Methods to identify time series abnormalities and predicting issues caused by component failures

Crina Narcisa Deac[1], *Gicu Calin* Deac[1] , *Florina* Chiscop[1*], and *Cicerone Laurentiu* Popa[1]

[1]University Politehnica of Bucharest, Robots and Production System Department, 313 Splaiul Indepedentei, Bucharest, Romania

Abstract. Anomaly detection is a crucial analysis topic in the field of Industry 4.0 data mining as well as knowing what is the probability that a specific machine to go down due to a failure of a component in the next time interval. In this article, we used time series data collected from machines, from both classes - time series data which leads up to the failures of machines as well as data from healthy operational periods of the machine. We used telemetry data, error logs from still operational components, maintenance records comprising historical breakdowns and replacement component to build and compare several different models. The validation of the proposed methods was made by comparing the actual failures in the test data with the predicted component failures over the test data.

1 Introduction

By using Predictive Maintenance, through a continuous monitoring of the asset, one can reduce operational risk of mission critical equipment, increase the rate of return on machines by predicting failures before they occur.

In this way, one can control the cost of the maintenance by enabling just-in-time maintenance interventions, with limited maintenance resources in a more cost-effective way. These solutions estimate the remaining lifetime of the asset, so they can recommend timely maintenance activities and enable just in time inventory by estimating orders for parts replacement. It optimises the Preventive Maintenance process which uses scheduled checks on machines from time-to-time, by enhancing quality and supply chain processes.

In order to achieve that, we need a record of the operational history of the equipment which contains both good and bad outcomes, the entire set of action taken to mitigate bad outcomes, errors logs, maintenance logs, repair and replace records and other information data about machines, relevant and sufficient enough to support the use case. Based on these historical data, predictive models learn patterns and predict future outcomes with certain probability.

* Corresponding author: florinachiscop@gmail.com

2 Our research

First we have studied the data, including a preliminary visualization and we have used some cloud datasets which contain information about 500 machines running over the course of a single year. Machine dataset, containing ID, years of operation and machine model, is completed by telemetry dataset, which contains parameters values for voltage, rotation, pressure and vibration, collected in real-time from the sensors, averaged over every hour. Other two useful datasets which complements our data sources to the model are maintenance history, comprising all scheduled inspection of machines and breakdowns and errors logs during operation.

After pre-visualising and understanding the data, the next step is feature engineering, which require to unify datasets, into a unique dataset of parameters which will represent features that describe ideally machine's health condition (machine behaviour) at any given hour. We are calculating lagging features. We are using a 6-hour window for the lag features and compute the rolling aggregate measures.

For telemetric data we compute mean and standard deviation on 6 and 12 hours, for errors data, because error IDs are categorical values, we count the number of errors for each type of error (error1, ..., error5) contained into a lagging window of 12 hours.

We also extract features from maintenance records, here possible features being the duration time since last replacement of a component, because it correlates better with component failures since the degradation becomes stronger if the component is more used. We have also the static features like model number of the machines and the age, they are treated as categorical variables for modeling and will be added to the final feature matrix.

Then the last step in feature engineering is the labelling of target variable, in our case this is the failure of our components (we have 4 types: component 1, 2, 3 or 4 for each ID machine). We created a categorical failure feature to serve as the label, because the model needs both, examples of failures with time series of observations leading up to failures and also examples of periods of healthy operation to be able to make the difference between the two states, using failures data from our historical dataset.

Once we do the labelling, the model must be able to give some advanced warning of an impending failure, it requires to modify the label definition from a failure event which occurs at a specific moment in time to a longer interval of time, a window of failure (future horizon period X). This interval is chosen by the business problem and the data at hand, in consultation with the domain expert [1]. We take an appropriate time window of 2 days prior to the component failure and label the feature as "failure expected" if it falls in this window, and label the rest of the records as normal, failure = "none".

This failure window will be equivalent to 8 time points in our case, because we are sampling data every 6 hours (6 x 8=48 hours), considering this time enough for the arrival of replacement parts and prevent the failure from happening. The prediction problem becomes estimating the probability of failure within this interval.

The next step is to prepare the training, test data and finally experimenting with different supervised machine learning models to make the predictions [3]. We have chosen Random Forest Classifier, Naïve Bayes and Gradient Boosting Classifier.

