), i \in [1, T]\}$. Here, *T* is the number of error types, and $\mathbf{D}_i$ is the *distance* set for the *i*-th error type in the observation window. $\text{Min}(\mathbf{D}_i)$ is the minimal distance to the prediction point for all error instances in the *i*-th error type. $\text{Max}(\mathbf{D}_i)$ and $\text{mean}(\mathbf{D}_i)$ are the maximal and average distances for the *i*-th error type; similarly, $\mathbf{V}_i$ stands for the *error interval* set for the *i*-th error type. The function *stdDev* is used to calculate the standard deviation of error intervals. Here, the length of **A** is $5 \times T$.

4.1.2. Pattern-based features

Pattern-based features represent the association relationships among error types. When using these features, we expect the model to learn the predictive insights regarding relationships between error patterns and failures.

In our method, we define a pattern as a combination of error types that repeats in different observation windows. We follow the method shown in Figure 3 to extract pattern-based features. The first step is to identify all pattern candidates from all observation windows. For example, if three error types (e1, e2, and e3) occur in an observation window, then all combinations of these errors, without consideration of occurrence order, e.g., (<e1>, <e2>, <e3>, <e1,e2>, <e1,e3>, <e2,e3>, <e1,e2,e3>), are candidate patterns. The second step is to evaluate each pattern's confidence in predicting failures. We define the confidence of a pattern as the ratio of the count of pattern instances in positive instances to the count of pattern instances in all instances. We select a pattern as a feature if its confidence exceeds a predefined threshold (e.g., we set a value of 80% as the threshold in all experiments). During instance generation, if a pattern is found in the observation window, the corresponding pattern-based feature is set to 1, otherwise it is set to 0. Pattern-based features can be denoted as a vector **P** ($\mathbf{P} = \{p_i, i \in [1,Q]\}$), where *Q* is the number of selected patterns. *Q* depends on the confidence threshold.

4.1.3. Failure similarity features

Failure similarity features represent the relationship between repeat failures, and we expect the prediction model to learn such relationships between any two failures using such features. Based on our observation of real-world (ATM) error and ticket data, many failures of a given type often repeat, and each failure is preceded by errors of similar types before it occurs. This is the motivation for using failure similarity as a type of feature to predict repeat failures. The different types of errors that appear in the observation window of a sample instance can be denoted as a collection **G**. The ticket created most recently with respect to the current instance also has an observation window with a collection of error types, denoted as collection **H**. We define the failure similarity feature (denoted as *F*) for the current instance as the Jaccard distance [18] between **G** and **H**.

4.1.4. Profile-based features

Aside from the three types of prediction-point-dependent features defined above, some equipment-profile-related information (such as the machine model and the installation date) is also helpful for building an effective prediction model. We refer to these types of features as profile-based features, denoted by the vector **R**.

As a summary, given a prediction point, an instance **S** can be generated where $\mathbf{S} = (\mathbf{B}, \mathbf{A}, \mathbf{P}, F, \mathbf{R}, L)$, and *L* is the label of the instance. Aside from these features, our method also has 5 parameters: *X*, *Y*, *Z*, *M*, and *N*. This degree of parameterization enables the model-building method to be sufficiently flexible to be applied to many different types of equipment. From a parameter tuning perspective, we list some experiential guidelines for effective tuning:

- Fix *Y*, *Z*, *M*, and *N* with some reasonable values, and then tune *X* to get the best *X*. (The phrase "best X" refers to the value of X to obtain the best model performance.)
- Set the best *X*, fix *Y*, *Z*, and *N* with some reasonable values, then tune *N* to get the best *N*.
- Set the best *X* and *N*, fix *Z* and *N*, and then tune *Y* to get the best *Y*.
- Then, tune *N* and *Z*.

4.2. Method and toolkits for automatic evaluation and tuning of prediction model performance

In order to optimize the prediction model for any type of equipment, we need to experiment with different parameter configurations and features. To facilitate this procedure, we designed a method and toolkit to automatically evaluate and tune prediction model performance.

