Revue d'Intelligence Artificielle
Vol. 37, No. 1, February, 2023, pp. 223-230
Journal homepage: http://iieta.org/journals/ria

Improving Existing CMMS Software Packages Using Association Rules

Samia Beldjoudi[1,2*]

[1] LTSE (Laboratoire de Technologies des Systèmes Energétiques), National Higher School of Technology and Engineering, Annaba 23000, Algeria
[2] Laboratory of Electronic Document Management LabGED, Badji Mokhtar University, Annaba 23000, Algeria

Corresponding Author Email: s.beldjoudi@esti-annaba.dz

https://doi.org/10.18280/ria.370128

Received: 21 January 2023
Accepted: 10 February 2023

Keywords:
RUL, CMMS, prognostics, association rules, data mining, predictive maintenance

ABSTRACT

Since their inception, industries have experienced the negative effects of downtime, lost productivity, lost revenue, as well as layoffs. The prediction of an item's remaining useful life (RUL) enables maintenance techniques to avoid costly and serious damage. As a result, a prognostic is now acknowledged as a crucial activity. Thanks to the Internet of Things (IoT) and IT solutions like Computer Aided Maintenance Management (CMMS) software packages, industries today have a vast amount of data gathered from on-site sensors. This offers real-time data on the equipment's state as well as each piece of equipment's history of interventions from the CMMS software database. By utilizing the vast amount of data that has accumulated over the years, we will be able to extract even more crucial information. The use of artificial intelligence (AI) methods can open up new possibilities for CMMS software packages. In this study, we try to predict RUL (Remaining Useful Time) using an artificial intelligence technique called association rules. This strategy is applied to enhance existing CMMS software programs. Experiment is carried out with a well-known dataset provided by the NASA Ames Research Center and the CoE "Center of Excellence". Experiment results indicate that our suggested approach performs well in forecasting the RUL of turbojet engines and that it also significantly improves the outcomes of predictive maintenance.

1. INTRODUCTION

The industrial maintenance industry is changing as a result of advances in predictivefigure maintenance, the Internet of Industrial Objects, and cutting-edge sophisticated technologies like machine learning (ML) and artificial intelligence (AI).

Real-time monitoring of the equipment's condition, made possible by the sensors it has, is the foundation of predictive maintenance. The information gathered includes, among other things, a number of signs that help predict impending failures, including temperature, vibration, cavitations, and oil analysis. Even while this data is quite helpful on its own, it has tremendous potential when combined with service history, spare parts knowledge, and all the reports accessible. This information can be utilized to produce forecasting algorithms when recorded and examined in a CMMS software program.

Having a modern and flexible CMMS is a sine qua non for setting up a prescriptive maintenance routine. As with any new technology, deploying a prescriptive strategy can be quite a daunting task at first. Computer hardware, software and training costs, the size of the company and its approach to innovative technologies can all constitute obstacles to the prescriptive. But this approach, which is ultimately quite emerging, has enormous potential and promises the entire sector to enter the world of Industry 4.0. In this world, Computer Assisted Maintenance Management is becoming Artificial Intelligence Assisted Maintenance Management in order to allow manufacturers to use AI as they operate a computer in the management of their maintenance (focus on the job and not on the means).

This is the context in which this study is positioned. Our goal is to introduce new functionalities to classic CMMS software packages in order to become software that helps in the management of predictive maintenance. This improvement allows the use of AI in the maintenance business of industrial machines in order to capitalize on knowledge, plan maintenance actions and better manage stocks.

Additionally, businesses today lament the excess data that is routinely retained on various media. Even though these massive databases have the potential to hold a wealth of important information, they are rarely or poorly utilized. In order to meet this need, it is necessary to deploy sizable resources in order to extract crucial information, leading to the development of data mining, which now serves as a crucial decision-making tool in industries that are highly competitive.

Making maintenance decisions is made possible by the ability to forecast Remaining Useful Life (RUL) through a combination of management, supervisory, technical, and related administrative procedures [1].

This goal will be accomplished by utilizing a very potent data mining technique, the aim of which is to find correlations between two or more variables recorded in very big databases that are of interest to a data analyst. By examining several data sources, including machine profiles,

and failure histories, the goal is to create an automated RUL prediction system.

This paper is organized as follows: Section 2 is an overview of the main contributions related to our work. Section 3 is dedicated to the presentation of the objectives of our approach. In section 4, we describe the suggested strategy and go over the findings of the experiment we ran to evaluate its efficacy. Conclusion and future works are described in Section 5.

2. RELATED WORK

Prognostics have received a lot of interest recently from both academic and industrial researchers. Model-based approaches, data-driven approaches and hybrid approaches are the three main categories of prognostic approaches.

