National College of Ireland

Improving predictive maintenance classifiers of industrial sensors' data using entropy. A case study.

MSc Research Project
Data Analytics

Eleonora Peruffo
14106761

School of Computing
National College of Ireland

Supervisor: Noel Cosgrave

National College of Ireland
Project Submission Sheet – 2017/2018
School of Computing

Student Name:	Eleonora Peruffo
Student ID:	14106761
Programme:	Data Analytics
Year:	2016
Module:	MSc Research Project
Lecturer:	Noel Cosgrave
Submission Due Date:	13/08/2018
Project Title:	Improving predictive maintenance classifiers of industrial sensors' data using entropy. A case study.
Word Count:	5175

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	
Date:	12th August 2018

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
3. Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Improving predictive maintenance classifiers of industrial sensors' data using entropy. A case study.

Eleonora Peruffo
14106761
MSc Research Project in Data Analytics

12th August 2018

Abstract

The increase in the availability of sensors' data in manufacturing (Industrial Internet of Things, IIOT) poses the challenge on how best to use this information. One of the emerging applications of data analysis in this field is predictive maintenance: being able to identify when and why a certain component breaks down and empower early intervention to prevent breakdowns. Imbalanced datasets literature shows that tree models perform better with entropy splits than Gini index splits. Entropy measures applied in previous studies in the domain of industrial sensors' data include not only Shannon's but also Renyi and Tsallis. This paper looks at the performance of classification trees using different entropies applied to the Scania trucks dataset. In this case, the best performing tree is a C5.0 model but we confirm that Renyi and Tsallis entropy trees can improve classification of the minority class in the data without excessively penalising the classification of the majority class. These models can therefore help to prevent companies costs by improving the identification of possible failures and avoiding unnecessary interventions on well-working equipment.

1 Introduction

The manufacturing industry has been implementing quality checks on stock production for a long time. Until recently, on top of visual and physical inspection, random sampling or processes similar to Six Sigma have been used to monitor the output of a production process (Lade et al.; 2017). Monitoring product quality in manufacturing is important because a prompt identification of defects (aesthetic, weight, length, treatment of raw material, production environmental factors, etc.) can help manufacturers to intervene and adjust the process to achieve required standards, for example measuring that a certain weight range is respected for each batch of biscuits boxes. The development of advanced manufacturing, or industry 4.0 as it is known in Europe, technologies such as Industrial Internet of Things (IIoT -sensors), advanced industrial robotics, additive manufacturing, deep learning, made available at a more affordable price sophisticated tools which can enhance and automate manufacturing processes. These technologies contributed to the introduction of the 'predictive maintenance' concept which goes beyond traditional quality control techniques (Spendla et al.; 2017). The application of sensors across the

production process allows to gather a huge and detailed amount of production process data (Yan et al.; 2017; Alam et al.; 2016) and ideally to monitor stock along the entire supply chain (Jayaram; 2016; Mourtzis et al.; 2016). These data can be analysed with specially designed algorithms which can deal with highly imbalanced dataset: in general, a production defect occurs only in a small part of the full productionbatch of products hence an industrial sensors dataset contains a very small number of observation related to defective parts.
Predictive maintenance aims at identifying and analysing trends of production data, trends that can be used to predict when the next process failure will occur (Gu et al.; 2017). The manufacturer can then intervene and fix the problem before it happens (Canizo et al.; 2017; Sharp et al.; 2018; Wang et al.; 2018). Predictive maintenance increases advantages deriving from traditional quality controls: reparation costs are reduced in terms of downtime, for example by fixing a metal part before tear-and-wear breaking point; by reducing the risk of product returns by unsatisfied customers; or by avoiding material waste caused by defective parts discharge (Lade et al.; 2017).
Predictive maintenance can be applied not only on the factory shop floor but also for the maintenance of machines which are sold. Sensors can transmit data to the manufacturer: the analysis of data usage is not limited anymore to computer and mobile devices but can be applied to vending machines, photocopiers and, on a bigger scale, to wind turbines, tractors, trucks, and aircrafts (El Afia and Sarhani; 2017; Li et al.; 2017). Similar to reparation costs in the factory, the identification of future failure on machinery and vehicles can save costs in terms of product life cycle and in terms of technicians' inspection routine trips since the technicians can be sent on demand. With the 'servitisation' of manufacturing (Herterich et al.; 2015) traditional manufacturers are now moving towards the provision of real time after-sales services, at the same time acquiring information which can be used to improve the production process. 'Virtual twins' or 'digital twins are digital copies of a machine or vehicles or even of an entire factory built by collecting sensors data, 'digital twins' can be used for example to simulate wind turbines' or airplanes' behaviours (Negri et al.; 2017).

This paper investigates the applicability of the Adaptive Renyi Decision Tree (ARDT) algorithm, previously used to analyse the Bosch production plant dataset, to the Scania trucks dataset to determine if a factory shop floor algorithm can be used for vehicles part preventive maintenance. The first section looks at the literature about previous studies dealing with predictive maintenance. The second section describes the Scania trucks dataset, justifies the selection of this dataset and discusses previous studies about this dataset. The third section presents the analysis and the results and the fourth part presents the conclusion and future work. Sources consulted consist of machine learning and data mining peer reviewed journal articles and conference proceedings on the topic of sensors data use for predictive maintenance purposes.

