Predicting the Remaining Useful Life of Plasma Equipment through XCSR

Liang-Yu Chen
Institute of Computer Science and Engineering, College of Computer Science, National Chiao Tung University
1001 University Road, Hsinchu 300
Taiwan R.O.C.
reno.iie02g@nctu.edu.tw

Jia-Hua Lee
Institute of Computer Science and Engineering, College of Computer Science, National Chiao Tung University
1001 University Road, Hsinchu 300
Taiwan R.O.C.
jessie1992a@gmail.com

Ya-Liang Yang
Department of Computer Science, College of Computer Science, National Chiao Tung University
1001 University Road, Hsinchu 300
Taiwan R.O.C.
alan.yaco@msa.hinet.net

Ming-Tsung Yeh
Smart Manufacturing Division, Architecture 4, United Microelectronics Corporation
No. 18, Nanke 2nd Rd., Xinshi Dist., Tainan City 744
Taiwan R.O.C.
madami@affiliation.org

Tzu-Chien Hsiao*
Department of Computer Science, College of Computer Science; Institute of Biomedical Engineering, College of Electrical and Computer Engineering, National Chiao Tung University
1001 University Road, Hsinchu 300
Taiwan R.O.C.
labview@cs.nctu.edu.tw

ABSTRACT

Predicting remaining useful life (RUL) of plasma equipment becomes an important issue for semiconductor manufacturing in this decade. If RUL can be accurately estimated, the schedule of maintenance can be proper to moderate the waste and cost of the production. Digital Radio Frequency Matching Box (RF-MB) is an essential equipment in the semiconductor manufacturing process. The status of RF-MB will be recorded by the Fault Detection and Classification (FDC). In order to establish the RUL of RF-MB, we use Fisher Discriminant Analysis (FDA) for feature selection to concentrate the leading variables in FDC. We marked the first 2 days of the RF-MB operation as "Good" and marked the last 2 days before the failure of RF-MB as "Bad". We used eXtended Classifier System with continuous-valued inputs (XCSR) to learn the well-labeled FDC data. The results show that XCSR can quickly find patterns and meaningful variables. The average accuracy of XCSR is 97.3% and the average missing rate of rules is only about 1.6%. The results confirmed that XCSR is capable of alerting related operator before the plasma component reaching its residual life. In the future, we will use XCS with Function approximation (XCSF) to more accurately approximate the function of RUL. We look forward to building a complete assessment of RUL.

CCS CONCEPTS

• **Computing methodologies → Machine learning → Machine learning approaches →** Rule learning

KEYWORDS

Fisher Discriminant Analysis (FDA), eXtended Classifier System (XCS), Digital Radio Frequency Matching Box (RF-MB), Remaining Useful Life (RUL)

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326879

1 INTRODUCTION

Industry 4.0 is a high-tech project proposed by the German government using computerized, digitalized, and intelligent ways to improve the production and quality of manufacturing. Semiconductor industry is an important indicator for the development of the four industrial evaluation. Many semiconductor manufacturers transform their manufacturing processes into fully automated intelligent controls. One of important issues for semiconductor manufacturing process is pre-diagnosis of plasma equipment failure. In the past, semiconductor manufacturer can only restore the plasma equipment after an anomaly occurred. Such process control brings two main cost losses. One is to scrap the semi-product when anomaly occurred, and the other is to suspend the production line during repair. If the remaining useful life (RUL) of plasma equipment could be accounted in advance, we can schedule the maintenance procedure to avoid manufacturing products without knowing the abnormality of the plasma equipment. Arranging a maintenance plan in advance can help us shorten the preparation time and reduce the scrap rate of products in manufacturing process. Therefore, it is necessary to use an intelligent way to predict the RUL of the plasma equipment to accurately control the production process, which is the most urgent issue for semiconductor manufacturing process.

Digital Radio Frequency Matching Box (RF-MB) machine is an essential equipment in the semiconductor manufacturing process. When operating, the Fault Detection and Classification (FDC) continuously record the statue of RC-MB. We want to figure out the appropriate algorithm to evaluate the RUL of RF-MB though FDC data. Since the RUL of the plasma equipment will change slightly with the environment, such as temperature and humidity, it is necessary to apply an algorithm that is suitable for adapting to dynamical changes. In the past, we have successfully applied Zeroth-level Classifier System with continuous-valued inputs (ZCSR) for targeting the optimal impedance in RF-MB [1]. Here, we evaluate an algorithm called eXtended Classifier System(XCS) for predicting the RUL of RF-MB. XCS is a rule-based machine learning method. XCS recalibrate its rule set by interacting with the environment and modifying its rule set by adopting Reinforcement Learning (RL) and Genetic Algorithm (GA) [2]. XCS has a special mechanism called "Covering". When the rule set of XCS does not match the condition of the current environment, the "Covering" mechanism will activate and generate a new rule that matches environmental condition. The "Covering" mechanism allows XCS to instantly recalibrate its rule set (model) for unseen data without re-training and re-testing the entire rule set. Therefore, XCS is suitable for online learning. In addition to applying online learning, XCS is also suitable for dealing with complex problems of both epistasis and heterogeneity [3]. Epistasis refers to the interaction between variables that affects the importance of the other variables. XCS proves its ability to deal with epistasis and generate mappings for multiplexer problems. The FDC data in RF-MB contains a total of 73 variables. It is difficult to check the dependence between variables one by one. We expect to find the dependencies between variables from the rules learned from XCS. Heterogeneity means that the same effect caused by the same action may contain different pattern. In the issues of predicting the RUL, the same predicting residual life time of the plasma equipment may be accompanied by different failure patterns. Another advantage of XCS is that there is a cooperative and competitive relationship between its rules. XCS can be matched by many rules with different conditions to the same prediction result. Therefore, XCS is suitable for patterning such heterogeneous problem domains. We expect that the rules learned by XCS can indicate the possible causes of the failure.