The method consists of seven steps: data preprocessing, pattern mining, instance generation, feature selection, model training, model testing, and model evaluation. For each type of equipment, the data preprocessing step needs to be manually customized to perform data cleansing, filtering, extraction of error events and ticket events, and consolidation of two categories of events to generate a single event stream. In pattern mining and instance generation, for each configurable parameter, we need to

provide the available value list for each parameter or parameter combination. For example, the size of the observation window can be 15, 30, 45, or 60 days, and the size of the prediction window can be 10, 20, 30, or 45 days. For a 30-day observation window, we selected some typical $X \times M$ combinations to perform experiments: 30×1, 15×2, 10×3, 6×5, 3×10, 2×15, and 1×30. Once these parameter configuration rules are set, pattern mining and instance generation can be automatically executed to generate a list of patterns and the learning instance sets. Then, for each instance set, the feature selection and model training toolkits can be configured to apply different feature ranking algorithms and classification algorithms in order to evaluate which algorithm is the most effective for the prediction problem and the specific equipment type. After testing and evaluation, a list of experiment reports is generated for review.

While the proposed features of our failure prediction method provide sufficient flexibility to be applied to a broad class of devices, one has to also address the issue of feature selection in order to eliminate redundant or noisy features, which can adversely affect model accuracy. Feature selection is necessary to improve the prediction and computation performance of the model. We use a two-step feature selection method: feature ranking and feature evaluation. First, by using a standard feature ranking algorithm, a ranked list of features is generated. Next we use a stepwise approach: by iteratively selecting the top N features from the ranked list, we train and test increasingly more complex models (i.e., with more features) until the model performance does not increase significantly when adding more features, which is the stopping criterion.

5. Experiments

To evaluate the effectiveness of our failure prediction method, we used ATM error event and device failure data to run a set of experiments. We targeted a large number of ATMs from a global bank headquartered in North America. The bank is also a customer of IBM's Technology Support Services. ATM error messages were provided by the bank, while ATM inventory and service request (ticket) data was provided by IBM TSS. This data includes event data for about 7 months from 1557 Wincor ATMs, covering five machine models: ProCash 2100 RL, ProCash 1500 RL, ProCash 2100 FL, ProCash 1500 FL, and ProCash 2250XE RL. The error messages include 128 error types. We used data from 1000 ATMs to train the prediction model, and tested the model with data from the remaining ATMs. Based on this data, we ran four categories of experiments.

5.1. Experiments on model parameters

The first category of experiments, falling into three groups, is used to find insights on how to configure model parameters to obtain the optimized prediction model. In our method, the most important parameters are the sizes of the observation window ($X \times M$) and prediction window (Y). In optimized prediction models for different types of equipment, the parameters ($X \times M$ and Y) will differ by equipment type since they depend on the nature of the equipment being modeled. For Wincor ATMs, we ran three groups of experiments. The result is shown in **Table 4**. For each group of experiments, we fixed some variables to evaluate how the target variable has an impact on the performance of the prediction model.

In the first group of experiments, the evaluation target is the observation window size, OW. According to the experimental results, we see that 30-day observation windows produce the best model performance if we use AUC as the evaluation metric, although the differences in performance are not significant for different OW sizes. This means that most of the significant error events that can predict failures occur within 30 days for Wincor ATMs.

In the second group of experiments, the evaluation target is the size of the prediction window Y. Results show that the bigger the prediction window, the better is the prediction performance, which has an obvious explanation. The AUC will be 1 if $Y = \infty$, because any ATM will eventually fail at some point in time. Although the explanation is simple, we cannot set $Y = \infty$ in our model. According to the business requirement provided by domain experts, the size of prediction window should not be bigger than 1 month.

The objective of the third group of experiments is to explore if for a given observation window size, different $X \times M$ combinations result in different prediction performances. The result shows that different combinations of $X \times M$ have no significant impact on the prediction performance. We chose 6×5 as the best combination of parameters since that choice resulted in the maximum value of AUC.

Note that the sizes of the training and test sets depend on the parameters. For the experiment with the best result, where $X = 6$, $M = 5$, $Y = 30$, and $Z = 0$, the size of training set is 13,383 samples, with 2,681 positive samples and 10,702 negative samples. The size of test set is 11,196 samples, with 1,104 positive samples and 10,092 negative samples. As discussed, if there is any ticket event in the prediction window, then the label of the instance, or sample, is positive; otherwise, the instance label is negative.

5.2. Experiments on feature groups

The second category of experiments is undertaken to evaluate the predictive effectiveness by individual feature types and combinations of types. According to Section 4, a learning instance can include five groups of features: basic statistics-based features (**B**), advanced statistics-based features (**A**), pattern-based features (**P**), failure similarity features (F), and profile-based features (**R**). As was the case

Table 4 Experimental results regarding how to configure model parameters to optimize prediction model for Wincor ATMs. All experiment data and experiment methods are described in the column experiment conditions. The first category of experiments is to tune X on Y, Z, with M fixed, where the best value for X is 6 (best in the sense of best model performance). In the second category, we observed that the bigger the Y value is, the better the performance is, but considering the limitation of the business requirement, we selected $Y = 30$. The third category is to tune M with X, Y, and Z fixed, and we obtained the best M at 5. For the experiment with the best result, where $X = 6$, $M = 5$, $Y = 30$, $Z = 0$, the size of training set is 13,383 samples, with 2,681 positive samples and 10,702 negative samples. The size of test set is 11,196 samples, with 1,104 positive samples and 10,092 negative samples.