2 Related Work

2.1 Sensors data and imbalanced datasets analysis

Datasets based on sensors data collect huge amounts of data, for example it has been calculated that a chemical plant can collect up to 2 million raw measurements per hour (Nino et al.; 2017) or a textile machine can produce up to 8.5 GB of measurements per hour (Baban et al.; 2016). These measurements are raw and the task of the data team

is to identify the data in the raw measurement which are needed to give useful insights about the production process. Its important to keep in mind that missing values are sometimes present (e.g. due to transmission faults) and that correct evaluation of those values should be done (Yang et al.; 2014). Then, one of the biggest challenges is to build a model that can identify failures. In a typical manufacturing dataset, the amount of failures is around 1% or less (Lade et al.; 2017), these small subsets pose challenges to the data scientist since even a model reaching 99% accuracy could fail to identify the 1% observations object of the investigation (Lade et al.; 2017; Maurya; 2016). Also, in practical terms for the company, the misclassification in the positive-negative matrix could result in yield loss in case of a good part eliminated (depending on its worth it could be a consistent damage) as bad, or in a bad part ending in the final product. The same logic applies to failure prevention: a false predicted failure might cause an unnecessary maintenance intervention but a failure classified as normal behaviour would entail a delay or possibly a halt of the production process (Lade et al.; 2017).

2.2 Challenges and solutions

While high data dimensionality, traceability and missing data are challenges encountered in any big data dataset, imbalance in observations and nonstationary data, that is data which changes in time due to changes to the production process, are present in industrial sensors data (Lade et al.; 2017). The high number of features, sometimes reaching thousands, makes it difficult to distinguish which features can be used as predictors (Lade et al.; 2017; Susto and Beghi; 2016); features reduction is often applied by either talking with the domain experts, by applying PCA or by using clustering or classification algorithms (Zhang et al.; 2016). Completeness of time series is important because the more data rich the time series is, the better the predictive models can be. Timestamps can be used to infer information for missing data (Lade et al.; 2017) and predictions based on repeated observations across time and presenting non-stationary data can be dealt with ARIMA or MIXED as demonstrated by Sanislav et al. (2016), in their study MIXED performed better. SAFE has been applied by Susto and Beghi (2016): the algorithm does not require features selection and can build very accurate time series. DDEN instead assigns weights to each feature thus preserving the information in changing across time and performs better than logistic regression (Ramakrishnanus and Ghosh; 2015).

2.3 Thresholding

Thresholding is one of the widely applied techniques proposed by data scientist to analyse high dimensional datasets:

> We define the concept of thresholding as a process of determining a decision boundary in the presence of a tunable parameter.

(Hong et al.; 2016). When the maximum value for the parameter is reached, that is the threshold for the decision we are looking at. When analysing the Bosch dataset , which as of June 2018 is one of the biggest IIoT datasets publicly available, a team of researcher which included Bosch's data analysts applied the thresholding method with the ARDT algorithm. This algorithm looks at outliers at every decision node and makes a classification decision (Hong et al.; 2016) by using Renyi entropy so that the maximum value of is selected for the prior class distribution; hence the threshold is given by

searching this value of which can vary from 0 to 1. The algorithm has been tested on the Bosch dataset and on other datasets Hong et al. (2016). The use of classification trees with Renyi entropy has not been tested on the Scania trucks dataset. Furthermore, the Bosch dataset has been exploited by a few authors, both Bosch employees and Kaggle competitions participants. Another manufacturing sensors' data dataset should be used to test the algorithms and techniques, it would allow better comparison and perhaps novel approaches to the same problems.

2.4 Scania trucks dataset

Five studies looked at the Scania truck dataset aiming at accurately predicting failures. In this section We compare the different decisions made by the data scientists who already worked on this dataset: treatment of missing values, algorithm, evaluation of results and cost achieved.
Gurung et al. (2015) didn't have to deal with missing data since they had used the proprietary version of the data. The rest of the papers were prepared as part of an industrial data challenge with anonymised data with some missing values. Different approaches were chosen: Gondek et al.(2016) went for replacement of missing values with the median value of the feature while Ozan et al.(2016) assumed that values where missing completely at random and didn't take any action. Cerqueira et al. (2016) deleted missing values rows completely and decided to use SMOTE to create synthetic examples. This choice is actually a good one since it eliminates incomplete cases and at the same time allows for a better distribution of the classes. There is a drawback in their approach though, deleting missing values reduces the dataset to 591 instances which we think is too small, even if synthetic examples are created from these 591 observations. A dataset should have a number of observations at least 10 times the number of features. Finally, a soft impute algorithm where maximum likelihood estimates are calculated for each fold was utilised by Ferreira Costa and Nascimento (2016).

Feature engineering consists in creating new features from functional combination of existing ones, this process brings advantages because by grouping or combining existing features this can reduce the number of columns in the dataset and thus reduce the training time of the model. Feature engineering can also be used to increase the number of features in the dataset and in case of the Scania dataset resulted in a final number of features of 282 (16 new features for each histogram)(Gondek et al.; 2016). This approach seems rather convoluted since after creating the new features, feature selection has to be used. Indeed, Gurung et al. (2015) preferred to consider adiacent bins only in histograms thus reducing the number of features. Another solution applied to the Scania dataset is metafeature engineering (Cerqueira et al.; 2016), advantageous because it doesn't require domain knowledge, and which in this case, consists in detecting outliers with three different techniques (Boxplot analysis, LOF, and clustering based outliers ranking). The combination of these measures can help to identify that probably there is an anomaly in an observation.