2.1 Finding features useful for our model. Data pre-processing

A set of 500 machines were monitored over a single year 2018. The dataset contains: Machine ID, Model No and Age in service. We tried to find which are the most important features which influences the patterns. Machine age could be an important static feature for analysis, the failures and errors type could depend on the number of age of operation. Maintenance dataset contains records of scheduled and unscheduled maintenance that may

arise from regular inspections or from mechanical failure (the case of unscheduled maintenance). It contains the components which were replaced for every machine during this year and the datetime of the replacement, a total of 16298 records. Errors Dataset contains errors that are not considered failures, errors logs while the machine is still operational, rounded to the closest hour, but which can predict a future failure event. Failures dataset contains machineID, component model that was broken on a certain machineID and the datetime of the failure. It correlates with the component replacement from Maintenance Dataset.

Maintenance, Errors and Failures datasets can supply important features for the model.

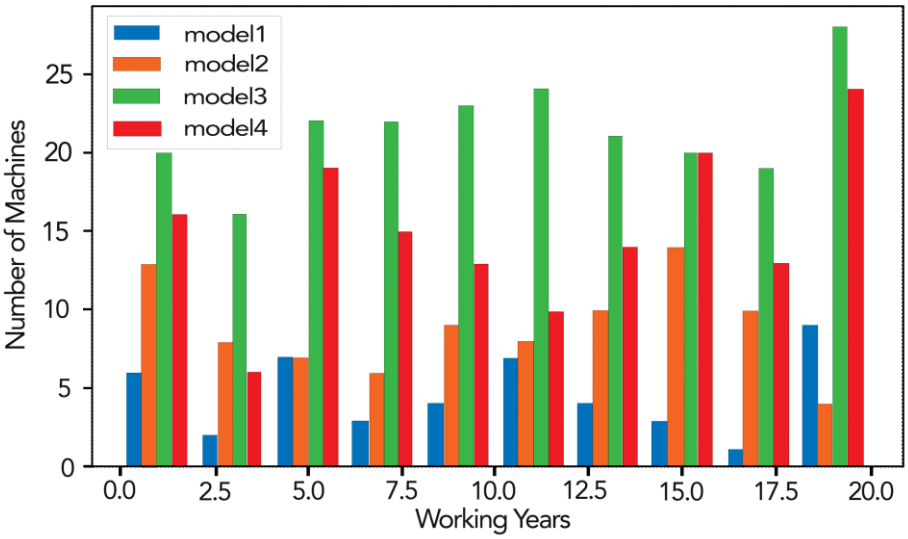

Fig.1 Graphic representation of Machine Dataset Histogram

Table 1. Telemetric Dataset Sample

	machineID	model	age
0	1	model2	17
1	2	model4	8
2	3	model3	10
3	4	model3	6
4	5	model2	2
5	6	model3	7
6	7	model4	20
7	8	model3	16
8	9	model1	7
9	10	model1	10
10	11	model4	6

Also, we have a large dataset containing Telemetric data, like Vibration, Pressure, Rotation and Voltage, measured by sensors in real time, during the same year 2018, averaged over an hour and finally stored in the telemetry history. We have a total of 8761 records for each machineID for an entire year collected by the remote server. All these telemetric data could influence the pattern of failure if their values become very distant from the mean. Vibration and rotation are indicators for the current asset condition and help us to calculate the life span of the component. Pressure and voltage must be kept at the indicated interval to prevent breakdown of the system.

Table 2. Telemetric data

	datetime	MachineID	volt	rotate	pressure	vibration
560704	2018-01-01 06:00:00	65	195.109442	451.726932	109.863744	44.309446
560705	2018-01-01 07:00:00	65	185.582806	558.981237	105.536069	34.986709
560706	2018-01-01 08:00:00	65	179.466761	481.638555	111.687588	41.651598
560707	2018-01-01 09:00:00	65	169.283632	466.060408	100.039113	41.288990
560708	2018-01-01 10:00:00	65	172.813428	459.477959	98.706602	34.867905
560709	2018-01-01 11:00:00	65	198.103990	405.470735	100.625217	38.396680
560710	2018-01-01 12:00:00	65	190.200668	491.837296	99.133892	43.703810
560711	2018-01-01 13:00:00	65	156.319395	381.780107	99.186956	34.599537

Next, we've organised a unified dataset such that the last column is the target (dependent variable) and for each training instance we've assigned a label as the value of this column. Because we are working with temporal data, like data series, we need to divide the duration of sensor data into time units such that each record to belong to a time unit for a machine ID and to offer a distinct information. Time units are defined function the business, in multiple of minutes, hours, days, months and does not to be the same as the frequency of the collection. If the frequency is high, there will be not important differences from one unit to the other.