When applying XCS to predict the RUL of plasma equipment, there are 2 important issues that need to be solved. The first issue is that the original XCS is only suitable for classification problems. This means that the original XCS is incompletely suitable for RUL modeling. To simplify the problem, we label the FDC data as good state and impending failure. XCS can be modeled with labeled data, alerting the relevant people to arrange maintenance before the machine fails. The other issue is that conventional XCS is only able to process binary input data. However, actual data is usually recorded in real numbers. In order to handle this problem, Prof. Wilson proposed the interval predicate as the expression of rule condition to extend the possibility of dealing with the real-word problem. The modified version is called XCS with continuous-valued inputs (XCSR) [4].

This study mainly uses XCSR to predict the RUL of RF-MB equipment. Section 2 introduces XCS and describes the difference between XCS and XCSR. The experimental setup and parameters setup of XCSR are explained. We also provide a method of feature selection called Fisher Discriminant Analysis (FDA). FDA can find out the leading variable and feature which has better classification ability. The statistics of the obtained data and the workflow of data pre-processing are illustrated in this section. Section 3 shows the experimental results of XCSR. Section 4 looks deeper and gives future work on this topic. Finally, a summary of this study was given in section 5.

2 MATERIAL AND METHODS

We selected the FDC data in a RF-MB machine from normal to abnormal as training material. The dates of FDC data range from February 21 to August 14, 2017. We used the first 2 days of FDC data which are February 21 and February 22 and labelled these as "Good". In addition, we selected the last 2 days of data which are August 13 and August 14 and labelled these as "Bad". The size of this FDC dataset is 207,629 for this 4-day period. However, the RF-MB machine has many different recipes (tasks) that need to be executed, as shown in Table 1. Different recipes have different process conditions. We could find that *Recipe1* has the most data in RF-MB machine, so we chose *Recipe1* as an online interactive environment for XCSR. To verify the ability of generalization for XCSR, we also use *Recipe1* for 10-fold cross-validation to verify the performance of XCSR. After that, we used *Recipe1* as training data and *Recipe2* as testing data, because the procedure for Recipe1 and Recipe2 are similar. This experiment mainly tests whether XCSR has the ability of generalization between similar recipes in the same RF-MB machine.

FDC data contains 73 variables in this machine. There are many steps in each recipe that must be executed and many variables are recorded. Considering that more days of data may be used in the future as training data to improve accuracy, we must concentrate the number of features to reduce the amount of data. Prof. Jay Lee mentioned in [7] that the average, standard deviation, maximum, and minimum values are features that can effectively distinguish categories. Therefore, we calculate these values for each variable as input features in each step. This also causes the number of features to change from 73 to 292, but the volume of data can be reduced from 148,541 to 3,260. To accelerate XCS learning, we used FDA to help us identify the leading features of variables. Table 2 list the top 50 features by FDA.

Table 1: Amount of each recipe data in RF-MB machine

Rank	Recipe	Count (*)
1	*Recipe1*	148541
2	*Recipe2*	33640
3	*Recipe3*	7036
4	...	...

Table 2: Top 50 features selected by FDA

Rank	Feature	Value	Rank	Feature	Value
1	~~*Var40* Step13, Mean~~	355250	26	*Var30* Step6, Max	139.6
2	*Var25* Step10, Max	12323.5	27	*Var29* Step10,Max	135.6
3	*Var27* Step17, Max	4571.4	28	~~*Var40* Step8, Mean~~	123.5
4	*Var25* Step10, Mean	4203.2	29	*Var30* Step17, Max	119.3
5	*Var27* Step21, Max	3156.1	30	*Var27* Step19, Max	112.5
6	*Var26* Step10, Mean	2420.2	31	*Var29* Step2, Min	92.2
7	*Var27* Step10, Max	1848.5	32	*Var26* Step8, Max	91.6
8	*Var27* Step10, Mean	1261.5	33	*Var30* Step19, Max	89.8
9	*Var26* Step10, Max	1181.5	34	*Var26* Step21, Mean	84.4
10	*Var27* Step17, Mean	723	35	*Var26* Step10, SD	76.7
11	*Var29* Step6, Mean	563.9	36	*Var27* Step13, Mean	75.7
12	*Var26* Step8, Mean	517	37	*Var29* Step19, Min	73.2
13	*Var26* Step19, Max	452.6	38	*Var26* Step4, Mean	71.3
14	*Var29* Step21, Mean	373.5	39	*Var30* Step13, Max	70.8
15	*Var26* Step19, Mean	347.1	40	*Var29* Step16, Min	69.9
16	*Var29* Step17, Mean	319.2	41	*Var59* Step10, Mean	69.5
17	*Var29* Step17, Max	284.9	42	*Var29* Step8, Mean	68.5
18	*Var27* Step21, Mean	283.8	43	*Var29* Step7, Max	66.3
19	*Var26* Step21, Max	269.6	44	*Var26* Step13, Mean	64.3
20	*Var29* Step19, Mean	250	45	*Var26* Step8, SD	63.2
21	*Var29* Step6, Min	246.1	46	*Var29* Step7, Mean	62.6
22	*Var29* Step17, Min	151	47	*Var29* Step6, Max	61.8
23	*Var26* Step13, Max	148.9	48	*Var30* Step15, Max	61.1
24	~~*Var40* Step13, Min~~	146	49	*Var29* Step2, Mean	59.5
25	*Var27* Step13, Max	143	50	*Var29* Step15, Min	55.3