The full list of algorithms tested on the dataset can be found in Figure 1. As this dataset was the object of a competition all the authors except for the creators (method area under the curve, AUC) used the cost of the prediction as the evaluation method. The dataset's creators, Gurung et al. (2015), looked at the possibility of predicting failures looking at the histograms classification although the algorithm didn't present clear prediction advantages on the other methods and was slower to train. Two studies (Fer-

Figure 1: Previous work on Scania dataset

Authors	Missing data	Feature engineering	Classifier applied to the dataset	Best result given by
Ferreira Costa & Nascimiento (2016)	Soft-Impute algorithm		Logistic regression, K-NN, SVM, Decision trees, Random Forests	Random forest was 92.6% better than the random baseline
Gondek et al. (2016)	Replace missing values with the median value	16 different features for each histogram.	Random forest with thresholding for every feature; changed it in steps of one percent.	Random forest
Cerqueira et al. (2016)	Deleted missing values. SMOTE: over-sampling technique that creates synthetic examples of the minority class. SMOTE enables to balance the class distribution of the data which leads to a better generalization of the classifier.	Metafeature engineering approach that does not require domain knowledge.	Random forest Random forest with meatfetures XGBoost XGboost with meatfeatures 10k fold cross -validation	XGboost with metafeatures
Ozan et al. (2016)	Assume that missing data is missing completely at random. Use k-NN		SVM, AdaBoost Random Forests, k-NN 5 fold cross-validation	K-NN with optimisation. This approach does not produce a model, difficult to get novel novel insights.
Gurung et al. (2015)	Used proprietary version of dataset with no missing values	Histograms and bin merges.	Histogram decision tree learning algorithm.	Histogram approaches did not result in clear advantages: longer training times, only captures linear patterns.

reira Costa and Nascimento; 2016; Gondek et al.; 2016) found that the random forest algorithm gave the best result while one found that boosted trees (Cerqueira et al.; 2016) was better. A different approach, meaning the use of an unsupervised algorithm, K-NN, was applied by Ozan et al. (2016), although this method does not allow to produce a model so it has limits in terms of long term applicability for preventive maintenance. The cost-wise evaluation of the dataset sees the methodology applied by Ferreira et al. (2016), winners of the competition, achieving a cost of 9920 while Cerqueira et al. (2016) achieved a cost between 3000 and 4000.

3 Methodology

The most common parameter configuration for classification trees is the use of information gain ratio measured with Gini which usually yields good results. If the Gini parameter doesn't give satisfying classification results, the parameter criterion that can be used is information gain calculated with Shannon entropy. As shown in the related work section of this paper (Hong et al.; 2016) and more widely in domains with similar issues, experts found that trees using Renyi entropy perform better for classification of highly imbalanced datasets.

3.1 Entropy

Entropy is a concept borrowed from physics where it expresses the measure of energy in a system. Information theory uses entropy as the measure of information present in a set

of data, entropy is at its maximum when the distribution of classes in the set is even (e.g. in a set with two classes the distribution should be 50/50). Classification trees, using entropy as the splitting criterion, split the tree according to the attributes which give more information at each split, that is the ones with highest entropy (Lima et al.; 2010). The difference between classification trees using Shannon entropy and using Gini index is that Gini index uses the measure of impurity of a node to achieve the minimum impurity possible (Kuhn and Johnson; 2013). The most used measure of entropy is Shannon entropy which is measured as the proportion of the number of positives multiplied by its log in base 2; and summed to the proportion of the number of negatives multiplied by its log in base 2. Given pP= portion of positive observations and pN portion of negative observations, the formula for Shannon entropy is:

$$-pP * log2(pP) - pN * log2(pN)$$

Renyi entropy is a generalisation of Shannon entropy and the formula is where α is a number between 0 and 1:

$$1/1 - \alpha * (-pP * log2(pP) - pN * log2(pN))$$

and Tsallis entropy formula is

$$1/1 - \alpha * (-pP * log2(pP) - pN * log2(pN))$$

where values of $\alpha > 1$.

> 'It is important to note that using Shannon entropy, events with high or low probability do not have different weights in the entropy computation'(Lima et al.; 2010).

The use of an entropy different from Shannon entropy can alleviate the problem of improving the precision for the minority class at the expense of the precision for the majority class (Hong et al.; 2016). For $\alpha > 1$ Lima et al. (2010) demonstrate that when using Tsallis entropy the value of α is proportionally determined by the bigger probability of classifying an event.

3.2 Description of the dataset

The Scania trucks dataset contains 60,000 observations in the training set and 1,600 in the test set. The dataset collects information about failures in the air pressure system(APS): in the training set 59,000 (not related to APS failure) observations are negative and 1,000 are positive (related to APS failure). Although the number of observations in this dataset is not in the order of millions we believe that it's enough to be used as a case example to test the tree using Renyi entropy: the imbalance (1% of positive observations) is present in a proportion similar to bigger sensors' data datasets. The model should take into account that there are different costs in predicting a true positive versus a false one: cost_1 is the cost of an unnecessary check while cost_2 is the cost for a truck that needed repair but has not been identified. The model should be evaluated according to the following formula:

$$Total\_cost = Cost\_1 \times No\_Instances + Cost\_2 \times No\_Instances$$

3.3 Missing values

The calculation of the prior distribution of the classes is 0.016 which is quite lower than the one used in previous works where it was around 0.16. If major issues are met in the analysis of the data as is, SMOTE could be used to improve the class balance in the dataset. However we will not use it in the same way as Cerqueira et al. (2016) but look at implementing it without eliminating all missing data. We think that 591 observations would be too small as a sample.