We'll use a time-point of 6 hours.

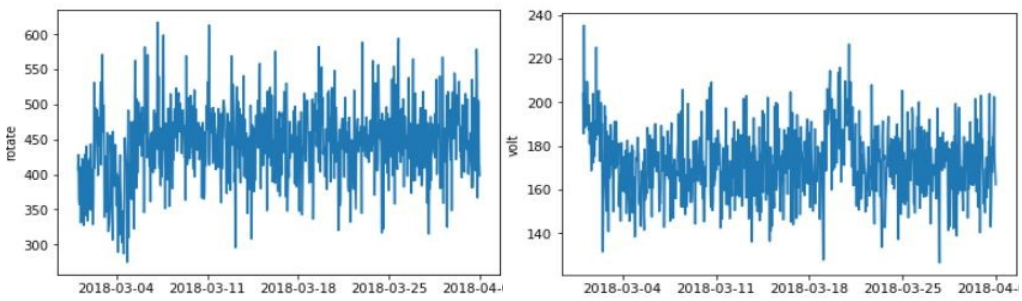

Fig.2 Run charts for Rotation and Voltage during a month for a machine ID

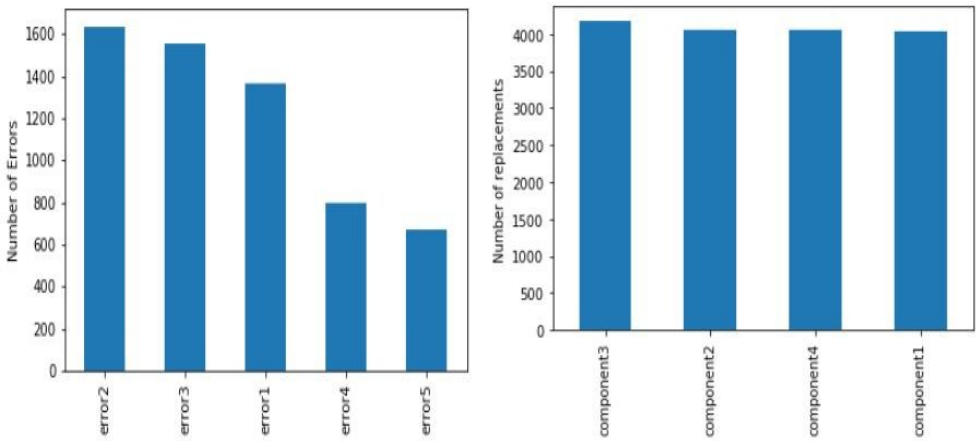

Fig.3 Histogram of components replacements with all four component types over the entire maintenance history

Table 3. Failures dataset sample

	datetime	machineID	failure
222	2018-01-02 06:00:00	32	component4
224	2018-01-18 07:00:00	32	component1
225	2018-02-02 08:00:00	32	component3
226	2018-03-04 09:00:00	32	component2
227	2018-04-03 10:00:00	32	component3
228	2018-05-03 11:00:00	32	component2
229	2018-06-02 12:00:00	32	component2
230	2018-07-17 13:00:00	32	component1
231	2018-08-16 09:00:00	32	component3
232	2018-08-31 10:00:00	32	component1
233	2018-09-30 11:00:00	32	component4
234	2018-10-15 12:00:00	32	component1
235	2018-11-14 13:00:00	32	component2
236	2018-11-14 13:00:00	32	component3

2.2 Feature engineering

In the feature engineering phase, after analysing data, we combined different dataset into a single dataset of features, which describe the machine's life cycle over a period, containing a time unit interval. Each record in the final dataset correspond to a unique time unit within each machine and will contain features and labels to be fed into the machine-learning algorithm.

2.2.1 Telemetry features

Telemetry data has 8761 hourly observations for every asset, so a total of 8761 x 500 = 4.380.500 records, being the largest dataset that we have. We enhance the performance of the model by aggregating average measures on a tumbling 6 hour window: we align the data on 6 hours observations and compute a rolling mean of the sensors measures and a rolling standard deviation, over the last 6, respectively 12 hour lags, for each telemetric data

like volt, rotation, pressure, vibration. Then we joined all aggregate measures (telemetry mean 6 hour data, telemetry standard deviation 6 hour data, telemetry mean 12 hour, telemetry standard deviation 12 hour) and replace raw data with the tumbling window data, picking up results every 6 hours. As a result, we reduce the numbers of records from 4.380.500 to 730.000 records, reducing also the computation time for the entire process.