Experiment conditions	*Evaluation target*	*Precision*	*Recall*	*AUC*
Parameters: $Y = 30, Z = 0, M = 5$	$X = 2, OW = 10$	0.794	0.473	0.81
Features: Basic, Advanced, Pattern-based, Failure Similarity, Profile	$X = 3, OW = 15$	0.792	0.486	0.813
	$X = 4, OW = 20$	0.796	0.488	0.814
Feature selection: Yes	$X = 6, OW = 30$	0.803	0.483	0.815
Classifier: Random Forest	$X = 9, OW = 45$	0.764	0.477	0.79
	$X = 12, OW = 60$	0.747	0.437	0.75
Parameters: $X = 6, M = 5, Z = 0$	$Y = 10$	0.742	0.466	0.797
Features: Basic, Advanced, Pattern-based, Failure Similarity, Profile	$Y = 20$	0.763	0.476	0.807
	$Y = 30$	0.803	0.483	0.815
Feature selection: Yes	$Y = 45$	0.798	0.482	0.815
Classifier: Random Forest	$Y = 60$	0.822	0.473	0.818
Parameters: $OW = 30, Y = 30, Z = 0$	$X \times M = 2 \times 15$	0.798	0.484	0.811
Features: Basic, Advanced, Pattern-based, Failure Similarity, Profile	$X \times M = 3 \times 10$	0.78	0.492	0.811
	$X \times M = 6 \times 5$	0.803	0.483	0.815
Feature selection: Yes	$X \times M = 10 \times 3$	0.789	0.476	0.814
Classifier: Random Forest	$X \times M = 15 \times 2$	0.81	0.475	0.812
	$X \times M = 30 \times 1$	0.794	0.475	0.815

Table 5 Evaluation of the predictive effectiveness by feature type and their combination. The experimental result shows that the order of importance of feature types is **A**, **B**, and **P**. The classification model trained with one of the first 4 feature combinations will have much better performance than trained with other feature combinations. This means all combinations with **A** are better than without **A**. All of them have the acceptable scores on *Precision*, *Recall*, and *AUC*.

Experiment conditions	*Evaluation target*	*Precision*	*Recall*	*AUC*
Parameters: $X \times M = 6 \times 5, Y = 30, Z = 0$	**B,A,P**	0.785	0.47	0.79
Features: failure similarity, profile	**A,P**	0.754	0.487	0.793
Feature selection: No	**A**	0.775	0.493	0.79
Classifier: Random Forest	**B,A**	0.775	0.466	0.789
	B,P	0.467	0.328	0.727
	B	0.457	0.322	0.727
	P	0.124	0.168	0.519

with the parameter tuning experiments, our goal was to understand which feature types or combination of feature groups are the most effective for ATM failure prediction. In this experiment, we mainly focused on evaluating feature types—**B**, **A**, and **P**. The experimental results in **Table 5** show that the order of importance of feature types is **A**, **B**,