Understanding the system physics-of-failure and underlying system degradation models is necessary for the deployment of general model-based prognostic techniques. In order to anticipate real-time RUL in the situation of fatigue crack growth while taking into account the uncertainties in both degradation processes and condition monitoring techniques, Myötyri et al. [2] proposed the usage of a stochastic filtering technique. Later, condition-based component replacement in the context of fatigue crack formation used a similar particle filtering approach [3]. A vehicle suspension system was the subject of a model-based prognostic approach developed by Luo et al. [4] that depends on a precise simulation model for system degradation prediction. For RUL forecasts of rolling element bearings under time-varying operational settings [5] or in the absence of prior degradation information [6], Gebraeel provided a degradation modeling methodology.

Understanding all conceivable physics-of-failures and their interactions for a complicated system is nearly hard since high-risk engineered systems typically consist of several components with multiple failure modes.

The use of data-driven approaches is particularly prevalent in the field of RUL prediction, where RUL is calculated using statistical and probabilistic techniques using historical data and data from the system that is regularly monitored [7].

The availability of multivariate historical data regarding system behavior, which must cover all stages of system operation and degradation scenarios under specified operating parameters, is a prerequisite for setting up the data-driven models for RUL prediction. Fielded applications, testing environments for experiments, and computer simulations are the three methods for obtaining these run-to-failure data [8].

Although physics-based approaches are favoured over data-driven ones because to their accuracy and precision, getting a physics of failure (PoF) degradation model is frequently exceedingly challenging. The data-driven approaches, in comparison, tend to be more user-friendly because they primarily rely on AI tool techniques that can be deployed directly or with few modifications. The RUL prediction field sees a significantly faster spread pace than the physics-based ones due to the rapid implementation and deployment of data-driven prognostics. Another major factor in the proliferation of these types of approaches among prognostics system makers is the lack of a demand for knowledge of the system's physics. Both the advantages and the disadvantages of these methods are obvious. According to Leser, among the many data-driven methodologies, extrapolation and data collection are two of the most frequently mentioned challenges [9]. One of the fields of machine learning known as "deep learning" developed from artificial neural networks and is characterized by several nonlinear processing layers (ANN). Given its capacity to capture the hierarchical relationship hidden in deep structures, DL has emerged as one of the primary study issues in the field of prognostics due to the rapid growth of computational infrastructure [10]. In the area of RUL prediction, it hasn't yet been completely utilized [11]. An end-to-end prognostic paradigm for estimating state-of-health (SOH) and predicting remaining useful life (RUL) was put forth by Li et al. [12]. A hybrid neural network (NN), which combines an active-state-tracking long-short-term memory NN and a one-dimensional convolutional NN, is created in such a framework to capture the hierarchical features between various variables affecting battery degeneration as well as the temporal dependencies contained in those features. A structured-effect neural network for estimating remaining useful life is suggested in Kraus and Feuerriegel [13]. Through the use of variational Bayesian inferences, the parameters are computed. In Hou et al. [14], the authors created a deep supervised learning method that makes use of similarity to boost prediction accuracy. Hou et al. [14] have devised an unsupervised learning strategy based on restricted Boltzmann machine (RBM) to create the health indicator (HI), as the health indicator (HI) generation procedures rely on manual labeling or expert opinion. They demonstrated that, in comparison to other conventional methodologies, their performance offers superior performance.

For the RUL prediction of rolling bearings, Chen et al. [15] created a recurrent neural network (RNN) model employing an encoder-decoder structure with an attention mechanism. They proved that their model can work with less prior knowledge and gives higher performance compared to other algorithms.

A transferable convolutional neural network (TCNN) was proposed by Cheng et al. [16] to train domain invariant characteristics for bearing RUL prediction. The authors demonstrated that their approach avoids the impact of kernel selection and offers improved RUL prediction performance.

For two rotating machinery datasets, Li et al. [17] work showed promising results. The authors of this study computed the distribution of the healthy state data using the generative adversarial network (GAN) technique and developed a health indicator.

In order to predict RUL, Li et al. [18] created a multi-scale deep convolutional neural network (MS-DCNN) and combined the MS-DCNN algorithm with min-max normalization. The authors demonstrated that the new model offers promising results on the NASA C-MAPSS dataset by comparing its performance with that of other cutting-edge models.

Recurrent convolutional neural network (RCNN) for RUL prediction was proposed by Wang et al. [19] who also demonstrated its efficacy using two case studies. The authors demonstrated how successfully the suggested model can forecast the RUL of rolling element bearings and milling cutters. Their model streamlines decision-making and offers a probabilistic result for RUL prediction.

Although data-driven approaches based on deep learning have produced promising results for RUL prediction tasks, these approaches require a significant amount of labeled

datasets for the network to be trained before it can provide a model that is accurate enough. However, it is frequently challenging to gather enough data with run-to-failure information for complicated systems. The training dataset and test dataset must also have identical distributions in order for the current deep learning methods to work, which means that the dataset must originate from the same feature space. However, because of the shifting environment in which equipment operates, changes in data distribution are pervasive during the actual application process, which lowers the RUL prediction accuracy. In other words, the performance on the test dataset may be subpar, and the RUL prediction model obtained through the training dataset may not have high generalization capacity.