We had obtained the following results:

Table 4. Features table phase I

	mID	datetime	voltmean_6h	rotatemean_6h	pressuremean_6h	vibrationmean_6h
1	1	2018-01-01 18:00:00	167.834206	474.626613	102.031913	39.791396
2	1	2018-01-02 00:00:00	172.961699	464.132387	96.886832	43.660143
3	1	2018-01-02 06:00:00	161.772937	439.937279	101.042766	45.066460
4	1	2018-01-02 12:00:00	161.865310	443.446884	102.280621	40.484160

	mID	datetime	voltsd_6h	rotatesd_6h	pressuresd_6h
1	1	2018-01-01 18:00:00	18.670712	62.057849	11.331620
2	1	2018-01-02 00:00:00	16.427561	70.300710	7.306678
3	1	2018-01-02 06:00:00	13.704264	23.056306	7.468164
4	1	2018-01-02 12:00:00	6.923907	23.966996	12.734399

2.2.2 Errors features

Because errors IDs are categorical values, we can't average them over time intervals like telemetry measures and then we'll count the number of error of each type for every machine within the lag window. From Errors dataset we form a matrix having like columns machine ID, datetime and errors types. Each Error Type column contains the total number of errors by every asset, summed by hour. Then we make the sum of the Error Type columns on a rolling 12 hour window and align the resulted error counts data by tumbling over 6 hour window using a join with telemetry data

2.2.3 Maintenance features

To obtain maintenance features expressed by the number of days since last replacement, we are using a matrix indicator obtained from maintenance dataset, containing datetime, machineID and component 1, …,4 replacements count for each machine at every hour. We unify indicator to telemetry datetimes, adding time points when no component was replaced. Then we replace the component replacement count, where exists, with the corresponding datetime of the replacement and fill all null fields with the most recent date of component change. Finally we compute the number of days since last replacement.

2.2.4 Machine features

Machine dataset contains the age and the model information, which is string type, therefor we create a set of dummy variables to specify the model. Finally we join to the Feature

table (Table 7), the Maintenance matrix (Table 9), the Error matrix (Table 8) and at the last, Machine table, doing a left join on datetime and machine ID key, in order to obtain the Features dataset.

Table 5. Maintenance matrix unified on telemetry datetime column

	datetime	mid	component1	component2	component3	component4
1	2018-01-01 06:00:00	1	109.000000	109.000000	49.000000	184.000000
2	2018-01-01 06:00:00	1	109.041667	109.041667	49.041667	184.041667
3	2018-01-01 06:00:00	1	109.083333	109.083333	49.083333	184.083333
4	2018-01-01 06:00:00	1	109.125000	109.125000	49.125000	184.125000
5	2018-01-01 06:00:00	1	109.166667	109.166667	49.166667	184.166667
6	2018-01-01 06:00:00	1	109.250000	109.250000	49.250000	184.250000

2.2.5 Label engineering

We have to label our Features dataset with operational condition of machine: healthy versus failed. After we completed the classification, by joining Failure Dataset to Features dataset, we obtained the label "failure" which is "None" or filled with the component type (1.2,3,4) that failed. We need to receive notification for the upcoming failure in a machine to prevent the breakdown. For this we must to change the failure event to a longer window which indicates failure duration, which is set and dependent by the business case. We set the range of time to 8 time units (8 x 6 hours = 48 hours =2 days) to obtain a failure warning window of 2 days. Then we label all observations within this failure warning window "as failed" and estimate the probability of failure within this window.

3 Model Building

3.1 Preparing Training and Testing data

In PdM problems, when working with time series data, it is recommended to split the examples into training, validation and test data sets in a time-dependent manner, such as validation examples are later in time than all training examples and all test examples should be later in time then all the training and validation examples. After the split, we train the algorithm over the training dataset and measure the model's performance over the validation and test set. Typically, 80% of data are used for training and 10%+10% to calibrate and evaluate the model. We choose the splitting point, 2018-09-30 12:00:00. We train the model with examples up to that point and validate the model using examples after this point.

3.2 Classification Models and Evaluation

We have used and compared three classification models: Random Forest Classifier, Gradient Boosting Classifier and Naïve Bayes Classifier. We will use evaluation metrics for computing the Precision level and the Accuracy:

Accuracy is a measure of how correctly we predicted the labelled data:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

Recall measures how well the model can find the positive samples:

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

Precision determines how well the model classifies truly positives samples:

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

TP = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negative

F1 score is the harmonic average of Precision and Recall:

F1 score = 2*((Precision*Recall) / (Precision + Recall))

The chosen algorithms will try to fill the "predicted_failure" column. Result will be compared with the actual label, "failure":

Table 6. Predicted failure

error 4	error 5	Comp. 1	Comp. 2	Comp. 3	Comp. 4	model	age	failure	predicted failure
0.0	0.0	208.75	28.75	28.75	13.75	(0,1,0,0)	17	Comp.1	Comp.1
0.0	0.0	209.75	29.25	29.25	14.24	(0,1,0,0)	17	none	none

3.2.1 Random Forest Classifier

It is one among the top performers for classification and regression tasks. A random forest is a sum of decision trees. Random forests combine many decision trees in order to reduce the risk of over fitting. Decisions trees are very used, easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions [4].

We read the confusion matrix numbers along the diagonal as correctly classifying the component failures. Numbers above the diagonal indicate the model incorrectly predicting a failure when non occurred, and those below indicate incorrectly predicting a non-failure for the row indicated component failure.

Table 7. Confusion Matrix

predicted_failure	component1	component2	component3	component4	none
failure					
component1	1298	48	2	3	700
component2	19	1885	5	4	764
component3	7	21	387	5	410
component4	26	35	2	929	306
none	126	254	85	106	179073

Confusion matrix:

Accuracy = 0.984301
Precision = 0.857442
Recall = 0.673604
F1 = 0.754486

3.2.2 Gradient Boosting Classifier

Gradient boosting combines weak "learners" into a single strong learner in an iterative fashion, typically decision trees [2]. GBDT training generally takes longer because of the fact that trees are built sequentially. However, benchmark results have shown GBDT are better learners than Random Forests.

Table 8. Confusion matrix:

predicted_failure	component1	component2	component3	component4	none
failure					
component1	1516	39	9	14	473
component2	30	2001	26	23	597
component3	7	13	534	15	261
component4	18	24	3	1022	231
none	224	315	143	126	178841

Confusion matrix:

Accuracy = 0.986108
Precision = 0.831367
Recall = 0.764582
F1 = 0.7965776

3.2.3 Naive Bayes Model

Naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. It only requires a small number of training data to estimate the parameters necessary for classification.

Despite it's simple architecture Naive Bayes classifiers works well in real world situations and that is the reason behind choosing this method.

Table 9. Confusion matrix:

predicted_failure	component1	component2	component3	component4	none
failure					
component1	745	51	70	48	1137
component2	33	1243	78	50	1273
component3	4	7	675	10	134
component4	8	13	14	1035	228
none	1149	2193	4368	2943	168996

Confusion matrix:

Accuracy = 0.925948
Precision = 0.250933
Recall = 0.571561
F1 = 0.348753

3.2.4 Evaluation

We compared the actual failures « failure » listed in the test data with the « predicted failure » of the component, having computed the confusion matrix which displays predicted failures in column and actual component failures in rows.

Analyzing the evaluation metrics, we can say that Gradient Boosting has highest F1 score =0.55 and therefor it's the best classifier for our data however it requires more time than other algorithms. Random Forest is also a good classifier having the F1 score = 0.52 It also requires significantly less time than Gradient Boosting classifier. Naïve Bayes converges very fast which is below 1 second.

References

1. Azure AI guide for predictive maintenance solutions
2. Wikipedia Gradient boosting
3. Advanced Machine Learning and Signal Processing, IBM, Coursera
4. Microsoft Decision Trees Algorithm
5. C.E. Coteț, C.L. Popa, G. Enciu, A. Popescu, T. Dobrescu. Using CAD and flow simulation for educationalplatform design and optimization, Int. J.Simul. Model, **15**(1) ISSN 1726-4529 pp. 5-16, DOI:10.2507/IJSIMM12(2)2.225, Accession Number: WOS: WOS:000377105400001 (2016)
6. D. Popescu, D. Anania, C.E. Coteț, C. Amza. Fullyautomatedliquidpenetrant inspection line simulation model for increasingproductivity, Int. J.Simul. Model **12**(2), ISSN online: 1740-2131, ISSN print: 1740-2123 pp. 82-93, DOI: 10.2507/IJSIMM12(2)2.225 (2013)
7. G. Dragoi, C.E. Cotet, L. Rosu, S.M. Rosu. Role of the virtual networks in the Virtual Enterprise, The Journal of Mechanical Engineering Nr. 7-8, pp. 526-531, Ljubljana, Slovenia, ISSN 0039-2480 (2006)