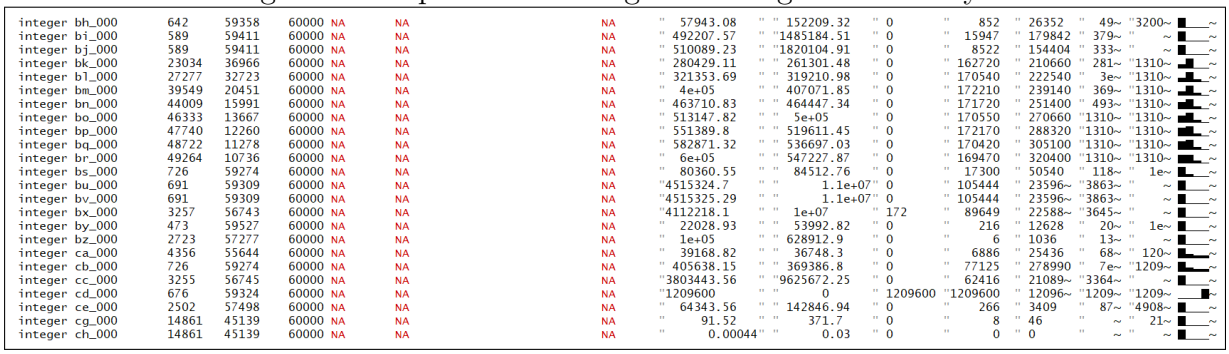
3.4 Data models

The aim of the paper is to check if different preprocessing and classification methods can improve the results obtained in previous work. In particular, since the work of Lima (2010) and Hong et al. (2016) showed that Renyi tree can improve classification performance on imbalanced data. However, it's important to check other models as well to be able to compare the Renyi tree performance. The work will entail building a C5.0 tree, an rpart tree, a random forest and a J48 (using RWeka (Hornik et al.; 2009)) which should allow to customise entropy. To check performance in different environments, scikit-learn (Pedregosa et al.; 2011) the classification trees module in Python, which also provides an implementation of J48, will be explored. Implementation methods which entail the modification of an existing classifier are suggested by Lima et al. (2010). The work of Lima et al. (2010) has been developed in a doctoral thesis by Acker (2015) where the project setup and a list of edits to WEKA java classes is proposed. Since this process is described in a way that makes it possible to reproduce it, this option will be kept in mind, but first attempts should take place in the R environment. All models will be tuned with 10 folds crossvalidation and then tested on the test set.

4 Implementation

Dataset preprocessing was done in R, the approach chosen follows Gondek et al. (2016) who consider the data missing completely at random and impute the median value of the column for missing data. We agree with this approach since the dataset has been anonymised and missing data serve the purpose of preventing too much insights on business sensitive data. We contemplated the use of the Multivariate Imputation by Chained Equations (MICE) package (van Buuren and Groothuis-Oudshoorn; 2011) but the package is more apt to impute data where there is more knowledge about each variable. In this case the post-imputation diagnostic would not be informative since we don't know which values could be plausible: they could be values included in the min-max range of the variable or outside these boundaries. Therefore, the middle value seems a reasonable choice. A good library to use for missing data when there is around a hundred of features is 'skimr', it allows to plot missing data for each variable and plots an inline histogram. In comparison to Amelia it allows to output a table where it is not necessary to zoom to be able to read the variables' names (Figure 2). The preprocessed dataset has been used in 3 different environments: R, Python and Weka. The 'class' column which in the original dataset was the first column of the dataset, has been moved to the end of the dataset to make it compatible with Weka. The two class values neg and pos have been changed from string to numerical '0' and '1' to allow easier manipulation in Python: if the class is left as a string the script does not run.

Figure 2: Snapshot of missing data using skimr library



integer	bh_000	642	59358	60000	NA	NA	NA	57943.08	152209.32	0	852	26352	49~	3200~	▇▁▁▁▁
integer	bi_000	589	59411	60000	NA	NA	NA	492207.57	1485184.51	0	15947	179842	379~	~	▇▁▁▁▁
integer	bj_000	589	59411	60000	NA	NA	NA	510089.23	1820104.91	0	8522	154404	333~	~	▇▁▁▁▁
integer	bk_000	23034	36966	60000	NA	NA	NA	280429.11	261301.48	0	162720	210660	281~	1310~	▃▇▁▁▁
integer	bl_000	27277	32723	60000	NA	NA	NA	321353.69	319210.98	0	170540	222540	3e~	1310~	▃▇▁▁▁
integer	bm_000	39549	20451	60000	NA	NA	NA	4e+05	407071.85	0	172210	239140	369~	1310~	▃▇▁▁▁
integer	bn_000	44009	15991	60000	NA	NA	NA	463710.83	464447.34	0	171720	251400	493~	1310~	▃▇▁▁▁
integer	bo_000	46333	13667	60000	NA	NA	NA	513147.82	5e+05	0	170550	270660	1310~	1310~	▃▇▁▁▁
integer	bp_000	47740	12260	60000	NA	NA	NA	551389.8	519611.45	0	172170	288320	1310~	1310~	▃▇▁▁▁
integer	bq_000	48722	11278	60000	NA	NA	NA	582871.32	536697.03	0	170420	305100	1310~	1310~	▃▇▁▁▁
integer	br_000	49264	10736	60000	NA	NA	NA	6e+05	547227.87	0	169470	320400	1310~	1310~	▃▇▁▁▁
integer	bs_000	726	59274	60000	NA	NA	NA	80360.55	84512.76	0	17300	50540	118~	1e~	▇▁▁▁▁
integer	bu_000	691	59309	60000	NA	NA	NA	4515324.7	1.1e+07	0	105444	23596~	3863~	~	▇▁▁▁▁
integer	bv_000	691	59309	60000	NA	NA	NA	4515325.29	1.1e+07	0	105444	23596~	3863~	~	▇▁▁▁▁
integer	bx_000	3257	56743	60000	NA	NA	NA	4112218.1	1e+07	172	89649	22588~	3645~	~	▇▁▁▁▁
integer	by_000	473	59527	60000	NA	NA	NA	22028.93	53992.82	0	216	12628	20~	1e~	▇▁▁▁▁
integer	bz_000	2723	57277	60000	NA	NA	NA	1e+05	628912.9	0	6	1036	13~	~	▇▁▁▁▁
integer	ca_000	4356	55644	60000	NA	NA	NA	39168.82	36748.3	0	6886	25436	68~	120~	▇▂▁▁▁
integer	cb_000	726	59274	60000	NA	NA	NA	405638.15	369386.8	0	77125	278990	7e~	1209~	▇▂▁▁▁
integer	cc_000	3255	56745	60000	NA	NA	NA	3803443.56	9625672.25	0	62416	21089~	3364~	~	▇▁▁▁▁
integer	cd_000	676	59324	60000	NA	NA	NA	1209600	0	1209600	1209600	12096~	1209~	1209~	▁▁▁▁▇
integer	ce_000	2502	57498	60000	NA	NA	NA	64343.56	142846.94	0	266	3409	87~	4908~	▇▁▁▁▁
integer	cg_000	14861	45139	60000	NA	NA	NA	91.52	371.7	0	8	46	~	21~	▇▁▁▁▁
integer	ch_000	14861	45139	60000	NA	NA	NA	0.00044	0.03	0	0	0	~	~	▇▁▁▁▁