By combining the knowledge from the two methodologies, hybrid approaches try to benefit from both the strengths of model-based and data-driven approaches. A relatively small number of studies focus on hybrid approaches, while the majority of studies focus on data-driven and physics-based approaches. Only 8% of the research in this area was focused on hybrid prognostics approaches as of 2017 [20]. The disadvantage of hybrid approaches is that they also have the drawbacks of both approaches and add to the complexity of finding a solution.

This work is a data-driven approach where aims to use a very potent data mining technique to find correlations between two or more variables recorded in very big databases. By examining several data sources, including machine profiles and failure histories, the goal is to create an automated RUL prediction system.

3. PRINCIPLES AND OBJECTIVES

Through-life engineering services are the technical services needed to guarantee that a complex engineering system operates as required, as predictably as possible for the duration of its anticipated operational life, and at the overall lowest cost [21].

This is fuelled by rapid decision-making process maintenance, repair, and overhaul [21] with the goal of returning assets to a state where they regularly satisfy design requirements. The capacity to forecast Remaining Useful Life (RUL) makes these maintenance decisions possible through a mix of management, supervisory, technical, and corresponding administrative operations [1].

Numerous predictors, or variable characteristics that could potentially have an impact on future behavior or outcomes, make up a predictive model [22]. Predictive modeling therefore includes any classifier that may be used to assign a certain class (such as spam) to a test object.

An emerging method called data mining seeks to identify important patterns or intriguing rules from transaction databases. Many domains, including user behavior analysis, network intrusion detection, event categorization and regression, etc., have successfully used it in recent years. A common data mining technique is association rule, which is used to uncover possibly significant connections between data itemsets. It can be divided into two smaller problems: finding all frequent item sets that satisfy the minimal support threshold, and creating all association rules that satisfy the minimum confidence level.

We can determine whether our data contains any relationships through association rule mining. Think about a simple illustration where one observes that bread and butter are frequently purchased together. When information about the other is available, this could be used to forecast sales of either bread or butter.

Prediction is a fascinating use case for association mining in context temporal databases. To forecast the consequence of a rule, one must use the antecedent of the rule. However, not every association rule may be appropriate for prediction.

In this study, we explore the characteristics of prediction rules and create a method for finding association rules that are helpful for RUL's prediction.

4. APPROACH DESCRIPTION AND EXPERIMENTAL RESULTS

We conducted an experiment utilizing a well-known dataset provided by the NASA Ames Research Center and the CoE "Center of Excellence" to estimate the RUL using the association rules method. With the help of this dataset, which simulates turbojet degradation, we can derive association rules that pinpoint a turbojet's status during the previous 30 cycles. As a result, we will be able to estimate whether the turbojet will hit its limit after a maximum of 30 cycles or not. This will make it possible for the maintenance crew to do the necessary repairs to further the turbojet engine's lifespan.

The dataset in use is an example of a C-MAPSS simulation of the deterioration of a turbojet engine. The operation of four separate sets of turbojets was simulated under various configurations of operational circumstances. In order to characterize the progression of the faults, the deterioration is recorded using multiple sensor channels. The dataset is supplied as a zip file that contains four training files, each with a unique set of operating settings, and a test file that is subjected to the same set of conditions. Only the first file, "train FD001.txt", will be worked on here.

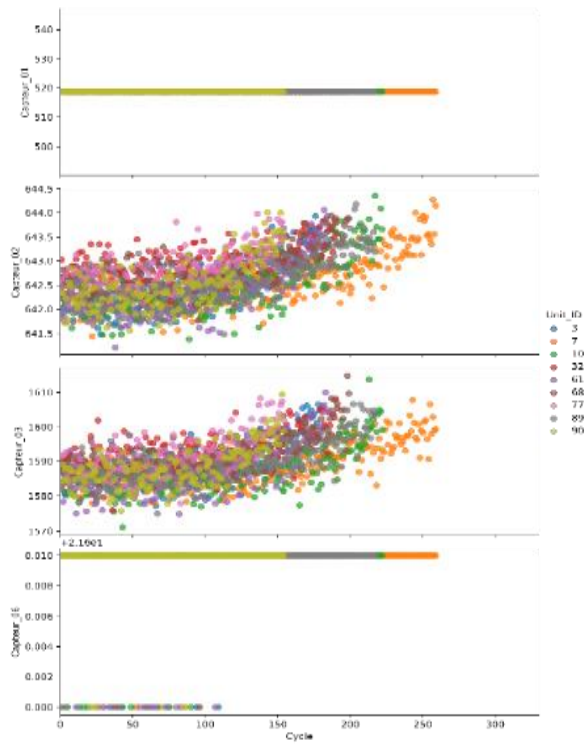

Figure 1. Visualization of the sensors 01, 02, 03, and 06's variation over time

A data table of 20631 rows (observations) and 27 columns makes up the file. These observations are based on 100 turbojets, and each observation comprises of a turbojet's life cycle. Columns include:

(1) Unit_ID: identifying the number of the turbojet;

(2) Cycle: the number of the cycle;

(3) PO_1, PO_2, PO_3: Three operational parameters;

(4) Sensor_01, Sensor_02…. Sensor_21: 21 sensors which each measure a specific characteristic during system operation.