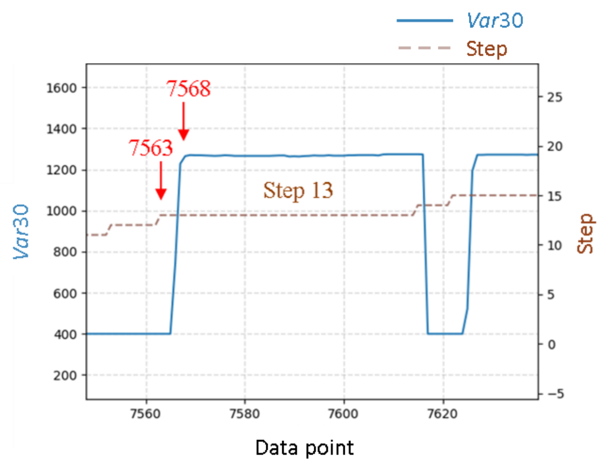

Figure 1: The descriptive statistics of Var30 for step 13 of Recipe 1.

Table 3: Raw data for each step

Step	Raw data	Size	Stable after *n* points
1	[400, 400, 400, 400, 400, 400, 400, 400]	8	0
2	[400, 400, 400, 400, 890, 1052, 1054, 1056, 1056]	9	6
3	[1061, 1043, 1043, 400, 400, 400, 400, 400]	8	4
4	[400, 400, 400, 468, 863, 904, 894, 877, ...]	22	7
5	[842, 1093, 684, 400, 400, 400, 400, 400, 400]	9	4
6	[400, 400, 400, 400, 940, 1208, 1208, 1208, ...]	43	6
7	[1218, 1216, 1216, 400, 400, 400, 400, 400]	8	4
8	[400, 400, 400, 662, 1182, 1195, ...]	18	5
9	[1144, 1144, 1144, 400, 400, 400, 400]	7	4
10	[400, 400, 400, 707, 1295, 1312, 1323, 1330, ...]	94	7
11	[400, 400, 400, 643, 1252, 1295, 1310, 1322, 1331, 1332, ...]	10	8
12	[400, 400, 400, ...]	10	0
13	[400, 400, 400, 400, 1088, 1227, 1238, 1238, ...]	53	7
14	[1249, 1248, 981, 400, 400, ...]	6	4
15	[400, 400, 400, 676, 1171, 1237, 1248, 1248, ...]	34	7
16	[1246, 1245, 1245, 400, 400, 400, ...]	8	4
17	[400, 400, 400, 400, 1190, 1206, 1212, 1214, ...]	24	7
18	[1222, 1221, 971, 400, 400, 400, ...]	8	4
19	[400, 400, 400, 400, 917, 965, 969, 969, ...]	24	6
20	[973, 972, 731, 400, 400]	5	3
21	[400, 400, 400, ...]	18	0
22	[400, 400, 400, ...]	31	0

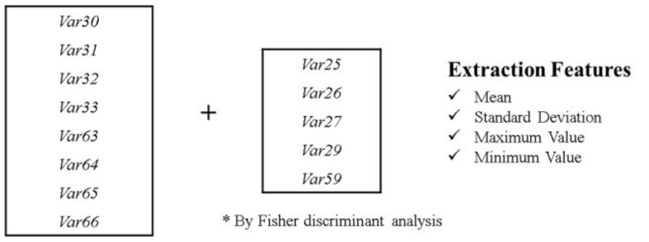

Figure 2: A total of 13 variables and 52 features were selected. The first 8 variables were selected by experts in the manufacturing production line. The last 5 variables were defined by FDA.

We find that *Var*25, *Var*26, *Var*27, *Var*29, *Var*30, and *Var*59 are leading variables, which are indicated in grey in Table 2. It can be seen that *Var*40 is also pointed out as an important feature by FDA, but the experts of semiconductor manufacturing have evaluated that *Var*40 has no significant impact on the entire process. Therefore, *Var*40 is ignored for further XCSR processing.