To the best of our knowledge, there are no ready-to-use algorithms using Renyi entropy as a parameter. So, the challenge to this part of the research was to either write a classification tree from scratch or find a way to modify an existing classification method. Writing the algorithm from scratch is not trivial since even if a tree is built, all the other functions from plotting to getting the summary need to be built as well. The literature review of the selected papers on sensors' data doesn't bring up detailed information about the implementation method.

Looking beyond the selected domain, information on trees using entropy is found especially for the intrusion detection domain, which also deals with imbalanced datasets. In order to better understand the Scania trucks dataset a few models were built with R as preferred tool. However even using parameter tuning, the possibility of entering a measure of entropy different from Shannon required a modification of the R library. The current implementation of the C5.0 algorithm only implements Gini splits. The library by Hornik et al. (2009) provides an R interface to build J48 trees within R, the criteria are Gini and Shannon entropy. In our quest to build a Renyi tree, the other language chosen for modelling is Python since the scikit-learn Pedregosa et al. (2011) module offers a variety of trees. Again the out-of-the box solution doesn't provide for a change in the entropy criterion.

The module python-weka-wrapper offers the implementation of WEKA's classifiers in Python and the possibility to call java classes from within a java virtual machine. But, since in this case WEKA seemed to be the interface that best allowed for the modification of existing models, it made sense to work on WEKA directly. A modified WEKA application has been implemented following Acker (2015): in his paper he proposes one modified java class and four new java classes to calculate Renyi and Tsallis entropy. We compiled his classes into a new jar file which can run the modified WEKA program through the Eclipse Java IDE. The WEKA interface appears as usual but when the J48 algorithm is selected the entropy is calculated reading the entropy type and values from a text file. Running the J48 tree just with Renyi and Tsallis and changing the α parameter didn't bring an overall improvement of the classifier when looking at the cost since a > 100 False positive are reported, see table 3. In the modified Weka jar the alpha parameter is an external input so it's not possible to run an automatic search on this. Trials were run for values of alpha from 0.1 to 0.9. In terms of feature selection, note that the nature of classification trees is to perform feature selection so it is no wonder that running the model with the top features didn't improve the performance, of course it sped up processing time. Using the J48 classifier with Renyi entropy features selection bestsplit results in the

list of the following features:ag_000, ag_002, ak_000, ay_005 while greedystepwise α=0.5 in the list of these features:ag_0002,ak_000,ay_005,da_000.

We followed Hong et al.(2016) and proceeded with thresholding. The WEKA 'Threshold Selector' function performs a threshold optimisation search before implementing the J48 classifier. The function was set to optimise the result for the minority class, with 5 fold crossvalidation. The optimisation threshold search did improve the performance of the classifier in comparison to the J48 model without thresholding, but again didn't beat the best model in C5.0.

Since the Renyi tree didn't perform better than other classifier we looked at the previous distribution for the data and decided to diminish the class imbalance by applying SMOTE to the processed data. The algorithm was set to give us a distribution of >= 0.16 which is the distribution that Hong et al. (Hong et al.; 2016) found in the datasets that they tested.

Up to this point we ran tests on the dataset with median imputing, we decided to test some models on the dataset with an improved distribution: we created synthetic examples to achieve a prior distribution of 0.16. The tree we used is J48 with $\alpha = 0.7$ and confidence factor = 0.25 which was the one that performed better in our α search. In this case, accuracy and Kappa would be two good metrics to take into consideration since while accuracy will remain high Kappa will give an indication of how the expected values fared.

5 Evaluation

The models' performance is evaluated through accuracy and through cost of the model. Accuracy is the standard measure to check how many instances are correctly classified but at a superficial look this paper would seem it contains exceptional results since all classifier have an accuracy > 90%. However, since this is an imbalanced dataset, 'cost' is the best indication of the classifier's efficacy,the lower the number of false positives, the more money the company can save by repairing trucks before they break down.

5.1 Experiment / Models built in R

Table 1 reports the list of the models built in R and their performance.