In this experiment, we used 15 turbojets as the test table and 85 turbojets as a training table.

For the first ten turbojets, a plot is created to observe the variation of a few variables with respect to the cycle. This will enable us to visualize our data more effectively. Figure 1 shows that some of these variables (Sensor_01) remain the same during the course of their lifetimes. The "Sensor_06" is another option, and its values only differ by two with regard to turbojets: of the 20631 values, 20225 are 21.61 and 406 are 21.60. Nevertheless, for the sake of this study, it can also be regarded as constant because, 98.03% of the time, it only takes on a constant value.

In actuality, there are other variables in the table that remain constant as well. Complete visualization reveals that sensors 1, 05, 10, 16, 18, and 19 are all constants in addition to the operational parameter 03; this gives us eight variables that can be disregarded because they are constant and won't affect the results of our study. Considering them would be a waste of time and memory.

4.1 RUL extraction

The RUL is the number of operational cycles left before a failure. Therefore, in order to locate it, we must first determine each turbojet's maximum cycle life, or "Max Cycle" as presented in Table 1. Then, calculating RUL will be far too easy; all we have to do is take the number of cycles that have already passed from "Max Cycle", and the output will be the RUL.

We must first read our table and then determine the RUL for each observation. For this purpose we take the highest cycle number for each unit and call it "Max Cycle".

Then, we subtract each cycle number in each observation of the “Max_Cycle” from each “Unit_ID”. The result presented in Table 2 is the RUL we were looking for.

Table 1. The maximum number of cycles reached by each turbojet

Unit_ID	1	2	3	4	5	6	7	8	9	10
Max_Cycle	192	287	179	189	269	188	259	150	201	222

Table 2. A sample data table displaying the RUL

	Unit_ID	**Cycle**	**PO_1**	**PO_2**	**Sensor _02**	**Sensor _03**	**Sensor_04**	**RUL**
0	1	1	-0,0007	-0,0004	641,82	1589,7	1400,6	191
1	1	2	0,0019	-0,0003	642,15	1591,82	1403,14	190
2	1	3	-0,0043	0,0003	642,35	1587,99	1404,2	189
3	1	4	0,0007	0	642,35	1582,79	1401,87	188
4	1	5	-0,0019	-0,0002	642,37	1582,85	1406,22	187
5	1	6	-0,0043	-0,0001	642,1	1584,47	1398,37	186
6	1	7	0,001	0,0001	642,48	1592,32	1397,77	185
7	1	8	-0,0034	0,0003	642,56	1582,96	1400,97	184
8	1	9	0,0008	0,0001	642,12	1590,98	1394,8	183
9	1	10	-0,0033	0,0001	641,71	1591,24	1400,46	182
10	1	11	0,0018	-0,0003	642,28	1581,75	1400,64	181

Table 3. The data table was encoded using one-hot.

	Unit_ID	**Cycle**	**RUL**	**50%<PO_01<60%**	**60%<PO_01<70%**	**70%<PO_01<80%**
0	1	1	191	0	0	1
1	1	2	190	0	0	0
2	1	3	189	1	0	0
3	1	4	188	0	0	0
4	1	5	187	0	0	1
5	1	6	186	1	0	0
6	1	7	185	0	0	0
7	1	8	184	0	1	0
8	1	9	183	0	0	0
9	1	10	182	0	1	0
10	1	11	181	0	0	0

4.2 Setting up the transaction table

A table must first be converted into a binary transaction table in order to extract the association rules from it. However, in this case, our values are numerical rather than category. Therefore, we must convert them into categorical values. We must first construct categories for each variable before we can do this. Ten categories will be established, as follows:

1. Each column is subject to the formula (1), where x_i stands for a column's values.

$$f(x_i) = \frac{x_i - \min_i(x_i)}{\max_i(x_i) - \min_i(x_i)} \quad (1)$$

2. When we're finished, all of the columns' values fall within the range [0: 1]. The values between 0.1*(i-1) and 0.1*i are now replaced with the corresponding values, where i is an integer between 1 and 10 inclusive. We will then have ten categories, each of which will represent 10% of the range [min: max] inclusive.

4.3 One-hot encoding

The one hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions. This encoding is a crucial part of feature engineering for machine learning.

Table 4. A sneak peek at the training table

Sensor_21(%)								Target
[20, 30]	[30, 40]	[40, 50]	[50, 60]	[60, 70]	[70, 80]	[80, 90]	[90, 100]	
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	1
0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	0	1

Thus, we must write our transaction table in binary by using one-hot encoding. To do this there is a simple example: if a variable x takes a value between 90% and 100% that implies it is greater than 70% and 80% at the same time, so we assign a value of 1 to each of them. The same procedure is followed for the three little values, i.e., we assign a value of 1 to each of the other two categories if the value is between 0% and 10%, which denotes that it is smaller than 30% and 20%. As a result, Table 3 provides a summary of the final table.