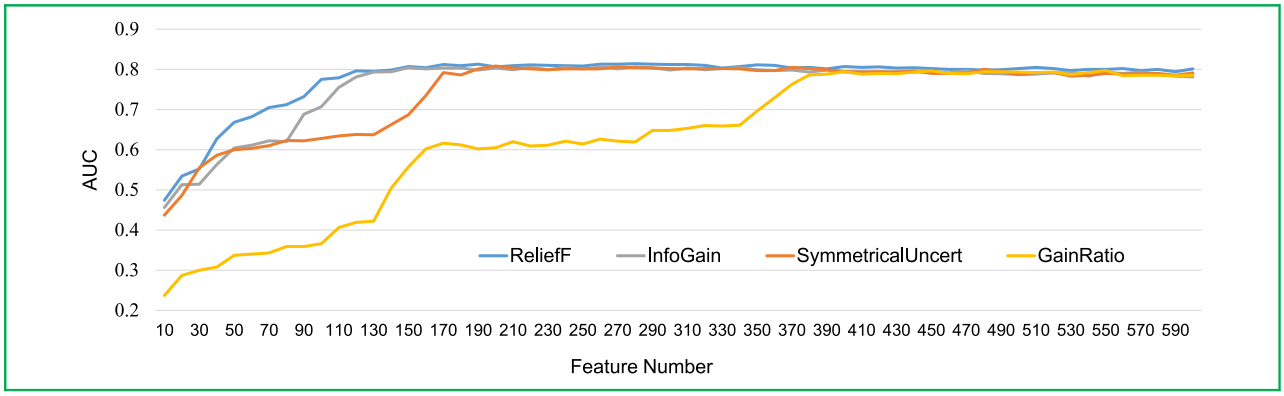

Figure 4

The evaluation of feature-selection methods. Four typical feature selection methods implemented in Weka are evaluated under the following model parameters: $X = 6$, $M = 5$, $Y = 30$, and $Z = 0$, using Random Forest as classification algorithm. The results show Relief and InfoGain can rapidly converge to the best feature set, where InfoGain only required 23 seconds, but ReliefF required 18,174 seconds. Thus, we selected InfoGain in our practice.

and **P**. The first 4 combinations are much better than the rest, but similar to each other. This means all combinations with **A** are better than without **A**. All of them have the acceptable scores on *precision*, *recall*, and *AUC*. In addition, although the similarity feature type consists of one feature only, our experiment shows that it can improve prediction performance (using AUC) by 2∼3%.

5.3 Experiments on feature selection

The third category of experiments is used to evaluate the effectiveness of applying feature selection technology. In our performance tuning method and toolkits, feature selection is one of the key steps used to improve prediction performance. In this experiment, we evaluated 4 feature ranking algorithms implemented in Weka (a machine learning open source project [19]): InfoGain [20, 21] GainRatio [22], ReliefF [23], and SymmetricalUncert [24]. The experiment (**Figure 4**) shows that the ranking results of ReliefF and InfoGain are more effective than SymmetricalUncert and GainRatio, because they selected the most important features at the top of the ranked feature list, thus enabling the feature set evaluation iterations to converge rapidly. However, to create the ranked feature list, InfoGain required 23 seconds while ReliefF required 18,174 seconds. Therefore, we chose to use InfoGain in our method and toolkits. In addition, applying feature selection technology improved AUC by 3% to 5% in our experiments.

Table 6 Classification algorithm comparison.

Classifier	*Implementation*	*AUC*
XGBoost	Python	0.835
RandomForest	Weka	0.815
AdaBoostM1	Weka	0.782
LibSVM	Weka	0.662

5.4. Experiments on classification algorithms

The fourth category of experiments was performed to evaluate different classification algorithms. We compared four classification algorithms: XGBoost [25], Random Forest [26], Ada Boost M1 [27], and LibSVM [28]. XGBoost, Random Forest, and Ada Boost demonstrated significant advantages over the other algorithms for our problem (shown in **Table 6**). We think this is because the first three are ensemble models—and generally speaking, ensemble learning is better than non-ensemble. In additional, XGBoost uses bag-based tree selection, but Random Forest uses optimization-based tree selection, so XGBoost is about 2% better than Random Forest.

6. Conclusion

PdM is a relatively new business model for managing enterprise equipment, which requires advanced enabling technologies. As the consumer of PdM technology, MSPs require not only high-performance prediction models, but also a reusable PdM solution and technology that can be customized and optimized for different types of equipment. In this paper, we proposed a novel and flexible parameterized PdM solution for event/log based equipment. Specifically, we solved the PdM problem by using machine learning techniques—classifying a piece of equipment into a binary class: likely to fail and not likely to fail. To train the prediction model, we systematically designed five types of features for generating sample

instances in order to cover the PdM problem for different types of equipment. Additionally, we applied feature-selection methods to eliminate redundant and noisy features in order to improve the performance of the model. Using the selected features, we adopted state-of-the-art classification approaches to build failure prediction models. We used ATMs as a typical example of a type of event/log-based equipment for which to validate our method. The experimental results demonstrated the effectiveness of the proposed method and toolkits for building PdM models. We also presented the salient findings from these experiments to guide the customization and performance tuning of failure prediction models for new types of equipment. In our model, the buffer window Z is an interesting concept. In future work, we plan to evaluate how it influences the prediction results.

In addition, in our practice, sometimes MSPs only have historical failure records, and have no system error logs. How to predict failures in this situation is a real business problem. For the new challenge, the authors in [17, 29–32] proposed some interesting approaches from a statistics and stochastic process perspective by using a Survival model, Cox model, and Hawkes process model. These areas will be our focus in the future work.