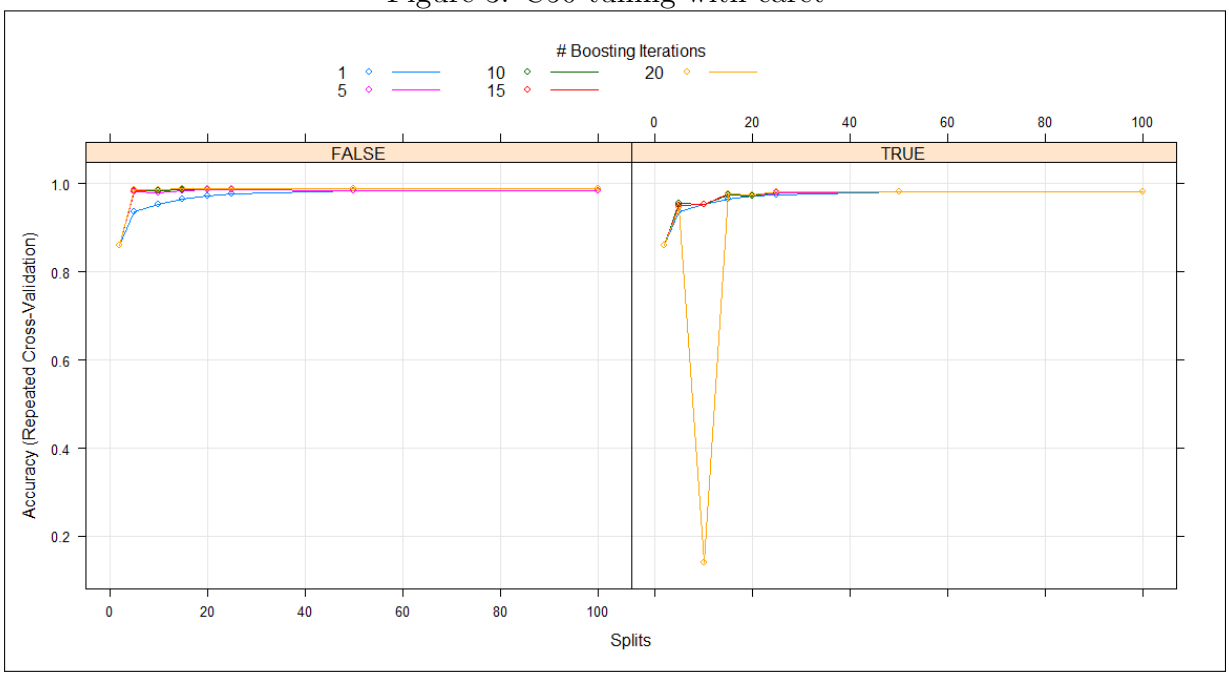
Model	**library**	**Accuracy**	**Cost**	**FN**	**FP**
rpart gini	rpart	0.98825	58,600	110	360
rpart information	rpart	0.985938	76,360	152	36
C 5.0	C 5.0	0.990188	16,760	31	126
C 5.0 100 trials	C 5.0	0.9928	9,980	18	98
random forest	random forest	0.992438	10,030	18	103
Naive Bayes	naiveBayes	0.96475	264,360	528	36
random forest	RWeka	0.96235	51,680	103	18
J48	RWeka	0.96235	36,160	72	16

Table 1: Models built in R

Parameter tuning was performed on the best model, C5.0. The tuning was performed with the caret package, 10 fold cross validation, on a grid looking at n tri-

als = $1, 5, 10, 15, 20$ and splits from a very shallow tree to a deep one (n splits = $2, 5, 10, 15, 20, 25, 50, 100$). The final values used for the model were trials = 20 and splits = 100 (Figure 3). However, when tested, this did not achieve the same result as the tree with 100 trials.

Figure 3: C50 tuning with caret



5.2 Experiment / Models built in Python

The package Scikit-learn in Python has been used to build and compare trees. The best model, with a cost of 19,730 (17 FN and 1123 FP) was using the criteria: entropy,class weight and minimum impurity decrease(0.5). This indicates that impurity, that is the disorder in the dataset, is a parameter that can be used to improve the tree. Table 2 presents the results obtained (Scikit_learn trees. *ent=entropy, w=class weight, i=impurity).

Models	**Accuracy**	**FN**	**FP**	**Cost**
DT gini	98.7375	64	129	65,140
DT ent	98.74375	67	115	58,170
DT ent w	98.74375	74	115	58,240
DT ent, w, i	92.875	17	1123	19,730

Table 2: Models built in Python.

Among the trees built with scikit-learn (Pedregosa et al.; 2011), the best performing model utilised Shannon entropy as split criteria, this change didn't improve the prediction compared with the Gini-based split. Similarly, the the introduction of weights for the classes did not improve the results. The final attempt, introducing the measure of impurity which in scikit-learn (see documentation) splits a node only if there is a decrease in the impurity of the node. This improved significantly the number of FP, only 17, but

greatly increased the FN to 1123. This solution is not optimal because it means that a high number of efficient vehicles would be taken off service for no reason.

5.3 Experiment / Models built on modified WEKA.gui

The WEKA implementation of J48 trees using different entropy measures does not achieve an improvement on the false negatives prediction compared to models built in R. The J48 tree which performed best was the one using Tsallis entropy for α=1.2 and confidence factor=0.25.Both Tsallis and Renyyi trees performed better than Shannon trees. Table 3 presents results obtained by changing entropy measures in WEKA. To note that the threshold search improved the false negative class prediction of the model but the cost of the classifier remained high ($> 50,000$). Feature selection was also implemented for 'bestsplit'(looks for global optimum by measuring all possible splits at each stage) and 'greedystepwise' (finds local optimum and moves on) but the J48 trees based on a reduced dataset performed of course faster but not better in terms of predictions (the features are the ones the classifier would use first in any case to do the split).