Since all of these turbojets are identical and belong to the same model and we have already retrieved the RUL for each turbojet and each observation, we can now remove the "Unit ID" and "Cycle" columns.

We still have a problem, even after doing all of this. Only the values "0" and "1" can be used in the "Apriori" association rules extraction technique. However, since there are 362 different RUL values here and they are all integers, if we treat them as categories, we will run into two issues:

1. With 361 new columns, our table will be much bigger and include 7,467,791 more values, which will use up too much RAM.

2. For each turbojet, the 362 new columns will only accept a value a maximum of once. In other words, for all turbojets, the value 1 will only appear 100 times at most in the same column among 20,631 observations, making it impossible for us to draw any conclusions about them. They cannot be regarded as categories, though.

The fact that association rules take into account the likelihood of two or more elements occurring simultaneously in a transaction is another issue. However, they are unable to foretell what the following sighting may bring. To address this, we suggest adding a new column called "Target" and inserting a limit "L" that specifies how many cycles will pass before the failure we're trying to forecast. Additionally, this column will receive the value 0 if the RUL is greater than L and the value 1 if it is lower than L. This indicates that if this column is given the value 1 in an observation, it follows that no more than L cycles are possible before failure.

Take L=30 Cycle and apply this to our table. After that, since the "RUL" column is no longer useful to us, we can delete it.

The association rules that describe the status of a turbojet in its final 30 cycles before failure are now ready to be extracted from our table.

4.4 Extracting association rules

Our objective is to produce association rules with support and confidence values greater than "min support" and "min confidence", respectively. As a result, we'll proceed in two steps: Identify the itemsets "E" that are most common and confirm the condition: supp (E)≥min_support, and from “E”, produce association rules to confirm the following: conf (R_i)≥min_confidence.

We'll utilize the "apriori" function from the Python 3.9 "mlxtend" module to accomplish this. The "prior" function accepts a DataFrame with columns that only contain the values 0 or 1 (false or true), together with the value of the minimal support, as input, and returns a DataFrame with two columns that contains the frequent item sets and their corresponding supports.

Using the "apriori" function of "frequent patterns", which takes the training table "Table 4" as input, we have determined the frequent itemsets after applying this function. Then, using mlxtend's "association rules" function, we extracted the rules that have a confidence level higher than min confidence.

Finding all the rules that adhere to specific minimal support and confidence requirements is the goal of association rules mining. The extracted rules become more obvious when the support value is increased, which makes them less useful to the user. As a result, in order to retrieve crucial information, the support value must be set low enough. Only an estimate of the rules' correctness in the future is provided by the confidence. It stands for the trust in the laws that we seek. Finding the appropriate support and confidence settings and obtaining the best rules that have an impact on the rate of F-measure require a certain level of competence. Two experiments are conducted to determine the ideal minimum support and minimum confidence values. In the first experiment, we use our dataset to find the ideal minimum support value. To determine the value for which our approach performs best, we consider several minimum support values between 0 and 1. As a consequence, 0.029 is the most appropriate minimum support number that results in the greatest F-measure metric value.

The second experiment uses the experimental dataset with

minimum support set to 0.029 in order to find the ideal value of minimum confidence. In this experiment, various minimal confidence values between 0 and 1 are employed. As a result, 0.9 is the ideal minimal confidence number that offers the optimum performance. The appropriate support and confidence settings in the experiments that follow are 0.029 and 0.9, respectively.

Each turbojet contains 30 true values; hence the training table contains 30x85=2550 true values. To go back to relative support, we divide it by 17,340, which is the number of observations for the first 85 turbojets, and we get a minimum support of 0.029.

This gives back an array of the most common itemsets with support higher than 0.029. This results in an excessively large table with an excessive number of useless item sets since the support is too tiny. Finding all the important itemsets is the first step in tidying up this mess (Table 5 presents a preview on frequent itemsets). Then, we simply delete all the itemsets that are not a part of or are equal to any of the itemsets that contain our target after finding all the itemsets that contain our target in them as presented in Table 6.