We observed the descriptive statistics of the feature selected by FDA and found that the machine had not reached its stable state before the start of each step. As shown in Fig. 1, step 13 begins at 7,563 data points and reaches a stable state at 7,568 data points. We listed all the data points of each step and observed that each step will reach the stable state at a maximum of 8 data points after the start of each step and will be out of the stable state at most a maximum of 2 data points before the end of each step. Table 3 lists the raw data of each step, the size of the raw data, and the number of data points needed to reach the stable state after the start of each step. Therefore, according to Table 3, we remove the first 8 data points and the last 2 data points of each step to calculate the value of the feature.

Finally, we selected *Var*30, *Var*31, *Var*32, *Var*33, *Var*63, *Var*64, *Var*65, and *Var*66 based on the experience of experts, and we selected *Var*25, *Var*26, *Var*27, *Var*29, and *Var*59 based on FDA. A total of 13 variables were selected as input variables based on the experience of experts and FDA as shown in Fig. 2. In the following, we briefly explained the FDA, the difference between XCS and XCSR, and the parameters used by XCSR.

2.1 Fisher discriminant analysis (FDA)

The purpose of FDA is to confirm whether there are statistically significant differences between the average scores of a set of variables for two or more predefined groups. The FDA is also a supervised dimensionality reduction method for two categories of samples. On the classification problem, FDA can be used to sort the ability of each feature to distinguish between two categories. The formula for FDA is shown in (1), where i and j are the class labels; $u_{i,fk}$ and $u_{j,fk}$ are the means of the k^{th} feature, and $(\sigma_{i,fk})^2$ and $(\sigma_{j,fk})^2$ are the variances of the k^{th} feature. A higher FDA score means that a feature has a better ability to distinguish between two categories.

$$J_{f_k} = \frac{\left\|\mu_{i,\,f_k} - \mu_{i,\,j_k}\right\|^2}{\sigma^2_{i,f_k} + \sigma^2_{i,j_k}} \tag{1}$$

In the Artificial Intelligence for Cyber-Enabled Industrial Systems project hosted by Prof. Jay Lee [5, 6], the collection of multiple variables in Industry 4.0 is considered as big data. When dealing with the issue of big data, we must consider the amount of data that can be processed by the computer and the complexity between the variables. To avoid the curse of dimensionality, the dimension reduction process will be performed before dealing with multiple variables. Prof. Jay Lee also applied FDA as a feature selection method for the analysis of semiconductor manufacturing processes [7].

2.2 eXtended Classifier System (XCS)

XCS is a rule-based machine learning method suitable for online learning systems. The core of XCS is composed of rules, and the interpretation of each rule is "IF condition, THEN action". The rules learned by XCS are very simple interpretations for humans. A collection of multiple rules in XCS is called a population set [P]. XCS finds the rules that satisfy the environmental conditions from [P] and executes the action of these rules to interact with the environment. The environment gives feedback or payoff based on the action of XCS. XCS evaluates the quality of the rules based on the payoff and determines whether the rules can be retained in [P]. XCS uses the ternary alphabet {0, 1, #} as encoding for the condition of the rule, where the # symbol stands for "Don't care". Regardless of whether the input value is 0 or 1, the environmental condition corresponding to the bit of # is considered to satisfy the condition. To evaluate the rules, XCS uses 3 parameters to calculate the quality of the rules. Rules contain evaluation parameters called classifiers. The three parameters contained in the classifier are prediction, prediction error, and fitness. Prediction is used to predict how many payoffs can be obtained from the environment after the action of the classifier is executed. Prediction error is the error between the predicted value of the classifier and the actual payoff obtained from the environment. Fitness is used to assess whether a classifier is suitable for retention in [P]. Fitness is defined as a function of prediction error. A classifier with smaller prediction error is easier to retain in [P]. After interacting with the environment, XCS uses the reinforcement learning method to update these 3 parameters according to the payoff given by the environment. In addition to adding 3 parameters to evaluate the quality of the rule, GA is applied to help XCS produce new rules. XCS evolves through iterations to find the most accurate and the most general rules.