Entropy	**Validation**	**Alpha**	**C**	**FP(cost 10)**	**FN(cost 500)**	**Cost**
Renyi	10 fold	0.5	0.5	57	229	115070
Renyi	10 fold	0.7	0.5	34	191	95840
Tsallis	10 fold	1.2	0.25	64	128	64640
Tsallis	10 fold	1.2	0.5	61	129	65110
Shannon	10 fold	na	0.25	188	318	160880
Shannon	10 fold	na	0.5	206	309	156560

Table 3: J48 models built in modified WEKA.gui.

5.4 Discussion of results

Creating trees in different environments allows to explore and test different solutions. Options don't always match so a direct comparison cannot be done but the cost of the model offers a way to compare results across models but also across environments (see configuration manual for details of each setup). Among the tested algorithms on the three applications (R, Python, Weka-jar) the best result, that is the one with the lowest cost, was achieved by the C5.0 algorithm with 100 trials. The cost was $9,980$. We also performed a grid search for other trials but 100 (the maximum in caret) was the best. This confirms, as found in the previous works on this dataset, that parameter tuning through boosting is a feasible option to improve predictions. Similarly, the adaboosting metaclassifier on J48 does display an improvement in the classification of false positives without increasing false negatives exponentially; so the observation of Hong et al. (2016) holds true, but in this case the cost of the J48 Renyi entropy tree remains high. Previous works looked at finding a model which best reduces the cost of repairing interventions: in our case the C5.0 model achieved the best results. In comparison with previous work, the results are similar to those of Ozan et al. (2016), positioned second after Cerqueira et al.(2016) whose prediction was in the $3000 - 4000$ cost range.

In this dataset the prior distribution is 0.016 while, in all the datasets tested in Hong et al. (2016), prior distribution was > 0.16. Following the creation of synthetic samples, we tested the difference on the best performing Renyi tree ($\alpha = 0.7$, $C = 0.25$, $M = 2$, 10

fold validation) on both datasets. The 'before' dataset had an accuracy of 98.62 an Kappa of 0.475 and 169 FP and 38 FN. The 'after' had accuracy of 98.01 and kappa of 0.61, observations misclassification resulted in 109 FP and 209 FN. In this case, this shows that an improvement in the prior distribution did have an effect on the performance of the Renyi tree and the majority class misclassification increase wasn't so steep. In the future, it would be worth performing more experiments on the synthetic examples dataset.

6 Conclusion and Future Work

This paper investigated the applicability of entropy classification trees on the Scania dataset. Results demonstrate that in this particular case, probably due to the very low prior distribution of the classes, Renyi trees did not improve prediction costs. Though, the distinctive advantage of the Renyi tree model, improving labelling of minority class without increasing the number of false negatives held true. Further processing of the dataset to achieve a class distribution of 0.16 showed better results but not enough to reduce the cost at the level of the C5.0 tree. Ultimately it's up to the analyst to decide which model best responds to the business needs. In this case, the best cost-wise solution resulted to be the C5.0 model.
As seen from the literature, the use of different entropies remains an important feature for data modelling. Future work could look at adapting the java classes to the current WEKA version or at developing a modified J48 with Python with the WEKA wrapper to allow easier selection of parameters, especially a grid search for α values. For R, the RWeka library could include an extension of the entropy criterion. Given that a Gini classification tree with 100 trees and a random forests have shown a good performance; a future work hypothesis could look at implementing a random forest with Renyi entropy to improve the sensitivity of a model by diminishing the number of false positives and at the same time by not increasing the false negatives out of proportion.
Classification trees and random forests are not the only solution applied to imbalanced datasets. Future work on this dataset could look at the performance of the approach proposed by Sonak et al.(2016) which combines genetic algorithm for feature selection, k-means for clustering and ANN to compare thresholds.

References

Acker, F. (2015). *Use of Entropy for Feature Selection with Intrusion Detection System Parameters*, Doctoral dissertation., Nova Southeastern University. Date accessed: 15/07/2018.
URL: *https://nsuworks.nova.edu/gscis_etd/370*

Alam, F., Mehmood, R., Katib, I. and Albeshri, A. (2016). Analysis of Eight Data Mining Algorithms for Smarter Internet of Things (IoT), *Procedia Computer Science*, Vol. 58, pp. 437–442. Date accessed: 06/11/2017.
URL: *https://www.sciencedirect.com/science/article/pii/S187705091632213X*

Baban, C. F., Baban, M. and Suteu, M. D. (2016). Using a fuzzy logic approach for the predictive maintenance of textile machines, *Journal of Intelligent & Fuzzy Systems* **30**(2): 999–1006.

Canizo, M., Onieva, E., Conde, A., Charramendieta, S. and Trujillo, S. (2017). Real-time predictive maintenance for wind turbines using Big Data frameworks, *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–8.

Cerqueira, V., Pinto, F., Sà, C. and Soares, C. (2016). Combining Boosted Trees with Metafeature Engineering for Predictive Maintenance, *in* H. Boström, A. Knobbe, C. Soares and P. Papapetrou (eds), *Advances in Intelligent Data Analysis XV*, Vol. 9897 of *Lecture Notes in Computer Science*, INESC TEC, Universidade do Porto, Springer International Publishing, Cham, pp. 393–397.

El Afia, A. and Sarhani, M. (2017). Particle Swarm Optimization for Model Selection of Aircraft Maintenance Predictive Models, *Proceedings of the 2nd international Conference on Big Data, Cloud and Applications - BDCA'17* pp. 1–12.

Ferreira Costa, C. and Nascimento, M. A. (2016). IDA 2016 Industrial Challenge: Using Machine Learning for Predicting Failures, *Advances in Intelligent Data Analysis XV 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings*, Vol. 9897, pp. 381–386.

Gondek, C., B, D. H. and Sampson, O. R. (2016). Prediction of failures in the air pressure system of scania trucks using a random forest and feature engineering, *Advances in Intelligent Data Analysis XV* **9897**: 398–402.