Table 5. Preview on frequent itemsets

	Support	Itemsets
0	0,980342	'70%<Sensor_06<80%'
1	0,980342	'80%<Sensor_06<90%'
2	0,980342	'90%<Sensor_06<100%'
3	0,895178	'20%<Sensor_09<30%'
4	0,866614	'20%<Sensor_14<30%'
5	0,980342	'80%<Sensor_06<90%', '70%<Sensor_06<80%'
6	0,980342	'90%<Sensor_06<100%', '70%<Sensor_06<80%'
7	0,875717	'20%<Sensor_09<30%', '70%<Sensor_06<80%'
8	0,847615	'20%<Sensor_14<30%', '70%<Sensor_06<80%'
9	0,980342	'80%<Sensor_06<90%', '90%<Sensor_06<100%'

Table 6. Insight into frequent itemsets related to "Target"

	Support	Itemsets
0	0,147059	'Target'
1	*0,042849*	'Target', *'40%<PO_01<50%'*
2	0,042042	'Target', '50%<PO_01<60%'
3	0,037255	'Target', '20%<PO_02<30%'
4	0,036217	'Target', '70%<PO_02<80%'
5	0,03489	'50%<Sensor_02<60%', 'Target'
6	0,049539	'Target', '60%<Sensor_02<70%'
7	0,051096	'Target', '70%<Sensor_02<80%'
8	0,04902	'Target', '50%<Sensor_03<60%'
9	0,048731	'Target', '60%<Sensor_03<70%'
10	0,052307	'Target', '60%<Sensor_04<70%'

Table 7. All association rules extracted with confidence>0.9

	Antecedents	Consequents	Support	Confidence	Lift
R0	['70%<Sensor_11<80%','70%<Sensor_15<80%']	'Target'	0,032987	0,996516	6,776307
R1	['70%<Sensor_11<80%','20%<Sensor_7<30%']	'Target'	0,029758	0,996139	6,773745
R2	['70%<Sensor_4<80%', '20%<Sensor_7<30%']	'Target'	0,032065	0,992857	6,751429
R3	['20%<Sensor_12<30%', '20%<Sensor_7<30%']	'Target'	0,029815	0,992322	6,747793
R4	['70%<Sensor_11<80%','20%<Sensor_12<30%']	'Target'	0,033852	0,991554	6,742568
R5	['20%<Sensor_12<30%' '70%<Sensor_15<80%']	'Target'	0,030392	0,988743	6,723452
R6	['70%<Sensor_11<80%', '70%<Sensor_4<80%']	'Target'	0,039273	0,986957	6,711304
R7	['70%<Sensor_11<80%' '20%<Sensor_21<30%']	'Target'	0,03391	0,986577	6,708725
R8	['20%<Sensor_12<30%','70%<Sensor_4<80%']	'Target'	0,037197	0,986239	6,706422
R9	['70%<Sensor_11<80%','20%<Sensor_20<30%']	'Target'	0,03564	0,985646	6,702392
R10	['70%<Sensor_4<80%', '70%<Sensor_2<80%']	'Target'	0,030854	0,983456	6,6875
R11	['70%<Sensor_15<80%', '70%<Sensor_4<80%']	'Target'	0,03564	0,982512	6,681081
R12	['20%<Sensor_20<30%','70%<Sensor_15<80%']	'Target'	0,033045	0,979487	6,660513
R13	['70%<Sensor_4<80%', '20%<Sensor_21<30%']	'Target'	0,038062	0,979228	6,658754
R14	['20%<Sensor_12<30%','20%<Sensor_21<30%']	'Target'	0,031084	0,978221	6,651906
R15	['70%<Sensor_15<80%','20%<Sensor_21<30%']	'Target'	0,030392	0,975926	6,636296
R16	['20%<Sensor_12<30%','20%<Sensor_20<30%']	'Target'	0,031949	0,973638	6,620738
R17	['20%<Sensor_20<30%', '70%<Sensor_4<80%']	'Target'	0,037889	0,970458	6,599114
R18	['70%<Sensor_11<80%']	'Target'	0,055709	0,966	6,5688
R19	['20%<Sensor_20<30%','20%<Sensor_21<30%']	'Target'	0,033737	0,948136	6,447326
R20	['20%<Sensor_7<30%']	'Target'	0,046021	0,945498	6,429384
R21	['20%<Sensor_12<30%']	'Target'	0,053403	0,938197	6,379737
R22	['70%<Sensor_4<80%']	'Target'	0,066032	0,919679	6,253815

We will use the "association rules" function on this new table of itemsets. The updated frequent itemset table and the minimum confidence value, which is 0.9, are the inputs for this function. These resulting rules will undoubtedly have a wide range of effects, but we are only concerned with those that will have our "Target" as a result.

All the rules from our training table that were extracted are in Table 7.

There are 23 association rules total, each having a confidence better than 0.9 and a lift between 6.25 and 6.77. Each row in this table represents an association rule. For instance, the first rule states that we are in the final 30 cycles 99.65% of the time if the readings of sensors 11 and 15 are higher than 70%.

4.5 Testing and assessing the rules

We must utilize the "test" table in order to verify these rules. We will take the "antecedents" of each rule and search for the observations in the "test" table where the antecedents for each rule are true.

Since these rules are only designed to detect situations in which the target is equal to 1, we will naturally concentrate our evaluation just on the predictions of the "Target=1" class. Precision, recall, and F-measure were calculated on the test data to evaluate our rules. When we apply these functions to the outcomes of the test of each rule separately, we obtain the outcomes shown in Table 8.