The performance cycle of XCS is divided into 6 stages. (i) Matching operation. XCS converts the detected environmental inputs into a binary string and creates an empty set called match set [M]. XCS scans [P] to find the classifiers that satisfy the classifier conditions for the given environmental inputs and put these classifiers into [M]. If [M] is still empty after scanning [P], XCS will activate the "Covering" mechanism. The covering mechanism will generate a new classifier whose condition must satisfy the environmental inputs. Each bit in the condition of the newly generated classifier will be covered by # with a certain probability. The action and 3 parameters of the newly generated classifier are given randomly. Therefore, re-scanning [P] will inevitably find a classifier that meets the environmental inputs and put it into [M]. (ii) Generating prediction array [PA]. Each classifier with the same action in [M] will weigh its prediction with fitness. Finally, the weighted values of the classifiers with the same action are averaged as the prediction of the action. [PA] can be generated after all the predictions for the action are calculated. (iii) Action selection. After [PA] is generated, XCS will decide which action to execute based on the prediction value of each action in [PA]. There are two mechanisms for action selection: exploration and exploitation. In exploration, XCS will randomly select an action based on the prediction of each action in [PA] as the probability of selection, or randomly select an action to execute. In exploitation, XCS will choose the largest predicted value of the action in [PA] as the executed action. In the process of interaction with the environment, XCS will alternately perform exploration and exploitation. The classifiers in [M] that match the chosen action will be selected and placed in action set [A]. (iv) RL. The update of the parameters mainly occurs in [A]. When the XCS performs an action through the effector, the environment provides the payoff. XCS will use Q-learning to update the parameters of the classifiers in [A] based on the payoff. (v) GA. If the threshold of GA is satisfied, GA will activate. GA acts on [A] to generate new classifiers for adapting to environmental changes or finding better solutions. GA mainly contains two important mechanisms: Crossover and Mutation. The Crossover uses a two-point Crossover or uniform Crossover to exchange the information of two classifiers for generating better classifiers. The Mutation is to scan each bit of the classifier's condition. The scanned bit will change the value or cover by # with a certain probability. (vi) Updating operation. XCS will scan the classifiers in [P] to find out which classifier's parameters need to be updated. If a new classifier generated by GA does not exist in [P], XCS will check the size of [P]. If the size of [P] does not reach the upper limit, XCS will insert the new rule directly into [P]. If the number of rules exceeds the size of [P], XCS will delete classifiers based on fitness. XCS will repeat the above six stages to find the most correct and most general classifiers.

XCS also has two major mechanisms, the macroclassifier and the subsumption mechanism, which help XCS learn the general classifiers. The concept of a macroclassifier is to speed up the matching operation of XCS. XCS may generate the same classifier in [P]. Therefore, a new parameter "num" is added to the classifier. When XCS generates a new classifier, XCS will scan [P] to find if there is a macroclassifier whose condition and action are the same as the newly generated classifier. If a macroclassifier exists, the num of the macroclassifer will be incremented by 1; otherwise, this new classifier is inserted in [P] and its num is set to 1. When a macroclassifier is selected to be deleted, its num is decremented by 1 if its num is greater than 1; otherwise, the macroclassifier is removed from [P]. Subsumption mainly occurs in [A] and GA; it can also be called *ActionSetSubsumption* and *GASubSumption* according to the location of activation. The concept of subsumption is to help XCS find accurate and general classifiers.

2.3 XCS with continuous-valued inputs (XCSR)

To process continuous-valued inputs, Pro. Wilson proposed XCSR. This method changes the original expression of a classifier's condition from ternary alphabet to interval predicate. Interval predicates are expressed as int_i = (c_i, s_i), where c_i represents the middle value of the interval and s_i represents the half range centered on c_i. For instance, the environmental inputs are x_1 and x_2, and both x_1 and x_2 are real numbers. A classifier's condition is expressed as [(0.3, 0.1), (0.7, 0.1)]. We can interpret this classifier's condition as "if ($0.2 \leq x_1 \leq 0.4$) and ($0.6 \leq x_2 \leq 0.8$) are true, then the classifier matches the environmental inputs". The performance cycle of XCSR is the same as that of XCS except for two mechanisms, which are Covering and GA.

Covering will activate when [M] is still empty after matching operation. XCSR will generate a new rule whose condition is expressed as int_i = (c_i, s_i), where c_i is equal to the environment input x_i; s_i is randomly taken from 0 to s_r; s_r is defined by the user and $s_r \geq 0$. The action of the classifier is given randomly. The difference between GA in XCSR and XCS are in the operation of Crossover and Mutation. XCSR uses Uniform Crossover. XCSR scans the classifier's condition for each interval predicate, and exchanges the interval predicates of the two rules with probability x. In Mutation, the center and spread values in each interval predicate are randomly added and subtracted from random values between 0 and m, where m is defined by user. We call this modified version of XCS, XCSR.

2.4 Parameters setup of the XCSR

The parameters setup of the XCSR mainly refer to [8], and the description of the setting is as follows: Population size n = 5000; learning rate β = 0.2; decline rate of fitness α = 0.1; the threshold of prediction error ε_0 = 10; fitness exponent υ = 5; the threshold of GA acts on [A] θ_{GA} = 25; the probability of Crossover χ = 0.8; the probability of Mutation μ = 0.04; the deletion threshold θ_{del} = 20; the fraction of the mean in [P] δ = 0.1; the subsumption threshold θ_{sub} = 20; the probability that a bit in the Covering mechanism is covered by the "don't care" symbol $P_{\#}$ = 0.33; initial prediction value p_I = 10; initial prediction value ε_I = 0; initial fitness value F_I = 10; the maximum range of spread s_r = the half range of values for each variable; the range of value that increase or decrease the value of the interval predicate m = s_r; Both *doActionSetSubsumption* and *doGASubsumption* are activated.

3 RESULTS

We calculated the mean, standard deviation, maximum, and minimum values of the data in *Recipe1* as the input features. A total of 3,260 data and 52 features for each data are inputted to XCSR. The output of the XCSR is "Good" (the first 2 days after the machine's operation) and "Bad" (the last 2 days before the machine's failure). The data was repeatedly trained 100 times, and the data will be shuffled before each training time. XCSR repeated 30 experiments for each data training, and all static reports were the average results for 30 runs. Fig. 3 shows the result of XCSR.

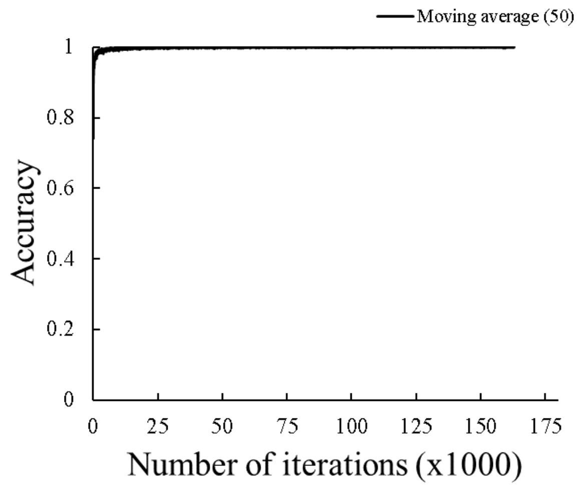

Figure 3: The moving average accuracy result per 50 exploitations for XCSR.

Table 4: 10-fold cross validation for XCSR

No	Accuracy	Number of missing classifiers in [P]	No	Accuracy	Number of missing classifiers in [P]
1	0.981250	7	6	0.974643	8
2	0.962145	9	7	0.965409	8
3	0.972050	4	8	0.987616	3
4	0.975155	4	9	0.972136	3
5	0.984568	2	10	0.959119	7
Average accuracy		**0.973409**	***Missing rate of classifiers in[P]***		**0.016871**

The result is the moving average with the accuracy of every 50 exploitations. We can find that XCSR reaches 100% accuracy after approximately 11,500 interactions. The result confirmed that XCSR can quickly adapt to the environment and find the patterns.

We used 10-fold cross-validation to test the performance and the generalization of XCSR. *Recipe1* has 3,260 data, of which 1,779 are marked as "Good" and 1,481 are marked as "Bad" . We cut the data into 10 folds, about 326 data each fold, and the ratio of "Good" and "Bad" in each fold is close to the ratio of the original data. XCSR will use the first fold as testing data, and the remaining nine folds will be used as training data. Each fold will become the testing data once, and the rest of the folds will become training materials. This process will be repeated 10 times. During the XCSR verification process, XCSR does not activate the mechanisms for Covering, GA, and Updating operations. XCSR only uses exploitation for selecting execution actions, which means that XCSR only uses the classifier that predicts the maximum payoff. If XCSR cannot find a classifier from [P] that matches the verification data, we will record it as a missing classifier and add 1 to the number of missing classifier. Missing classifiers are not included in the calculation of the accuracy rate, because the normal operation of XCSR will activate the Covering mechanism to generate a new classifier. Table 4 shows the results of 10-fold cross-validation. The average accuracy rate of XCSR is 0.973409, and the missing rate of classifiers is only 0.016871. The results confirmed that XCSR has good performance in predicting unseen data, which also confirmed that XCSR has the ability of generalization in dealing with multivariable issues.

Finally, we used the data of *Recipe2* as the input of the testing data, which has a similar process to *Recipe1*. We also calculated the mean, standard deviation, maximum, and minimum values of the data in *Recipe2* as the input features. The amount of input data for *Recipe2* has been reduced from 33,640 to 686. In the input data, 249 data are marked as "Good" and 437 data are marked as "Bad". It can be observed that the ratio of two classes in *Recipe2* is unbalanced. We used the 686 data of *Recipe2* as input data to verify the effectiveness of the [P] learned by XCSR. Table 5 shows the results of 30 verifications. The average accuracy rate of XCSR is 0.97399, and the missing rate of classifiers is only 0.02074. The verification results give us more confidence to confirm that XCSR can handle complex real-world problems of such multiple variables. XCSR can still find patterns to accurately predict RUL, even if there are slight differences between similar recipes.

We go further to see the classifiers learned in [P]. In order to understand what XCSR learns, when the range of values covered by the interval predicate is greater than or equal to the maximum value of the variables and less than or equal to the minimum value of the variables, we will use the "don't care" symbol (#) to express the allele of interval predicate. This means that any input value can match this allele of interval predicate, indicating that the classifier believes that the variable does not affect its output. We analyze a classifier learned by XCSR as an example. Table 6 is the classifier that XCSR learned for step 6 in *Recipe1* of RF-MB machine. There are a total of 52 features in the input data. In this classifier, XCSR only retains 3 features and the rest are covered with #. This classifier predicts that RF-MB machines may be failure after 2 days. We draw descriptive statistics for the "Good" and "Bad" of the mean value of Var29 in step 6 as shown in Fig. 4. Compared with the classifier learned by XCSR, it can be found that "Good" and "Bad" do have significant differences, and XCSR can accurately cut the "Bad" range, from 17.5 to 23. The mean value of Var29 in step 6 also has the same result as shown in Fig. 5. XCSR can accurately cut the "Bad" range, from 19 to 21.6. In the mean value of Var59 in step 6, it is shown that XCSR can accurately cut out the "Bad" range value, from 50.1 to 50.5, as shown in Fig. 6. However, it can be found that the range of "Bad" learned by XCSR overlaps with the range of "Good". We think the possible reason is that the number of 100 iterations is too short, and it is not enough to make XCSR converge to the optimal range. However, XCSR can reduce features from 52 to 3 and accurately concentrate the exact range, which is an exciting result.

Table 5: Verify the accuracy and missing rate of [P] learned by XCSR

No	Accuracy	Number of missing classifiers in [P]	No	Accuracy	Number of missing classifiers in [P]
1	0.946824	9	16	0.979228	12
2	0.988166	10	17	0.988113	13
3	0.961652	8	18	0.986547	17
4	0.985251	8	19	0.953869	14
5	0.965672	16	20	0.984985	20
6	0.973451	8	21	0.968085	28
7	0.967213	15	22	0.986706	9
8	0.942857	21	23	0.964126	17
9	0.983558	17	24	0.982143	14
10	0.986587	15	25	0.986726	8
11	0.953453	20	26	0.96875	14
12	0.979259	11	27	0.976155	15
13	0.991031	17	28	0.962853	13
14	0.968935	10	29	0.976366	9
15	0.987786	31	30	0.973451	8
Average accuracy	**0.97399**		*Missing rate of classifiers in[P]*	**0.02074**	

Table 6: Classifiers learned by XCSR for step 6 in recipe 1 of RF-MB machine

Condition						Action
Step 6						Bad
Var	30	31	32	33	63	
Mean	#	#	#	#	#	
Std	#	#	#	#	#	
Max	#	#	#	#	#	
Min	#	#	#	#	#	
Var	64	65	66	25	26	
Mean	#	#	#	#	#	
Std	#	#	#	#	#	
Max	#	#	#	#	#	
Min	#	#	#	#	#	
Var	27	29	59			
Mean	#	17.5-23.0	50.1-50.5			
Std	#	#	#			
Max	#	19.0-21.6	#			
Min	#	#	#			

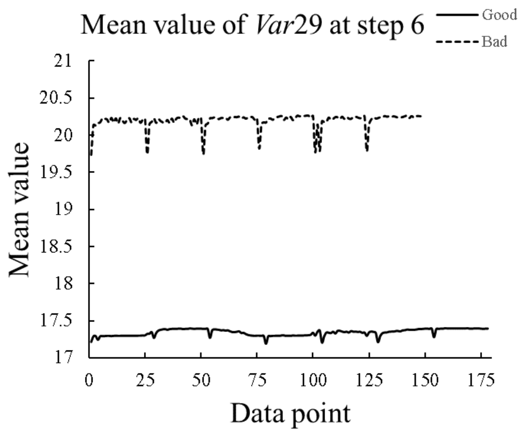

Figure 4: Descriptive statistics for the mean value of *Var*29 in step 6.

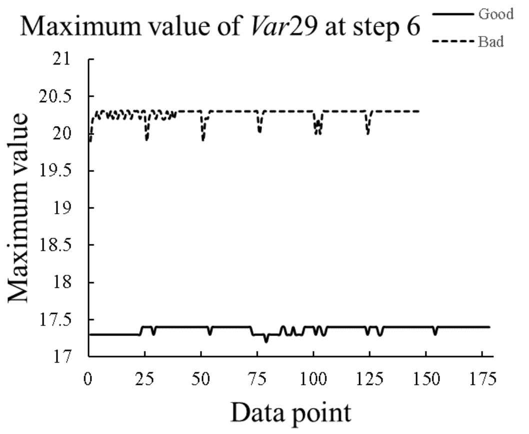

Figure 5: Descriptive statistics for the maximum value of *Var*29 in step 6.

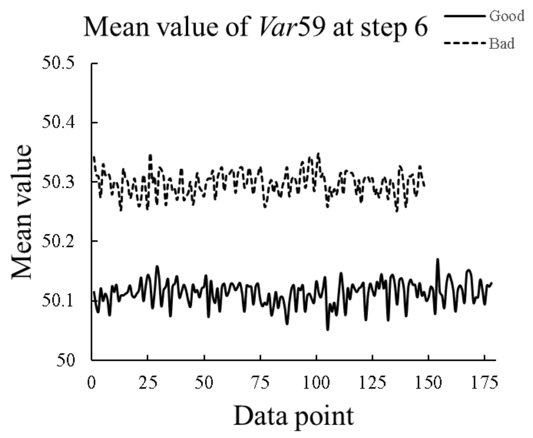

Figure 6: Descriptive statistics for the mean value of *Var*59 in step 6.

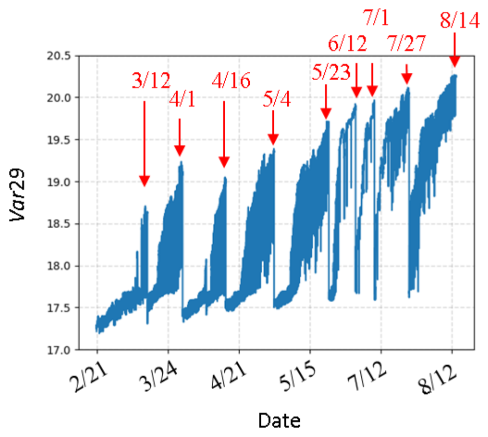

Figure 7: Descriptive statistics for the mean value of *Var*29 in step 6.

4 DISSCUSSIONS

The XCSR learns "Good" and "Bad" labels as RUL indicators for RF-MB machine on the 4-day training materials. However, the trend of data for the rest of days have not yet been taken into account. We have found complete data of Recipe1 in RF-MB machine for about six months, and plotted the statistical description of *Var*29. Complete data means that the RF-MB machine is from a new normal state to a failure. Fig. 7 shows the variation of the mean value of *Var*29 in step 6, where illustrates an occurrence of periodic reset. Periodic reset is about half a month. This is a periodic maintenance action that the operator will perform based on past operation experience. Periodic reset seems to be effective in returning *Var*29 to its normal state. However, we found that the overall change in *Var*29 presents a trend of increasing values when the time is close to the "Bad" that we have labeled. It can be observed that it is impossible to return to the situation on March 12 after maintenance in June 12. This represents an interference problem with other anomalies. At this time, we only know the cause of RF-MB machine failure. Other possible reasons need to be further discussed with relevant experts and operators.

XCSR can find patterns and accurately predict state of the RF-MB machine in similar recipes of the same RF-MB. In the future, we hope to verify the performance of XCSR under the same recipe of different RF-MB machines. However, learning and testing in different RF-MB machines may encounter some difficulties. For example, the difference in the baseline of each variable for different RF-MB machines can be very large due to different environmental status. We must find an appropriate normalization method for XCSR to find patterns in the same recipe between different RF-MB machines. We hope to find the pattern on different RF-MB machines in the future.

We successfully used XCSR to exam the features of the normal state and abnormal state of the RF-MB machine. If we want to calculate RUL of plasma equipment more accurately, it is necessary to find an algorithm to approximate the function of RUL. XCS with function approximation (XCSF) is another modified version of XCS that can approximate unknown functions [9, 10]. We think that we can use the remaining time of RF-MB from normal to failure as the payoff given by the environment. We use Var29 and the information of periodic reset as input data expecting XCSF to approximate the RUL function of RF-MB machine. We hope to build a more complete evaluation method of RUL in the future.

5 CONCLUSIONS

We use XCSR to predict the RUL of plasma equipment. We use the *Recipe1* in RF-MB machine as training data. The data is recorded from February 21, 2017 to August 14, 2017, where a component of RF-MB machine is reinstalled and continuously operates until the component is damaged. Due to the large number of variables, we use FDA to help us concentrate important variables. We marked the first 2 days of the RF-MB operation as "Good" and marked the last 2 days before the failure of RF-MB as "Bad". In order to reduce the amount of data, we use the mean, standard deviation, maximum, and minimum of each variable in a single step as input features. XCSR can quickly find patterns from these features. XCSR has the ability to reduce features from 52 to 3. We used 10-fold cross-validation in *Recipe1* for XCSR. The averaged accuracy of XCSR is 97.3% and the averaged missing rate of rules is only about 1.6%. We also used similar process of recipe in the same RF-MB machine as testing data. The averaged accuracy of XCSR is 97.3% and the averaged missing rate of rules is about 2%. The result shows that XCSR performs the ability to handle complexity problems with multiple variables. In the future, we will conduct the XCSF to approximate the optimization function for more accurately predicting the RUL of plasma equipment.

ACKNOWLEDGMENTS

This project is fully supported by United Microelectronics Corporation (UMC).

REFERENCES

[1] L. Y. Chen, Y. L. Yang, and T. C. Hsiao. 2017. ZCSR for targeting the optimal impedance in digital radio frequency matching box. In *Proceedings of ACM GECCO conference, Berlin, Germany* (*GECCO'17*), 269–270.
[2] S. W. Wilson. 1995. Classifier fitness based on accuracy. *Evolutionary Computation.* Vol. 3, No. 2, 149–175.
[3] R. J. Urbanowicz and W. N. Browne. 2017. Introduction to learning classifier systems. Springer.
[4] S. W. Wilson. 1999. Get real! XCS with continuous-valued inputs. In *International Workshop on Learning Classifier Systems.* Springer, 209–219.
[5] J. Lee, B. Bagheri, and H. A. Kao. 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, Vol. 3, 18–23.
[6] J. Lee, H. A. Kao, and S. Yang. 2014. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia CIRP.* Vol. 16, 3–8.
[7] J. Lee, D. Siegel, and E. R. Lapira. 2013. Development of a predictive and preventive maintenance demonstration system for a semiconductor etching tool. *ECS Transactions.* Vol. 52, No. 1, 913–927.
[8] M. V. Butz and S. W. Wilson. 2000. An algorithmic description of XCS. In *International Workshop on Learning Classifier Systems.* Springer. 253–272.
[9] S. W. Wilson. 2001. Function approximation with a classifier system. In *International Workshop on Learning Classifier Systems* (*GECCO'01*). 974–981.
[10] M. V. Butz, P. L. Lanzi, and S. W. Wilson. 2008. Function approximation with XCS: Hyperellipsoidal conditions, recursive least squares, and compaction. *IEEE Transactions on Evolutionary Computation.* Vol. 12, No. 3, 355–376.