Gu, C., He, Y., Han, X. and Chen, Z. (2017). Product quality oriented predictive maintenance strategy for manufacturing systems, *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, IEEE, Harbin, pp. 1–7.

Gurung, R. B., Lindgren, T. and Bostr, H. (2015). Learning decision trees from histogram data using multiple subsets of bins, *in* R. Stahlbock and G. M. Weiss (eds), *Proceedings of the 2015 International Conference on Data Mining: DMIN 2015*, CSREA Press, pp. 139–145.

Herterich, M. M., Uebernickel, F., Brenner, W. and Boucher, X. (2015). The impact of cyber-physical systems on industrial services in manufacturing, *in* X. Boucher and D. Brissaud (eds), *Procedia CIRP 7th Industrial Product-Service Systems Conference - PSS, industry transformation for sustainability and business*, Vol. 30, Elsevier, pp. 323–328.
URL: *https://www.sciencedirect.com/journal/procedia-cirp/vol/30*

Hong, C., Ghosh, R. and Srinivasan, S. (2016). Dealing with class imbalance using thresholding, *ACM SIGKDD 2016 Workshop on Outlier Definition, Detection, and Description on Demand*, San Francisco.

Hornik, K., Buchta, C. and Zeileis, A. (2009). Open-source machine learning: R meets Weka, *Computational Statistics* **24**(2): 225–232.

Jayaram, A. (2016). Lean six sigma approach for global supply chain management using industry 4.0 and iiot, *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, IEEE, pp. 89–94.

Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modeling [Hardcover]*, Springer, New York, NY.

Lade, P., Ghosh, R. and Srinivasan, S. (2017). Manufacturing analytics and industrial Internet of Things, *IEEE Intelligent Systems* **32**(3): 74–79.

Li, S., Yang, Y., Yang, L., Su, H., Zhang, G. and Wang, J. (2017). Civil Aircraft Big Data Platform, *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*, IEEE, pp. 328–333.

Lima, C. F. L., d. Assis, F. M. and d. Souza, C. P. (2010). Decision tree based on shannon, rnyi and tsallis entropies for intrusion tolerant systems, *2010 Fifth International Conference on Internet Monitoring and Protection*, pp. 117–122.

Maurya, A. (2016). Bayesian optimization for predicting rare internal failures in manufacturing processes, *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 2036–2045.

Mourtzis, D., Vlachou, E. and Milas, N. (2016). Industrial big data as a result of iot adoption in manufacturing, *Procedia CIRP* **55**: 290 – 295. 5th CIRP Global Web Conference - Research and Innovation for Future Production (CIRPe 2016).

Negri, E., Fumagalli, L. and Macchi, M. (2017). A Review of the Roles of Digital Twin in CPS-based Production Systems, *Procedia Manufacturing*, Vol. 11, pp. 939–948.

Nino, M., Saenz, F., Blanco, J. M. and Illarramendi, A. (2017). Requirements for a big data capturing and integration architecture in a distributed manufacturing scenario, *IEEE International Conference on Industrial Informatics (INDIN)* pp. 1326–1329.

Ozan, E. C., Riabchenko, E. and Kiranyaz, S. (2016). Advances in Intelligent Data Analysis XV, *15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings*, Vol. 9897, pp. 387–392.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**: 2825–2830.

Ramakrishnanus, N. and Ghosh, R. (2015). Distributed Dynamic Elastic Nets : A Scalable Approach for Regularization in, *2015 IEEE International Conference on Big Data*, Santa Clara, CA, USA, pp. 2752–2761.

Sanislav, T., Merza, K., Mois, G. and Miclea, L. (2016). Cyber-physical system dependability enhancement through data mining, *2016 20th IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2016 - Proceedings*, pp. 0–4.

Sharp, M., Ak, R. and Jr, T. H. (2018). A survey of the advancing use and development of machine learning in smart manufacturing, *Journal of Manufacturing Systems* .

Sonak, A., Patankar, R. and Pise, N. (2016). A new approach for handling imbalanced dataset using ann and genetic algorithm, *2016 International Conference on Communication and Signal Processing (ICCSP)*, pp. 1987–1990.

Spendla, L., Kebisek, M., Tanuska, P. and Hrcka, L. (2017). Concept of predictive maintenance of production systems in accordance with industry 4.0, *SAMI 2017 - IEEE 15th International Symposium on Applied Machine Intelligence and Informatics, Proceedings*, pp. 405–410.

Susto, G. A. and Beghi, A. (2016). Dealing with time-series data in Predictive Maintenance problems, *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Vol. 2016-Novem, pp. 0–3.

van Buuren, S. and Groothuis-Oudshoorn, K. (2011). mice: Multivariate imputation by chained equations in r, *Journal of Statistical Software, Articles* **45**(3): 1–67.
URL: *https://www.jstatsoft.org/v045/i03*

Wang, J., Ma, Y., Zhang, L., Gao, R. X. and Wu, D. (2018). Deep learning for smart manufacturing : Methods and applications, *Journal of Manufacturing Systems* pp. 1–13.

Yan, J., Meng, Y., Lu, L. and Li, L. (2017). Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes and Applications for Predictive Maintenance, *IEEE Access* **5**.

Yang, H., Park, M., Cho, M., Song, M. and Kim, S. (2014). A system architecture for manufacturing process analysis based on big data and process mining techniques, *Big Data (Big Data), 2014 IEEE International Conference on* pp. 1024–1029.

Zhang, D., Xu, B. and Wood, J. (2016). Predict failures in production lines: A two-stage approach with clustering and supervised learning, *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 2070–2074.