Table 8 displays the precision of the forecast we made utilizing all of the previously extracted association rules. To determine if there were predicted false values or actual values, this table is only calculated for those instances where a value of 1 was predicted.

Table 8. Each rule's precision, recall, and F-score

Rule	Precision	Recall	F-measure
R0	1	0,222989	0,182331
R1	0,988095	0,190805	0,159923
R2	0,980198	0,227586	0,184701
R3	1	0,216092	0,177694
R4	0,991667	0,273563	0,214414
R5	0,989796	0,222989	0,181989
R6	1	0,312644	0,238179
R7	0,989796	0,222989	0,181989
R8	0,977444	0,298851	0,228873
R9	0,97479	0,266667	0,209386
R10	0,989691	0,22069	0,180451
R11	0,975806	0,278161	0,216458
R12	1	0,222989	0,182331
R13	0,974138	0,25977	0,205082
R14	0,990291	0,234483	0,189591
R15	1	0,2	0,166667
R16	0,964912	0,252874	0,200364
R17	0,977444	0,298851	0,228873
R18	0,969231	0,434483	0,3
R19	0,961165	0,227586	0,184015
R20	0,911765	0,285057	0,217163
R21	0,946809	0,409195	0,285714
R22	0,898374	0,508046	0,324523

We identified rules that enabled us to predict 435 values with a minimum confidence level of 0.9, or 90%, of which 384 were correctly predicted and 51 were incorrectly forecasted. However, we are aware that there are a total of 450 values, or 30x15 true values. We can easily observe the outcomes of our predictions and the areas where our rules became confused during the prediction by utilizing the "confusion- matrix" function of the "metrics" module in the "sklearn" library. Figure 2 shows the resulting confusion matrix.

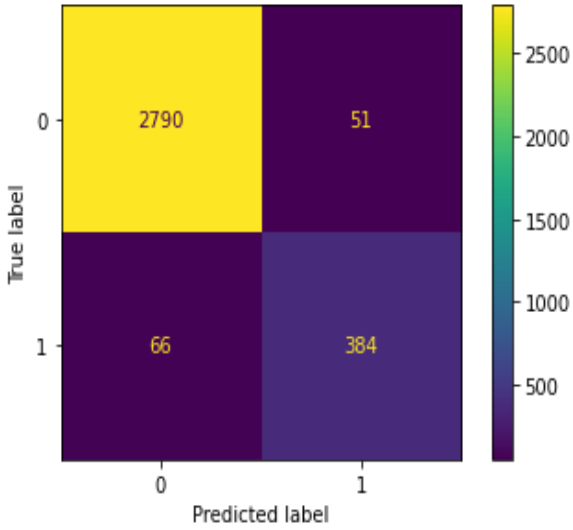

Figure 2. Confusion matrix of all the rules

We can thus calculate the precision of the recall and the F-score of our predictions:

Starting with class 0 (RUL≥30): Precision0=0.9768 and Recall0=0.9820

Now let's move on to class 1 (RUL<30): Precision1=0.8827, Recall1=0.8533

Now let's calculate the F-score: F=0.9237

And if we focus on class 1 only: the F-score would be: F_1=0.8677

This means that our model has an accuracy of 88.27%, a recall of 85.33% and an F-score of 0.86 in the prediction of class 1 (RUL<30), which can be considered a good result.

This means that now we can predict that there will be a maximum of 30 life cycles left for the turbojet engine with an accuracy of 88.27% if we use all these rules.

5. CONCLUSIONS

The purpose of this study was to use one of the AI technologies to optimize the CMMS software packages. As a result, we chose association rules to estimate how many turbojet life cycles are left.

We were able to take advantage of the data gathered by the sensors thanks to the Internet of Things and extrapolate more information from it by using association rules, one of many AI tools. In this study, we focused on the most recent 30 cycles in order to be able to roughly predict when the last cycle will be.

This prediction will give us the advantage of being able to make the necessary repairs before the equipment malfunctions, prolonging its life in particular so that we can get the most out of these services before purchasing another. We have tested our approach on a baseline dataset where we have obtained good results.

Other AI methods, such as neural networks and deep learning, can also forecast how long an item will last.

We plan to use one of these tools to attempt to forecast the RUL, then we'll compare the outcomes to see which strategy

performed the best.

ACKNOWLEDGMENT

The author would like to express his very great appreciation to M. Teffahi Rabah Master degree in the higher school of industrial technologies of Annaba, for his valuable performing of the algorithm.