Acknowledgments

We thank Rich Dye and Kevin Wahlmeier (IBM Technical Support Services - TSS) for providing funding for the research work. We also thank John Bird, Richard Bernstein, Andrew Mouck, and Donald Hamilton [IBM Global Technology Services (GTS)] for providing important experimental data, business requirements, and insightful domain discussions. We thank Ea-Ee Jan (IBM GTS) for providing technical guidance and engaging us in many insightful modeling discussions. We thank Shao Chun Li, Wei Sun, and Alan Bivens (IBM Research) for their support and help on this research.

References

1. J. S. Dagpunar and N. Jack, "Preventative maintenance strategy for equipment under warranty," *Microelectron. Reliab.*, vol. 34, no. 6, pp. 1089–1093, 1994.
2. S. Diamond and A. Marfatia, *Predictive Maintenance for Dummies*. Hoboken, NJ, USA: Wiley, 2013.
3. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.
4. T. Li, F. Liang, S. Ma, and W. Peng, "An integrated framework on mining logs files for computing system management," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2005, pp. 776–781.
5. W. Xu, "Detecting large scale system problems by mining console logs," Ph.D. dissertation, Univ. California, Berkeley, CA, USA, 2010.
6. W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Mining console logs for large-scale system problem detection," in *Proc. 3rd Workshop Tackling Comput. Syst. Problems Mach. Learn. Techn.*, San Diego, CA, USA, 2008, p. 4.
7. W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online system problem detection by mining patterns of console logs," in *Proc. 9th IEEE Int. Conf. Data Mining*, Miami, FL, USA, 2009, pp. 588–597.
8. R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, "Log-based predictive maintenance," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2014, pp. 1867–1876.
9. O. Maron and T. Lozano-Pérez, "A framework for multiple-instance learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1998, pp. 570–576.
10. N. R. Mabroukeh and C. I. Ezeife, "A taxonomy of sequential pattern mining algorithms," *ACM Comput. Surveys (CSUR)*, vol. 43, no. 1, 2010, Art. no. 3.
11. G. Dong and J. Pei, *Sequence Data Mining*. New York, NY, USA: Springer, 2007.
12. M. J. Zaki, N. Lesh, and M. Ogihara, "PlanMine: Sequence mining for plan failures," in *Proc. 4th Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 1998, pp. 369–373.
13. K. L. Butler, "An expert system based framework for an incipient failure detection and predictive maintenance system," in *Proc. Int. Conf. Intell. Syst. Appl. Power Syst.*, Orlando, FL, USA, 1996, pp. 321–326.
14. R. C. M. Yam, P. W. Tse, L. Li, and P. Tu. "Intelligent predictive decision support system for condition-based maintenance," *Int. J. Adv. Manuf. Technol.* vol. 17, no. 5, pp. 383–391, 2001.
15. S. J. Richards, "A handbook of parametric survival models for actuarial use," *Scand. Actuarial J.*, vol. 4, pp. 233–257, 2012.
16. R. G. Miller, Jr., *Survival Analysis*. Hoboken, NJ, USA: Wiley, 2011.
17. Z. Li, S. Zhou, S. Choubey, and C. Sievenpiper, "Failure event prediction using the cox proportional hazard model driven by frequent failure signatures," *IIE Trans.*, vol. 39, no. 3, pp. 303–315, 2007.
18. J. Paul, "The distribution of the flora in the alpine zone," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912.
19. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Wintten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newslett.*, vol. 11, no. 1, pp. 10–18, 2009.
20. J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986
21. Y. M. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proc. Int. Conf. Mach. Learn.*, Nashville, TN, USA, 1997, vol. 97, pp. 412–420.
22. J. Han and M. Kamber, *Data Mining Concepts and Techniques*. San Mateo, CA, USA: Morgan Kaufmann, 2001.
23. K. Kira and L. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Proc. 10th Nat. Conf. Artif. Intell.*, San Jose, CA, USA, 1992, vol. 2, pp. 129–134.
24. S. S. Kannan and N. Ramaraj, "A novel hybrid feature selection via symmetrical uncertainty ranking based local memetic search algorithm," *Knowl.-Based Syst.*, vol. 23, no. 6, pp. 580–585, 2010.
25. T. Q. Chen and T. He, xgboost: eXtreme Gradient Boosting, 2017. [Online]. Available: http://cran.hafro.is/web/packages/xgboost/vignettes/xgboost.pdf
26. T. K. Ho, "Random decision forests," in *Proc., 3rd Int. Conf. Document Anal. Recognit.*, Montreal, QC, Canada, 1995, pp. 278–282.
27. Y. Freund and R. E. Schapire, "A short introduction to boosting," *J. Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, 1999.
28. C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 389–396, 2011.
29. J. Yan, Y. Wang, K. Zhou, J. Huang, C. Tian, H. Zha, and W. Dong, "Towards effective prioritizing water pipe replacement and rehabilitation," in *Proc. Int. Joint Conf. Artif. Intell.*, Beijing, China, 2013, pp. 2931–2937.
30. J. Yan, C. Zhang, H. Zha, M. Gong, C. Sun, J. Huang, S. Chu, and X. Yang, "On machine learning towards predictive sales pipeline analytics," in *Proc. 29th AAAI Conf. Artif. Intell.*, Austin, TX, USA, 2015, pp. 1945–1951.

31. J. Yan, M. Gong, C. Sun, J. Huang, and S. Chu, "Sales pipeline win propensity prediction: A regression approach," in *Proc. 2015 IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, Ottawa, ON, Canada, 2015, pp. 854–857.
32. S. Xiao, J. Yan, C. Li, B. Jin, X. Wang, H. Zha, and X. Yang, "Modeling contagious M&A via point processes with a profile regression prior," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016.
33. S. Xiao, J. Yan, C. Li, B. Jin, X. Wang, H. Zha, and X. Yang, "On modeling and predicting individual paper citation count over time," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, New York, NY, USA, 2016.

Received February 23, 2016; accepted for publication March 22, 2016. Date of current version March 8, 2017.

Jian Wang *IBM Research, Beijing, China (wangwj@cn.ibm.com).* Dr. Wang is a Research Staff Member in the Internet of Things and Services Research department at IBM Research - China. He received his B.S. and M.S. degrees in aircraft manufacturing engineering in 1994 and 1997, respectively, and his Ph.D. degree in aeronautics and astronautics manufacturing engineering in 2000, all from the Northwestern Polytechnic University. He subsequently joined IBM Research - China, where he has worked on e-commerce, model-driven business transformation, immersive web, Internet of Things, and cloud service scalability research. In 2013, he received the title of IBM Master Inventor. He is author or coauthor of 36 patents and 14 technical papers.

Changsheng Li *IBM Research, Beijing, China (lcsheng@cn.ibm.com).* Dr. Li is a Research Staff Member in the Internet of Things department at IBM Research - China. He received his B.E. degree from the University of Electronic Science and Technology of China (UESTC) in 2008, and his Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences in 2013. He also studied as a research assistant in The Hong Kong Polytechnic University from 2009 to 2010. He joined IBM Research - China in 2013. His research interests include machine learning, pattern recognition, and data mining.

Siyao Han *IBM Research, Beijing, China (bjhansi@cn.ibm.com).* Ms. Han is a candidate for a Master's Degree at Beijing Language and Culture University, majoring in computer science. Her research interests include machine learning, data mining, and recommendation systems. She received her B.S degree in computer science from Beijing Language and Culture University in 2013. She joined IBM Research - China as a summer intern in the Blue Pathway program. During her internship, she worked on cognitive predictive maintenance topics. She is author or coauthor of five technical papers.

Soumitra (Ronnie) Sarkar *IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598 USA (sarkar@us.ibm.com).* Dr. Sarkar is a Senior Technical Staff Member in IBM Research. He received undergraduate and graduate degrees in electrical engineering and computer science from the Indian Institutes of Technology, and a Ph.D. degree in computer science from the Ohio State University. He led architecture, development, and strategy initiatives in several IBM product divisions—networking, application integration and middleware, and storage—prior to joining IBM Research in 2006, where he worked on text mining and enterprise search, continuous availability using synchronous storage replication. He also worked on monitoring, automated incident management, and modular managed services for the IBM Cloud Division, and on analytics for IBM's Technology Support Services. He is a co-inventor/author on 44 patents and 19 papers.

Xin Zhou *IBM Research, Beijing, China (zhouxin@cn.ibm.com).* Dr. Zhou is a Research Staff Member in the Cognitive Internet of Things Industries and Services Operation department at IBM Research - China. She received her Ph.D. degree in computer science from the Peking University in 2003. She subsequently joined IBM at IBM Research - China, where she has worked on software development governance, project portfolio optimization, and service operation analytics. She is author or coauthor of 29 patents and 15 technical papers.