REFERENCES

[1] Xiongzi, C., Jinsong, Y., Diyin, T., Yingxun, W. (2011). Remaining useful life prognostic estimation for aircraft subsystems or components: A review. In IEEE 2011 10th International Conference on Electronic Measurement & Instruments, IEEE, 2: 94-98. https://doi.org/10.1109/ICEMI.2011.6037773

[2] Myötyri, E., Pulkkinen, U., Simola, K. (2006). Application of stochastic filtering for lifetime prediction. Reliability Engineering & System Safety, 91(2): 200-208. https://doi.org/10.1016/j.ress.2005.01.002

[3] Cadini, F., Zio, E., Avram, D. (2009). Model-based Monte Carlo state estimation for condition-based component replacement. Reliability Engineering & System Safety, 94(3): 752-758. https://doi.org/10.1016/j.ress.2008.08.003

[4] Luo, J., Pattipati, K.R., Qiao, L., Chigusa, S. (2008). Model-based prognostic techniques applied to a suspension system. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 38(5): 1156-1168. https://doi.org/10.1109/TSMCA.2008.2001055

[5] Gebraeel, N., Pan, J. (2008). Prognostic degradation models for computing and updating residual life distributions in a time-varying environment. IEEE Transactions on Reliability, 57(4): 539-550. https://doi.org/10.1109/TR.2008.928245

[6] Gebraeel, N., Elwany, A., Pan, J. (2009). Residual life predictions in the absence of prior degradation knowledge. IEEE Transactions on Reliability, 58(1): 106-117. https://doi.org/10.1109/TR.2008.2011659

[7] Kabir, A., Bailey, C., Lu, H., Stoyanov, S. (2012, May). A review of data-driven prognostics in power electronics. In 2012 35th International Spring Seminar on Electronics Technology, IEEE, pp. 189-192. https://doi.org/10.1109/ISSE.2012.6273136

[8] Elattar, H.M., Elminir, H.K., Riad, A.M. (2016). Prognostics: a literature review. Complex & Intelligent Systems, 2(2): 125-154. https://doi.org/10.1007/s40747-016-0019-3

[9] Leser, P.E. (2017). Probabilistic prognostics and health management for fatigue-critical components using high-fidelity models. North Carolina State University. https://ui.adsabs.harvard.edu/abs/2017PhDT.....176L/abstract, accessed on Dec. 12, 2022.

[10] Ma, M., Sun, C., Chen, X. (2017). Discriminative deep belief networks with ant colony optimization for health status assessment of machine. IEEE Transactions on Instrumentation and Measurement, 66(12): 3115-3125. https://doi.org/10.1109/TIM.2017.2735661

[11] Sateesh Babu, G., Zhao, P., Li, X.L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. Database Systems for Advanced Applications, pp. 214-228.

[12] Li, P., Zhang, Z., Grosu, R., Deng, Z., Hou, J., Rong, Y., Wu, R. (2022). An end-to-end neural network framework for state-of-health estimation and remaining useful life prediction of electric vehicle lithium batteries. Renewable and Sustainable Energy Reviews, 156: 111843. https://doi.org/10.1016/j.rser.2021.111843

[13] Kraus, M., Feuerriegel, S. (2019). Forecasting remaining useful life: Interpretable deep learning approach via variational Bayesian inferences. Decision Support Systems, 125: 113100. https://doi.org/10.1016/j.dss.2019.113100

[14] Hou, M., Pi, D., Li, B. (2020). Similarity-based deep learning approach for remaining useful life prediction. Measurement, 159: 107788. https://doi.org/10.1016/j.measurement.2020.107788

[15] Chen, Y., Peng, G., Zhu, Z., Li, S. (2020). A novel deep learning method based on attention mechanism for bearing remaining useful life prediction. Applied Soft Computing, 86: 105919. https://doi.org/10.1016/j.asoc.2019.105919

[16] Cheng, H., Kong, X., Chen, G., Wang, Q., Wang, R. (2021). Transferable convolutional neural network based remaining useful life prediction of bearing under multiple failure behaviors. Measurement, 168: 108286. https://doi.org/10.1016/j.measurement.2020.108286

[17] Li, X., Zhang, W., Ma, H., Luo, Z., Li, X. (2020). Data alignments in machinery remaining useful life prediction using deep adversarial neural networks. Knowledge-Based Systems, 197: 105843. https://doi.org/10.1016/j.knosys.2020.105843

[18] Li, H., Zhao, W., Zhang, Y., Zio, E. (2020). Remaining useful life prediction using multi-scale deep convolutional neural network. Applied Soft Computing, 89: 106113. https://doi.org/10.1016/j.asoc.2020.106113

[19] Wang, B., Lei, Y., Yan, T., Li, N., Guo, L. (2020). Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery. Neurocomputing, 379: 117-129. https://doi.org/10.1016/j.neucom.2019.10.064

[20] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. Mechanical systems and signal processing, 104: 799-834. https://doi.org/10.1016/j.ymssp.2017.11.016

[21] Roy, R., Shaw, A., Erkoyuncu, J.A., Redding, L. (2013). Through-life engineering services. Measurement and Control, 46(6): 172-175. https://doi.org/10.1177/0020294013492283

[22] Geisser S. (1993). Predictive Inference: An Introduction. Chapman & Hall.

NOMENCLATURE

RUL	Remaining Useful Life
CMMS	Computer Aided Maintenance Management
IoT	Internet of Things
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning